Signature Search Time Evaluation in Flat File Databases

KWANGIL KO

THOMAS G. ROBERTAZZI, Fellow, IEEE
Stony Brook University

For the first time, divisible load scheduling theory is used to solve for the expected time for searching for both single and multiple signatures in certain multiple processor database architectures. The target architectures examined for illustrative purposes are linear daisy chains and single level tree networks with single and multiple installment load distribution. The use of divisible load modeling and analysis yields elegant expressions for expected search time.

Manuscript received October 18, 2005; revised May 26, 2006; released for publication March 23, 2007.

IEEE Log No. T-AES/44/2/926535.

Refereeing of this contribution was handled by W. D. Blair.

Authors' current addresses: K. Ko, Samsung Electronics, Suwon, Korea; T. G. Robertazzi, Dept. of Electrical and Computer Engineering, Stony Brook University, Light Engineering Bldg., Stony Brook, NY 11794, E-mail: (tom@ece.sunysb.edu).

I. INTRODUCTION

Flat file records are the most basic of database records. In the work presented here, records consisting of very long linear data sequences are considered. It is desired to find certain distinctive patterns or "signatures" in the records by a direct search on a parallel computing system. This may be done in aerospace applications such as radar and sensor data processing, signature and pattern recognition, and signal processing.

Related to this is string matching [6, 11] and template matching [10]. Of course, there are far more sophisticated database record approaches than flat files and in fact early flat file systems are often converted to such models [13, 14]. However for initial raw data processing flat files are a natural choice [15].

A. Aerospace Database Applications

Database technology has been used for many years in aerospace applications. Databases are used both in flight systems and ground support systems.

In 1993 Roth et al. [20] and Glickstein et al. [21] discussed database management as the heart of integrated avionics systems. Real time databases for avionics are examined in Peng and Lin [22]. Spatial and temporal databases are the subject of Bonnor [23] and Vladlamani and de Haag [24]. Finally security and avionics databases are discussed in Roark [25].

Databases for space applications have been discussed in terms of shuttle database integration by Stevens and Componation [26], in terms of the use of XML (extensible markup language) as the underlying structure of the James Webb Space Telescope effort by Detter et al. [27] and in terms of NASA's New Millenium project by Some et al. [28]. Moreover the NASA Technical Report Server contains material on a wide variety of aerospace-oriented databases, ranging from those for turbulence studies, for images, for astronomical data and for flight tests.

Ground support databases include those for air traffic flow management as in Sripada et al. [29]. Support for ad hoc queries for heterogeneous databases is discussed in Adams et al. [30]. Databases for human systems integration in use in the early 1990s appears in Gentner [31]. Finally, information challenges for the aerospace industry are discussed in Sripada [32].

B. Divisible Load Modeling

The novelty of this paper is in applying divisible load scheduling theory for the first time to the problem of finding expected signature search time (i.e., expected time to a "match(es)"). The paradigm of divisible loads has been considered during the past 18 years as a tool for understanding parallel

system performance. Divisible loads are loads which can be arbitrarily partitioned among a number of processors. Scheduling divisible load in parallel system is surveyed in Bharadwaj [4, 5] and Robertazzi [18]. The computation of schedules providing the minimal amount of solution time for linear daisy chain networks of communicating processors is examined in Cheng [7], Mani [16] and Robertazzi [17]. An example of the inclusion of reporting time, the time taken for processors to report the solution back to the originator, is also presented in Cheng [7]. 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 Bataineh [2], Bharadwaj [4] and Cheng [8]. Optimal load allocation for load sharing a divisible job over processors connected by a bus network is considered in Bataineh [1, 2] and Sohn [19]. Drozdowski [34] investigated signature searching, among other computational problems, on networks of transputers in the 1990s.

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.

Elegant expressions are presented here for the expected time to find both single and multiple signatures in specific target architectures. 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 signature search time on other architectures. This work is significant for demonstrating the power of divisible load scheduling theory for predicting search times.

II. 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 situations to be considered involve a linear daisy chain of processors and a single level tree network, as illustrated in Figs. 1 and 2, respectively. The model of a linear daisy chain network or a single level tree network consists of $(M + 1)$ processors. Processor 0 is assumed to be the originating processor

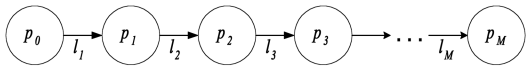


Fig. 1. Model of linear daisy chain network with communication links.

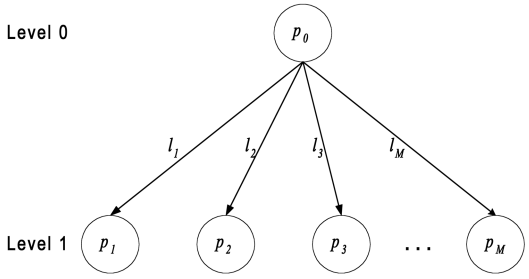


Fig. 2. Model of single level tree network with communication links.

which distributes the fractions of the entire load to M processors. Without loss of generality, it is 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 are used throughout this paper.

α_i: Fraction of entire processing load assigned to ith processor.

w_i: Constant that is inversely proportional to computation speed of ith processor.

z_i: Constant that is inversely proportional to channel speed of ith link.

T_{cp}: Computation intensity: time taken to process a unit (the entire) load on ith processor when $w_i = 1$.

