Estimating the Execution Time Distribution for a Task Graph in a Heterogeneous Computing System*

Yan Alexander Li
Intel Corporation, SC9-15
2200 Mission College Blvd.
Santa Clara, CA 95052-8119 USA
ali2@mipos3.intel.com

John K. Antonio
Department of Computer Science
Texas Tech University
Lubbock, TX 79409-3104 USA
antonio@cs.ttu.edu

Abstract

The problem of statically estimating the execution time distribution for a task graph consisting of a collection of subtasks to be executed in a heterogeneous computing (HC) system is considered. Execution time distributions for the individual subtasks are assumed to be known. A mathematical model for the communication network that interconnects the machines of the HC system is introduced, and a probabilistic approach is developed to estimate the overall execution time distribution of the task graph. It is shown that, for a given matching and scheduling, computing the exact distribution of the overall execution time of a task graph is very difficult, and thus impractical. The proposed approach approximates the exact distribution and requires a relatively small amount of calculation time. The accuracy of the proposed approach is demonstrated mathematically through the derivation of bounds that quantify the difference between the exact distribution and that provided by the proposed approach. Numerical studies are also included to further validate the utility of the proposed methodology.

1 Introduction

A heterogeneous computing (HC) system provides a variety of architectural capabilities, orchestrated to perform an application whose subtasks have diverse execution requirements [1]. HC has become a subject of intensive research within the high-performance computing community in the quest of systems that are versatile and provide good performance. For a description of example HC applications and a list of related references, refer to [1].

*This work was supported by Rome Laboratory under grant number F30602-96-1-0098.

Throughout this paper, an HC system is assumed to consist of a suite of independent machines of different types interconnected by a high-speed network. HC requires the effective use of diverse hardware and software components to meet the distinct and varied computational requirements of a given application. Implicit in the concept of HC is the idea that subtasks with different machine architectural requirements are embedded in the applications executed by the HC system. The concept of HC is to decompose a task into computationally homogeneous subtasks, and then assign each subtask to the machine where it is best suited for execution [1].

Unlike in distributed homogeneous systems (e.g., a network of workstations of the same type and configuration), it is generally difficult and impractical to suspend the execution of a subtask on one machine and resume that subtask's execution on another machine of a different type in an HC system. Thus, a challenge in making effective use of an HC system is to minimize the need for such dynamic subtask migration, which implies an increased importance on the static problems of assigning subtasks to machines (matching) and ordering the execution of subtasks assigned to the same machine (scheduling).

Performance prediction is the basis of matching and scheduling techniques for HC systems. Many matching and scheduling algorithms make the simplifying assumption that the execution time for each subtask is a known constant for each machine in the system (e.g., [2, 3]). However, there are elements of uncertainty, such as the uncertainty in input data values or in inter-machine communication time, which can impact the execution times. Machine choices for executing subtasks can also affect the execution time and its degree of uncertainty.

In this paper, the task to be executed on the HC system is modeled as a task graph consisting of a collection of subtasks. A mathematical model for the communication network that interconnects the machines of the HC system is introduced, and a probabilistic approach is developed to estimate the overall execution time distribution of the task graph. This overall distribution depends on the individual subtask execution time distributions, the inter-machine communication time distributions, the data dependency structure among the subtasks, the matching of subtasks to machines, and the scheduling of subtasks matched to a common machine. It is shown that, for a given matching and scheduling, computing the exact distribution of the overall execution time of a task graph is very difficult, and thus impractical. The proposed approach approximates the exact distribution and requires a relatively small amount of calculation time. The accuracy of the proposed approach is demonstrated mathematically through the derivation of bounds that quantify the difference between the exact distribution and that provided by the proposed approach. Numerical studies are also included to further validate the utility of the proposed methodology.

Section 2 presents the basic assumptions and an overview of the approach. A mathematical framework for the approach is presented in Section 3. Section 4 demonstrates the generic difficulty associated with calculating the exact execution time distribution for a task graph. An approximate approach is then proposed based on the conditions set forth by the Kleinrock independence approximation [4]. Section 4 concludes with a mathematical derivation of a bound for quantifying the difference between the exact distribution and that associated with the proposed approach. In Section 5, execution time distributions determined using the proposed approach are compared with corresponding distributions obtained by simulation of example task graphs. These studies indicate that the proposed approach predicts the execution time distribution for a large class of practical task graphs with high accuracy.

2 Assumptions and Overview

It is assumed that the HC system consists of a dedicated network of machines under the control of a single matching/scheduling mechanism. The type of application task considered is composed of a number of subtasks, each of which is to be executed on a particular machine in the HC system. The execution time distribution for each individual subtask is assumed to be known or estimated for the machine on which it is to be executed. Previous approaches for determining the execution time distribution of (parallel) programs (e.g., [5, 6]) could be applied for estimating the execution time distribution of subtasks. Estimates of subtask execution time distributions based on empirical measurements could also be utilized in the framework assumed here.

The subtask-to-machine matching and the order of execution for multiple subtasks assigned to the same machine (i.e., the subtask scheduling for each machine) are assumed to be static and known. The problems of optimal matching and scheduling represent large bodies of research in the field of HC, e.g., see [3, 7]. How to determine good solutions to the matching and scheduling problems is beyond the scope of this paper. The goal here is to develop a new probabilistic approach for analyzing the overall task execution time for given matching and scheduling policies. From this probabilistic modeling foundation, future matching and scheduling techniques may be developed that are based on probabilistic metrics for performance.

For each subtask, all input data items must be present at the subtask's designated machine before computation starts, and output data items can be transferred to other machines only after computation of the subtask is completed. Data-dependencies among the subtasks are represented by a task graph. A task graph is a directed acyclic graph in which nodes represent subtasks and arcs represent the data-dependencies among the subtasks.

To compute the execution time distribution of the entire task, the assumed execution time distributions of all subtasks (on their designated machines) are utilized. These distributions correspond to the computation time of the subtasks only; distributions for the times required to input and output any data structures are defined separately.

For each subtask, the machine from which each required input data item is fetched is assumed to be specified. In general, these machines will depend on the subtask-to-machine matching that is used. For example, fetching a data item from the machine at which it was first generated (or was initially stored) is a simple rule that is often assumed for this type of analysis. However, the model devised here allows for more general refinements in how the data is distributed and retrieved. For example, the model is general enough to account for the data-reuse and multiple data-copies situations [8], which allow the fetching of data items from a machine other than the one where it was generated.

