Robust Sequential Resource Allocation in Heterogeneous Distributed Systems with Random Compute Node Failures

Vladimir Shestak*, Edwin K. P. Chong†§, Anthony A. Maciejewski†, and Howard Jay Siegel†‡

*InfoPrint Solutions Company
6300 Diagonal Highway Boulder, CO 80301
Email: vshestak@us.ibm.com

†Department of Electrical and Computer Engineering
‡Department of Computer Science
§Department of Mathematics
Colorado State University
Fort Collins, CO 80523–1373
Email: {echong, aam, hj}@colostate.edu

***Abstract*—The problem of finding efficient workload distribution techniques is becoming increasingly important today for heterogeneous distributed systems where the availability of compute nodes may change spontaneously over time. Therefore, the resource-allocation policy must be designed to be robust with respect to absence and re-emergence of compute nodes so that the performance of the system is maximized. Such a policy is developed in this work, and its performance is evaluated on a model of a dedicated system composed of a limited set of heterogeneous Web servers. Assuming that each HTML request results in a "reward" if completed before its hard deadline, the goal is to maximize a cumulative reward obtained in the system. A failure rate for each server is set relatively high to simulate its operation under harsh conditions. The results demonstrate that the proposed approach based on the concepts of the Derman–Lieberman–Ross theorem outperforms other policies compared in our experiments for inconsistent, processor-consistent, and task-processor-consistent types of heterogeneity.**

This research was supported in part by the InfoPrint Solutions Company, the National Science Foundation under Grants No: CNS-0615170 and No: ECCS-0700559, and the Colorado State University George T. Abell Endowment.

978-1-4244-3750-4/09/$25.00 ©2009 IEEE

I. Introduction

Distributed computing systems are widely used today for execution of large workloads composed of independent tasks that are divided among multiple heterogeneous compute nodes. The assignment of tasks to compute nodes is referred to in the literature as resource allocation or mapping. Effective mapping policies must account for multiple factors, e.g., the set of available compute nodes in the system, characteristics of these nodes, and links between them. Multiple scenarios can be identified where there is an uncertainty in the availability of functional compute nodes over time. Due to this uncertainty, any compute node may randomly fluctuate between the "failure" ("down") and "working" ("up") states. Examples include Internet computing systems [8], systems operating under harsh conditions, systems experiencing 'software aging" [18], [29], etc.

The fault tolerant aspect of distributed computing systems has been extensively explored in the last few years [10]. Available literature on distributed

computing in such uncertain environments primarily considers reactive techniques, where a node failure is addressed only after its occurrence. Checkpoint-resume or terminate-restart mechanisms are often used to recover unprocessed tasks at the failed nodes [28], [45].

Clearly, the uncertainty in working compute nodes is expected to degrade the performance of any resource-allocation policy that does not account for node failure and recovery. In this study, we model a dedicated system composed of a limited set of heterogeneous Web servers. The availability of servers is described with exponential distributions with failure rates that are relatively high to simulate harsh operating conditions. Each HTTP request made by a user of the website is assumed to have a hard deadline, limiting the total time available to process it, and an associated reward. The goal is to design a resource-allocation policy that maximizes the expected cumulative reward received from the requests that finish before their deadlines, with the uncertainty of compute node failures.

The concept of robustness of a resource allocation was introduced in [7], and later efforts applied the concept to resource management [5], [6], [31], [38], [39], [42]–[44]. The mathematical model developed in this research does incorporate many of the same basic principles addressed in our earlier work on robustness. In analyzing systems where compute resources fail in a random fashion, the operational periods of the compute resources are uncertain. This uncertainty can impact the system by causing tasks to fail during execution and finally miss their deadlines. Often in practice, if the number of failed tasks reaches a certain level a service provider becomes subject to profit losses and penalties. In this environment, the system is said to be robust against these uncertainties if the profit continues to be above a user-specified level of profit despite the failures of compute resources. Therefore, a simplistic quantification for robustness can be defined as the difference between the actual (or expected) profit and the lowest level that justifies the operation of the system.

The major contribution of this paper is the design of a resource-allocation policy for the above environment based on the concepts of the Derman-Lieberman-Ross theorem [13]. Our simulation results demonstrate that the proposed solution outperforms other policies considered, and its superior performance is sustainable in environments with different types of heterogeneity among tasks and compute nodes.

II. DLR Policy

Derman, Lieberman, and Ross introduced the following optimal policy for the sequential stochastic assignment problem [13]. Suppose there are N workers available to perform N jobs. The N jobs arrive in sequential order, i.e., job 1 arrives first, followed by job 2, etc. Associated with the jth ($j = 1, 2, ..., N$) job is a real-valued random variable X_j representing its worth. It will be assumed that the X_j are independent and identically distributed random variables with a cumulative density function (cdf) $G_X(z)$ with a finite mean value. If a "perfect" worker is assigned to jth job, a reward X_j is obtained. However, none of the N workers are perfect, and whenever the ith worker is assigned to the jth job, the reward is given by $q_i X_j$, where $0 \leq q_i \leq 1, i = 1, 2, ..., N$[1]represents the probability of worker i successfully completing any job. Each worker is assigned to one and only one job. The goal is then to assign the N workers to the N jobs so as to maximize the total expected reward. Let a policy be any rule for sequentially assigning workers to jobs. In particular, if $m(i)$ is defined to be the job to which ith worker is assigned, then the total expected reward is given by

$$E\left[\sum_{i=1}^{N} q_i X_{m(i)}\right]. \tag{1}$$

The desired policy is the one that maximizes this expected cumulative reward. The following Derman–Lieberman–Ross (DLR) theorem embodies this intuition providing the *optimal* resource-allocation policy using the available stochastic information (see [13] for further details).

DLR Theorem. *For each $N \geq 1$, there exist numbers $-\infty = a_{0,N} \leq a_{1,N} \leq a_{2,N} \leq \ldots \leq a_{N,N} = +\infty$, such that whenever there are N assignments to make and probabilities $q_1 \leq q_2 \leq ... \leq q_N$ then the optimal choice in the first assignment is*

[1]The constraint, $0 \leq q_i \leq 1$, is given for clarity of application, and none of the ensuing results is dependent upon it.

to use the worker i such that X_1 is contained in the interval $(a_{i-1,N}, a_{i,N}]$. The $a_{i,N}$ depend on G_X but are independent of the q_i values and calculated recursively for N as follows:

$$a_{i,N} = \int_{a_{i-1,N-1}}^{a_{i,N-1}} z\, dG_X(z) + a_{i-1,N-1} G_X(a_{i-1,N-1}) + a_{i,N-1}[1 - G_X(a_{i,N-1})],$$

with the convention that $a_{0,N} = -\infty$, $a_{N,N} = +\infty$, $-\infty \times 0 = 0$, and $\infty \times 0 = 0$.

