

# Greedy Signature Processing with Arbitrary Location Distributions: A Divisible Load Framework

Yuntai Kyong, *Student Member, IEEE* and Thomas G. Robertazzi, *Fellow, IEEE*

Department of Electrical and Computer Engineering

Stony Brook University

Stony Brook, NY, 11794

email: Yuntai.Kyong@sunysb.edu, tom@ece.sunysb.edu

## Abstract

The optimal partition of a huge, linear (flat) file among processing nodes in a network to minimize the time to search for signatures of interest in the file is considered. First, an expression is developed for the expected time of finding the  $k$ th signature (including the last signature) of  $K$  signatures for a uniform distribution of signatures in the file. Secondly, for a single signature we propose processing data in the file in order from that with the most probability mass (i.e. data with the most a priori likelihood of containing the signature) to that with the least probability mass in a “greedy” manner to speed processing time. Applications of this work include radar, sensors, image processing and search.

## I. INTRODUCTION

A “signature” is a data pattern of interest in a large data file. Signature searching involves finding such signatures in the voluminous amount of data that can be produced with aerospace technology. This process is complicated by the possible presence of noise in the data and because sometimes ideal signatures may only be approximate in actual instances of data. There are natural radar and sensor applications for signature searching. However, as examples, three representative applications involving image processing are:

- 1) An automated lander for Mars may process views of the surface for the signatures of good landing locations (e.g., flat areas, no boulders or trenches, etc...).
- 2) Archaeologists may process satellite imagery of jungle areas for signatures of overgrown ancient structures.
- 3) Astronomers may search through images from a space based telescope, such as the Hubble, automatically cataloging distinctive astronomical features (e.g., spiral galaxies).

In this paper divisible load theory is applied to the problem of signature search time evaluation in flat file databases. That is, one can view the entire data set as a huge, linear (flat) file that is to be optimally partitioned

among processing nodes in a network so that processing time is minimal. Flat (linear) files are a natural choice for early database implementations though more sophisticated database models are often later used [1].

In this paper after establishing the model and notation (section II), the expected time for finding multiple signatures in a flat file is developed (section III). In doing so, a uniform distribution of the signatures within the file is first assumed. An expression is developed for the expected time for finding the  $k$ th signature out of  $K$  signatures, including the last signature. This work extends the earlier work of Ko and one of the authors [1]. That work considered single signatures with a uniform distribution of signature location as well as finding the last signature of a number of uniformly distributed multiple signatures.

However signatures may not always be uniformly distributed. For instance a satellite may record images of the ocean in a search and rescue operation. Based on elapsed time and known ocean currents it may be possible to associate a probability density function to the current location of a lost individual. As a second example, probability density functions may also be associated with the positions of a vessel or a vehicle one has lost contact with. Here one would also take into account the potential speed of the vessel/vehicle.

In this paper we propose for such situations processing data in order from that with the most probability mass (i.e. data with the most a priori likelihood to find a signature) to that with the least probability mass in a “greedy” manner to speed the processing time. Naturally for a small data set on a single computer, time may not be an issue. We are thus considering voluminous data sets, with significant processing and transmission time, that can be processed on a number of computers (such as a cluster of computers). We use the term “greedy” in the spirit of greedy combinatorial optimization algorithms which choose, each algorithm iteration, the combinatorial choice that yields the largest possible improvement possible in maximizing or minimizing an optimization (objective) function.

In section IV distributing load for greedy processing for the case of a single signature is examined. Greedy processing for arbitrary location densities for a single signature is considered in section V. An example involving a truncated normal probability density function appears in section VI. This is followed by conclusions and future work in section VII.

#### *A. Divisible Load Theory*

In this paper we use divisible load analysis because of its tractability and appropriateness for the model considered. Divisible load theory was introduced by [2] and [3] and, also independently in [4] as “large-grained parallelism”. Divisible load modeling is concerned with massive computational and communication loads without any precedence relationships that must be optimally scheduled/allocated to processors and links. Here optimality is defined as processing the load in a minimal amount of time for a given scheduling policy and interconnection network topology.

Divisible load theory is a theory of proportions. If one has two processors and a very fast connection between them, with one processor twice as fast as the other, it makes sense that two thirds of the load should be assigned to the faster processor and one third of the load to the slower processor to process the load in a minimal amount of time. Given many processors interconnected by channel speed limited links in some sort of interconnection

network one has a more complex problem of proportions but one that can still be solved through linear equations, mathematical programming or, in some cases, by simple algebraic recursions.

Ever since its introduction, divisible load theory has served as an effective modeling tool for data-intensive applications [5][6][7][8][9][10] and a large amount of work has been published for various network structures including linear networks [11], bus networks [12], single and multi level tree-networks [13][14][15], mesh networks [16][17], hypercubes [18] and arbitrary graphs [19] with different constraints such as different release times, buffer constraints [20], communication start-up costs and time-varying network capacity. Also, various distribution strategies have been studied including multi-installment of load distribution [21]. In [22], [23] and [24], scheduling strategies without knowledge of network resources are examined. In [25], signature searching is investigated on bus networks experimentally. In [1], a uniform distribution of the signature in the dataset is assumed as it is a feasible model of the distribution of signatures in large databases as discussed in [26].

## II. SYSTEM MODEL

### A. Network Model

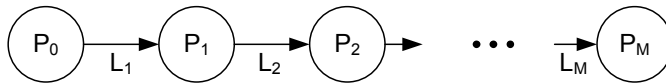


Fig. 1. Model of linear daisy chain network with communication links.

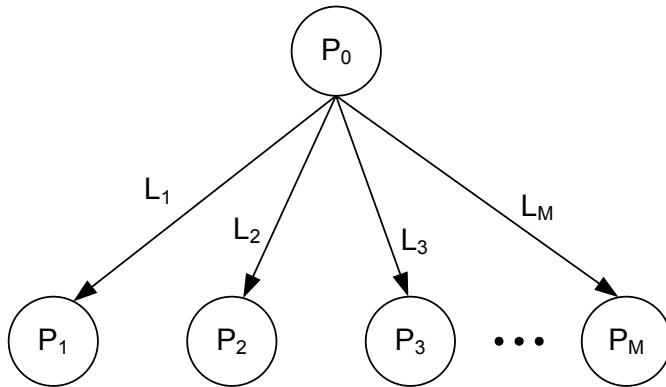


Fig. 2. Model of single level tree network with communication links.

We consider signature search time on two network models using divisible load theory. Fig. 1 describes a linear daisy chain model and, in Fig. 2, a single level tree network model is described. In this second network model, one root processor,  $P_0$ , is connected to the rest of the processors via links. In both models, the load is originated from  $P_0$ . Communication links connected to  $P_j$  are shown as  $L_j$ . The notation used in this paper is summarized in TABLE I. Here  $\alpha_i$  is the optimal fraction of load distributed to processors and links, calculated with divisible load theory

$K$	The number of signatures in a data file when there are multiple signatures
$M$	The number of processors besides the load originating processor (root)
$B$	The total number of bins
$\alpha_i$	Fraction of entire processing load assigned to $i$ th processor
$w_i$	Constant that is inversely proportional to the computation speed of the $i$ th processor
$z_i$	Constant that is inversely proportional to the communication speed of the $i$ th link
$T_{cp}$	Computation intensity: time taken to process a unit load on the $i$ th processor when $w_i = 1$
$T_{cm}$	Communication intensity: time taken to communicate a unit load over link $l_i$ when $z_i = 1$
$S_i$	Processing start time of the $i$ th processor
$\beta^i$	Fraction of load that is processed during $S_i$ and $S_{i+1}$
$ \beta^i $	The size of $\beta^i$
$l(\beta^i), u(\beta^i)$	The lower and the upper boundaries of $\beta_i$
$\alpha_j^i$	The fraction of load distributed to $P_i$ taken from $\beta^i$
$ \alpha_i^j $	The size of $\alpha_i^j$
$l(\alpha_i^j), u(\alpha_i^j)$	The lower and the upper boundaries of $\alpha_i^j$
$Y$	Random variable describing the amount of time to find a single signature contained in a data file
$Y_k$	Random variable describing the amount of time to find the $k$ th of $K$ signatures in a data file
$A_m$	Event when $k$ th of $K$ signatures is found in processor $m$ .

TABLE I  
SUMMARY OF NOTATION FOR DIVISIBLE LOAD THEORY

using the rest of system parameters,  $w_i, z_i, T_{cp}$  and  $T_{cm}$  which are given as constant values. For the scheduling policy for a single level tree network, we consider only the single-installment case, where communication of the load to each processor takes place only once for each processor. We also assume that the originating node also computes the load. It is also assumed that a child can only begin processing its load fraction after it receives it in its entirety (“staggered start”). Furthermore, each processor is equipped with a front-end processor for off-loading communication, so that communication and computation can take place concurrently. A closed-form solution for the optimal load distribution fractions ( $\alpha_i$ ) and an expression for the finish time (makespan) appear in [1] and [5] and are reproduced in the next subsection.