Network I/O at each machine is assumed to be non-blocking (e.g., handled by a stand-alone I/O processor). Therefore, computation and inter-machine communication can be overlapped in time. Each subtask is assumed to start computation when its designated machine is ready and all its input data items are available on this machine. Immediately after computation for a subtask completes, the machine is made available for executing the next subtask scheduled for execution on that machine. Also, the output data items produced by the completed subtask are made available for all subsequent subtasks to be executed on that machine. If multiple data items produced by a subtask are to be transferred to other machines, the order in which they are sent is assumed to be specified. All outgoing data items at a given machine are assumed to be buffered when the machine is transmitting another data item.

In general, the network transmission time for a given data item (including the uncertain delay caused by network contention) depends on factors such as the size of the data item being transferred, the topology and bandwidth of the network, the type of switching used, the types of machine-network interfaces used, and the overall network load. For the purpose of this analysis, a network model in which the transmission time is modeled according to a probabilistic distribution is assumed. In this model, the shape of the distribution is fixed (e.g., the variance is fixed), and the mean of the distribution is defined to be the sum of a fixed overhead and a term proportional to the size of data item transfered. The fixed overhead corresponds to the latency of the network, and the coefficient for the second term corresponds to the inverse of the channel bandwidth. This represents one possible model for the transmission time for a network. Other network models are possible and could be used in place of the one assumed here. Furthermore, the transmission time can be source-destination dependent. The only requirement is that the transmission time be modeled according to a probabilistic distribution.

Three separate random variables are used to represent the start time, execution time, and completion time for each subtask. The values of the start and finish times are defined with respect to a global time-line, and the subtask execution time represents the length of an interval on this time-line. The task is assumed to start execution at time 0. A subtask is called a terminal subtask if it corresponds to a node with no successors in the task graph and is the last subtask executed on its designated machine. The maximum of the completion times over all terminal subtasks defines the finish time of the entire task.

3 Mathematical Model of Task Graph Execution in an HC System

In this section, random variables are used to model the data communication times among the machines and the start time, execution time, and finish time for each subtask. The relationships among these random variables are derived. These are used in the next section to compute the overall task execution time distribution by performing appropriate operations to the distribution functions of these random variables.

It is assumed that there are $\underline{m}$ machines in the HC system, labeled $\underline{M_i}$, $i = 0, 1, \ldots, m-1$. The task consists of $\underline{n}$ subtasks, labeled $\underline{S_j}$, $j = 0, 1, \ldots, n-1$. The subtask-to-machine matching is defined by the function

$$\underline{\mathcal{M}} : \{0, \ldots, n-1\} \to \{0, \ldots, m-1\}. \qquad (1)$$

Thus, subtask S_j is to be executed on machine $M_{\mathcal{M}(j)}$. For each machine M_i, the number of subtasks assigned to M_i is denoted as $\underline{\alpha_i}$, and the execution schedule for these α_i subtasks is defined by the function $\underline{\mathcal{X}_i}$: $\{0, \ldots, \alpha_i - 1\} \to \mathcal{M}^{-1}(i)$, which is a bijection. Thus, $\mathcal{X}_i(k)$, $0 \le k < \alpha_i$, defines the $(k+1)$-th task to be executed; the sequence of execution on machine M_i is from subtask $S_{\mathcal{X}_i(0)}$ to $S_{\mathcal{X}_i(\alpha_i - 1)}$.

For each subtask S_j, $0 \le j < n$, define $\underline{n_j^I}$ and $\underline{n_j^O}$ to be the number of associated input and output data items, respectively. Input data items of S_j are labeled $\underline{D_{j,v}^I}$, $0 \le v < n_j^I$. Output data items of S_j are labeled $\underline{D_{j,u}^O}$, $0 \le u < n_j^O$. If multiple output data items of a subtask need to be transmitted to other machine(s), then they are transmitted serially in ascending index order, i.e., $D_{j,u}^O$ is transmitted before $D_{j,\ell}^O$, for $u < \ell$.

For each subtask S_j, the times at which computation starts and finishes is modeled by random variables $\underline{T_j^S}$ and $\underline{T_j^F}$, respectively. The execution time of S_j is modeled by random variable $\underline{\tau_j^E}$, defined as $\tau_j^E = T_j^F - T_j^S$. Note that throughout this paper, values of random variables involving the letter "T" correspond to points on the global time-line, and those that use "τ" represent lengths of intervals on this time-line. It is assumed that the execution times of all subtasks are independent, i.e., τ_j^E, $0 \le j < n$, form a set of mutually independent random variables. This has been an assumption made by other researchers as well, e.g., [9]. Based on the definition of τ_j^E, a useful expression for T_j^F is given by:

$$T_j^F = T_j^S + \tau_j^E. \qquad (2)$$

For each subtask S_j, the time at which input data item $D^I_{j,v}$, $0 \le v < n^I_j$, becomes available on machine $M_{\mathcal{M}(j)}$ is defined by $\underline{T^I_{j,v}}$. It is assumed that all initial data items are pre-loaded to the machines that will first use them, i.e., the time required to load these data items is not considered in the analysis. Thus, the available time for all pre-loaded data items is defined as 0. The sum of the queuing time, denoted by $\underline{\tau^Q_{j,u}}$, and the network time, denoted by $\underline{\tau^N_{j,u}}$, defines the time period starting at time T^F_j and ending when data item $D^O_{j,u}$ is available at its destination subtask. If the destination subtask of $D^O_{j,u}$ is on the same machine as S_j, then both $\tau^Q_{j,u}$ and $\tau^N_{j,u}$ are defined to be 0. Otherwise, $\tau^Q_{j,u}$ represents the time $D^O_{j,u}$ waits in the queue before machine $M_{\mathcal{M}(j)}$ begins to transmit it, and $\tau^N_{j,u}$ represents the amount of time (including any delay caused by network contention) to transfer $D^O_{j,u}$ through the network to its destination. Network times for different output data items are assumed to be independent (i.e., the random variables $\tau^N_{j,u}$ are independent).