Suppose that $N = 5$, and $G_X(z)$ is as illustrated in Fig. 1. When the DLR policy is applied, the $a_{i,5}$ values are calculated recursively starting from $N = 1$. These values divide the domain of all possible job worths into five intervals. Once the first job has arrived, a resource-management system identifies which of these five intervals this job falls in based on the job's worth X_1. Assume that this happened to be the third interval. Then, the resource-allocation system assigns the arrived job to the third worker in the sorted list of probability values $q_1 \le q_2 \le q_3 \le q_4 \le q_5$, as shown in Fig. 1. When the next job arrives, N is decremented, and the same procedure is repeated by recalculating intervals.

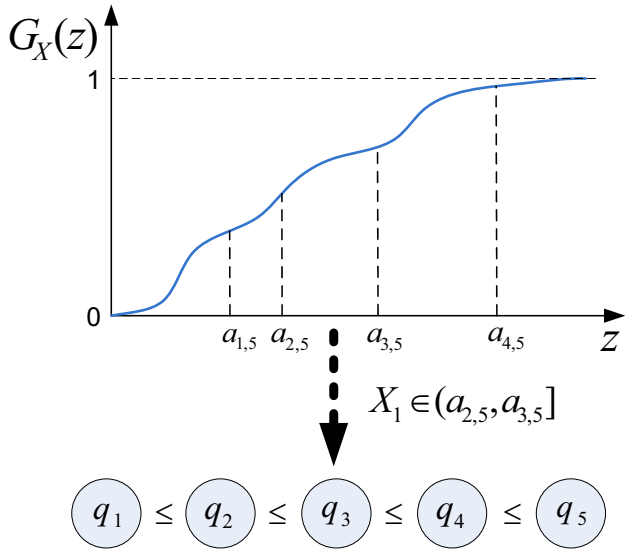


Fig. 1. The distribution described with cdf $G_X(z)$ is divided into five intervals. According to the DLR policy, as the worth X_1 of the arrived job belongs to the third interval, this job will be assigned to the third worker in the list of workers ranked based on probability values q_i.

III. Simplified Resource Allocation

First, consider the following problem statement for a *simplified* resource allocation problem in a distributed system. Suppose there is a batch of N tasks. Each task i, $1 \le i \le N$, is characterized by: (a) the number of instructions it contains, denoted n_i, and (b) reward r_i for completing this task. Suppose that N processors become available sequentially in time. Whenever a new processor becomes available, we need to assign one of the remaining (unassigned) tasks to this processor. Suppose that the jth processor to arrive is characterized by a random variable T_j that specifies the time that processor takes to execute each instruction (making a simplifying assumption that all instructions in a task take the same time to execute on the given processor). Moreover, the processor might randomly fail some time during the execution of a task, modeled as follows. The probability of failure is defined by its failure rate λ_j, so that the probability of failure during the execution of a task is given by λ_j multiplied by the execution time (we discuss this failure model further below).

Once task i is assigned to a processor, its execution begins, and reward r_i will be obtained if the task is completed successfully. If a processor fails while executing the task, no reward is earned and the task is removed from the batch, i.e., the task will never be executed again with this simplifying model. Let $\mathbb{P}[i, m(i)]$ denote the probability that task i is successfully completed when assigned to processor $m(i)$. Then, the goal is to maximize the total expected reward accumulated through N sequential assignments:

$$\text{maximize } E\left[\sum_{i=1}^{N} r_i \mathbb{P}[i, m(i)]\right]. \qquad (2)$$

Using the failure model above, $\mathbb{P}[i, m(i)] = 1 - \lambda_{m(i)} n_i T_{m(i)}$. Substituting this into Eq. 2 and simplifying, the optimization problem becomes

$$\text{maximize } E\left[\sum_{i=1}^{N} \left([r_i n_i] \times \left[-\lambda_{m(i)} T_{m(i)}\right]\right)\right]. \qquad (3)$$

The set of workers and arriving jobs in the sequential stochastic assignment problem considered in Section II correspond to tasks in the batch and sequentially arriving processors considered here, respectively. Moreover, the performance goal given

by Eq. 3 is structured identically to Eq. 1. Therefore, if we assume that the random variables $X_j = \lambda_j T_j$, $j = 1, \ldots, N$, have known cdf $G_X(x)$ with a finite mean, then the policy derived in the DLR theorem can be applied.

The failure model above can be viewed as an approximation to the standard model where the time it takes for failure is exponentially distributed with parameter λ_j. In this case, the probability that processor $m(i)$ fails while executing task i is given by $e^{-\lambda_{m(i)} n_i T_{m(i)}}$. Using the Taylor series expansion for the exponential function,

$$e^{-\lambda_{m(i)} n_i T_{m(i)}} = 1 - \lambda_{m(i)} n_i T_{m(i)} + \frac{(-\lambda_{m(i)} n_i T_{m(i)})^2}{2!} + \cdots .$$

Ignoring second order and higher terms, we arrive at the failure model used in the derivation above.

IV. Tasks with Exponentially Distributed Execution Times

In the simplified resource-allocation case described in the previous section, observe that (1) if tasks are ordered according to their $r_i n_i$ values this order does not depend on processors j, and (2) the objective function for the resource-allocation problem is based on the product $[r_i n_i] \times \left[-\lambda_{m(i)} T_{m(i)}\right]$. Fig. 2 illustrates this relationship between tasks and processors in a matrix form as the Cartesian product of two sequences: $r_i n_i,\ 1 \leq i \leq N$ and $-\lambda_j T_j,\ 1 \leq j \leq N$. Note that the matrix is ordered in both row and column directions. The DLR theorem provides the optimal assignment policy for such matrices given that processor's characteristic $-\lambda_j T_j$ is a random variable with a known distribution.