### B. Divisible Load Modeling

The solutions for the optimal load fraction to distribute to each processor and the starting time and the finish time of processing distributed load by each processor is obtained using recursive equations in [5]. Here optimality

is defined as the processing of the load in a minimal amount of time for a given interconnection network and given scheduling policy.

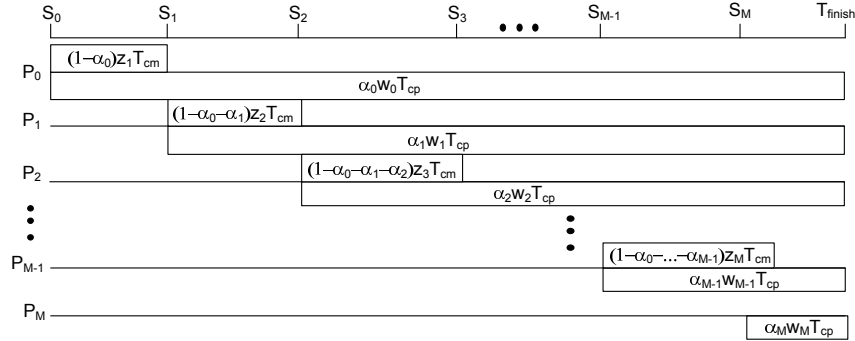


Fig. 3. Timing diagram for load distribution on a linear daisy chain network.

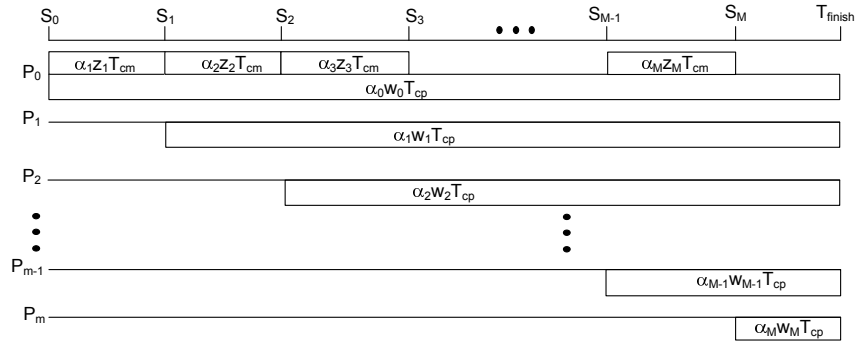


Fig. 4. Timing diagram for load distribution on a single level tree network with single installment.

Fig. 3 and Fig. 4 presents timing diagrams of load distribution for a linear daisy chain network model and a single level tree network model, respectively. It is shown that the processing time of the  $i$ th processor is given as  $\alpha_i w_i T_{cp}$  and the load communication time for the  $i$ th link is the amount of load to transfer multiplied by constant factor  $z_i T_{cm}$ . For example, in Fig. 4, the amount of time to transfer  $\alpha_1$  to  $P_1$  is given as  $\alpha_1 z_1 T_{cm}$ . As the optimality criterion of divisible load theory states, the computation time has to be finished at the same instant by all processors,  $T_{finish}$ . This criterion is intuitively reasoned in [5] - while distributing arbitrarily divisible loads, one should keep all the processors utilized until the last moment. If all processors do not stop at the same time, certainly the load can be transferred from busy processors to idle processor to improve the solution. A rigorous proof of this optimality criterion for various network models is presented in [5]. With this intuition concerning the nature of an optimal solution,  $M$  recursive equations can be written for the two network models of this paper as,

$$\alpha_i w_i T_{cp} = \left(1 - \sum_{j=0}^i \alpha_j\right) z_{i+1} T_{cm} + \alpha_{i+1} w_{i+1} T_{cp}, \quad (1)$$

$$i = 0, 1, \dots, M - 1,$$

for the linear daisy chain network model and

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

$$i = 0, 1, \dots, M-1,$$
(2)

for the single level tree network model. Each set of recursive equations, along with a normalization equation, form a system of  $(M+1)$  linear equations with  $(M+1)$  unknowns,  $\{\alpha_0, \alpha_1, \dots, \alpha_M\}$ . The normalization equation is given as,

$$\sum_{i=0}^M \alpha_i = 1,$$
(3)

for both network models. The starting time,  $S_i$ , of  $i$ th processor,  $P_i$ , is determined by communication time of the

Network Model	$S_0$	$S_m$ ( $m = 1, \dots, M$ )
Linear Daisy Chain	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	0	$\sum_{j=1}^m \alpha_j \cdot z_j T_{cm}$

TABLE II  
TIME TO START SEARCHING

load to the previous processors, from  $P_0$  to  $P_{i-1}$  and  $P_i$  itself. The starting time for each network model is given in [1] and is reproduced in TABLE II.

### III. EXPECTED SIGNATURE SEARCHING TIME FOR UNIFORM DISTRIBUTION

#### A. Problem Description

We define a set of random variables  $\{\mathbf{X}_i\}$  that describes the positions of  $K$  signatures in the dataset with the normalized size 1. We assume that the positions of the signatures have a certain distribution,

$$\{\mathbf{X}_i\} \sim \mathbf{F},$$
(4)

where  $\mathbf{F}$  is joint cumulative density function (CDF) defined on  $[0, 1]^K$ . We denote  $\mathbf{Y}_k$  as a random variable describing the amount of time to find the  $k$ th signature. The objective of this section of this paper is to find the expectation value of  $\mathbf{Y}_k$ .

#### B. Uniform Distribution with Single Installment

Let  $\{\mathbf{X}_i\}$  be independent and identically distributed random variables with uniform distribution,  $\mathbf{X}_i \sim U(0, 1)$ ,  $i \in [1, K]$ . The position of the  $k$ th signature is given as  $\mathbf{X}_{(k)}$ , the  $k$ th order statistics of  $\{\mathbf{X}_i\}$ . The  $E[\mathbf{Y}_k]$  can be expressed as,

$$E[\mathbf{Y}_k] = \sum_{m=0}^M E[\mathbf{Y}_k | A_m] \Pr(A_m),$$
(5)

where  $A_m$  denotes the event when the  $k$ th signature is found on  $P_m$ . For the single installment case, we have,

$$\begin{aligned} \Pr(A_0) &= \Pr(0 \leq \mathbf{X}_{(k)} \leq \alpha_0) \\ \Pr(A_m) &= \Pr\left(\sum_{j=0}^{m-1} \alpha_j \leq \mathbf{X}_{(k)} \leq \sum_{j=0}^m \alpha_j\right), \quad m \in [1, M]. \end{aligned} \quad (6)$$

Given that the  $k$ th signature is found on  $P_m$ ,

$$\begin{aligned} \mathbf{Y}_k|A_m &= g_m(\mathbf{X}_{(k)}|A_m) \\ &= (\mathbf{X}_{(k)}|A_m - \sum_{i=1}^{m-1} \alpha_i)w_m T_{cp} + S_m, \end{aligned} \quad (7)$$

where  $g_m(\cdot)$  is the transformation function presented in [1], which takes the position of a signature as an argument and gives the signature search time depending on which processor the signature is found. The search times for the processors are partially overlapping and the start times are generally increasing in signature position in the file. Plotted as a graph search time versus position exhibits a characteristic saw tooth type shape. Here,  $S_m$  denotes the starting time of  $P_m$  as described in TABLE II and  $(\mathbf{X}_{(k)}|A_m - \sum_{i=1}^{m-1} \alpha_i)$  is the offset of the position of the  $k$ th signature from the beginning of load fraction distributed to  $P_m$ .