In the following discussion, let output data item u of S_j (i.e., $D^O_{j,u}$) be input data item v of subtask S_g (i.e., $D^O_{j,u} = D^I_{g,v}$). Also, if S_j and S_g are assigned to the same machine, i.e., $\mathcal{M}(g) = \mathcal{M}(j)$, then $\tau^Q_{j,u} = 0$ for all u. If $u = 0$ (i.e., $D^O_{j,u} = D^O_{j,0}$ is the first data item transmitted), then the queuing time is zero, i.e., $\tau^Q_{j,0} = 0$. Hence, the general expression for $\tau^Q_{j,u}$ is:

$$\tau^Q_{j,u} = \begin{cases} 0 & \text{if } u = 0 \\ & \text{or } \mathcal{M}(j) = \mathcal{M}(g), \\ \tau^Q_{j,u-1} + \tau^N_{j,u-1} & \text{otherwise.} \end{cases} \quad (3)$$

It is assumed that $D^O_{j,u}$ is available to S_g immediately upon arriving at machine $M_{\mathcal{M}(g)}$. Let $\underline{T^A_{j,u}}$ define this arrival time, then

$$T^A_{j,u} = T^I_{g,v} = T^F_j + \tau^Q_{j,u} + \tau^N_{j,u}. \quad (4)$$

Assume now that subtask S_j is to be executed on machine M_i, i.e., $\mathcal{M}(j) = i$, and is the $(k+1)$-th subtask scheduled for execution, i.e., $\mathcal{X}_i(k) = j$. Let $\underline{T^M_j}$ denote the time that M_i becomes available for executing S_j. If $k = 0$, i.e., S_j is the first subtask scheduled to execute on M_i, then T^M_j is defined to be 0. Otherwise, $S_{\mathcal{X}_i(k-1)}$ is the previous subtask that executes on M_i, and T^M_j is defined to be the finish time of $S_{\mathcal{X}_i(k-1)}$. Therefore, the general relation for the time when machine M_i becomes available for executing subtask $S_{\mathcal{X}_i(k)}$ is:

$$T^M_{\mathcal{X}_i(k)} = \begin{cases} 0 & \text{if } k = 0, \\ T^F_{\mathcal{X}_i(k-1)} & \text{otherwise.} \end{cases} \quad (5)$$

The start time of a subtask is the maximum of the available time of the designated machine and the maximum of all arrival times of its input data items:

$$T^S_j = \max\left\{T^M_j, \max_{v=0}^{n^I_j - 1}\{T^I_{j,v}\}\right\}. \quad (6)$$

Equations (2) through (6) establish the relationships among the defined random variables, and are used to derive their associated probability distribution and/or density functions. In particular, distributions for the random variables τ^E_j and $\tau^N_{j,u}$ are assumed to be specified, and distributions for the other random variables are defined based on the relationships derived in this section. The overall execution time distribution of a task graph is analyzed in the next section.

4 Calculating the Execution Time Distribution for a Task Graph

4.1 Difficulty with exact calculation

In a task graph, subtasks that require input data items from a common subtask have correlated start and finish times, and thus their associated random variables are not independent. This correlation can propagate to the start and finish times of subsequent subtasks that get data from these subtasks. All such subtasks correspond to nodes in the task graph that have a common ancestor. It is shown in this subsection that this type of correlation generally makes the exact derivation of the overall execution time distribution of a task graph difficult and impractical.

Before demonstrating the difficulty of performing basic operations on correlated random variables, the summation and maximum operations for independent random variables are first reviewed. From basic probability theory, recall that the density function of the summation of independent random variables is the convolution of the density functions of the individual random variables [10]. Thus, for two independent random variables, say R and V with density functions $f_R(\cdot)$ and $f_V(\cdot)$, the density function of $Y = R + V$ is given by the convolution of $f_R(\cdot)$ and $f_V(\cdot)$, denoted by $\underline{f_Y(\cdot) = f_R(\cdot) * f_V(\cdot)}$, which is defined by:

$$f_Y(y) = \int_{-\infty}^{\infty} f_R(y-t) f_V(t) dt. \quad (7)$$

Also recall that the distribution function of the maximum of independent discrete random variables is the product of the distribution functions of the individual random variables [10]. Thus, for two independent discrete random variables, say R and V with distribution functions $F_R(\cdot)$ and $F_V(\cdot)$, the distribution function of $Z = \max\{R, V\}$ is given by

$$F_Z(z) = F_R(z) \cdot F_V(z). \tag{8}$$

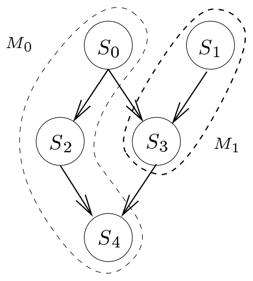


Figure 1: An example task graph whose overall execution time distribution is difficult to derive.

Consider the task graph with five subtasks shown in Fig. 1. Assume there are two machines in the HC system. Subtasks S_0, S_2 and S_4 are assigned to machine M_0, and subtasks S_1 and S_3 are assigned to machine M_1. To simplify the presentation, the network communication times are assumed to be zero, i.e., $\tau_{j,u}^N = 0$ for all j and u. Recall that τ_j^E denotes the given execution time distribution of S_j on its designated machine. The start time of subtask S_4 can be derived by using Equations (2)–(6) as follows.

$$\begin{aligned}
T_0^F &= \tau_0^E, \\
T_1^F &= \tau_1^E, \\
T_2^S &= T_0^F = \tau_0^E, \\
T_3^S &= \max\{T_0^F, T_1^F\} = \max\{\tau_0^E, \tau_1^E\}, \\
T_2^F &= \tau_0^E + \tau_2^E, \\
T_3^F &= \max\{\tau_0^E, \tau_1^E\} + \tau_3^E, \\
T_4^S &= \max\{T_2^F, T_3^F\} \\
&= \max\{\tau_0^E + \tau_2^E, \max\{\tau_0^E, \tau_1^E\} + \tau_3^E\}. \qquad (9)
\end{aligned}$$

Because T_2^F and T_3^F are not independent, Equation (8) is not applicable for computing the distribution of T_4^S. The only way to compute the exact distribution for T_4^S is to consider T_4^S as a function of τ_0^E, τ_1^E, τ_2^E, and τ_3^E (which are assumed to be independent random variables) and use direct integration. To simplify the notation, let $\tau_0^E = A$, $\tau_1^E = B$, $\tau_2^E = C$, $\tau_3^E = D$, and $T_4^S = X$. With these substitutions, Equation (9) is

$$X = \max\{A + C, \max\{A, B\} + D\}.$$

An exact derivation of the distribution function for X (i.e., T_4^S) based on basic probability theory is as follows.

$$\begin{aligned}
&F_X(x) \\
&= \Pr[\max\{(A+C), \max\{A, B\} + D\} \le x] \\
&= \Pr[A + C \le x, \max\{A, B\} + D \le x] \\
&= \int \Pr[A + C \le x, \max\{A, B\} + D \le x | D = d] \\
&\qquad dF_D(d) \\
&= \int \Pr[A + C \le x, \max\{A, B\} \le x - d]\; dF_D(d) \\
&= \iint \Pr[A + C \le x, \max\{A, B\} \le x - d | B = b] \\
&\qquad dF_B(b)\; dF_D(d) \\
&= \iint \Pr[A + C \le x, \max\{A, b\} \le x - d] \\
&\qquad dF_B(b)\; dF_D(d) \\
&= \iiint \Pr[A + C \le x, \max\{A, b\} \le x - d | A = a] \\
&\qquad dF_A(a)\; dF_B(b)\; dF_D(d) \\
&= \iiint \Pr[C \le x - a, \max\{a, b\} \le x - d] \\
&\qquad dF_A(a)\; dF_B(b)\; dF_D(d) \\
&= \iiint F_C(x - a) I(\max\{a, b\} \le x - d) \\
&\qquad dF_A(a)\; dF_B(b)\; dF_D(d), \qquad (10)
\end{aligned}$$

where $I(\cdot)$ is the "indicator function," defined for this case as follows: if $\max\{a, b\} \le x - d$, then $I(\max\{a, b\} \le x - d) = 1$; otherwise $I(\max\{a, b\} \le x - d) = 0$.

