A Stochastic Approach to Measuring the Robustness of Resource Allocations in Distributed Systems

Vladimir Shestak*, Jay Smith†*, H. J. Siegel*‡, and Anthony A. Maciejewski*
*Electrical and Computer Engineering Department
‡Computer Science Department
Colorado State University, Fort Collins, CO 80523–1373
Email: {shestak, hj, aam}@engr.colostate.edu.

†IBM
6300 Diagonal Highway Boulder, CO 80301
Email: bigfun@us.ibm.com

Abstract

Often, parallel and distributed computing systems must operate in an environment replete with uncertainty. Determining a resource allocation that accounts for this uncertainty in a way that can provide a probabilistic guarantee that a given level of quality of service (QoS) is achieved is an important research problem. This paper defines a stochastic methodology for quantifiably determining a resource allocation's ability to satisfy QoS constraints in the midst of uncertainty in system parameters. Uncertainty in system parameters and its impact on system performance are modeled stochastically. This stochastic model is then used to derive a quantitative expression for the robustness of a resource allocation. The paper investigates the utility of the proposed stochastic robustness metric by applying the metric to resource allocations in a simulated distributed system. The simulation results are then compared with deterministically defined metrics from the literature.

1. Introduction and Problem Statement

Often, parallel and distributed computing systems must operate in an environment replete with uncertainty while providing a required level of quality of service (QoS). This reality has inspired an increasing interest in robust design. The following are some examples. The Robust Network Infrastructures Group at the Computer Science and Artificial Intelligence Laboratory at MIT takes the position that "... a key challenge is to ensure that the network can be robust in the face of failures, time-varying load, and various errors." The research at the User-Centered Robust Mobile Computing Project at Stanford "concerns the hardening of the network and software infrastructure to make it highly robust." The Workshop on Large-Scale Engineering Networks: Robustness, Verifiability, and Convergence (2002) concluded that the "Issues are ... being able to quantify and design for robustness ..." There are many other projects of similar nature at other schools and organizations.

To provide insight into the target systems operating under uncertainty that must maintain a certain level of QoS, consider the following two examples.

Fig. 1 schematically depicts part of a total ship computing environment in the Adaptive and Reflective Middleware Systems (ARMS) program supported by DARPA's Information Exploitation Office [4]. This part of the ARMS example represents a large class of systems that operate on *periodically updated* data sets, e.g., surveillance for homeland security, monitoring vital signs of medical patients. Typically, in such systems, sensors (e.g., radar, sonar, video camera) produce data sets with a constant period of $\underline{\Lambda}$ time units. Periodic data updates imply that the total processing time for any given data set must not exceed Λ, i.e., Λ is an imposed timing QoS constraint for the system. Suppose that each input data set must be processed by a collection of $\underline{N}$ independent applications that can be executed in parallel on the available set of $\underline{M}$ heterogeneous compute nodes. Due to the changing physical world, the periodic data sets produced by the system sensors typically vary in such parameters as the number of observed objects present in the

*This research was supported by the DARPA Information Exploitation Office under contract No. NBCHC030137, by the Colorado State University Center for Robustness in Computer Systems (funded by the Colorado Commission on Higher Education Technology Advancement Group through the Colorado Institute of Technology), and by the Colorado State University George T. Abell Endowment.

radar scan and signal-to-noise ratio. Variability in the data sets results in variability in the execution times of processing applications. Due to an inability to precisely predict application execution times, they can be considered uncertainty parameters in the system.

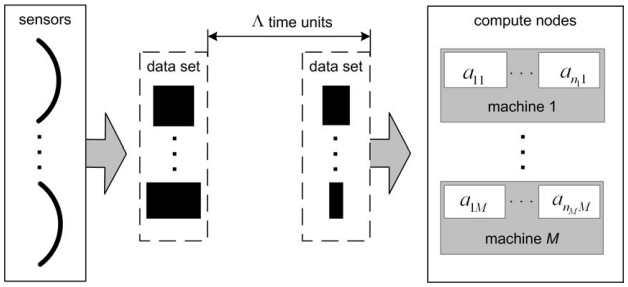


Figure 1. Major functional units and data flow for a class of system that operates on periodically updated data sets. The a_{ij}'s denote applications executing on machine j. Processing of each data set must be completed within Λ time units.