T_{cm}: Communication intensity: time taken to communicate a unit (the entire) load over link l_i when $z_i = 1$.

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

III. EXPECTED TIME OF SEARCHING FOR A SINGLE SIGNATURE

In this section we develop expressions for the optimal amount of load (i.e., flat file load) to transmit over each link and place on each processor for a time optimal solution for various architectures. 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. 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 reporting time is small compared with the load distribution time.

It is assumed that all processors will eventually stop searching their records at the same time if the desired signature is not found. This is the required condition [17, 19] 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 signature, if present, will definitely be found by the finish time.

A. Recursive Equations

In this subsection recursive equations modeling the different scheduling protocols covered here are described. Existing work has developed expressions for the finish time (makespan) of such schedules [4, 33]. For these equations and associated figures, expressions for the time to start searching are listed in the following subsection. This leads to an analysis for expected search time if there is a single signature in the data file.

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 $1,2,\ldots,M$ from left to right. Fig. 3 is a timing diagram for a linear daisy chain network. Communication appears above the horizontal axis and computation is below it. It can be seen that the processing time $\alpha_i w_i T_{cp}$ of the ith processor equals the transmission time $(1-\alpha_0-\alpha_1-\cdots-\alpha_i)z_{i+1}T_{cm}$, from the ith 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_i is the optimal fraction of the load assigned to the ith processor. Thus a set of recursive equations can be written

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

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 [4].

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 signature in its allocated fraction. Let the links and children processors be numbered $1,2,\ldots,M$. Each processor at level 1 will search for a signature as soon as it receives the entire single installment dataset fraction from the root processor. Fig. 4 is the timing diagram of communication and computation for a single level tree network with single installment load distribution. Let the children processors be numbered $1,2,\ldots,M$. For Fig. 4, the corresponding recursive load distribution equations are

$$\alpha_i w_i T_{cp} = \alpha_{i+1} z_{i+1} T_{cm} + \alpha_{i+1} w_{i+1} T_{cp}, \quad i = 0,1,\ldots,M-1 \tag{2}$$

and the normalization equation is

$$\sum_{i=0}^{M} \alpha_i = 1. \tag{3}$$

Recursive solutions to equations such as these appear in Bharadwaj [4] and Ko [33].

3) *Multi-Installment Distribution in Single Level Tree Network*: The expected time of searching for a signature is expected to decrease as the finish time becomes smaller. Sending the load fraction in more than one installment so that a processor can begin its computation earlier in time will reduce the finish time and the time to start searching [3].

Consider a system in which the load is distributed in N installments. The installments are such that all

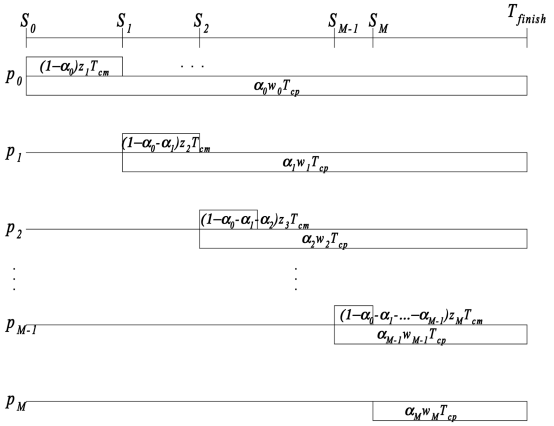


Fig. 3. Timing diagram for linear daisy chain network.

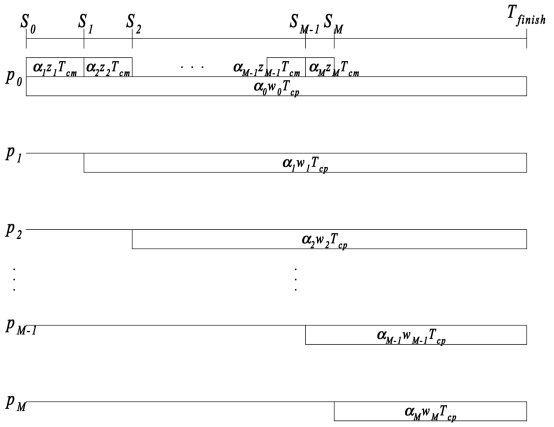


Fig. 4. Timing diagram for single installment distribution in single level tree network.

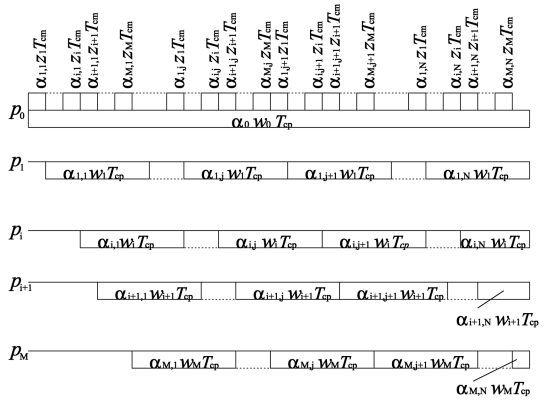


Fig. 5. Timing diagram for multi-installment distribution in single level network.