The above example illustrates that even for a simplified model (i.e., ignoring the communication overhead) of the considered task graph, the derivation of the exact execution time distribution is non-trivial. In particular, the production of a string of equalities is required (based on basic principles of probability theory) in order to derive the final expression given in Equation (10). Thus, although the final expression can be evaluated in this case, it was not straightforward to derive.

Practical task graphs will be more complicated than that of Fig. 1, and dependencies imposed by machine availability could further complicate the relationships among the start and finish times of subtasks. Although exact derivation for general task graphs may be possible, there is no clear systematic approach for automating such a derivation. A goal of this paper is to devise an approach for systematically determining (or suitably approximating) the execution time distribution of a task graph. In the remainder of this section, such a technique is proposed for estimating the overall execution time distribution based on the Kleinrock independence approximation. This approximation enables the usage of the simple formulas for summation and maximum of random variables (i.e., Equations (7) and (8)).

4.2 Independence assumption

As demonstrated in the previous subsection, subtasks corresponding to nodes in the task graph that have a common ancestor can have correlated random variables associated with the start and finish times. In such cases, deriving an expression for the exact distribution of the overall execution time distribution can become unrealistic for general task graphs. However, conditions exist for which the associated random variables can nevertheless be treated as uncorrelated despite this type of interaction. The Kleinrock independence approximation [4] is a well-known condition for describing this situation, and is used here as the basis for assuming independence among random variables that may technically be correlated.

To understand the original rationale of the Kleinrock independence approximation, consider a data network in which there are many interacting transmission queues. A traffic stream departing from one queue enters one or more other queues, perhaps after merging with portions of other traffic streams departing from yet other queues. Although packet inter-arrival times of data packets can become dependent (i.e., correlated) after a traffic stream leaves the first queue, the Kleinrock independence approximation concludes that the merging of many different packet streams on a transmission line has an effect similar to restoring the independence of inter-arrival times and packet lengths [11].

Similarly, in a task graph, a subtask may take input from many other subtasks, and multiple subtasks may be assigned to the same machine, where they must wait for the machine to become available before execution. The effect is analogous to that of merging many traffic streams in a data network. Thus, it is assumed that the input of data from many other subtasks has the effect of restoring independence among the start and finish times of subtasks that have a common ancestor in the task graph. This approximation of independence is the basis for justifying the use of Equations (7) and (8) to compute the distribution of start and finish times of subtasks. The degree to which this independence approximation is violated (or not) will influence the resulting accuracy of the calculated distribution. The estimation error is analyzed mathematically in Subsection 4.4, and is investigated further in Section 5 through numerical simulation studies.

4.3 Proposed approach for calculating the execution time distribution

Subtask start and finish time distributions are calculated in an order determined by the data dependency structure of the task graph and machine availability, which depends on the given matching of subtasks to machines and local scheduling for each machine. The key to calculating the start and finish time distributions is to partition the subtasks into layers, which requires the definition of an immediate predecessor. Subtask S_j is an immediate predecessor of subtask S_g if either of the following conditions is satisfied: (1) S_g requires data from S_j (i.e., there is an arc in the task graph from S_j to S_g) or (2) S_j and S_g are assigned to the same machine and S_j is scheduled to execute immediately before S_g. Those subtasks without an immediate predecessor are put into layer 1. A subtask is put into layer $k+1$ if the highest layer number of its immediate predecessors is k. Based on this definition (which implies a constructive procedure), there is no data dependence among subtasks of the same layer. The main difference between this layering approach and those found in the literature (e.g., Cluster-M model in [12]) is the resource dependence determined by scheduling is also considered, i.e., condition (2) above, is also considered.

Subtask start and finish time distributions are first computed for subtasks in layer 1. Distributions for the time each output data item is available on its target machine are then calculated. These steps are repeated for subtasks in layer 2, and so on. In this way, when the start time distribution of each subtask is computed, the distributions for available times of the designated machine and all input data items are known.

For each subtask S_j considered in the "layered" order, the following four calculations are performed.

1. Compute distribution function for subtask start time:

$$F_{T_j^S}(\cdot) = F_{T_j^M}(\cdot) \prod_{v=0}^{n_j^I-1} F_{T_{j,v}^I}(\cdot). \qquad (11)$$

2. Compute density function for subtask completion time:
$$f_{T_j^F}(\cdot) = f_{T_j^S}(\cdot) * f_{\tau_j^E}(\cdot). \tag{12}$$

3. Let the next subtask to be executed on machine $M_{\mathcal{M}(j)}$ be S_g. Define the density function for machine available time for S_g:
$$f_{T_g^M}(\cdot) = f_{T_j^F}(\cdot). \tag{13}$$

4. For each u from 0 to $n_j^O - 1$, let output data item $D_{j,u}^O$ be the input data item h of subtask S_g (i.e., $D_{g,h}^I$). Compute the distributions for queuing time, arrival time for $D_{j,u}^O$, and the available time for $D_{g,h}^I$:
$$f_{\tau_{j,u}^Q}(\cdot) = \begin{cases} \delta(\cdot) & \text{if } u = 0 \text{ or } \mathcal{M}(j) = \mathcal{M}(g), \\ f_{\tau_{j,u-1}^Q}(\cdot) * f_{\tau_{j,u-1}^N}(\cdot) & \text{otherwise.} \end{cases} \tag{14}$$
$$f_{T_{j,u}^A}(\cdot) = f_{T_{g,h}^I}(\cdot) = f_{T_j^F}(\cdot) * f_{\tau_{j,u}^Q}(\cdot) * f_{\tau_{j,u}^N}(\cdot). \tag{15}$$
($\delta(\cdot)$ represents the Dirac delta function [13].)