An important task for a mapper (resource management system) is to distribute processing applications across compute nodes such that a produced resource allocation is *robust*, i.e., it can guarantee (or has a high probability) that the imposed QoS constraint is satisfied despite uncertainties in application execution times.

Another example of a distributed computing system that must accommodate uncertainty under tight timing QoS constraints is a web search engine. In the Google search engine [5], the user query response time is required to be at most 0.5 seconds—including network round trip communication latency. Query execution in this system consists of two major phases. The first phase produces an ordered list of document identifiers (docids). This list is a result of merging the responses from multiple index servers, each searching over a particular subset (index shard) of the entire index database. The second phase uses the list of docids and computes the actual title and uniform resource locators of these documents, along with any query-specific document summary information. Document servers perform this job, each processing a certain part of the docids list.

Consider the first phase of the system where a fork-join job [17] must be performed, as shown in Fig. 2 (similar analysis can be derived for phase 2). To speed up overall execution time, each query is split into multiple copies which are processed in parallel by a subset of the available index servers—chosen by the cluster manager such that they cover the entire index database. Each copy queues to a different index server, and each index server has its own input buffer where it serves requests in the order of their arrival (for simplicity of analysis, sequential query processing at each index server is considered in this study). The cluster manager must be able to accommodate uncertainty in query processing times because the exact time required to process a query is not known *a priori*. However, it is possible for the fork node to use the attributes of an incoming query to identify a subset of the past queries that have similar attributes and share a common distribution of execution times. These past execution times taken from the identified subset of queries can be used to create a probability density function (pdf) that describes the possible execution times for the incoming query.

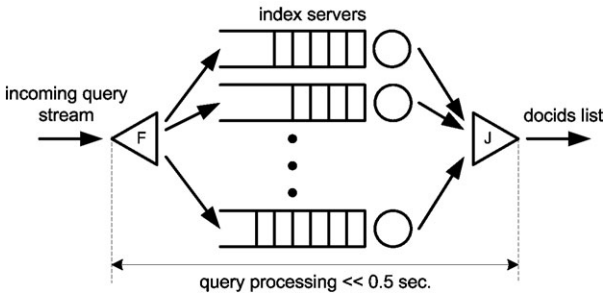


Figure 2. Fork (F) and Join (J) query processing in the first phase of Google Web search engine.

In both examples, simple load balancing algorithms may be sufficient when a distributed system is not oversubscribed, i.e., the number of queued tasks at each compute node is small. However, more sophisticated stochastic analysis is required for resource allocation as the system experiences workload surges or loss of resources.

Robust design for such systems involves determining a resource allocation that can account for uncertainty in a way that enables the system to provide a probabilistic guarantee that a given QoS is achieved. This paper defines a stochastic methodology for quantifiably determining a resource allocation's ability to satisfy QoS constraints in the midst of uncertainty in system parameters.

In this work, a new stochastic robustness metric is presented where the uncertainty in system parameters and its impact on system performance are modeled stochastically. This stochastic model is then used to derive a quantitative evaluation of the robustness of a given resource allocation, which is interpretable as the likelihood that the resource allocation will satisfy the expressed QoS constraints. The problem of deriving a resource allocation represents a large body of research in the field of parallel and distributed computing (e.g., [1, 8, 9, 11–14, 18, 21, 23, 24]), but lies outside the scope of this paper. This paper analyzes the utility of the proposed stochastic robustness metric by applying the metric to resource allocations in a simulated prototype of the

distributed system described in the ARMS Example. The simulation results of the application are also compared with a deterministic approach for determining the robustness of a resource allocation.

The major contribution of this work is a mathematical model for a stochastic robustness metric that utilizes available information to quantifiably determine a resource allocation's ability to satisfy expressed QoS constraints. In addition, the utility of the proposed metric is demonstrated by comparison with a common performance measure of resource allocations, as well as a comparison with a similar metric taken from the literature. We show that when the additional information required by the stochastic model is available, a better selection among resource allocations is possible. Further, this work presents two alternative means for computing the metric that render the required computation practical in a number of common environments.