In the literature on distributed systems, Estimated Time to Compute (ETC) matrices often are assumed to be given [21], [25], [46]. Each entry in an ETC matrix is an estimate of time, t_{ij}, to compute task i on processor j. The ETC values can be based on user supplied information, experimental data, or task profiling and analytical benchmarking [4], [17], [19], [22], [24], [30], [48]. Determination of ETC values is a separate research problem; the assumption of such ETC information is a common practice in resource allocation research [9], [16], [19], [23], [24], [27], [41], [47].

Clearly, for the simplified resource-allocation problem presented in Section III, the ETC matrix would be based on a Cartesian product between n_i and T_j sequences because $t_{ij} = n_i T_j$. However, in practice, it is difficult or even impossible to represent t_{ij} with such a product. This is mainly because the exact number of executed instructions is usually unknown in advance due to conditional branching, uncertain input data, etc. Furthermore, instruction execution rates of participating processors may depend on the mix of the types of executed instructions [20]. Therefore, an ETC matrix based on empirical measurements typically provides a more accurate means of capturing the relationship among tasks and processors in distributed systems.

The rest of this paper will focus on tasks with execution times exponentially distributed for each task-processor pair. This assumption implies that an ETC matrix contains expected execution times that are assumed to be known (expected values, i.e., means, are sufficient to fully describe exponential distributions [14]). Eq. 3 can be restructured as:

$$\text{maximize } E\left[\sum_{i=1}^{N} \left(r_i \times \left[-\lambda_{m(i)} t_{im(i)}\right]\right)\right]. \quad (4)$$

In this study, we distinguish between three major classes of heterogeneity in distributed systems: task-processor-consistent, processor-consistent, and inconsistent. The first class of heterogeneity describes systems where two conditions are satisfied: (1) if task A requires more time to execute than task B on one processor then the same is true for any other processor in the system; (2) if processor A requires more time to execute one task than processor B then it is true for any other task in the batch. The second type of heterogeneity implies condition (2) only [11]. The third class describes systems where neither of these two conditions are met [11].

Fig. 2 demonstrates a matrix where all rows and columns are ranked in ascending order. This may not be the case for many ETC matrices where t_{ij} entries are not represented by products of two independent components. As a result, the processors may not be ranked identically for each row. To satisfy the "global" ordering required to apply the DLR framework, the average mean execution values

task characteristics	processor characteristics					
	$-\lambda_1 T_1$	$\leq$	$-\lambda_2 T_2$	$\cdots$	$\leq$	$-\lambda_N T_N$
$r_1 n_1$	$-\lambda_1 T_1 r_1 n_1$	$\leq$	$-\lambda_2 T_2 r_1 n_1$	$\cdots$	$\leq$	$-\lambda_N T_N r_1 n_1$
$\leq$	$\leq$		$\leq$			$\leq$
$r_2 n_2$	$-\lambda_1 T_1 r_2 n_2$	$\leq$	$-\lambda_2 T_2 r_2 n_2$	$\cdots$	$\leq$	$-\lambda_N T_N r_2 n_2$
$\vdots$	$\vdots$		$\vdots$			$\vdots$
$\leq$	$\leq$		$\leq$			$\leq$
$r_N n_N$	$-\lambda_1 T_1 r_N n_N$	$\leq$	$-\lambda_2 T_2 r_N n_N$	$\cdots$	$\leq$	$-\lambda_N T_N r_N n_N$

Fig. 2. An example matrix based on the Cartesian product for the simplified resource-allocation example.

t_j^{av}, $1 \leq j \leq N$, used in this work are computed for each processor j across all the tasks left unassigned in the batch, i.e., $t_j^{av} = \sum_{i=1}^{N} t_{ij}/N$. Because the set of such tasks changes as a resource-allocation process continues, the t_j^{av} values change as well, which may change the ordering of processors.

Once the processors and tasks are sorted in ascending order of $-\lambda_j t_j^{av}$ and r_i rewards, respectively, matrix entry products $-\lambda_j t_j^{av} r_i$ in some rows and columns might not completely follow these "global" orders as shown in Fig. 3. To evaluate qualitatively how much, on average, a given ETC matrix deviates from each of these "global" orders, Δ_r and Δ_c metrics are introduced for rows and columns, respectively. Consider the Δ_r metric. First, the Bubble sort algorithm [26] is hypothetically applied to each row, and the number of swaps required to match the corresponding "global" order for processors is counted. Then, Δ_r is calculated as the average number of swaps across all the rows in the matrix. Similarly, the Δ_c metric is computed for the columns.

V. Distribution of Processors

The DLR theorem assumes that the distribution describing the relative random availability of each processor is known and constant over time. In the "Internet computing" model [45], where processors arrive (i.e., become available) from a large pool of external users, the required distribution can be constructed based on the past history of processor arrivals.

Let $N(t_0)$ denote the initial number of tasks in the batch. Consider a dedicated system composed of M processors, where $M \leq N(t_0)$. Fig. 4 illustrates

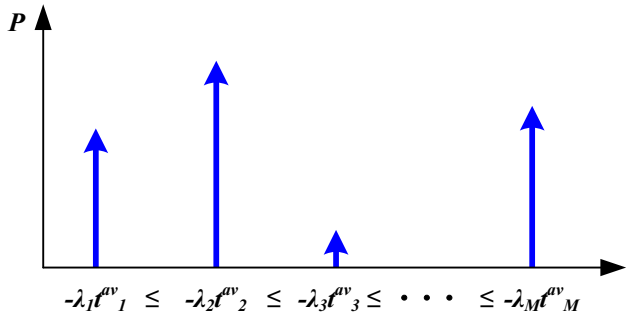


Fig. 4. A general probability mass function for a dedicated system with M processors.

a probability mass function (pmf) where values $-\lambda_k t_k^{av}$, $1 \leq k \leq M$, arranged in ascending order, can be used to compute the "bin boundaries" $a_{i,N}$ according to the DLR policy. Let p_k, $1 \leq k \leq M$, denote a probability associated with each processor, characterized by $-\lambda_k t_k^{av}$, in the pmf shown in Fig. 4. To evaluate the p_k values, consider the following models describing the availability of processors in a dedicated distributed system. The first two models are of little practical value; they are presented solely to provide a transition to the third model actually used in this study.