the processors will nominally stop computing at the same time instant. Let $\alpha_{i,j}$ be the fraction of the jth installment assigned to the ith processor and α_i be the fraction assigned to the ith processor

$$\alpha_i = \sum_{j=1}^{N} \alpha_{i,j}. \tag{4}$$

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

$$\alpha_{i,N} w_i T_{cp} = \alpha_{i+1,N}(z_{i+1} T_{cm} + w_{i+1} T_{cp}),$$

$$i = 1,2,\ldots,M-1 \tag{5}$$

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

$$i = 1,2,\ldots,M, \quad j = 1,2,\ldots,N-1 \tag{6}$$

$$\alpha_0 w_0 T_{cp} = \sum_{i=1}^{M} \sum_{j=1}^{N} \alpha_{i,j} z_i T_{cm} + \alpha_{M,N} w_M T_{cp}. \tag{7}$$

The normalizing equation is given by

$$\alpha_0 + \sum_{i=1}^{M} \sum_{j=1}^{N} \alpha_{i,j} = 1. \tag{8}$$

Solutions to multi-installment load scheduling appear in Bharadwaj [4] and Ko [33].

B. Time to Start Searching

Defining the time for each processor to start searching for a signature is important since this gives the lower limit of the amount of time until a signature is found on a given processor. Let S_m be the time for the mth processor to start searching. Then this time is given in Table I.

TABLE I
Time to Start Searching

Model	S_0	S_m $(m = 1,\ldots,M)$
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 signature position in the dataset. Here, a signature position variable is described in terms of the uniform distribution and is denoted as $\mathbf{X}$, where:

$$\mathbf{X} \sim U(0,1). \tag{9}$$

Since the dataset is normalized, a signature is positioned between zero and one. Certainly more complex assumptions on the statistics of signature position are possible. However we examine the uniform assumption case because it is canonical and because of its tractability. A good discussion of such assumptions appears in [9].

It is further assumed that the dataset is very large. Thus, the distribution of a signature position is regarded as a continuous variable. Now, considering the quantity that must be described as the amount of time until a signature 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 signature position $\mathbf{X}$

$$\mathbf{Y} = g(\mathbf{X}). \tag{10}$$

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

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

$$\mathbf{Y}_m = g_m(\mathbf{X}) \tag{11}$$

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

$$g_0(\mathbf{X}) = \mathbf{X} w_0 T_{cp} + S_0, \qquad 0 \leq \mathbf{X} \leq \alpha_0. \tag{12}$$

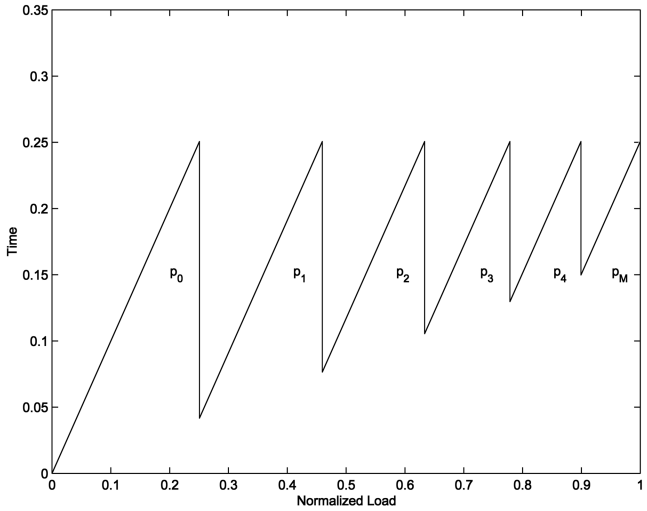


Fig. 6. Transformation function: linear daisy chain network, single level tree network; number of child processors, $M = 5$. Here $wT_{cp} = 1$ and $zT_{cm} = 1$.

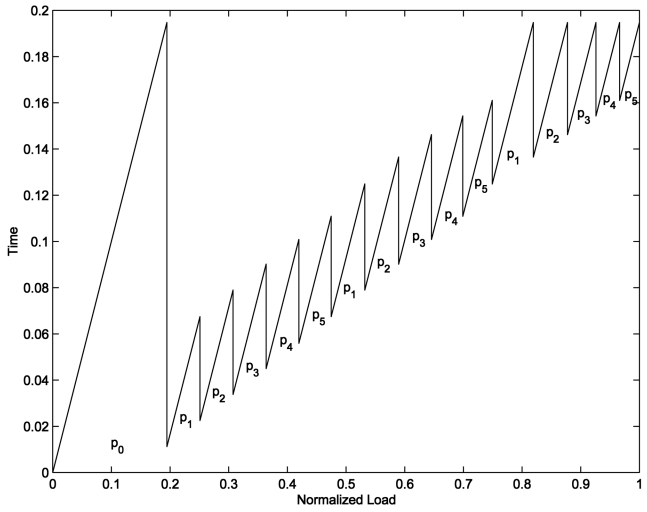


Fig. 7. Transformation function: multi-installment distribution in single level tree network; number of child processors, $M = 5$, number of installments, $N = 3$. Here $wT_{cp} = 1$ and $zT_{cm} = 1$.