The  $k$ th order statistics of the uniform distribution follows the well known beta distribution (see equation (19)) of applied mathematics,

$$\mathbf{X}_{(k)} \sim \mathbf{B}(k, K + 1 - k). \quad (8)$$

If we denote  $f_{k,K}(x)$  as the probability density function of the  $k$ th signature of  $K$  signatures, given that  $\mathbf{X}_{(k)}$  is on  $P_m$ , the conditional distribution of  $\mathbf{X}_{(k)}$  is given as,

$$\begin{aligned} \mathbf{X}_{(k)}|A_m &\sim \frac{f_{k,K}(x)}{\int_{\gamma_{m-1}}^{\gamma_m} f_{k,K}(x)dx} \\ &= \frac{f_{k,K}(x)}{I_{\gamma_m}(k) - I_{\gamma_{m-1}}(k)}, \quad \gamma_{m-1} \leq \gamma_m, \end{aligned} \quad (9)$$

where the limits of the integration  $\gamma_{m-1}$  and  $\gamma_m$  are  $\sum_{j=0}^{m-1} \alpha_j$  and  $\sum_{j=0}^m \alpha_j$ , respectively. We set  $\gamma_{-1} = 0$  for convenience.  $I_x(k)$  is the shorthand notation for  $I_x(k, K + 1 - k)$ , which is the cumulative distribution function of the beta distribution,  $f_{k,K}(x)$ .

$$I_x(k, K + 1 - k) = \int_0^x f_{k,K}(\zeta)d\zeta \quad (10)$$

It follows  $I_{\gamma_{-1}} = I_0 = 0$ . Taking the expectation of equation (7),

$$E[\mathbf{Y}_k|A_m] = (E[\mathbf{X}_{(k)}|A_m] - \sum_{i=0}^{m-1} \alpha_i)w_m T_{cp} + S_m. \quad (11)$$

From (9),

$$E[\mathbf{X}_{(k)}|A_m] = \frac{\int_{\gamma_{m-1}}^{\gamma_m} x f_{k,K}(x)dx}{I_{\gamma_m}(k) - I_{\gamma_{m-1}}(k)}, \quad (12)$$

From (6),

$$\begin{aligned}\Pr(A_m) &= \int_{\gamma_{m-1}}^{\gamma_m} f_{k,K}(x)dx \\ &= I_{\gamma_m}(k) - I_{\gamma_{m-1}}(k).\end{aligned}\tag{13}$$

Using these results into equation (5), we have an expression for the expected time for finding the  $k$ th signature out of  $K$  signatures:

$$\begin{aligned}E[\mathbf{Y}_k] &= \sum_{m=0}^M \left( \left( \frac{\int_{\gamma_{m-1}}^{\gamma_m} x f_{k,K}(x)dx}{I_{\gamma_m}(k) - I_{\gamma_{m-1}}(k)} - \gamma_{m-1} \right) w_m T_{cp} + S_m \right) \\ &\quad * (I_{\gamma_m}(k) - I_{\gamma_{m-1}}(k)).\end{aligned}\tag{14}$$

### Single Signature Case

When there is only one signature, we have  $K = k = 1$ . We use the identity,

$$I_x(a, b) = \sum_{j=a}^{a+b-1} \frac{(a+b-1)!}{j!(a+b-1-j)!} x^j (1-x)^{a+b-1-j}.\tag{15}$$

When  $a = k = 1$  and  $b = K + 1 - k = 1$ ,

$$I_{\gamma_m}(1, 1) = \gamma_m = \sum_{j=1}^m \alpha_j.\tag{16}$$

Then from equation (13)  $\Pr(A_m) = I_{\gamma_m}(k) - I_{\gamma_{m-1}}(k) = \sum_{j=0}^m \alpha_j - \sum_{j=0}^{m-1} \alpha_j = \alpha_m$ . Since  $\mathbf{X}_{(1)}$  is uniformly distributed, when  $K = 1$ ,  $E[\mathbf{X}_{(1)}|A_m] = (\gamma_m + \gamma_{m-1})/2$ . Also,  $(\gamma_m + \gamma_{m-1})/2 - \gamma_{m-1} = (\gamma_m - \gamma_{m-1})/2 = \alpha_m/2$ . Substituting these results into equation (5),

$$\begin{aligned}E[\mathbf{Y}_1] &= \sum_{m=0}^M \alpha_m \left( \frac{\alpha_m w_m T_{cp}}{2} + S_m \right) \\ &= \sum_{m=0}^M \alpha_m \left( \frac{\alpha_m w_m T_{cp} + 2S_m}{2} \right) \\ &= \sum_{m=0}^M \alpha_m \left( \frac{T_{finish} + S_m}{2} \right),\end{aligned}\tag{17}$$

where  $T_{finish}$  denotes the time when the computation finishes and is given as  $T_{finish} = \alpha_m w_m T_{cp} + S_m$ . The expected time of single signature search time is found as the weighted average of midpoints of the processing times of each processor. This special case confirms the result presented in [1].

### Time to find the last signature

If the file contains multiple, uniformly distributed, signatures then the distribution of the position of the last signature is given as,

$$\mathbf{X}_{(K)} \sim \mathbf{B}(K, 1).\tag{18}$$

The standard form of the probability density function of the beta distribution is given as,

$$f(x; a, b) = \frac{1}{B(a, b)} x^{a-1} (1-x)^{b-1},\tag{19}$$



where  $B(a, b)$  is the beta function with the parameters  $a$  and  $b$ . That is,  $B(a, b) = \int_0^1 t^{a-1}(1-t)^{b-1} dt$ . When  $k = K, a = K, b = K + 1 - k = 1$ ,  $f_{K,K}(x) = f(x; K, 1) = \frac{1}{B(K,1)} x^{K-1}$ . Also, from equation (15),

$$I_x(K, 1) = x^K \quad (20)$$

Using these with equation (12), with  $m$  being processor number,

$$\begin{aligned} E[\mathbf{X}_{(K)}|A_m] &= \int_{\gamma_{m-1}}^{\gamma_m} x \frac{f(x; K, 1)}{\gamma_m^K - \gamma_{m-1}^K} dx \\ &= \int_{\gamma_{m-1}}^{\gamma_m} x \frac{x^{K-1}}{B(K, 1)} \frac{1}{\gamma_m^K - \gamma_{m-1}^K} dx \\ &= \int_{\gamma_{m-1}}^{\gamma_m} \frac{x^K}{B(K, 1)} \frac{1}{\gamma_m^K - \gamma_{m-1}^K} dx \\ &= \frac{K}{K+1} \frac{\gamma_m^{K+1} - \gamma_{m-1}^{K+1}}{\gamma_m^K - \gamma_{m-1}^K}, \end{aligned} \quad (21)$$

where we use  $B(K, 1) = \frac{1}{K}$  from  $B(a, b) = \int_0^1 t^{a-1}(1-t)^{b-1} dt$ . With equations (5), (11), (13) and (20),

$$\begin{aligned} E[\mathbf{Y}_K] &= \\ &= \sum_{m=0}^M \left( \left( \frac{K}{K+1} \frac{\gamma_m^{K+1} - \gamma_{m-1}^{K+1}}{\gamma_m^K - \gamma_{m-1}^K} - \gamma_{m-1} \right) w_m T_{cp} + S_m \right) \\ &= (\gamma_m^K - \gamma_{m-1}^K). \end{aligned} \quad (22)$$

In [1], the expected search time of the last signature is derived using the concept of an equivalent processor for the uniformly distributed case. Here, a closed form solution using the beta distribution is derived.

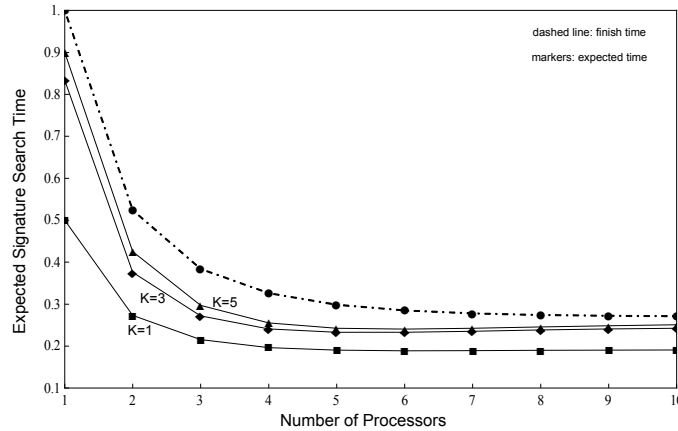


Fig. 5. Linear daisy chain network: time to find the last signature versus number of processors,  $K$  signatures,  $w = 1, z = 0.2, T_{cp} = 1, T_{cm} = 1$ .

Fig. 5 shows the search time for the last signature when there are  $K$  signatures. The dotted line at the top is the finish time when the processors do not stop processing after the last signature is found. It can be seen, as expected, search time can be decreased by adding processors up to a saturation limit. However, as shown in the

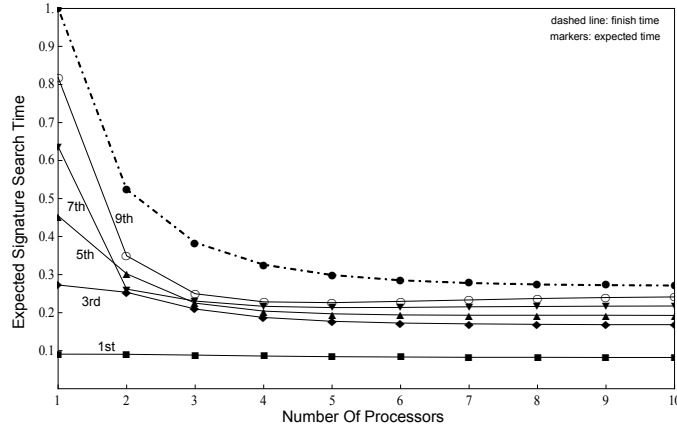


Fig. 6. Linear daisy chain network: time to find the  $k$ th signature versus number of processors, 10 signatures,  $w_i = 1, z_i = 0.2, T_{cp} = 1, T_{cm} = 1$ .

plot, as more processors are added the expected signature search time is slightly increasing after the search time is saturated (i.e. saturates at four processors). While the search time through parallel processing is not decreased because of the communication overhead, when calculating the expected value of the signature search time, the residual probability mass of finding signatures on the processors that are added after the saturation contributes to decreasing the expectation value albeit in a negligible way.