Identical failure rates, processor becomes available just after failure: Assume that each processor has the same failure rate λ, it becomes available for the next assignment immediately after a failure, and task completions do not make a processor available. Clearly in this situation, each processor fails, on average, the same number of times, i.e., probabilistic components p_k are the same and can be computed as $p_k = 1/M$.

Individual failure rates, processor becomes

task characteristics	processor characteristics				
	$-\lambda_1 t_1^{av}$	$\leq$	$-\lambda_2 t_2^{av}$	$\leq$	$-\lambda_3 t_3^{av}$
r_1	$-\lambda_1 t_{11} r_1$	$\leq$	$-\lambda_2 t_{12} r_1$	$\leq$	$-\lambda_3 t_{13} r_1$
$\leq$	$\geq$		$\leq$		$\leq$
r_2	$-\lambda_1 t_{21} r_2$	$\leq$	$-\lambda_2 t_{22} r_2$	$\geq$	$-\lambda_3 t_{23} r_2$
$\leq$	$\leq$		$\leq$		$\leq$
r_3	$-\lambda_1 t_{31} r_3$	$\leq$	$-\lambda_2 t_{32} r_3$	$\leq$	$-\lambda_3 t_{33} r_3$

Fig. 3. The example matrix demonstrates that the orders established for processors and tasks required in the DLR framework are not followed completely for the first processor and for the second task.

available just after failure: When each processor has its own failure rate λ_k, it fails over time, on average, every λ_k^{-1} time units. Therefore, the probability components p_k can be calculated as $p_k = \lambda_k / \sum_{k=1}^{M} \lambda_k$.

Individual failure rate, processor available just after failure and task completion: Let $\underline{w_k}$ denote the total availability rate for processor k. Assuming that times between failures and task completions are exponentially distributed for each processor, w_k can be computed as:

$$w_k = \lambda_k + \frac{1 - e^{-\lambda_k t_k^{av}}}{t_k^{av}}. \tag{5}$$

The second term in the sum in Eq. 5 estimates the processor's availability rate from task completions. On average, processor k would complete a task every t_k^{av} if there were no failures. The actual availability rate from task completions is lower than that because the probability of having no failures during t_k^{av} is accounted for in $1 - e^{-\lambda_k t_k^{av}}$. As before, the probability components p_k can be calculated by normalizing the w_k values, i.e., $p_k = w_k / \sum_{k=1}^{M} w_k$.

Based on our simulation studies, the following weighted iterative scheme to compute the pmf was found to give the best cumulative reward. According to this scheme, the distribution, described with a weighted sum of two pmfs, is recalculated before the assignment of the next task takes place. The first pmf is a normalized histogram constructed from the actual number of times that each processor has become available since the execution of the batch started. For the second pmf, the processor availability rates w_j are recomputed because the number of tasks left in the batch decreases with time. If $\overline{N(t)}$ denote the number of tasks left in the batch at time t, then $N(t)N(t_0)^{-1}$ and $1 - N(t)N(t_0)^{-1}$ are the weighting factors for the first and second pmfs in the sum, respectively. As such, these weighting factors are adjusted at each iteration proportionally to the progress made.

VI. Simulation setup and resource-allocation policies

The system, used as a prototype in this research, is a dedicated heterogeneous computer cluster processing HTTP requests (tasks). Typically, new tasks arrive dynamically into such systems, and their arrival times are not known in advance. In this study, we address a simplified case assuming that a batch is filled with HTTP requests without any new arrivals.

For ETC matrix generation, we assume that each HTTP request (i.e., task) in the batch belongs to one of five classes. Each task class is defined by a set of exponential distributions, where each distribution describes the probability of all execution times for that class on a given processor. To specify each distribution, the mean execution time is generated randomly in the range of $[0.5, 4]$ sec. for each task-class–processor pair. As a result of this random approach, an unsorted ETC matrix models the inconsistent heterogeneity described in Section IV. Subsequent sorting of elements in ascending order in the row and column directions result in the processor-consistent and task-processor-consistent heterogeneity models, respectively. After each sorting, the ETC matrix represents a completely different distributed system.

A batch for each simulation trial consisted of 100 tasks. Integer task rewards r_i were generated randomly in the range of $[1, 100]$. The deadline for each task was set at 300% over maximum, across

all processors, of the mean execution time for that task class. If a processor fails while executing a task then task returns to the batch, so it can be reassigned again. When the deadline expires for a task, it is dismissed from the batch. As the goal of our research is to maximize system performance under harsh operating conditions, the simulated mean time between failures was set for each processor relatively high, i.e., within the range of $[0.6, 1]$.

The performance of the following three different resource-allocation policies was explored in the simulated distributed system consisted of six processors.

1) In the **reward** policy, a task with the highest reward r_i is assigned to the next available processor.
2) In the **deadline** policy, a task with the closest deadline d_i is assigned to the next available processor.
3) In the **DLR** policy, once a processor becomes available, a resource-allocation system makes the following steps to select the next task for assignment:

    a) sorts the remaining $N(t)$ tasks in ascending order of r_i values;
    b) recomputes the distribution of processors, as explained in Section V;
    c) calculates "bin boundaries," for $N(t)$ according to the DLR theorem;
    d) determines the index i of the "bin" that the arrived processor corresponds to;
    e) selects task with index i for assignment from the list of tasks formed in step (1).

VII. Experimental Results

The resource-allocation policies were compared based on their ability to maximize the cumulative reward over 100 simulation trials. For each of these trials performed for the same set of parameters (i.e., distributions, number of tasks, etc.), a random number generator was seeded differently. This allowed the performance to be explored over a broad range of samples drawn differently in each trial from the corresponding distributions. In each simulation trial, the ETC matrix was generated in a random fashion to model inconsistent heterogeneity. The same matrix was sorted in the row direction and, then, in the column direction for the processor-consistent and task-processor-consistent scenarios, respectively. For each ETC matrix, the Δ_r and Δ_c metrics were computed for the **DLR** policy.

The average performance across 100 simulation trials along with the corresponding 95% confidence intervals are presented in Fig. 5(b) for all resource-allocation policies. The **DLR** policy consistently delivers the best results for all three types of heterogeneity explored in this study. Its superior performance is based on incorporating the stochastic information describing the availability of processors in the system into the decision making process.