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

$$g_m(\mathbf{X}) = (\mathbf{X} - \sum_{i=0}^{m-1} \alpha_i) w_m T_{cp} + S_m, \qquad \sum_{i=0}^{m-1} \alpha_i < \mathbf{X} \le \sum_{i=0}^{m} \alpha_i. \tag{13}$$

Here, S_m is the time for the mth processor to start searching as before. Fig. 6 and Fig. 7 depict this relationship between $\mathbf{X}$ and $\mathbf{Y}$ (based on their respective earlier timing diagrams). For the linear network and single level, single installment tree network the transformation function of Fig. 6 maps the signature position (a continuous variable from 0 to 1) into being found on one of the M+1 processors. Similarly the transformation function of Fig. 7 maps the signature position into a processor for multi-installment distribution on 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 signature in time when the signature is found in mth processor

$$g(\mathbf{X}) = \sum_{m=0}^{M} g_m(\mathbf{X}). \tag{14}$$

Note that in Figs. 6 and 7, $wT_{cp} = 1$ and $zT_{cm} = 1$. From (11), (12), and (13), the probability density function of $\mathbf{Y}_m$, denoted as $f(y \mid p_m)$, is obtained as

$$f(\mathbf{y} \mid p_m) = \frac{1}{T_{\text{finish}}(M) - S_m}, \qquad S_m < \mathbf{y} \le T_{\text{finish}}(M). \tag{15}$$

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

Assuming that a single signature exists in the dataset, this signature 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 signature is in the fraction assigned to the mth processor, its probability equals to α_m

$$\Pr\{\text{a signature exists at } p_m\} = \alpha_m. \tag{16}$$

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

$$\mathrm{E}[\mathbf{Y}] = \int_0^{T_{\text{finish}}(M)} \mathbf{y} \cdot f(\mathbf{y}) d\mathbf{y}. \tag{17}$$

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

$$f(\mathbf{y}) = \sum_{m=0}^{M} f(\mathbf{y} \mid p_m) \cdot \Pr\{\text{a signature exists at } p_m\} \tag{18}$$

$$= \sum_{m=0}^{M} f(\mathbf{y} \mid p_m) \cdot \alpha_m. \tag{19}$$

Here, the probability that a signature exists at p_m is already defined and equals to α_m. From (19), (17) can be rewritten as

$$\mathrm{E}[\mathbf{Y}] = \int_0^{T_{\text{finish}}(M)} \mathbf{y} \cdot \left[\sum_{m=0}^{M} f(\mathbf{y} \mid p_m) \cdot \alpha_m \right] d\mathbf{y} \tag{20}$$

$$= \sum_{m=0}^{M} \alpha_m \left[\int_0^{T_{\text{finish}}(M)} \mathbf{y} \cdot f(\mathbf{y} \mid p_m) d\mathbf{y} \right]. \tag{21}$$

From (15), $f(y \mid p_m)$ is only defined on $(S_m, T_{\text{finish}}(M))$. Thus the lower bound of the integral on the right side of (21) can be substituted as S_m instead of as 0:

$$\int_0^{T_{\text{finish}}(M)} \mathbf{y} \cdot f(\mathbf{y} \mid p_m) d\mathbf{y} = \int_{S_m}^{T_{\text{finish}}(M)} \mathbf{y} \cdot f(\mathbf{y} \mid p_m) d\mathbf{y}. \tag{22}$$

Solving the integral of the right-hand side of the above equation by substituting (15) yields

$$\int_{S_m}^{T_{\text{finish}}(M)} \mathbf{y} \cdot f(\mathbf{y} \mid p_m) d\mathbf{y} = \frac{[T_{\text{finish}}(M)]^2 - [S_m]^2}{2} \cdot \frac{1}{T_{\text{finish}}(M) - S_m} \tag{23}$$

$$= \frac{T_{\text{finish}}(M) + S_m}{2}. \tag{24}$$

Therefore the closed form of the expected time of searching for a signature is

$$\mathrm{E}[\mathbf{Y}] = \sum_{m=0}^{M} \alpha_m \frac{T_{\text{finish}}(M) + S_m}{2}. \tag{25}$$

Intuitively, this equation holds that the average signature search time is the weighted sum of the mid point of each segment weighted by the size of the segment. The previous derivation, though, is more general and establishes a framework for finding expected search time for other interesting networks and scheduling policies.

IV. EQUIVALENT EXPECTED TIME OF SEARCHING FOR MULTIPLE SIGNATURES

The expected time of searching for a single signature was considered above. The solution of the expected time of searching for multiple signatures quickly becomes unmanageable since the last signature to appear in the dataset is not always the last signature to be detected. This occurs because the processors shift through the data concurrently changing the order in which detections are made. However, in the case of a single processor, the last signature is detected last. Furthermore, since this is a linear model, the expected time of finding all of the signatures can be obtained by applying the mean of the last signature position

$$\mathrm{E}[\mathbf{Y}_L] = g(\mathrm{E}[\mathbf{X}_L]). \tag{26}$$

Here, $\mathrm{E}[\mathbf{X}_L]$ is the mean of the last signature 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 $g_{\text{eq}}(\cdot)$. Accordingly let

$$\mathbf{Y} = g_{\text{eq}}(\mathbf{X}). \tag{27}$$

Furthermore, since it is clear that $g_{\text{eq}}(\cdot)$ depends on w_{eq}, the inverse speed of the equivalent processor, one can use the (12) as follows:

$$g_{\text{eq}}(\mathbf{X}) = \mathbf{X} w_{\text{eq}} T_{cp} + S_{\text{eq}}. \tag{28}$$