The remainder of this work is organized in the following manner. Section 2 develops the general framework upon which the stochastic robustness metric is built. Specifically, in Subsection 2.1 a formal definition of stochastic robustness is given. Subsection 2.2 discusses methods of computing the stochastic robustness metric given the independence of input parameters. A bootstrap method for estimating probabilities in distributed systems is analyzed in Subsection 2.3. A numerical study is included in Section 3 to further validate the utility of the proposed methodology. Section 4 presents this work in relation to the published work from the literature. Section 5 concludes the paper.

2. Mathematical Model for a Stochastic Robustness Metric

2.1. Definition of Stochastic Robustness

The derivation of a stochastic robustness metric for a given distributed computing environment requires a mathematical model that accounts for the existing uncertainty to reasonably predict the performance of the system. To emphasize the distinction between the system and its mathematical model, any new terminology related to the model will explicitly reference the model.

In the ARMS example, let S_j be the sequence of n_j applications assigned to compute node j in the order they are to be executed, i.e., $S_j = [a_{1j}, a_{2j}, ..., a_{n_j j}]$. In the Google example, sequence S_j represents n_j queries assigned to index server j. Let random variable T_{ij} denote the execution time of each individual application (query) a_{ij} on compute node (index server) j. In a variety of systems, the execution time T_{ij} represents the time required for a_{ij} to process an individual data set on compute node j. The random variables T_{ij} serve as the inputs to the mathematical model that characterize the uncertainty in execution time for each of the applications in the system. These random variables will be referred to as the uncertainty parameters of the mathematical model. The performance of the considered distributed system is measured according to an established performance metric and may be different on a per system basis [1, 16]. In the mathematical model, the system performance ψ referred to as the performance characteristic, is an output of the mathematical model of the system.

In the ARMS example, the evaluation of system performance is based on the makespan value (total time required for all applications to process a given data set) [8] achieved by a given resource allocation, i.e., a smaller makespan equates to better performance. The functional dependence between the uncertainty parameters and the performance characteristic in the model can be expressed mathematically as

$$\psi = \max\{\sum_{i=1}^{n_1} T_{i1}, ..., \sum_{i=1}^{n_M} T_{iM}\}. \quad (1)$$

In the Google example, the performance in phase 1 is measured for each individual query. Unlike the ARMS example where the evaluation of makespan values occurs at each Λ prior to the execution of any application, query performance evaluation in the Google example is performed while the system is busy processing queries. Assume that M copies of a query arrive at index servers at wall-clock time t, and n_j is the number of queries pending execution or being executed by index server j at that time. Let t_{0j} denote the wall-clock start time of execution for the query being processed by index server j at time t. In the corresponding mathematical model, the functional dependence between the uncertainty parameters and the performance characteristic at time t, denoted as $\psi(t)$, can be stated as

$$\psi(t) = \max\{T_{11} - (t - t_{01}) + \sum_{i=2}^{n_1} T_{i1}, \\ ..., T_{1M} - (t - t_{0M}) + \sum_{i=2}^{n_M} T_{iM}\}. \quad (2)$$

Due to its functional dependence on the uncertainty parameters T_{ij}, the performance characteristic ψ is itself a random variable.

Let the QoS constraints be quantitatively described by the values β_{min} and β_{max} limiting the acceptable range of possible variation in the system performance [2], i.e., $\beta_{min} \leq \psi \leq \beta_{max}$. **The stochastic robustness metric is the probability that the performance characteristic of the system is confined to the interval $[\boldsymbol{\beta_{min}}, \boldsymbol{\beta_{max}}]$, i.e., $\mathbb{P}[\boldsymbol{\beta_{min}} \leq \boldsymbol{\psi} \leq \boldsymbol{\beta_{max}}]$.** For a given resource allocation, the stochastic robustness quantitatively measures the likelihood that the generated system performance will satisfy the stipulated QoS constraints. Clearly, unity is the most desirable stochastic robustness metric value, i.e., there is zero

probability that the system will violate the established QoS constraints.