Fig. 6 demonstrates the progress of each policy over time in one of the simulation trials for the processor-consistent heterogeneity. As expected, the **reward** policy quickly accumulates reward in the beginning as it selects the most profitable tasks first. When the number of such tasks becomes smaller, the total reward accumulation slows down due to more frequent task failures and deadline expirations. Fig. 5(a) shows the total number of successfully completed tasks averaged over 100 simulation trials. The fact that the **DLR** policy has rather moderate numbers of completed tasks highlights its ability to select tasks on a "more intelligent" basis.

The results demonstrate a significant performance improvement from the inconsistent to processor-consistent heterogeneity. For the **DLR** policy, observe the correlation between the reduction in average Δ_r and Δ_c values in Table I and this performance improvement.

This can be explained by the fact that t_j^{av} values dominate in $-\lambda_j t_j^{av}$ expressions, used to characterize processors. Thus, the majority of rows in the ETC matrix for the processor-consistent heterogeneity are ranked as required in the DLR framework. In contrast, an additional consistency among tasks in the **DLR** policy does not result in a performance improvement because only rewards r_i are used to rank tasks. The above analysis suggests that if the reward r_i for each task i was generated proportionally to the task's execution time then the majority of columns in the task-processor-consistent ETC matrix would be ranked in the same manner as the tasks in the DLR framework, that would result in a substantial performance improvement. The described scenario appears to be plausible in

TABLE I
AVERAGE Δ_r AND Δ_c VALUES ALONG WITH THE CORRESPONDING 95% INTERVALS COMPUTED ACROSS 100 SIMULATION TRIALS FOR 100 TASKS IS THE BATCH AND SIX PROCESSORS.

policy	class of heterogeneity	Δ_r		Δ_c	
		average value	95% conf. interval	average value	95% conf. interval
DLR	inconsistent	2.93	[2.27, 3.59]	365.41	[300.33, 430.47]
	processor-consistent	1.47	[0.82, 2.12]	277.77	[216.30, 339.24]
	task-processor-consistent	0.87	[0.29, 1.45]	274.46	[216.62, 332.30]

many practical applications as it implies that tasks with longer execution times are more valuable.

VIII. RELATED WORK

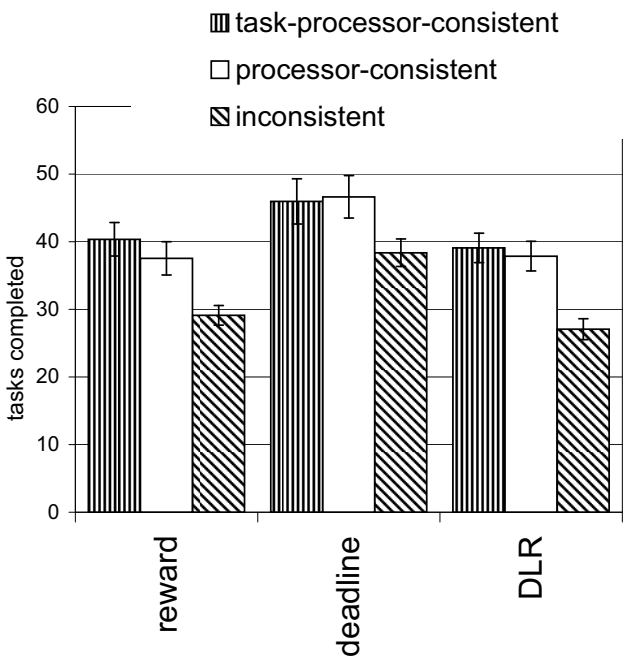
The original work of Derman et al. [13] inspired research in various areas. Example applications of the DLR theorem include selling houses and job-search strategy [1]. In 1972, Albright and Derman determined the limiting behavior of the $a_{i,N}$'s as N becomes large [3]. The derived closed form solution can be used to avoid lengthy computation of $a_{i,N}$'s in such cases. Later, these authors addressed cases where the arrival process is a non-homogeneous Poisson process with a general discount function [2]. Nakai permits the distribution of resources to change according to a partially observable Markov process and allows the number of resources to be random [32]. Sakaguchi allows a fixed time horizon [36] and permits resource values to be dependent [37]. The results of these studies were applied to investment strategies [35], firing torpedoes at randomly arriving targets [37], allocating organs for transplants [12], and manufacturing and telecommunications [34]. Similar to our work, all aforementioned studies address a sequential assignment problem in different in problem domains, i.e., at each time when the resource-allocation system observes a realization of random variable, it must select the best action, one at a time in sequential order.

As mentioned before, the fault tolerant aspect of modern distributed computing systems has been extensively explored. However, the available literature on distributed computing in such uncertain environments primarily considers reactive techniques, where a node failure is addressed only after its occurrence [10]. One of the few exceptions is the paper of Dhakal et al. [15] that presents two preemptive load-balancing policies for a heterogeneous distributed computing system with wireless links between nodes. Preemptiveness in this case implies adjusting actions to compensate for the possibility of node failure/recovery. The main goal for these policies is to avoid a scenario where a node fails while having a large amount of unprocessed load. The data transfer of such load to other nodes may result in a large random delay over wireless channels with following idle times on other nodes. A probabilistic model, based on the concept of regenerative processes, is presented to assess the overall performance of the system under these policies. The experiments show that preemptively utilizing the statistical information about the failure and recovery processes to adjust the load-balancing gain to an optimal value, allows one to minimize the mean of the overall completion time of the total workload. Although the problem domain differs from ours, [15] exemplifies an effective integration of the available node failure/recovery statistics into the resource-allocation process.

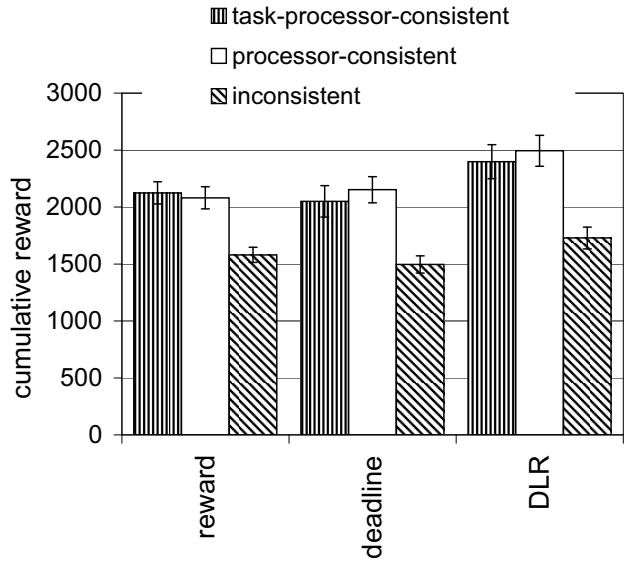
IX. CONCLUSION

This paper presents a method for robust static resource allocation in distributed systems where the compute resources fail in a random fashion. Given a batch of tasks, a resource-allocation system assigns tasks to compute resources that become available just after recovery from failures or task completions. The major contribution is the design of a resource-allocation policy for the above environment based on the concepts of the Derman-Lieberman-Ross theorem. The derived policy maximizes the expected cumulative reward received from the tasks that finish before their deadlines, given that the compute resources fail in a random fashion.