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 multiprocessor to an equivalent single processor. The general relation between $\mathbf{X}$ and $\mathbf{Y}$ is given below by combining the last two equations:

$$\mathbf{Y} = \mathbf{X} w_{\text{eq}} T_{cp} + S_{\text{eq}}. \tag{29}$$

Since a multiprocessor is considered as a single equivalent processor, the conditional probability density function in (15) can be directly applied to the probability density function of $\mathbf{Y}$

$$f(\mathbf{y}) = \frac{1}{T_{\text{finish}}(M) - S_{\text{eq}}}, \qquad S_{\text{eq}} < \mathbf{y} \le T_{\text{finish}}(M). \tag{30}$$

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

$$\mathrm{E}[\mathbf{Y}] = \int_{S_{\text{eq}}}^{T_{\text{finish}}(M)} \mathbf{y} \cdot f(\mathbf{y}) d\mathbf{y} \tag{31}$$

$$= \int_{S_{\text{eq}}}^{T_{\text{finish}}(M)} \mathbf{y} \frac{1}{T_{\text{finish}}(M) - S_{\text{eq}}} d\mathbf{y} \tag{32}$$

$$= \frac{S_{\text{eq}} + T_{\text{finish}}(M)}{2}. \tag{33}$$

To obtain S_{eq}, let (33) equal to (25). Then:

$$S_{\text{eq}} = \sum_{m=0}^{M} \alpha_m (T_{\text{finish}}(M) + S_m) - T_{\text{finish}}(M). \tag{34}$$

Taking the expectation of both sides of (29) yields

$$\mathrm{E}[\mathbf{Y}] = \mathrm{E}[\mathbf{X}] w_{\text{eq}} T_{cp} + S_{\text{eq}}. \tag{35}$$

The expected value of the distribution $\mathbf{X}$ is μ (= 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 (35):

$$w_{\text{eq}} = \frac{1}{\mu T_{cp}} \left[\sum_{m=0}^{M} \alpha_m \frac{T_{\text{finish}}(M) + S_m}{2} - S_{\text{eq}} \right]. \tag{36}$$

From (34), S_{eq} is known. Therefore (36) is rewritten as follows:

$$w_{\text{eq}} = \frac{1}{\mu T_{cp}} \left[T_{\text{finish}}(M) - \sum_{m=0}^{M} \alpha_m \frac{T_{\text{finish}}(M) + S_m}{2} \right]. \tag{37}$$

However, (29) is still not linearly proportional. To create a linearly proportional equation, change the variable $\mathbf{Y}$ as

$$\mathbf{Y}' = \mathbf{Y} - S_{\text{eq}}. \tag{38}$$

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

$$\mathbf{Y}' = \mathbf{X} w_{\text{eq}} T_{cp}. \tag{39}$$

This equation is linearly proportional.

Assuming that L signatures are distributed uniformly in the normalized data, the expected value of the last signature location is [12]

$$\mathrm{E}[\mathbf{X}_L] = 2\mu\frac{L}{L+1}. \tag{40}$$

Similarly, $E[\mathbf{Y}'_L]$ can be expressed like (26)

$$\mathrm{E}[\mathbf{Y}'_L] = \mathrm{E}[\mathbf{X}_L]w_{\mathrm{eq}}T_{cp} \tag{41}$$

$$= 2\mu\frac{L}{L+1}w_{\mathrm{eq}}T_{cp} \tag{42}$$

or

$$\mathrm{E}[\mathbf{Y}_L] = 2\mu\frac{L}{L+1}w_{\mathrm{eq}}T_{cp} + S_{\mathrm{eq}}. \tag{43}$$

Substituting (34) and (37) into (43), the explicit form of above equation is represented as

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

The above equation is the expected time of finding all of L signatures in the dataset under uniformly distributed signature placement.

V. SIGNATURE 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 Fig. 8 and Fig. 10, espectively. Note that the horizontal axis of Fig. 8 represents the number of processors in the linear daisy chain while the horizontal axis in Fig. 10 represents the number of children processors in a single level tree network (not counting the root processor). Thus in the special cases of a linear daisy chain with two processors and a single level tree network with one child processor (and one root node), the graphs show identical results.

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

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

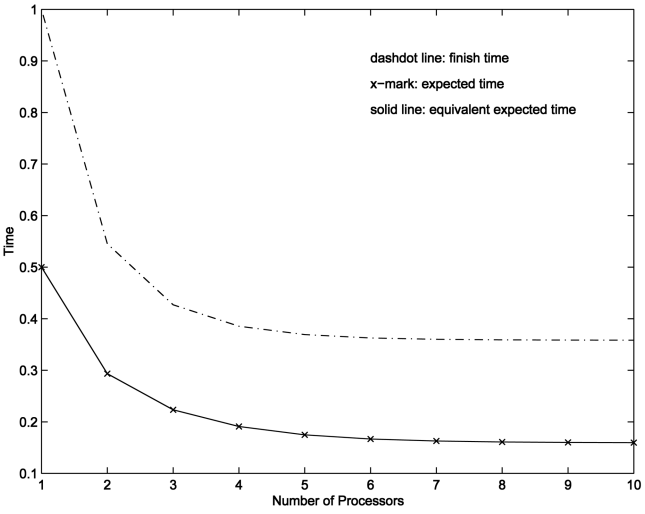


Fig. 8. Linear daisy chain network: time versus number of processors; one signature; $w_i = 1$, $z_i = 0.2$, $T_{cp} = 1$, $T_{cm} = 1$.