After completing the above four steps for each subtask (ordered according to the layer numbers), the overall distribution of the task execution time is computed. Let $\underline{\phi}$ be the number of terminal subtasks, and let $\{S_{j_0}, S_{j_1}, \ldots, S_{j_{\phi-1}}\}$ be the set of terminal subtasks. Then the probability distribution function of the completion time of the entire task is:
$$F_{T^C}(\cdot) = \prod_{\psi=0}^{\phi-1} F_{T_{\psi}^F}(\cdot). \tag{16}$$

4.4 Error analysis

In this subsection, the difference between the distribution computed by the proposed approach (Subsection 4.3) and the exact distribution is analyzed for a special class of task graphs for which the Kleinrock independence assumption is (apparently) violated. An analytical expression for the difference of the means of these distributions is derived. Based on this expression, it is shown that the proposed approach always overestimates the actual mean of the execution time. A bound for the difference of the means is also derived that depends on the parameters of the assumed subtask distributions involved. Finally, conditions are determined for which the distribution of the proposed approach equals the exact distribution for this class of task graph.

In a task graph, a fork-join structure between two subtasks S_f and S_j contains a set of two or more disjoint paths from S_f to S_j ("f" is for fork and "j" is for join). Let $\underline{W}$ denote the set of subtasks in a fork-join structure, excluding S_f and S_j. For a given subtask-to-machine matching and a given scheduling for each machine, a fork-join structure is called an isolated fork-join structure if every immediate predecessor of a subtask in W belongs to $W \cup \{S_\mathrm{f}\}$. Examples of isolated fork-join structures are shown in Fig. 2, in which each subtask is assumed to be assigned to a distinct machine. The conditions of the Kleinrock independence approximation are clearly violated in an isolated fork-join structure. This is because the data that flows from S_f to S_j (e.g., S_0 to S_3 in Fig. 2(a)) does not merge with other data originating from subtasks outside that fork-join structure. Therefore, the effect of restoring independence of arrival times of input data items for S_j by merging data flows from different subtasks/machines does not occur. Perhaps surprisingly, it is shown that even for this "worst case" structure (i.e., the isolated fork-join structure), the proposed approach still can provide reasonably accurate estimate of the exact distribution. Under certain conditions, it is shown that the distribution from the proposed approach actually coincides with the exact distribution.

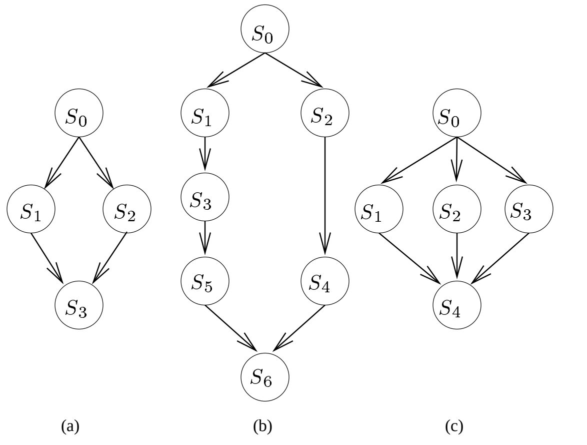


Figure 2: Examples of isolated fork-join structures. Each subtask is assumed to be assigned to a distinct machine for each example.

Isolated fork-join structures are characterized by the number of disjoint paths that connect S_f to S_j. In Fig. 2(b), note that the chains $S_1 \to S_3 \to S_5$ and $S_2 \to S_4$ can each be reduced to a single subtask, resulting in a structure identical to that in Fig. 2(a).

Therefore, without loss of generality, only structures in which there is exactly one subtask on each path between S_f and S_j (such as Figs. 2(a) and (c)) are studied here.

In general, for multiple subtasks matched to the same machine, a fork-join structure may not be an isolated fork-join structure. For example, consider Fig. 3 in which the subtasks are matched to machines as indicated by the ovals. Although S_0 through S_3 form a fork-join structure, the scheduling of S_1 and S_5 on M_1 determines whether it is an isolated fork-join structure. In particular, if S_5 is scheduled before S_1, then it is not an isolated fork-join structure, because S_5 is an immediate predecessor of S_1 and $S_5 \notin W \cup \{S_0\} = \{S_0, S_1, S_2\}$.

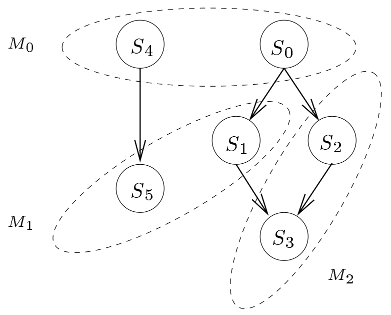


Figure 3: Example task graph in which its characterization as an isolated fork-join structure depends on the scheduling.

Consider the isolated fork-join structure shown in Fig. 2(a). For clarity of presentation and without loss of generality, network communication times are ignored. Let A, B, and C denote the execution time distribution of S_0, S_1, and S_2 on their designated machines, respectively. The start time of S_3 can be derived as:

$$\begin{aligned} T_0^F &= T_1^S = T_2^S = A, \\ T_1^F &= A + B, \\ T_2^F &= A + C, \\ T_3^S &= \max\{T_1^F, T_2^F\} = \max\{A+B,\ A+C\}. \end{aligned}$$

Note that the distributions for T_1^F and T_2^F are obtained by convolving the appropriate density functions (i.e., $f_{T_1^F}(\cdot) = f_A(\cdot) * f_B(\cdot)$ and $f_{T_2^F}(\cdot) = f_A(\cdot) * f_C(\cdot)$). The proposed approach estimates the distribution of T_3^S as $F_{T_1^F}(\cdot)F_{T_2^F}(\cdot)$. For notational convenience, let X denote the random variable with distribution function $F_{T_1^F}(\cdot)F_{T_2^F}(\cdot)$, and let $X^* = T_3^S$, i.e., X and X^* represent the estimated and exact value of T_3^S, respectively.

In [6], the exact distribution for X^* is derived and the difference between X and X^* is analyzed mathematically. Due to space limitations, only the results of this analysis is included here (refer to [6] for the detailed derivation). To state the results, some notation is needed for quantifying the ranges of the random variables A, B, and C. Because these random represent the execution times of S_0, S_1, and S_2, it is assumed that they are finite and have finite range. Thus, there exists constants $0 \le a_1 \le a_2$, $0 \le b_1 \le b_2$, and $0 \le c_1 \le c_2$, such that $\Pr[A < a_1] = \Pr[A > a_2] = 0$, $\Pr[B < b_1] = \Pr[B > b_2] = 0$, $\Pr[C < c_1] = \Pr[C > c_2] = 0$.

The following is a summary of the results proven in [6].