The resource-allocation policy was derived for the simplified case where tasks and processors are categorized by the number of instructions and the time



(a)



(b)

Fig. 5. Performance results for inconsistent, processor-consistent, and task-processor-consistent types of heterogeneity averaged across 100 simulation trials are plotted for: (a) the number of successfully completed tasks, (b) the cumulative reward. The error bars correspond to 95% confidence intervals. Although the **DLR** policy has relatively low percent of successfully completed tasks, its better performance wit respect to the stated metric of cumulative reward is based on the integration of the available stochastic information into the decision making process.

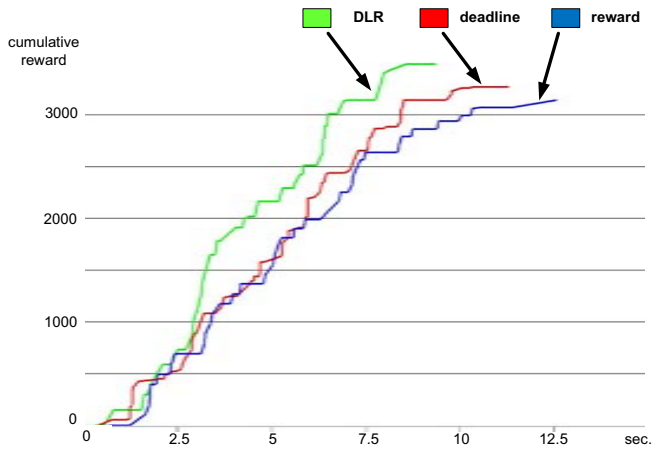


Fig. 6. Progress over time of the three resource-allocation policies with respect to the cumulative reward captured in one simulation trial for processor-consistent heterogeneity.

required to execute one instruction, respectively. Further, this policy was adapted to accommodate ETC matrices that provide a more accurate means of capturing the relationship among tasks and processors in distributed systems. As this study focused on dedicated distributed systems with a *limited* set of compute resources, the distribution describing the relative random availability of each processor was derived in Section V based on the characteristics of the tasks batched and the parameters of the processors available in the system.

The superior performance demonstrated by the **DLR** resource-allocation policy reveals its great potential for a variety of applications in a broad spectrum of distributed systems. For example, the problem of maximizing the performance when the system experiences temporal failures of compute resources is an important issue for embedded systems (e.g., [33], [40]), sensor networks (e.g., [49]), and special purpose cluster-based systems (e.g., [39]). Similar to the environment considered in this work, such systems sometimes are employed under harsh conditions but must deliver a certain level of performance to remain in operation. The application domains include surveillance for homeland security, monitoring vital signs of medical patients, and automatic target recognition systems. Thus, the proposed DLR-based resource-allocation scheme can be adapted in those systems.

REFERENCES

[1] S. C. Albright, "Optimal sequential assignments with random arrival times," *Management Science*, vol. 21, no. 1, pp. 60–67, Sep. 1974.

[2] ——, "A Bayesian approach to a generalized house selling problem," *Management Science*, vol. 24, no. 4, pp. 432–440, Dec. 1977.

[3] S. C. Albright and C. Derman, "Asymptotic optimal policies for the stochastic sequential assignment problem," *Management Science*, vol. 19, no. 1, pp. 46–51, Sep. 1972.

[4] S. Ali, T. D. Braun, H. J. Siegel, A. A. Maciejewski, N. Beck, L. Boloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao, "Characterizing resource allocation heuristics for heterogeneous computing systems," in *Advances in Computers*, A. R. Hurson, Ed. Amsterdam, the Netherlands: Elsevier, 2005, vol. 63: Parallel, Distributed, and Pervasive Computing, pp. 91–128.

[5] S. Ali, J.-K. Kim, H. J. Siegel, and A. A. Maciejewski, "Static heuristics for robust resource allocation to continuously executing applications," *Journal of Parallel and Distributed Computing*, vol. 68, no. 8, pp. 1070–1080, Apr. 2008.

[6] S. Ali, A. A. Maciejewski, and H. J. Siegel, "Perspectives on robust resource allocation for heterogeneous parallel systems," in *Handbook of Parallel Computing: Models, Algorithms, and Applications*, S. Rajasekaran and J. Reif, Eds. Boca Raton, FL: Chapman & Hall, 2008, pp. 41–1–41–30.

[7] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim, "Measuring the robustness of a resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 7, pp. 630–641, Jul. 2004.

[8] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer, "SETI at home: An experiment in public-resource computing," *Communications of the ACM*, vol. 45, no. 11, pp. 56–61, Nov. 2002.

[9] H. Barada, S. M. Sait, and N. Baig, "Task matching and scheduling in heterogeneous systems using simulated evolution," in *Heterogeneous Computing Workshop*, Apr. 2001, pp. 875–882.

[10] K. Birman, *Reliable Distributed Systems*. New York, NY: Springer-Verlag, 2005.

[11] T. D. Braun, H. J. Siegel, N. Beck, L. Bölöni, R. F. Freund, D. Hensgen, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810–837, Jun. 2001.

[12] I. David and U. Yechiali, "A time-dependent stopping problem with application to live organ transplants," *Operations Research*, vol. 33, no. 3, pp. 491–504, Jun. 1985.

[13] C. Derman, G. J. Lieberman, and S. M. Ross, "A sequential stochastic assignment problem," *Management Science*, vol. 18, no. 7, pp. 349–355, Mar. 1972.

[14] J. L. Devore, *Probability and Statistics for Engineering and Sciences*, 5th ed. Los Angeles, CA: Duxbury Press, 1999.