Fig. 6 shows the search time for the  $k$ th signature when the number of signatures is fixed to  $K = 10$ . It can be observed that before the speed up saturates, the order of expected times to find the signature changes as shown in the figure.

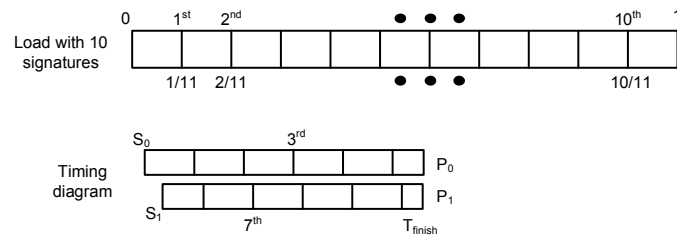


Fig. 7. Load with 10 uniformly distributed signatures and mini timing diagram for two processors,  $w = 1, z = 0.2, T_{cp} = 1, T_{cm} = 1$

The reason for this can be explained with Fig. 7. In the figure, the upper part describes the normalized load with size 1 with 10 uniformly distributed signatures of those expected locations are shown as vertical lines. With the same system parameters for generating the plot for Fig. 6, when there are two processors the load distribution is given as  $\{0.52381, 0.47619\}$ . The bottom part shows a mini timing diagram when the load is distributed with this proportion and the expected location of signatures. As shown in the timing diagram, the expected time to find the 7th signature is smaller than the expected time of the 3rd signature because of parallel processing. Although this diagram is not exact as it uses the expected positions, it explains why the order of expected search time may be

switched as shown in Fig. 6. A similar effect is also pointed out in [1].

This all assumes a load is mapped continuously and sequentially into each load fraction for each processor. The signatures are not necessarily detected in the original order they appear in the original flat data file. In the case of signature searching if the original flat file is indexed by time, there may be some concern that signatures spanning two sides of a load partition boundary are correctly detected. This can be handled by some overlapping of data fragments near partition boundaries. If this can be handled, then the original load could conceivably be multiplexed into each processing node sequentially and repetitively so that signatures are more likely to be detected closer to their original order if that is important.

#### IV. GREEDY PROCESSING: DISTRIBUTING LOAD

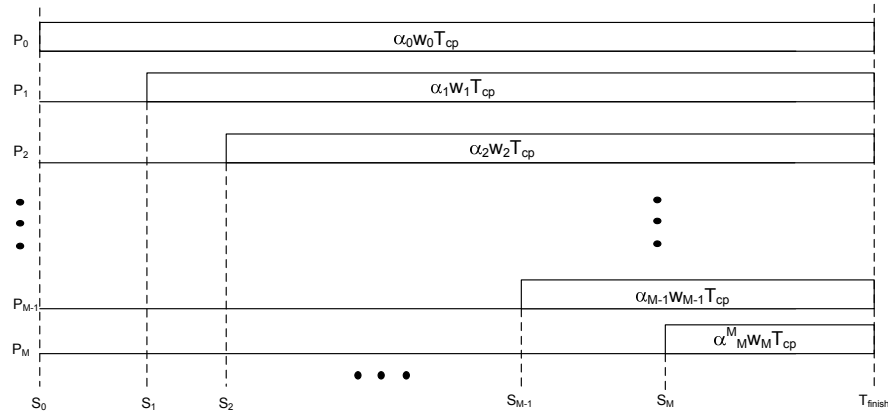
##### A. Load Classification based on Starting Time

Consider a search of a massive flat file for a single signature. Fig. 8(a) shows the timing diagram of the computation time of distributed load for a single level tree network (in fact since the figure only involves computation it is also appropriate for a linear daisy chain network but for this discussion the figure can be thought of as representing a single level tree network). The load is distributed in sequential order as it has been assumed in the literature of divisible load theory, where the position of the fraction has not been considered. In sequential load distribution, once  $\alpha_i$ , the fraction for  $P_i$  is determined,  $P_0$  gets the first part of load and  $P_1$  receives the next part of load. The ranges of the fractions inside the total load are  $[0, \alpha_0]$  and  $[\alpha_0, \alpha_0 + \alpha_1]$ , respectively, for  $P_0$  and  $P_1$ .

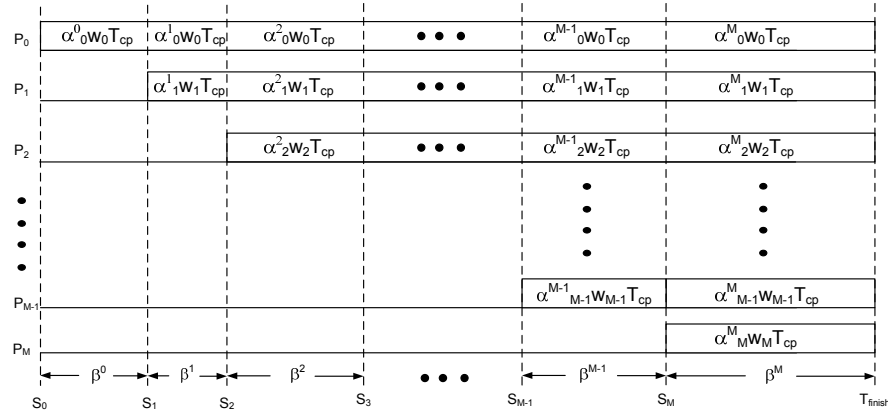
However, when the likelihood of finding a signature is not uniform, by greedily computing earlier the fractions of load with a higher likelihood of finding the signatures, the expected time of finding the signature will be shown to be reduced. Fig. 8(b) shows the classification of load based on their starting time of computation. Here  $\beta^i$  denotes the fraction of load which is computed between  $S_i$  and  $S_{i+1}$ . Since  $\beta^0$  is processed at the earliest time, it should contain the fraction of load that has highest probability of containing signatures. The fraction,  $\beta^1$ , having the highest probability excluding  $\beta^0$  is processed between  $S_1$  and  $S_2$ . Since both  $P_0$  and  $P_1$  process their fraction during that time period,  $\beta^1$  is separated into two fractions,  $\alpha_0^1$  and  $\alpha_1^1$ , where  $\alpha_j^i$  denotes the fraction of load distributed to  $P_j$  from  $\beta^i$ . We assume that  $\beta_i$  is from one contiguous region of load and later this assumption will be relaxed. Note that  $\beta_i$  is divided into  $i + 1$  fractions because there are  $i + 1$  processors computing their fraction between  $S_i$  and  $S_{i+1}$ . Also, each processor,  $P_j$ , receives the load from  $M - j$  different  $\beta^i$ . Note that there are events  $S_0, S_1, \dots, S_M, T_{finish}$ . Thus there are  $(M + 1)\beta_i$ , hence each processor  $P_j$  receives load from  $M + 1 - j$  different  $\beta_i$ . For example,  $P_0$  computes its fraction from  $S_0$  to  $T_{finish}$  so it receives the fractions from all  $\beta^i$ .

In the DLT literature,  $\alpha_i$  usually means the size of the fraction of the load and its position in load is not considered. Since  $\beta_i$  and  $\alpha_j^i$  are from different parts of total load, their position needs to be taken account as well as their size.

For the size of  $\beta_i$  and  $\alpha_j^i$ , we use  $|\beta_i|$  and  $|\alpha_j^i|$  and  $l(\cdot)$  and  $u(\cdot)$  for the boundaries of the range of fractions. For notational consistency, we also use  $|\alpha_i|$  to denote the size of fraction of load distributed using the usual sequential



(a) Sequential load distribution.



(b) Load distribution based on starting time.

Fig. 8. Comparison of Load Distribution Schemes.

distribution although  $\alpha_i$  means the same quantity. For summary, our notation  $\beta_i$  and  $\alpha_j^i$  consist of pairs of values,  $(|\beta_i|, l(\beta_i))$  and  $(|\alpha_j^i|, l(\alpha_j^i))$ , respectively, and  $u(\cdot) = l(\cdot) + |\cdot|$ .

The values of  $\{|\alpha_j^i|\}$  and  $|\beta^i|$  can be obtained from  $|\alpha_i|$  which can be obtained from well-known solutions of the literature of divisible load theory and can be calculated using recursive equations presented in Section II.