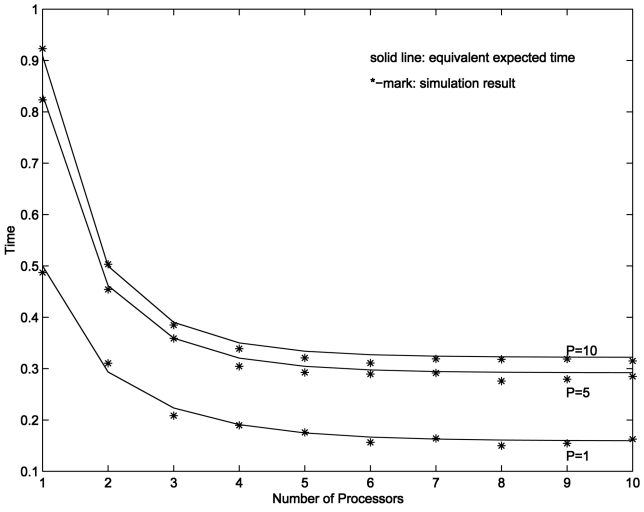


Fig. 9. Linear daisy chain network: time versus number of processors and varying number of signatures P; $w_i = 1$, $z_i = 0.2$, $T_{cp} = 1$, $T_{cm} = 1$.

VI. CONCLUSION AND OPEN PROBLEMS

In this paper the expected search time for single and multiple signatures is investigated in certain database architectures. Open problems include the following.

1) Finding an exact analytical expression for the mean search time to find the last signature using probabilistic means;

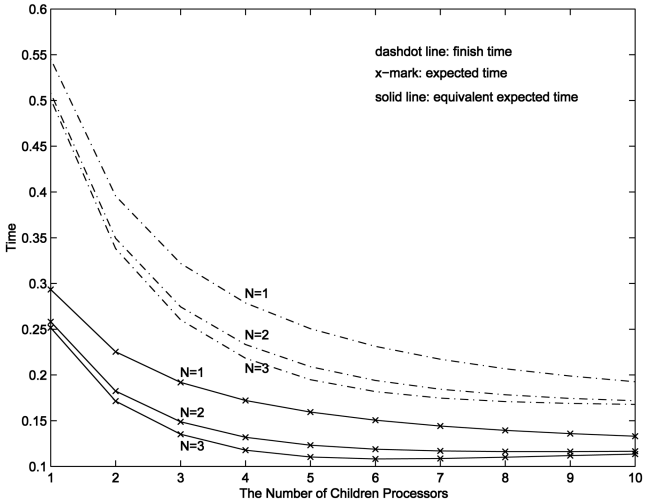


Fig. 10. Single level tree network: time versus number of processors and varying number of installments N; one signature; $w_i = 1$, $z_i = 0.2$, $T_{cp} = 1$, $T_{cm} = 1$.

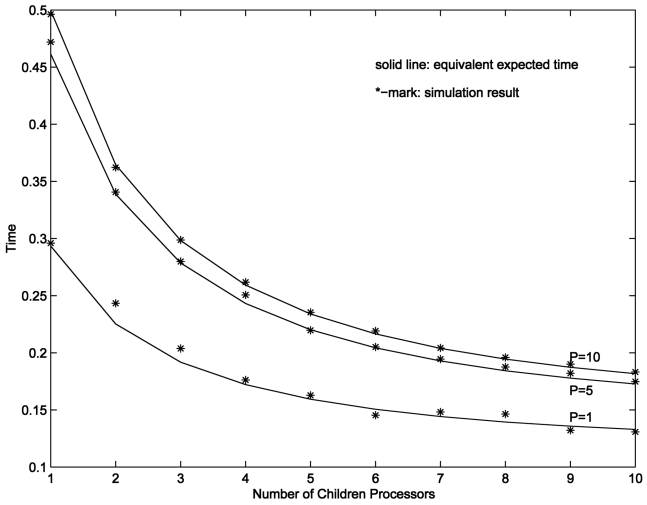


Fig. 11. Single level tree network: time versus number of processors and varying number of signatures P; one installment ($N = 1$); $w_i = 1$, $z_i = 0.2$, $T_{cp} = 1$, $T_{cm} = 1$.

2) A technique was proposed here for determining the mean time to find the last of a number of multiple signatures to be found. It is an open problem to find a tractable means of determining the mean time to find the nth signature;

3) Developing a technique to calculate the distribution of search time, not just the mean search time;

4) Accommodating in the analysis correlated signature positions;

5) Extending techniques proposed here to other scheduling policies and architectures.

Signature searching is a common problem for such fields as DNA sequence analysis, network intrusion detection, biometrics, and large scientific experiments. Thus the progress reported here may be of interest to others as well as the aerospace community.