1. The proposed approach always overestimates the mean, and the estimation error for the mean is upper-bounded by the length of the range of A:
$$0 \le EX - EX^* \le a_2 - a_1, \qquad (17)$$
where EX and EX^* denote the expected values (i.e., means) of the approximate and exact distributions, respectively.

2. The proposed approach yields the exact distribution if the length of the range of A is shorter than the length of combined range of B and C. Mathematically, this condition on the length of the range of A is stated as:
$$a_2 - a_1 < \max\{b_2, c_2\} - \min\{b_1, c_1\}.$$
Thus, if the above inequality is satisfied, then the distribution produced by the proposed approach equals the exact distribution (i.e., $X = X^*$).

These two results share a common theme – the smaller the range of possible values for A, the smaller the estimation error. The second result is most interesting, and perhaps most surprising. It states that if the range of values for A is sufficiently small (with respect to the corresponding ranges for B and C), then the estimated distribution actually equals the exact distribution (i.e., there is no estimation error). An important key in deriving these results is the finite range assumption for the random variables A, B, and C.

5 Numerical Studies

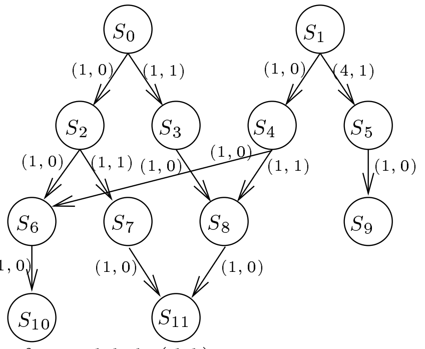
5.1 Overview

In this section, the accuracy of the execution time distributions determined from the proposed approach

(Subsection 4.3) is evaluated through numerical studies. Due to the size and complexity of the task graphs considered, derivation of the exact distributions (as done for the simple task graphs considered in the previous section) is not feasible. Thus, detailed simulations of the task graphs are performed as a means of determining the actual distributions. The results show that the proposed approach predicts simulated task graph execution time distribution with high accuracy. Task graph structures for which the independence among the random variables is apparently violated to a substantial degree are also studied. Even for these task graph structures, the distributions computed by the proposed approach match those from the simulation reasonably well. On a Sun SPARCstation 5, the time required to compute distributions based on the proposed approach is about 10 times less than that based on simulating the task graph over 4000 instances.

For this study, each subtask execution time distribution (associated with the random variable τ_j^E for subtask S_j) is assumed to be either a uniform or a normal distribution. (This is for convenience only; any distribution could be used in this framework.) Each network transmission time distribution (associated with the random variable $\tau_{j,u}^N$) is assumed to be a normal distribution with a fixed standard deviation, and the mean defined as the sum of a fixed overhead and a term proportional to the size of data item to be transfered. (Each normal distribution $N(\mu, \sigma)$ with mean μ and standard deviation σ is discretized in the range of $(\max\{\mu - 4\sigma, 0\}, \mu + 4\sigma)$.) The parameters of these distributions are included in an input file. Also included in this file is the subtask-to-machine matching and execution schedule of each individual machine. A program was written in C to parse this input file and output a Matlab program for simulating the execution time of task graph (see [6] for details).

For each instance of the simulation, the execution time of a subtask is determined by generating a random number according to its assumed distribution, and network transmission times are determined similarly. The overall execution time of the task graph is calculated according to data dependency structure of the task graph and execution schedule of the individual machines. Subtask start and finish times are first computed for subtasks in layer 1. The time each output data item of layer 1 subtasks is available on its target machine are then calculated. These steps are repeated for subtasks in layer 2, and so on. In this way, when the start time of each subtask is computed, the available times of the designated machine and all



Key for arc labels (d, k):

d: amount of data to be transfered

k: order index of output data item

Figure 4: Example task graph.

input data items are known. The overall execution time of the task graph is the maximum of the finish times over all terminal subtasks. For each task graph studied, 4000 simulation instances were performed to collect the sample distribution of the overall execution time of the task graph.

5.2 Comparison of estimated distribution and simulated sample distribution

The first task graph studied is shown in Fig. 4. There are 12 subtasks in the graph, labeled from S_0 to S_{11}. Each arc is labeled with an ordered paired (d, k), where d is the amount of data to be transfered and k is the output data index for the source subtask. (Recall if a subtask has multiple output data items to be transmitted to other machine(s), they are transmitted in ascending index order.) It is assumed that there are four machines in the HC system, labeled from M_0 to M_3. The assumed subtask-to-machine matching, execution schedule of each machine, and execution time distribution of each subtask on its designated machine are defined in Table 1. Three subtasks are assigned to each machine. Each row corresponds to parameters for a subtask; the first column is the label of the subtask, the second column is the label of its designated machine, the third column is the order of execution on that machine, and the forth column parameterizes its execution time distribution.

The network transmission time is modeled by a normal distribution with a standard deviation of 3. Two separate models for the mean of the network network transmission time were used. In the first, the mean is equal to $10 + (5 \times d)$, and in the second, the mean is equal to $20 + (30 \times d)$. These two models represent different computation to communication ratios. For

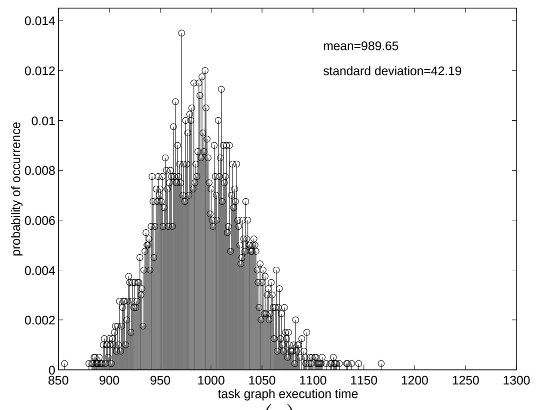
subtask	machine	schedule	exe. time distr.
0	0	0	$U(125, 146)$
1	1	0	$N(203, 10.3)$
2	2	0	$U(244, 325)$
3	0	1	$N(301, 24.3)$
4	3	0	$U(203, 248)$
5	1	1	$N(350, 27.3)$
6	2	1	$U(271, 324)$
7	0	2	$N(283, 26.1)$
8	3	1	$N(201, 34.1)$
9	1	2	$U(278, 321)$
10	2	2	$U(130, 183)$
11	3	2	$N(231, 29.4)$

Table 1: Assumed matching, scheduling, and execution time distribution for subtasks of Fig. 4. $U(a, b)$: uniform distribution between a and b. $N(\mu, \sigma)$: normal distribution with mean μ and standard deviation σ.