From  $\{|\alpha_i|\}$ ,  $\{|\alpha_j^i|\}$  is calculated with the following equations

$$|\alpha_j| = \sum_{i=j}^M |\alpha_j^i|, \quad j = 0, \dots, M \quad (23)$$

with a constraint,

$$|\alpha_0^i| w_0 T_{cp} = \dots = |\alpha_j^i| w_i T_{cp}, \quad i = 1, \dots, M, \quad j \leq i \quad (24)$$

and,

$$|\beta^0| = |\alpha_0^0| \quad |\beta^i| = \sum_{j=1}^i |\alpha_j^i|, \quad i = 1, \dots, M. \quad (25)$$

From (23),

$$\begin{aligned} \begin{pmatrix} |\alpha_0| \\ |\alpha_1| \\ \vdots \\ |\alpha_{M-1}| \\ |\alpha_M| \end{pmatrix} &= \begin{pmatrix} |\alpha_0^0| & |\alpha_0^1| & \dots & |\alpha_0^{M-1}| & |\alpha_0^M| \\ & |\alpha_1^1| & \dots & |\alpha_1^{M-1}| & |\alpha_1^M| \\ & & \vdots & \vdots & \vdots \\ & & & |\alpha_{M-1}^{M-1}| & |\alpha_{M-1}^M| \\ & & & & |\alpha_M^M| \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} |\alpha_0^0| & |\alpha_0^1| & \dots & |\alpha_0^{M-1}| & |\alpha_0^M| \\ & \frac{w_0}{w_1} |\alpha_0^1| & \dots & \frac{w_0}{w_{M-1}} |\alpha_0^{M-1}| & \frac{w_0}{w_M} |\alpha_0^M| \\ & & \vdots & \vdots & \vdots \\ & & & \frac{w_0}{w_{M-1}} |\alpha_0^{M-1}| & \frac{w_0}{w_M} |\alpha_0^M| \\ & & & & \frac{w_0}{w_M} |\alpha_0^M| \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 1 & \dots & 1 & 1 \\ & \frac{w_0}{w_1} & \dots & \frac{w_0}{w_{M-1}} & \frac{w_0}{w_M} \\ & & \vdots & \vdots & \vdots \\ & & & \frac{w_0}{w_{M-1}} & \frac{w_0}{w_M} \\ & & & & \frac{w_0}{w_M} \end{pmatrix} \begin{pmatrix} |\alpha_0^0| \\ |\alpha_0^1| \\ \vdots \\ |\alpha_0^M| \end{pmatrix} \end{aligned} \quad (26)$$

Here, the second equality is from (24).

Taking the inverse of the  $(M+1) \times (M+1)$  matrix,

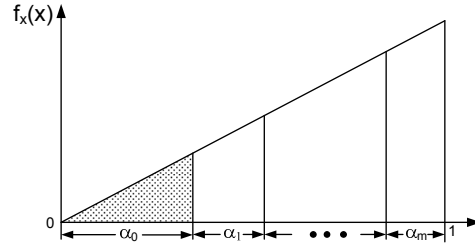
$$\begin{pmatrix} |\alpha_0^0| \\ |\alpha_0^1| \\ \vdots \\ |\alpha_0^M| \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 & 1 \\ & \frac{w_0}{w_1} & \dots & \frac{w_0}{w_{M-1}} & \frac{w_0}{w_M} \\ & & \vdots & \vdots & \vdots \\ & & & \frac{w_0}{w_{M-1}} & \frac{w_0}{w_M} \\ & & & & \frac{w_0}{w_M} \end{pmatrix}^{-1} \begin{pmatrix} |\alpha_0| \\ |\alpha_1| \\ \vdots \\ |\alpha_M| \end{pmatrix}. \quad (27)$$

The matrix form is given in (27) for exposition purposes. The  $\alpha$ 's may be solved more efficiently using standard divisible load theory algebraic recursions in the  $\alpha$ 's. Once  $\{|\alpha_0^0|, |\alpha_0^1|, \dots, |\alpha_0^M|\}$  is found, other  $\{|\alpha_j^i|, j \leq i \text{ and } |\beta^i|\}$  follows from equations (24) and (25).

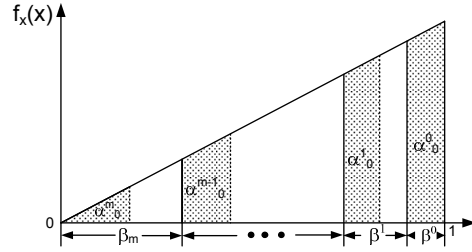
$$|\alpha_j^i| = |\alpha_0^i| \frac{w_0}{w_j}. \quad (28)$$

The values of the  $\beta^i$  can be determined from equation (25).

When the probability distribution function for the location of signatures is monotonically increasing or decreasing as shown in Fig. 9, for all  $\beta_i$ , one contiguous fraction suffices to give the optimal range. In the next subsection, we will give a motivational example with the probability distribution shown in the figure and discuss the case of arbitrary shapes of distributions in the following section.



(a) Sequential distribution.



(b) Distribution based on the probability mass.

Fig. 9. A comparison between the load distribution schemes.

### B. Motivational Example

In this example, a linearly increasing probability density for the location of a single signature is considered. We describe a more efficient processing of the load based on the shape. Consider Fig. 9. It shows that the probability mass of the position of the signature linearly increases toward the end of the load. In the usual sequential load distribution, the load is partitioned in sequence as shown in Fig. 9(a). The load distributed to  $\alpha_0$  is chosen from the beginning of the whole load of which computation begins at  $S_0$ . Fig. 9(b) is a distribution scheme based on the timing diagram of Fig. 8(b), where the fraction of the load containing the larger mass of probability of finding the signature is distributed to  $P_0$  and processed between  $S_0$  and  $S_1$ , during the earliest time. The grayed area of the figures identify the portion of load distributed to  $P_0$ . In the figure, multiple fractions of load contribute to  $\alpha_0$ , the load distributed to  $P_0$ . Note that, as probability mass monotonically increases, the range of  $\beta^i$  is continuous and each  $\beta^i, i > 0$  is partitioned for multiple processors.

The probability distribution function of the signature position shown in the figure is given as

$$f_x(x) = 2x, \quad 0 \leq x \leq 1 \quad (29)$$

For a linear daisy chain network with 5 processors, the  $\{|\alpha_m|\}$  and  $\{S_m\}$  is calculated using recursive equations presented in section II and TABLE II. The calculated values are shown in TABLE III.

In TABLE IV, the size of the fraction of the load distributed to  $P_j$  from  $\beta^i$ ,  $|\alpha_j^i|$  is calculated using equation (27) and equation (28). Although the sizes are determined for  $\alpha_j^i$  and  $\beta_i$ , with the aid of the DLT solution, the

$m$	0	1	2	3	4
$ \alpha_m $	0.299	0.229	0.181	0.152	0.139
$S_m$	0	0.0701	0.117	0.146	0.160

TABLE III

$\alpha_m$  AND  $S_m$  FOR LINEAR DAISY CHAIN NETWORK WITH  $w = 1, z = 0.1, T_{cm} = T_{cp} = 1$

$i$	0	1	2	3	4
$ \alpha_0^i $	0.0701154	0.0472423	0.0290935	0.0138541	0.138541
$ \alpha_1^i $		0.0472423	0.0290935	0.0138541	0.138541
$ \alpha_2^i $			0.0290935	0.0138541	0.138541
$ \alpha_3^i $				0.0138541	0.138541
$ \alpha_4^i $					0.138541
$ \beta^i $	0.0701154	0.0944847	0.0872806	0.0554162	0.692703

TABLE IV

$\alpha_j^i$ , LINEAR DAISY CHAIN NETWORK WITH  $w = 1, z = 0.1, T_{cm} = T_{cp} = 1$

range of the fractions of load still needs to be determined. In Fig. 9(b), the shaded area indicates the fractions of load that need to be distributed to  $P_0$ . Here, the load distributed to  $P_0$  are from multiple parts of the load based on their probability mass. Now  $\alpha_0^0$  which is processed between  $S_0$  and  $S_1$  is taken from the part of load with highest probability mass. The fraction of load processed during the period between  $S_M$  and  $T_{finish}$ ,  $\beta_M$ , is taken from the beginning part (leftmost area in Fig. 9) of the original load because of its small probability mass. Since all the processors process the fraction the load during that period,  $\beta_M$  is partitioned and distributed to all of the processors. As the ranges of fractions are dispersed in the load, the position of each fraction needs to be obtained. To begin, the range of  $\beta_i$  needs to be obtained and then the positions of all  $\alpha_j^i$  for  $j = 1 \dots i$  are calculated.