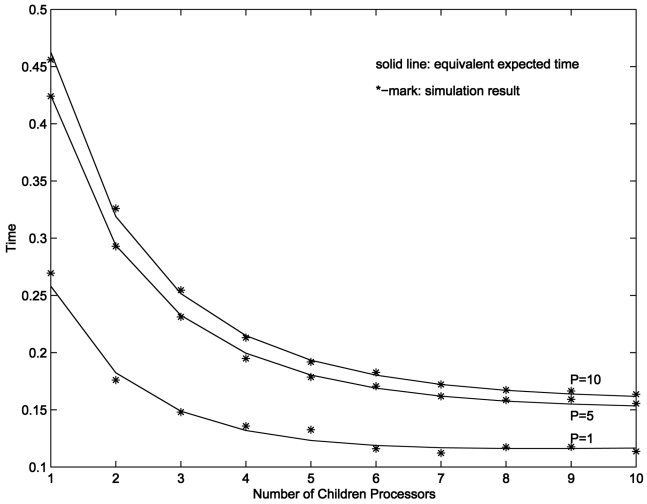


Fig. 12. Single level tree network: time versus number of processors and varying number of signatures P; two installments ($N = 2$); $w_i = 1$, $z_i = 0.2$, $T_{cp} = 1$, $T_{cm} = 1$.

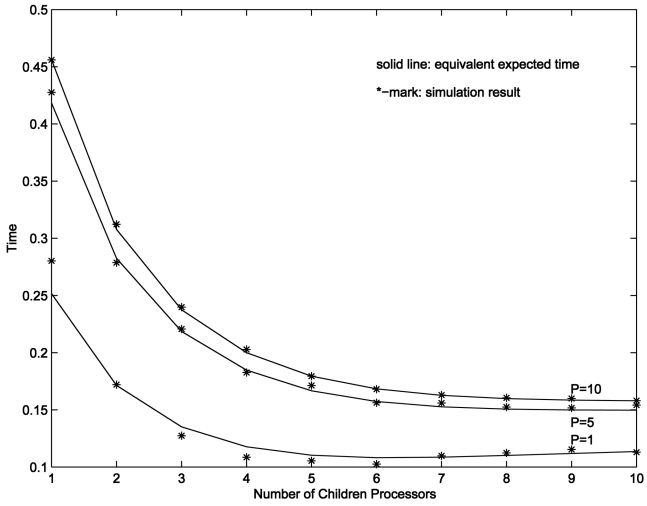


Fig. 13. Single level tree network: time versus number of processors and varying number of signatures P; three installments ($N = 3$); $w_i = 1$, $z_i = 0.2$, $T_{cp} = 1$, $T_{cm} = 1$.

REFERENCES

[1] Bataineh, S., and Robertazzi, T. G.
Bus-oriented load sharing for a network of sensor driven processors.
IEEE Transactions on Systems, Man, and Cybernetics, **21**, 5 (1991), 1202–1205.

[2] Bataineh, S., Hsiung, T., and Robertazzi, T. G.
Closed form solutions for bus and tree networks of processors load sharing a divisible job.
IEEE Transactions on Computers, **43**, 10 (1994), 1184–1196.

[3] Bharadwaj, V., Ghose, D., and Mani, V.
Multi-installment load distribution in tree networks with delays.
IEEE Transactions on Aerospace and Electronic Systems, **31**, 2 (1995), 555–566.

[4] Bharadwaj, V., Ghose, D., Mani, V., and Robertazzi, T. G.
Scheduling Divisible Loads in Parallel and Distributed Systems.
Los Alamitos, CA: IEEE Computer Society Press (now distributed by Wiley), 1996.

[5] Bharadwaj, V., Ghose, D., and Robertazzi, T. G.
Divisible load theory: A new paradigm for load scheduling in distributed systems.
Cluster Computing, **6**, 1 (2003), 7–18.

[6] Boyer, R. S., and Moore, J. S.
A fast string searching algorithm.
Communications of the ACM, **20**, 10 (1977), 762–772.

[7] Cheng, Y. C., and Robertazzi, T. G.
Distributed computation with communication delay.
IEEE Transactions on Aerospace and Electronic Systems, **24**, 6 (1988), 700–712.

[8] Cheng, Y. C., and Robertazzi, T. G.
Distributed computation for a tree network with communication delays.
IEEE Transactions on Aerospace and Electronic Systems, **26**, 3 (1990), 511–16.

[9] Christodoulakis, S.
Implications of certain assumptions in database performance evaluation.
ACM Transactions on Database Systems, **9**, 2 (1984), 163–186.

[10] Fujita, S., Yamashita, M., and Ae, T.
Parallel template matching in a restricted addressing mode.
In *Proceedings of the 24th Annual Hawaii International Conference on Systems Sciences*, 1991, 27–35.

[11] Galil, Z.
Optimal parallel algorithms for string matching.
In *Proceedings of the 16th ACM Symposium on Theory of Computing*, 1984, 240–248.

[12] Harter, H. L.
Ordering Statistics and Their Use in Estimation and Testing, vols. 1 and 2.
Washington, D.C.: Superintendent of Documents, U.S. Government Printing Office, 1969.

[13] Hoffman, M. A., and Carver, D. L.
Reverse engineering data requirements.
In *Proceedings of 1996 Aeropsace Applications Conference*, 1996, 269–277.

[14] Kitakami, H., Shin-I, T., Ikeo, K., et al.
YAMATO and ASUKA: DNA database management system.
In *Proceedings of the 28th Annual Hawaii International Conference on System Sciences*, 1995, 72–80.

[15] Lübeck, M.
An overview of a large-scale data migration.
In *Proceedings of the 20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, 2003, 49–55.