[15] S. Dhakal, M. M. Hayat, J. E. Pezoa, C. T. Abdallah, J. D. Birdwell, and J. Chiasson, "Load balancing in the presence of random node failure and recovery," in *The 20th International Parallel and Distributed Processing Symposium (IPDPS 2006)*, 2006, pp. 25–29.

[16] M. K. Dhodhi, I. Ahmad, and A. Yatama, "An integrated technique for task matching and scheduling onto distributed heterogeneous computing systems," *Journal of Parallel and Distributed Computing*, vol. 62, no. 2, pp. 1338–1361, Sep. 2002.

[17] R. F. Freund and H. J. Siegel, "Heterogeneous processing," *IEEE Computer*, vol. 26, no. 6, pp. 13–17, Jun. 1993.

[18] S. Garg, A. Moorsel, K. Vaidyanathan, and K. S. Trivedi, "A methodology for detection and estimation of software aging," in *The Ninth International Symposium on Software Reliability Engineering*, Nov. 1998, pp. 283–292.

[19] A. Ghafoor and J. Yang, "A distributed heterogeneous supercomputing management system," *IEEE Computer*, vol. 26, no. 6, pp. 78–86, Jun. 1993.

[20] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed. San Francisco, CA: Morgan Kaufmann, 2003, ch. 8.

[21] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on non-identical processors," *Journal of the ACM*, vol. 24, no. 2, pp. 280–289, Apr. 1977.

[22] J.Yang, A. Khokhar, S. Sheikh, and A. Ghafoor, "Estimating execution time for parallel tasks in heterogeneous processing (HP) environment," in *Heterogeneous Computing Workshop*, Apr. 1994, pp. 23–28.

[23] M. Kafil and I. Ahmad, "Optimal task assignment in heterogeneous distributed computing systems," *IEEE Concurrency*, vol. 6, no. 3, pp. 42–51, Jul. 1998.

[24] A. Khokhar, V. K. Prasanna, M. E. Shaaban, and C. Wang, "Heterogeneous processing: Challenges and opportunities," *IEEE Computer*, vol. 26, no. 6, pp. 18–27, Jun. 1993.

[25] J.-K. Kim, S. Shivle, H. J. Siegel, A. A. Maciejewski, T. Braun, M. Schneider, S. Tideman, R. Chitta, R. B. Dilmaghani, R. Joshi, A. Kaul, A. Sharma, S. Sripada, P. Vangari, and S. S. Yellampalli, "Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment," *Journal of Parallel and Distributed Computing*, vol. 67, no. 2, pp. 154–169, Feb. 2007.

[26] D. Knuth, *The Art of Computer Programming*. Reading, MA: Addison-Wesley, 1997.

[27] C. Leangsuksun, J. Potter, and S. Scott, "Dynamic task mapping algorithms for a distributed heterogeneous computing environment," in *4th IEEE Heterogeneous Computing Workshop (HCW '95)*, Apr. 1995, pp. 30–34.

[28] H. M. Lee, S. H. Chin, J. H. Lee, D. W. Lee, K. S. Chung, S. Y. Jung, and H. C. Yu, "A resource manager for optimal resource selection and fault tolerance service in grids," in 10^{th} *IEEE International Symposium on Cluster Computing and the Grid*, 2004.

[29] L. Li, K. Vaidyanathan, and K. Trivedi, "An approach for estimation of software aging in a web server," in *The 2002 International Symposium on Empirical Software Engineering (ISESE'02)*, 2002, pp. 91–100.

[30] M. Maheswaran, T. D. Braun, and H. J. Siegel, "Heterogeneous distributed computing," in *Encyclopedia of Electrical and Electronics Engineering*, J. Webster, Ed. New York, NY: John Wiley, 1999, vol. 8, pp. 679–690.

[31] A. M. Mehta, J. Smith, H. J. Siegel, A. A. Maciejewski, A. Jayaseelan, and B. Ye, "Dynamic resource allocation heuristics that manage tradeoff between makespan and robustness," *Journal of Supercomputing, Special Issue on Grid Technology*, vol. 42, no. 1, pp. 33–58, 2007.

[32] T. Nakai, "A sequential stochastic assignment problem in a partially observable Markov chain," *Mathematics of Operations Research*, vol. 11, no. 2, pp. 230–240, May 1986.

[33] S. I. Park, V. Raghunathan, and M. B. Srivastava, "Energy efficiency and fairness tradeoffs in multi-resource, multi-tasking embedded systems," in *The 2003 International Symposium on Low Power Electronics and Design (ISLPED'03)*, Aug. 2003, pp. 2–13.

[34] R. Righter, "Stochastically maximizing the number of successes in a sequential assignment problem," *Journal of Applied Probability*, vol. 27, no. 2, pp. 351–364, Jun. 1990.

[35] V. Saario, "Limiting properties of the discounted house-selling problem," *European Journal of Operational Research*, vol. 20, no. 2, pp. 206–210, May 1985.

[36] K. Sakaguchi, "A sequential stochastic assignment problem associated with a non-homogeneous markov process," *Japanese Journal of Mathematics*, vol. 29, pp. 13–22, 1984.

[37] ——, "Best choice problems for randomly arriving offers during a random lifetime," *Japanese Journal of Mathematics*, vol. 31, pp. 107–117, 1986.

[38] V. Shestak, E. K. P. Chong, A. A. Maciejewski, H. J. Siegel, L. Benmohamed, I.-J. Wang, and R. Daley, "A hybrid branch-and-bound and evolutionary approach for allocating strings of applications to heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 68, no. 4, pp. 410–426, Apr. 2008.

[39] V. Shestak, J. Smith, A. A. Maciejewski, and H. J. Siegel, "Stochastic robustness metric and its use for static resource allocations," *Journal of Parallel and Distributed Computing*, vol. 68, no. 8, pp. 1157–1173, Aug. 2008.

[40] I. Shin and I. Lee, "Periodic resource model for compositional real-time guarantees," in 24^{th} *IEEE Real-Time System Symposium (RTSS 2003)*, Dec. 2003, pp. 469–474.

[41] H. Singh and A. Youssef, "Mapping and scheduling heterogeneous task graphs using genetic algorithms," in 5^{th} *IEEE Heterogeneous Computing Workshop (HCW '96)*, 1996, pp. 86–97.