$l(\beta^4)$	$l(\beta^3)$	$l(\beta^2)$	$l(\beta^1)$	$l(\beta^0)$
0	0.692703	0.748119	0.8354	0.929885

TABLE V

LOWER BOUNDARY OF  $\beta^i$  WITH LINEAR DAISY CHAIN NETWORK WITH  $w = 1, z = 0.1, T_{cm} = T_{cp} = 1, u(\beta^i) = l(\beta^i) + |\beta^i|$

In this example with a monotonically increasing distribution, the range of  $\beta_i$  can be obtained in a straightforward manner with known  $|\beta_i|$  as we can take  $\beta_i$  from the end of the load toward the beginning sequentially as shown in 9(b) and the lower boundaries of  $\beta_j^i$  are calculated and shown in TABLE V. The table shows the lower boundary of  $\beta_i$ , but the upper boundary can be easily derived with  $|\beta_i|$ :  $u(\beta_i) = l(\beta_i) + |\beta_i|$ , where  $u(\cdot)$  denotes the upper

boundary of the load fraction.

$i$	0	1	2	3	4
$l(\alpha_0^i)$	0.929885	0.8354	0.748119	0.692703	0
$l(\alpha_1^i)$		0.882642	0.777213	0.706557	0.138541
$l(\alpha_2^i)$			0.806306	0.720411	0.277082
$l(\alpha_3^i)$				0.734265	0.415623
$l(\alpha_4^i)$					0.554164

TABLE VI

LOWER BOUNDARY OF  $\alpha_m^i, l(\alpha_m^i)$  WITH LINEAR DAISY CHAIN NETWORK WITH  $w = 1, z = 0.1, T_{cm} = T_{cp} = 1,$   
 $u(\alpha_m^i) = l(\alpha_m^i) + |\alpha_m^i|$

The lower boundary of  $\{\alpha_m^i\}$  is shown in TABLE VI. Inside the range of  $\beta_i$ , we partition the  $\beta_i$  in the order of the processor index for simplicity. Therefore,  $l(\alpha_0^1) = l(\beta_1)$  and  $l(\alpha_1^1) = l(\beta_1) + |\alpha_0^1|$  and other lower boundaries are obtained similarly. Other ways of partitioning  $\beta^i$  are possible but are not considered in this paper.

In order to find the expected signature search time for this distribution scheme, equation (5) can be re-written as

$$E[\mathbf{Y}] = \sum_{m=1..M, j \leq m} E[\mathbf{Y}|A_m^j] Pr(A_m^j), \quad (30)$$

where  $A_m^i$  denotes an event when the signature is found in  $\alpha_m^i$ . Similarly to equation (7),

$$\begin{aligned} \mathbf{Y}|A_m^i &= g_m((X)|A_m^i) \\ &= (\mathbf{X}|A_m^i - l(\alpha_m^i))w_m T_{cp} + S_m, \end{aligned} \quad (31)$$

where  $l(\alpha_m^i)$  denotes the starting position of  $\alpha_m^i$  in load.

Taking expectation of  $\mathbf{Y}|A_m^i$  gives

$$E[\mathbf{Y}|A_m^i] = (E[\mathbf{X}|A_m^i] - l(\alpha_m^i))w_m T_{cp} + S_m. \quad (32)$$

The conditional expectation of the location of the signature given that it is distributed to  $P_m$  and from  $\beta^i$ ,

$$E[\mathbf{X}|A_m^i] = \frac{\int_{l(\alpha_m^i)}^{u(\alpha_m^i)} x f_x(\zeta) d\zeta}{\int_{l(\alpha_m^i)}^{u(\alpha_m^i)} f_x(\zeta) d\zeta}. \quad (33)$$

Also, the probability of the event  $A_m^i$  is,

$$Pr(A_m^i) = \int_{l(\alpha_m^i)}^{u(\alpha_m^i)} f_x(\zeta) d\zeta. \quad (34)$$

Finally, substituting the above equations into (30),

$$\begin{aligned} E[\mathbf{Y}] &= \sum_{(m,i)} \left( \left( \frac{\int_{l(\alpha_m^i)}^{u(\alpha_m^i)} x f_x(\zeta) d\zeta}{\int_{l(\alpha_m^i)}^{u(\alpha_m^i)} f_x(\zeta) d\zeta} - l(\alpha_m^i) \right) w_m T_{cp} + S_m \right) \\ &\quad * \int_{l(\alpha_m^i)}^{u(\alpha_m^i)} f_x(\zeta) d\zeta. \end{aligned} \quad (35)$$



When probability distribution is given as (29), we have,

$$\begin{aligned} \int_{l(\alpha_m^i)}^{u(\alpha_m^i)} x f_x(\zeta) d\zeta &= \int_{l(\alpha_m^i)}^{l(\alpha_m^i)+|\alpha_m^i|} 2x^2 dx \\ &= \frac{6|\alpha_m^i|^2 l(\alpha_m^i) + 6|\alpha_m^i| l^2(\alpha_m^i) + 2|\alpha_m^i|^3}{3} \end{aligned} \quad (36)$$

and

$$\begin{aligned} \int_{l(\alpha_m^i)}^{u(\alpha_m^i)} f_x(x) dx &= \int_{l(\alpha_m^i)}^{l(\alpha_m^i)+|\alpha_m^i|} 2x dx \\ &= 2|\alpha_m^i| l(\alpha_m^i) + |\alpha_m^i|^2. \end{aligned} \quad (37)$$

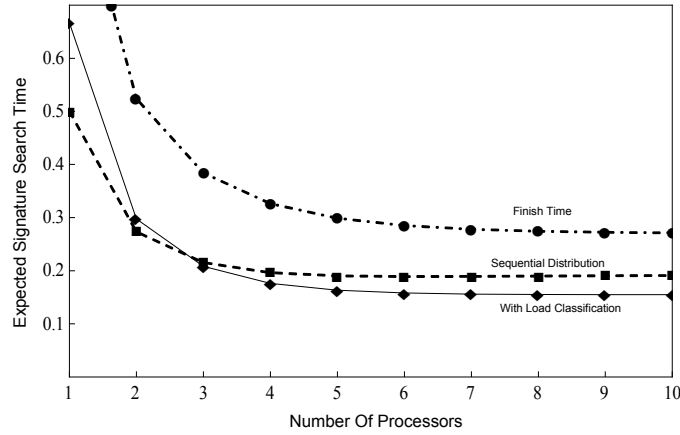


Fig. 10. Comparison of expected time to find a single signature between sequential distribution and distribution based on load classification based on the probability distribution on single level tree network,  $w_i = 1$ ,  $z_i = 0.2$ ,  $T_{cp} = 1$ ,  $T_{cm} = 1$ .

By using the calculated values from the above tables, we plot the expected search time using the usual sequential distribution in the DLT literature and when the load is classified by probability mass in Fig. 10. As shown in the figure, when the load is classified by the probability mass and scheduled with consideration of the starting time of each processor the expected time to find signatures is faster. The finish time of computation is shown in the plot for comparison.

Similar curves for a linear daisy chain of processors as in Figure 10 could be produced using equations (23) through (37).

## V. GREEDY PROCESSING: ARBITRARY DENSITIES

### A. Greedy Load Rearrangement Procedure

In this section, we present a greedy procedure to distribute load based on the distribution of a single signature as introduced in the previous section. We assume that the probability distribution of the location of a single signature is known. In this approach, the load is rearranged first in order of the likelihood of finding the signature. Fig. 11 describes the procedure. As shown in Fig. 11(a) and Fig. 11(b), the load is sliced into equal sized bins. Practically

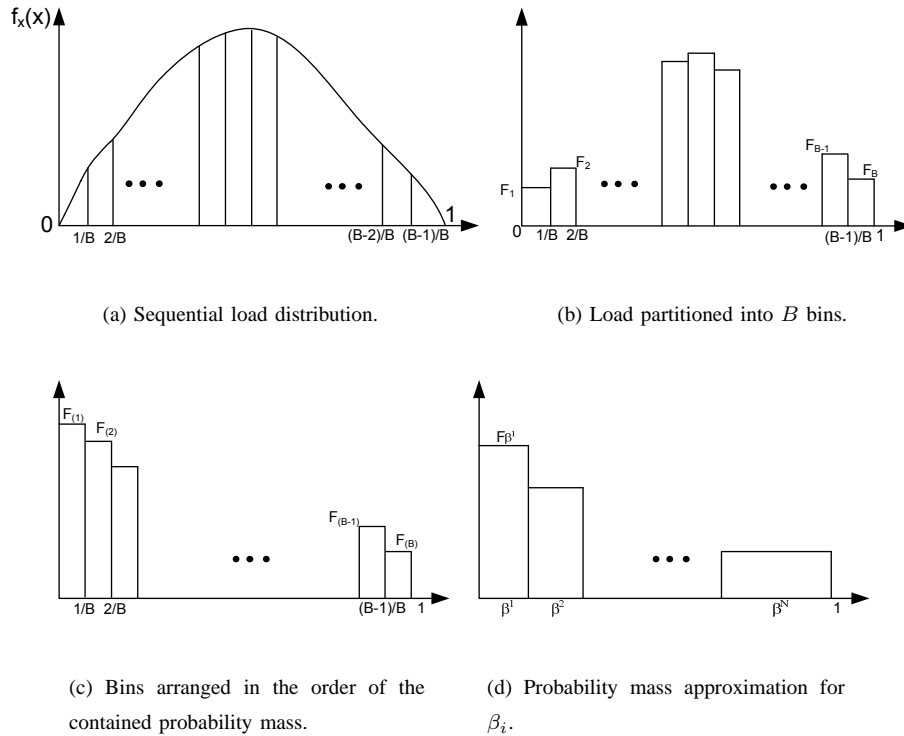


Fig. 11. Greedy rearrangement of loads based on ranking.

bins can be rearranged by a server that feeds load to a number of parallel processors according to an a priori known pattern (distribution) of signature location(s). The choice of bin size should take into account the number of processors available, the desired solution time and the saturating nature of the search time curves. These bins are arranged in decreasing order of probability mass (see Fig. 11(c)). A further approximation step (Fig. 11(d)) is described in the next subsection. Note that once we get as far as the steps of Fig. 11(c) and 11(d) then the results of the previous section apply.