[16] Mani, V., and Ghose, D.
Distributed computation in linear networks: Closed-form solutions.
IEEE Transactions on Aerospace and Electronic Systems, **30**, 2 (1994), 471–83.

[17] Robertazzi, T. G.
Processor equivalence for daisy chain load sharing processors.
IEEE Transactions on Aerospace and Electronic Systems, **29**, 4 (1993), 1216–21.

[18] Robertazzi, T. G.
Ten reasons to use divisible load theory.
Computer, **36**, 5 (2003), 63–68.

[19] Sohn, J., and Robertazzi, T. G.
Optimal divisible job load sharing for bus networks.
IEEE Transactions on Aerospace and Electronic Systems, **32**, 1 (1996), 34–39.

[20] Roth, M. A., Ruberg, S. A., and Eldridge, B. L.
Database management: The heart of integrated avionics.
In *Proceedings of the National Aerospace and Electronics Conference*, 1993, 535–541.

[21] Glickstein, I., Ruberg, S., and Marsh, J.
Database management for integrated avionics systems.
In *Proceedings of the National Aerospace and Electronic Conference*, 1992, 617–622.

[22] Peng, C.-S., and Lin, K.-J.
A performance study of the concurrency control algorithms for real-time avionics systems.
In *Proceedings of the AIAA/IEEE Digital Avionics Systems Conference*, 1997, 1.2-23–30.

[23] Bonnor, N.
The development of digital databases for airborne applications.
In *Proceedings of the Colloqium on Serious Low Flying*, 1998, 3/1–3/3.

[24] Vladlamani, A., and de Haag, M. V.
A 3-D spatial integrity monitor for terrain databases.
In *Proceedings of the Digital Avionics Systems Conference*, 2004, 4.C.2-1–4.C.2-13.

[25] Roark, M. B.
Reconciling avionics database management with security.
In *Proceedings of the AIAA/IEEE Digital Avionics Systems Conference*, 1997, 1.2-1–1.2-7.

[26] Stevens, J., and Componation, P. J.
Shuttle performance improvement through multiple database integration.
IEEE Transactions on Aerospace and Electronic Systems, **40**, 2 (2004), 478–490.

[27] Detter, R., Mooney, M., and Fatig, C. C.
XML—James Webb space telescope database issues, lessons and status.
In *Proceedings of the IEEE Aerospace Conference*, 2004, 3306–3312.

[28] Some, R. R., Czikmantory, A., et al.
XML hierarchical database for missions and technologies.
In *Proceedings of the IEEE Aerospace Conference*, 2004, 292–303.

[29] Sripada, S. M., Rosser, B. L., Bedford, J. M., and Kowalski, R. A.
Temporal database technology for air traffic flow management.
In *Proceedings of the 1st International Conference on Applications of Databases*, 1994, 28–41.

[30] Adams, T., Dullea, J., et al.
Semantic integration of heterogeneous information sources using a knowledge-based system.
In *Proceedings of the Fifth International Conference on Computer Science and Informatics*, 2000, 289–294.

[31] Gentner, F. C.
Survey of air force MPTS tools and databases for human systems integration.
In *Proceedings of the 1991 National Aerospace and Electronics Conference*, 1991, 824–831.

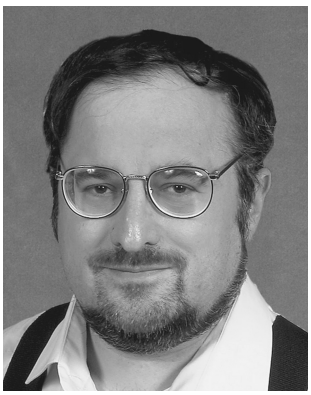
[32] Sripada, S. M.
Information management challenges from the aerospace industry.
In *Proceedings of the 28th Very Large Database Conference*, 2002.

[33] Ko, K., and Robertazzi, T. G.
Signature search time evaluation in flat file parallel databases.
In Stony Brook University College of Engineering and Applied Sciences, Technical Report 822, Apr. 25, 2006; available from T. Robertazzi.

[34] Drozdowski, M., and Wolniewicz, P.
Experiments with scheduling divisible tasks in clusters of workstations.
In A. Bode, et al. (Eds.), *Proceedings of EURO-Par2000*, Lecture Notes in Computer Science, LNCS 1900, New York: Springer-Verlag, 2000, 311–319.

Kwangil Ko received the M.S. and Ph.D degrees in 1996 and 2000 from Stony Brook University, Stony Brook, NY.

He is currently with Samsung Electronics, Suwon, Korea, where he is working as a traffic engineer on radio access networks. His research interests include the performance measurement of networks, scheduling algorithms, flow control, QoS, and resource management.



Thomas G. Robertazzi (S'75—M'77—SM'91—F'06) received the Ph.D. from Princeton University, Princeton, NJ, in 1981 and the B.E.E. from the Cooper Union, New York, NY, in 1977.

He is presently a professor in the Department of Electrical and Computer Engineering at Stony Brook University, Stony Brook, NY. He is also the faculty director of the Science and Engineering Living Learning Center at Stony Brook. In supervising a very active research area, he has published extensively in the areas of parallel processor and grid scheduling, ad hoc radio networks, telecommunications network planning, ATM switching, queueing, and Petri networks.

Dr. Robertazzi has authored, coauthored or edited five books in the areas of performance evaluation, scheduling, and network planning.