[42] J. Smith, L. D. Briceño, A. A. Maciejewski, and H. J. Siegel, "Measuring the robustness of resource allocations in a stochastic dynamic environment," in *21st International Parallel and Distributed Processing Symposium (IPDPS 2007)*, Mar. 2007.

[43] J. Smith, H. J. Siegel, and A. A. Maciejewski, "Robust resource allocation in heterogeneous parallel and distributed computing systems," in *Wiley Encyclopedia of Computing*, B. Wah, Ed. New York, NY: John Wiley & Sons, 2009, vol. 4, pp. 2461–2470.

[44] P. Sugavanam, H. J. Siegel, A. A. Maciejewski, M. Oltikar, A. Mehta, R. Pichel, A. Horiuchi, V. Shestak, M. Al-Otaibi, Y. Krishnamurthy, S. Ali, J. Zhang, M. Aydin, P. Lee, K. Guru, M. Raskey, and A. Pippin, "Robust static allocation of resources for independent tasks under makespan and dollar cost constraints," *Journal of Parallel and Distributed Computing*, vol. 67, no. 4, pp. 400–416, Apr. 2007.

[45] T. Tannenbaum, D. Wright, K. Miller, and M. Livny, *Condor - A Distributed Job Scheduler*, ser. Beowulf Cluster Computing with Linux, T. Sterling, Ed. Cambridge, MA: The MIT Press, 2002.

[46] L. Wang, H. J. Siegel, V. P. Roychowdhury, and A. A. Maciejewski, "Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach," *Journal of Parallel and Distributed Computing*, vol. 47, no. 1, pp. 8–22, Nov. 1997.

[47] D. Xu, K. Nahrstedt, and D. Wichadakul, "QoS and contention-aware multi-resource reservation," *Cluster Computing*, vol. 4, no. 2, pp. 95–107, Apr. 2001.

[48] J. Yang, I. Ahmad, and A. Ghafoor, "Estimation of execution times on heterogeneous supercomputer architectures," in *International Conference on Parallel Processing*, vol. 1, Aug. 1993, pp. 219–225.

[49] T. Yuan, J. Boangoat, E. Ekici, and F. Ozguner, "Real-time task mapping and scheduling for collaborative in-network processing in dvs-enabled wireless sensor networks," in 20^{st} *International Parallel and Distributed Processing Symposium (IPDPS 2006)*, Apr. 2006, pp. 21–27.

Vladimir Shestak received a B.S. degree from the Moscow Engineering Physics Institute in 1998, M.S. degree from New Jersey Institute of Technology in 2003, and Ph.D. degree from Colorado State University in 2008. In 2006 and 2007 he was a recipient of an IBM Ph.D. Fellowship. He is currently a Software Engineer at InfoPrint Solution Company, Boulder, CO. His research interests include resource management within distributed computing systems, algorithm parallelization, and optimization in image processing. For more information, please visit www.engr.colostate.edu/~chestak.

Edwin K. P. Chong received the B.E.(Hons.) degree with First Class Honors from the University of Adelaide, South Australia, in 1987; and the M.A. and Ph.D. degrees in 1989 and 1991, respectively, both from Princeton University, where he held an IBM Fellowship. He joined the School of Electrical and Computer Engineering at Purdue University in 1991, where he was named a University Faculty Scholar in 1999, and was promoted to Professor in 2001. Since August 2001, he has been a Professor of Electrical and Computer Engineering and a Professor of Mathematics at Colorado State University. His research interests span the areas of communication and sensor networks, stochastic modeling and control, and optimization methods. He coauthored the recent best-selling book, An Introduction to Optimization, 3rd Edition, Wiley-Interscience, 2008. He is currently on the editorial board of the IEEE Transactions on Automatic Control, Computer Networks, Journal of Control Science and Engineering, and IEEE Expert Now. He is a Fellow of the IEEE, and served as an IEEE Control Systems Society Distinguished Lecturer. He received the NSF CAREER Award in 1995 and the ASEE Frederick Emmons Terman Award in 1998. He was a co-recipient of the 2004 Best Paper Award for a paper in the journal Computer Networks. He has served as Principal Investigator for numerous funded projects from NSF, DARPA, and other funding agencies.

Anthony A. Maciejewski received the B.S.E.E, M.S., and Ph.D. degrees in Electrical Engineering in 1982, 1984, and 1987, respectively, all from The Ohio State University. From 1988 to 2001, he was a Professor of Electrical and Computer Engineering at Purdue University. In 2001, he joined Colorado State University where he is currently the Head of the Department of Electrical and Computer Engineering. He is a Fellow of IEEE. An up-to-date vita is available at www.engr.colostate.edu/~aam.

Howard Jay Siegel was appointed the Abell Endowed Chair Distinguished Professor of Electrical and Computer Engineering at Colorado State University (CSU) in 2001, where he is also a Professor of Computer Science. He is the Director of the CSU Information Science and Technology Center (ISTeC), a university-wide organization for promoting, facilitating, and enhancing CSUs research, education, and outreach activities pertaining to the design and innovative application of computer, communication, and

information systems. From 1976 to 2001, he was a professor at Purdue University. Prof. Siegel is a Fellow of the IEEE and a Fellow of the ACM. He received a B.S. degree in electrical engineering and a B.S. degree in management from the Massachusetts Institute of Technology (MIT), and the M.A., M.S.E., and Ph.D. degrees from the Department of Electrical Engineering and Computer Science at Princeton University. He has co-authored over 350 technical papers. His research interests include robust computing systems, resource allocation in computing systems, heterogeneous parallel and distributed computing and communications, parallel algorithms, and parallel machine interconnection networks. He was a Coeditor-in-Chief of the Journal of Parallel and Distributed Computing, and was on the Editorial Boards of both the IEEE Transactions on Parallel and Distributed Systems and the IEEE Transactions on Computers. He was Program Chair/Co-Chair of three major international conferences, General Chair/Co-Chair of seven international conferences, and Chair/Co-Chair of five workshops. He is a member of the Eta Kappa Nu electrical engineering honor society, the Sigma Xi science honor society, and the Upsilon Pi Epsilon computing sciences honor society. He has been an international keynote speaker and tutorial lecturer, and has consulted for industry and government. For more information, please see www.engr.colostate.edu/~hj.