To be specific, we denote  $B$  the total number of bins and, therefore, the normalized size of each bin is  $1/B$ . In the next step, shown in 11(b), the probability mass of each bin is calculated as,

$$F_n = \int_{b_{n-1}}^{b_n} f_x(x) dx, \quad (38)$$

where  $F_n$  denotes the probability mass of  $n$ th bin and  $b_n = n * \frac{1}{B}$ . Once the mass of each bin is calculated the load is sliced and rearranged in the order of the decreasing magnitude of the probability mass as shown in Fig. 11(c). We call this the sorted load.

Finally, the load is distributed according to pre-calculated  $\{\beta^i\}$ . Once the load is sorted according to the probability mass in the previous step, the range and the size of  $\{\beta_i\}$  does not vary for a given network model with the same system parameters independently from the probability distribution of signatures. For example, the range and size of

	$\beta^0$	$\beta^1$	$\beta^2$	$\beta^3$	$\beta^4$
$ \beta^i $	0.0701154	0.0944847	0.0872806	0.0554162	0.692703
$u(\beta^i)$	0.0701154	0.1646	0.251881	0.307297	1

TABLE VII

$|\beta^i|$  AND UPPER BOUNDARIES OF  $\beta^i$  WITH LINEAR DAISY CHAIN NETWORK FOR SCALED NORMAL DISTRIBUTION WITH  $w = 1, z = 0.1$ ,  
 $T_{cm} = T_{cp} = 1$

$\{\beta^i\}$  can be calculated for the linear daisy network with  $w = 1, z = 0.1$  and  $T_{cm} = T_{cp} = 1$  as shown in TABLE VII. The size of  $\beta_i$  are obtained using (23), (24) and (25). The range of  $\beta_i$  is taken from the beginning of the sorted load. Therefore, the upper boundaries shown in TABLE VII are the values in the sorted load, not in the original load.

### B. Approximation of Expected Signature Search Time

Fig. 11(d) illustrates the approximated probability mass assigned to the portion of load with the same ranking. Since the exact distribution for a portion of the load with the same ranking is not specified, the actual calculation of the expected search time can only be approximated. The approximated value of the expected time of signature search time is given as,

$$\hat{E}[X] = \sum_{i=1}^N \frac{S_{i-1} + S_i}{2} * \hat{F}_{\beta^i} \quad (39)$$

where  $\hat{F}_i$  denotes the approximated probability mass calculated for  $\{\beta^i\}$ .

$$\hat{F}_{\beta^i} = F_{(k-1)} \frac{(b_k - l(\beta^i))}{1/B} + \sum_{j=k}^p F_{(j)} + F_{(p+1)} \frac{(u(\beta^i) - b_p)}{1/B} \quad (40)$$

Here  $b_{k-1} \leq l(\beta^i) \leq b_k$  and  $b_p < u(\beta^i) \leq b_{p+1}$ . Here the b's are the integration limits of equation (38). Here  $F_{(k)}$  is the probability mass of the  $k$ th bin after sorting.

Intuitively, this equation shows that the approximated expected signature search time is the weighted sum of the mid points of processing time of  $\{\beta_i\}$  with their approximated probability masses.

## VI. EVALUATION AND ANALYSIS

As an example, assume that the distribution of the location of signature time is given as a truncated normal distribution on the range  $[0, 1]$  centered at the  $1/2$ . The distribution can be written as,

$$f(x; \mu, \sigma, a, b) = \frac{\frac{1}{\sigma} \phi\left(\frac{x-\mu}{\sigma}\right)}{\Phi\left(\frac{b-\mu}{\sigma}\right) - \Phi\left(\frac{a-\mu}{\sigma}\right)} \quad (41)$$

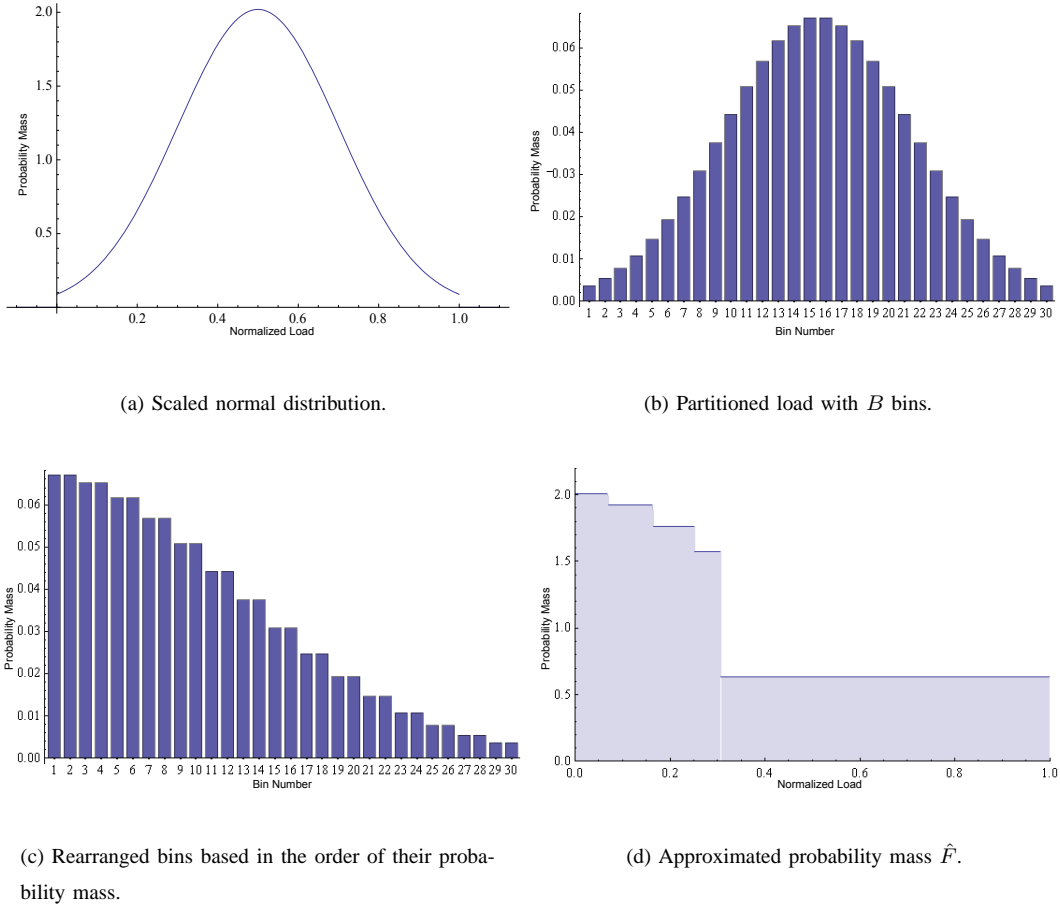


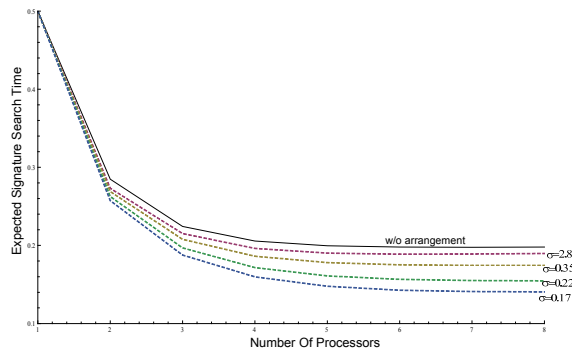
Fig. 12. Greedy rearrangement of loads based on ranking for a truncated normal distribution.