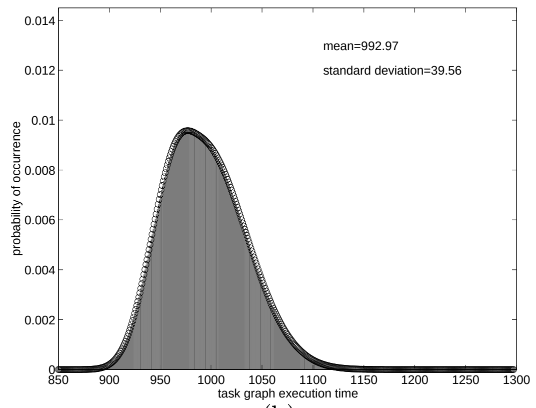
each model, the distribution for the execution time of the entire task graph are obtained through simulation, and the corresponding distribution is also computed by the proposed approach.

The results of these studies are shown in Figs. 5 and 6. For each study, the difference between the simulated mean execution time and the estimated mean execution time is less than 0.4%, and the difference for standard deviation is less than 6.3%. From this, it is concluded that the proposed approach accurately estimates the distribution of execution time for the task graph considered.

Simulation was also performed for the task graph shown in Fig. 7. The task graph is nearly the same structure as that of Fig. 4. The only difference is the arc $S_4 \rightarrow S_6$ (in Fig. 4) has been changed to become arc $S_6 \rightarrow S_{11}$ (in Fig. 7). However, this small change in the structure of the task graph creates an isolated fork-join structure (S_2, S_6, S_7, S_{11}). The subtask-to-machine matching, subtask execution time distribution, and execution schedules of individual machines remain unchanged. Simulations are performed using the same two network communication models as used for Fig. 4. The resulting distributions are compared with estimates in Figs. 8 and 9. Still, the proposed approach provides a good approximation for the task graph execution time distribution. The error between estimated and simulated results is less than 1.08% for the mean execution time and less than 10.4% for the standard deviation. This example demonstrates the robustness of the proposed approach when some assumptions are violated. (Recall that even for isolated fork-join structure, the estimation error for the mean execution time is upper-bounded by the width of the density of execution time of the fork node, and conditions exists under which exact results could be obtained.)



(a)

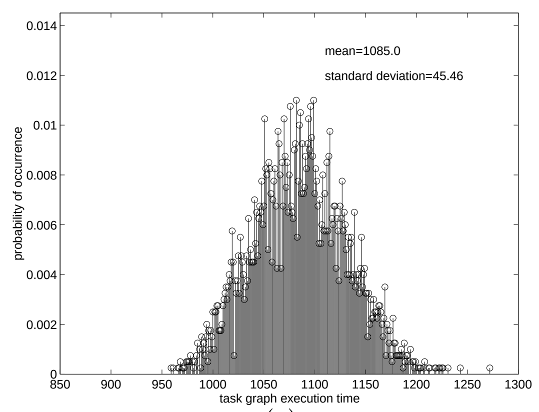


(b)

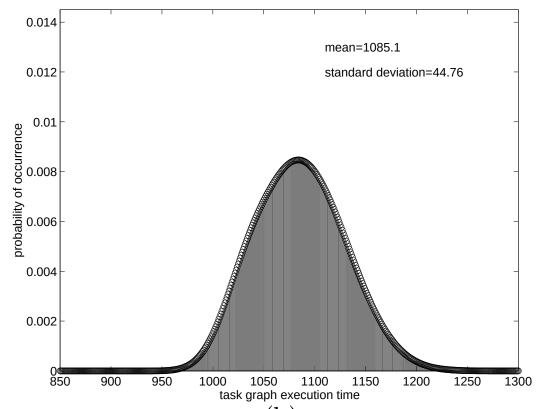
Figure 5: Distributions of execution time for task graph in Fig. 4 where the mean of network transmission time is equal to $10 + (5 \times d)$. (a) Sample distribution. (b) Estimated distribution.

6 Summary

In this paper, a methodology for estimating the execution time distribution for a task graph executed in an HC system is introduced. Individual subtask execution time distributions are assumed to be known or estimated using analytical or empirical techniques. A probabilistic model for the data transmission time is developed. Random variables are used to represent

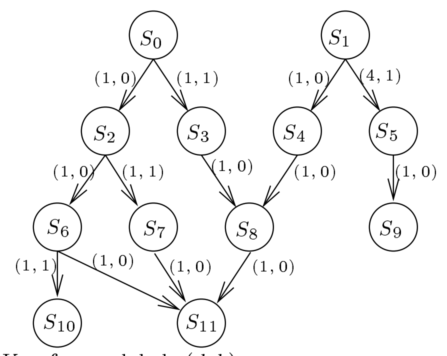


(a)



(b)

Figure 6: Distributions of execution time for task graph in Fig. 4 where the mean of network transmission time is equal to $20 + (30 \times d)$. (a) Sample distribution. (b) Estimated distribution.

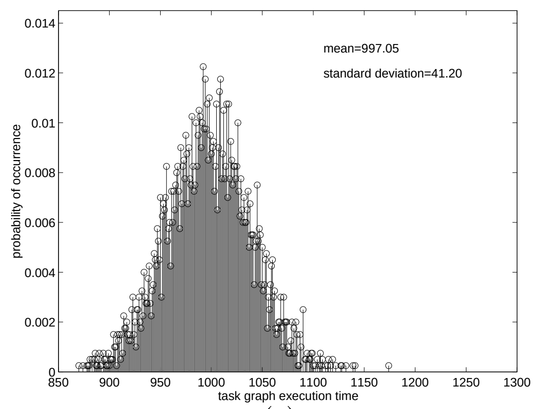


Key for arc labels (d, k):

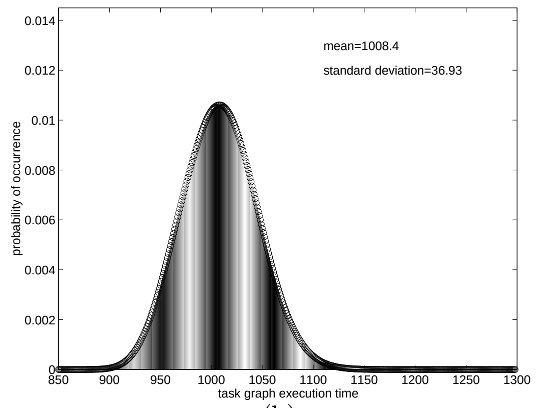
d: amount of data to be transfered

k: order index of output data item

Figure 7: Example task graph in Fig. 4 with arc $S_4 \rightarrow S_6$ moved to $S_6 \rightarrow S_{11}$.