Here  $\phi(\cdot)$  is the probability density function of the standard normal distribution, and  $\Phi(\cdot)$  is its cumulative distribution function. Also,  $a$  and  $b$  denote the limits of the range of truncation, and  $\mu$  denotes the average and  $\sigma$  is the standard deviation value. With  $a = 0$ ,  $b = 1$  and  $\mu = 1/2$ ,

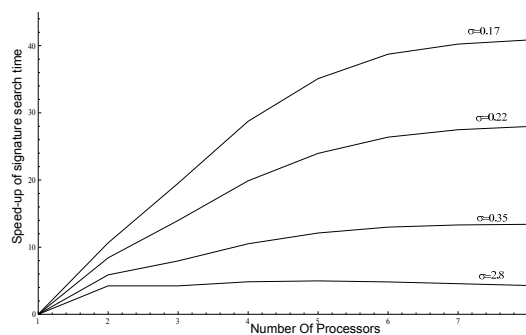
$$f(x; 1/2, \sigma, 0, 1) = \frac{\frac{1}{\sigma} \phi\left(\frac{x-1/2}{\sigma}\right)}{\Phi\left(\frac{1}{2\sigma}\right) - \Phi\left(-\frac{1}{2\sigma}\right)}, \quad (42)$$

This distribution is shown in Fig. 12(a). Following the procedure presented in the previous section, the load is sliced into bins as shown in Fig. 12(b) and rearranged based on the probability mass of the bins as shown in Fig. 12(c). The figures are based on 30 bins being used. Finally, the approximated probability mass of  $\beta^i$  is plotted in Fig. 12(d).

Fig. 13(a) shows the expected signature search time with and without load arrangement procedure with various standard deviations,  $\sigma$ . The figure was computed using (39) and (40) and the definition of speedup. Fig. 13(b) shows the performance improvement in terms of percentage, where the speed up is defined as the ratio of signature search time on one processor to signature search time on  $N$  processors. Naturally speedup is greater than one but



(a) Expected signature search time.



(b) Speed up of signature search time.

Fig. 13. Performance improvement (%) through greedy load rearrangement for various  $\sigma$ .

is often less than  $N$  because of inefficiencies in parallel processing. “Speedup” is a common parallel processing advantage performance metric. As shown in the figures, as the probability of finding a signature is concentrated in a smaller area (lower  $\sigma$  value) the performance gain increases.

## VII. CONCLUSION AND FUTURE WORK

In the first part of this work, closed form solutions of the expected search time of the  $k$ th signature of  $K$  signatures, with a uniform distribution of signature locations, are derived. In the latter part, with a prior knowledge of the signature distribution, it is shown that the expected time of finding a single signature can be improved and a greedy procedure to speed up the signature search time and a simulation result are presented. With highly concentrated distributions, the improvement in the speed of finding a signature is shown to be significant. As extensions of this work one can examine how database operations can be mapped to divisible load scheduling and investigate how a knowledge of input data can be used to create distribution strategies that improve performance.

The trend in data processing is the use of parallel processing to speed execution even to the level of a single machine (i.e. multicore architectures). Thus for radar, sensor and image data processing, the use of divisible load

theory leads to a better quantitative understanding of processing performance and processing options. Thus the result in this paper should be of interest for quite some time.

#### A. Acknowledgment

T. Robertazzi acknowledges the support of DOE grant DE-SC0003361.

#### REFERENCES

- [1] Ko, K. and Robertazzi, T.G.  
Signature search time evaluation in flat file databases.  
*IEEE Transactions on Aerospace and Electronic Systems*,  
**44**, 2 (April 2008), 493-502.
- [2] Cheng, Y.C. and Robertazzi, T.G.  
Distributed computation with communication delays.  
*IEEE Transactions on Aerospace and Electronic Systems*,  
**24**, 6 (1988), 700-712.
- [3] Cheng, Y.C. and Robertazzi, T.G.  
Distributed computation for tree networks with communication delays.  
*IEEE Transactions on Aerospace and Electronic Systems*,  
**26** (1990) 511-516.
- [4] Agrawal, R. and Jagadish, H.  
Partitioning techniques for large-grained parallelism.  
*Proceedings of the Seventh Annual International  
Phoenix Conference on Computers and Communications*  
March 1988, 31-38.
- [5] Bharadwaj, V., Ghose, D., Mani, V. and Robertazzi, T.  
*Scheduling Divisible Loads in Parallel and Distributed Systems*.  
Los Alamitos, CA, USA: IEEE Computer Society Press, 1996.
- [6] Robertazzi, T.  
Ten reasons to use divisible load theory.  
*Computer*,  
**36**, 5 (May 2003), 63-68.
- [7] Bharadwaj, V., Ghose, D. and Robertazzi, T.  
A new paradigm for load scheduling in distributed systems.  
*Cluster Computing*,  
**6**, 1 (Jan. 2003) 7-18.
- [8] Drozdowski, M.  
*Scheduling for Parallel Processing*.  
New York, NY: Springer, 2009.
- [9] Casanova, H., Legrand, A. and Robert, Y.  
*Parallel Algorithms*  
Florida: CRC Press, 2009.
- [10] Robertazzi, T.  
*Networks and Grids: Technology and Theory*.  
New York: Springer, 2007.
- [11] Veeravalli, B. and Min, W.H.

- Scheduling divisible loads on heterogeneous linear daisy chain networks with arbitrary processor release times.  
*IEEE Transactions on Parallel and Distributed Systems*,  
**15**, 3 (March 2004), 273-288.
- [12] Veeravalli, B., Li, X. and Ko, C.C.  
On the influence of start-up costs in scheduling divisible loads on bus networks.  
*IEEE Transactions on Parallel and Distributed Systems*,  
**11**, 12 (2000), 1288-1305.
- [13] 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 (April 1995), 555-567.
- [14] Charcranoon, S., Robertazzi, T. and Luryi, S.  
Parallel processor configuration design with processing/transmission costs.  
*IEEE Transactions on Computers*,  
**49**, 9 (Sept. 2000), 987-991.
- [15] Jingxi, J., Veeravalli, B. and Ghose, D.  
Adaptive load distribution strategies for divisible load processing on resource unaware multilevel tree networks.  
*Computers, IEEE Transactions on*,  
**56**, 7 (July 2007), 999-1005.
- [16] Drozdowski, M. and Glazek, W.  
Scheduling divisible loads in a three-dimensional mesh of processors.  
*Parallel Computing*,  
**25**, 4 (1999), 381-404.
- [17] Chang, Y.K., Wu, J.H., Chen, C.Y. and Chu, C.P.  
Improved methods for divisible load distribution on k-dimensional meshes using multi-installment.  
*IEEE Transactions on Parallel and Distributed Systems*,  
**18**, 11 (2007) 1618-1629.
- [18] Li, X., Veeravalli, B. and Ko, C.C.  
Divisible load scheduling on a hypercube cluster with finite-size buffers and granularity constraints.  
*First IEEE/ACM International Symposium on Cluster Computing and the Grid, 2001 Proceedings*.  
2001, 660-667.
- [19] Yao, J. and Veeravalli, B.  
Design and performance analysis of divisible load scheduling strategies on arbitrary graphs.  
*Cluster Computing*,  
**7**, 2 (2004), 191-207.
- [20] Bharadwaj, V. and Barlas, G.  
Scheduling divisible loads with processor release times

- and finite size buffer capacity constraints in bus networks.  
*Cluster Computing*,  
**6**, 1 (2003), 63-74.
- [21] Drozdowski, M. and Lawenda, M.  
Multi-installment divisible load processing in  
heterogeneous distributed systems.  
*Concurrency and Computation: Practice and Experience*,  
**19**, 17 (2007) 2237-2253.
- [22] Zeng, Z. and Veeravalli, B.  
Divisible load scheduling on arbitrary distributed  
networks via virtual routing approach.  
*Proceedings of the Tenth International Conference  
on Parallel and Distributed Systems (ICPADS'04)*,  
2004, 161.
- [23] Ghose, D., Kim, H.J. and Kim, T.H.  
Adaptive divisible load scheduling strategies for  
workstation clusters with unknown network resources.  
*IEEE Transactions on Parallel and Distributed Systems*,  
**16**, 10 (Oct. 2005), 897-907.
- [24] Li, H., Sun, G. and Xu, Y.  
Distributed scheduling strategies for processing  
multiple divisible loads with unknown network resources.  
*IFIP International Conference on Network and  
Parallel Computing, NPC Workshops, 2007*.  
Sept. 2007, 824-829.
- [25] Drozdowski, M. and Wolniewicz, P.  
Experiments with scheduling divisible tasks  
in clusters of workstations.  
*Proceedings of the 6th International  
Euro-Par Conference on Parallel Processing*.  
2000, 311-319.
- [26] Christodoulakis, S.  
Implications of certain assumptions in database  
performance evaluation.  
*ACM Transactions Database Systems*,  
**9**, 2 (1984) 163-186.