(a)

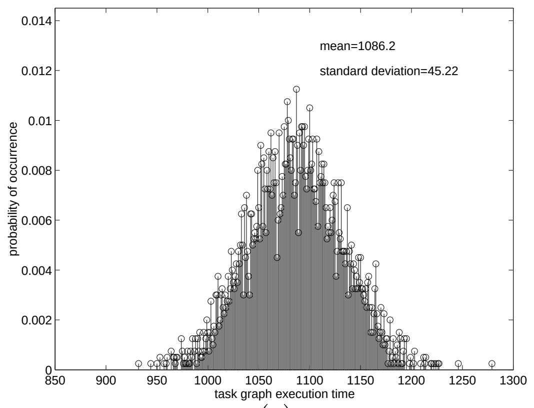


(b)

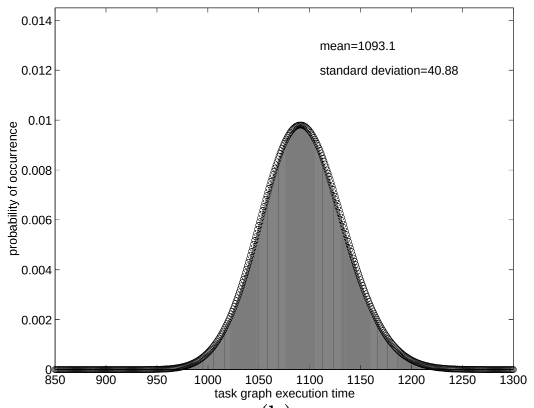
Figure 8: Distributions of execution time for task graph in Fig. 7 where the mean of network transmission time is equal to $10 + (5 \times d)$. (a) Sample distribution. (b) Estimated distribution.

the duration of subtask executions and network transmissions, as well as the start and finish times. Data dependency and machine availability are used to derive the relationships among these random variables.

It is demonstrated that deriving the exact execution time distribution for general task graphs is extremely difficult. The Kleinrock independence approximation is applied to make the computation of associated probability distributions tractable. Graph structures for which the independence assumption is violated are identified, and an upper bound of the estimation error for the mean execution time is derived for this case. Simulations were performed for various task graphs. The simulation results indicate that the proposed approach provides accurate estimates for execution time distribution.



(a)



(b)

Figure 9: Distributions of execution time for task graph in Fig. 7 where the mean of network transmission time is equal to $20 + (30 \times d)$. (a) Sample distribution. (b) Estimated distribution.

References

[1] H. J. Siegel, J. K. Antonio, R. C. Metzger, M. Tan, and Y. A. Li, "Heterogeneous Computing," in *Handbook of Parallel and Distributed Computing*, A. Y. Zomaya, ed., pp. 725–761, McGraw-Hill, New York, NY, 1996, (also Purdue EE School technical report TR-EE 94-37).

[2] D. W. Watson, J. K. Antonio, H. J. Siegel, and M. J. Atallah, "Static Program Decomposition Among Machines in an SIMD/SPMD Heterogeneous Environment with Non-Constant Mode Switching Costs," in *Proceedings of the Heterogeneous Computing Workshop (HCW '94)*, pp. 58–65, Apr. 1994.

[3] R. F. Freund, "Optimal Selection Theory for Superconcurrency," in *Proceedings of Supercomputing '89*, pp. 47–50, Nov. 1989.

[4] L. Kleinrock, *Communication Nets: Stochastic Message Flow and Delay*, McGraw-Hill, New York, NY, 1964.

[5] Y. A. Li, J. K. Antonio, H. J. Siegel, M. Tan, D. W. Watson, "Estimating the Distribution of Execution Times for SIMD/SPMD Mixed-Mode Programs," in *Proceedings of the Heterogeneous Computing Workshop (HCW '95)*, pp. 35–46, Apr. 1995.

[6] Y. A. Li, *A Probabilistic Framework for Estimation of Execution Time in Heterogeneous Computing Systems*, Ph.D. Dissertation, School of Electrical and Computer Engineering, Purdue University, Aug. 1996.

[7] S. Chen, M. M. Eshaghian, A. Khokhar, and M. E. Shaaban, "A Selection Theory and Methodology for Heterogeneous Supercomputing," in *Proceedings of the Workshop on Heterogeneous Processing*, pp. 15–22, Apr. 1993.

[8] M. Tan, J. K. Antonio, H. J. Siegel, and Y. A. Li, "Scheduling and Data Relocation for Sequentially Executed Subtasks in a Heterogeneous Computing System," in *Proceedings of the Heterogeneous Computing Workshop (HCW '95)*, pp. 109–120, Apr. 1995.

[9] A. B. Tayyab and J. G. Kuhl, "Stochastic Performance Models of Parallel Task Systems," in *Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pp. 284–285, May 1994.

[10] A. M. Mood, F. A. Graybill, and D. C. Boes, *Introduction to the Theory of Statistics*, McGraw-Hill, New York, NY, 1974.

[11] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[12] M. M. Eshaghian and R. F. Freund, "Cluster-M Paradigms for High-Order Heterogeneous Procedural Specification Computing," in *Proceedings of the Workshop on Heterogeneous Processing*, pp. 47–49, May 1992.

[13] J. B. Thomas, *An Introduction to Applied Probability and Random Processes*, Robert E. Krieger Publishing Company, Huntington, NY, 1981.

Biographies

Yan A. Li received his B.E. degree from Tsinghua University, Beijing, China, in 1991, and his MSEE and Ph.D. degrees, both from Purdue University, West Lafayette, Indiana, U.S.A., in 1993 and 1996, respectively. He is currently a Senior System Architect at Intel Corporation. He is a member of IEEE and Eta Kappa Nu. His major research interest includes parallel processing, high-performance heterogeneous computing, computer architecture, and computer systems simulation.

John K. Antonio received the B.S., M.S., and Ph.D. degrees in electrical engineering from Texas A&M University, College Station, TX. He currently holds the position of Associate Professor of Computer Science within the College of Engineering at Texas Tech University. Before joining Texas Tech, he was with the School of Electrical and Computer Engineering at Purdue University. During the summers of 1991-94 he participated in a faculty research program at Rome Laboratory, Rome, NY, where he conducted research in the area of high performance computing. His current research interests include heterogeneous systems, configuration techniques for embedded parallel systems, and computational aspects of control and optimization. He has co-authored over 50 publications in these and related areas. For the past four years, he has organized the Industrial Track and Commercial Exhibits portions of the International Parallel Processing Symposium. He is a member of the IEEE computer society and is also a member of the Tau Beta Pi, Eta Kappa Nu, and Phi Kappa Phi honorary societies. Organizations that have supported his research include the Air Force Office of Scientific Research, National Science Foundation, Naval Research Laboratory, Orincon, Inc., and Rome Laboratory.