2.2. Independence Assumption

In the model of compute node j, the functional dependence between the set of local uncertainty parameters $\{T_{ij}|1 \leq i \leq n_j\}$ and the local performance characteristic ψ_j can be stated in the ARMS example as $\psi_j = \sum_{i=1}^{n_j} T_{ij}$; in the Google example as $\psi_j = T_{1j} - (t - t_{0j}) + \sum_{i=2}^{n_j} T_{ij}$.

Independence of the local performance characteristics implies that the random variables $\psi_1, \psi_2, ..., \psi_M$ are mutually independent. If such independence is established, the stochastic robustness in a distributed system can be expressed as the product of the probabilities of each compute node meeting the imposed QoS constraints. Mathematically, this is given as

$$\mathbb{P}[\beta_{min} \leq \psi \leq \beta_{max}] = \prod_{j=1}^{M} \mathbb{P}[\beta_{min} \leq \psi_j \leq \beta_{max}]. \quad (3)$$

Specifically in (3), $\beta_{max} = \Lambda$ in the ARMS example and $\beta_{max} \ll 0.5$ sec. in the Google example. In both examples β_{min} is set to zero because there is no minimum time constraint on execution.

If the execution times T_{ij} of applications mapped on a compute node j are mutually independent (e.g., this assumption is valid for non-multitasking execution mode commonly considered in the literature [8, 11, 17, 21, 24]), then $\mathbb{P}[\beta_{min} \leq \psi \leq \beta_{max}]$ can be computed using an n_j-fold convolution of probability density functions (pdfs) $f_{T_{ij}}(t_i)$ [19]

$$\mathbb{P}[\beta_{min} \leq \psi_j \leq \beta_{max}] = \int_{\beta_{min}}^{\beta_{max}} [f_{T_{ij}}(t_1) * \ldots * f_{T_{n_j j}}(t_{n_j})]dt. \quad (4)$$

An n_j-fold convolution of (4) requires $n_j - 1$ computations of the convolution integral [19]; thus, a direct numerical integration may become a formidable task when n_j is a relatively large number. However, a high quality approximation to the n_j-fold convolution can be obtained, at a low computational expense, by applying Fourier transforms. Thus, if $\Phi_{T_{ij}}(\omega)$ denotes the characteristic function [22] of T_{ij}, then (4) can be computed as follows

$$\mathbb{P}[\beta_{min} \leq \psi_j \leq \beta_{max}] = \int_{\beta_{min}}^{\beta_{max}} \Phi_{\psi_j}^{-1}\{\Phi_{T_{1j}}(\omega) \times \ldots \times \Phi_{T_{n_j j}}(\omega)\}. \quad (5)$$

From this point on we assume that each pdf $f_{T_{ij}}(t_i)$ is expressed as a discrete probability mass function (pmf) utilizing Ω points—this is common in practical implementations. As such, the calculation can be performed in the frequency domain using a Fast Fourier Transform (FFT) that reduces the computational cost of finding the corresponding characteristic functions $\Phi_{T_{ij}}$. The FFT method is a discrete Fourier transform algorithm that reduces the number of computations needed for Ω points from $2\Omega^2$ to $2\Omega \log \Omega$ [22]. Thus, the computational complexity of determining the local performance characteristic can be drastically reduced, making the approach reasonable to compute.

Table I shows the empirical computation times required to execute n-fold convolution with respect to two different levels of n and four different levels of Ω. The table demonstrates that, as the number of sequential convolution operations grows, the corresponding computation time increases at a reasonable rate. This result reflects a potential applicability of the stochastic robustness metric for a broad spectrum of distributed systems where the imposed QoS constraints are either substantially longer than the total time needed for a mapper to execute a required number of convolutions, or a mapping is generated in off-line fashion, i.e., this time is not an issue.

Table 1. Computation times (sec.) required to achieve different levels of n-fold convolution computed with the Fast Fourier Transform method.

n in n-fold convolution	number of points Ω in T_{ij}'s pmf			
	62	128	256	512
10	0.0216	0.0462	0.0953	0.2059
100	1.2	1.79	3.57	7.28

In dynamic systems, processing a continuous stream of tasks (e.g., in the Google example), the number of convolutions required at each mapping event is relatively low. For example, evaluating a potential allocation of a given task on a particular compute node requires only one convolution of the execution time distribution for the task with the completion time distribution of the the task assigned last to the considered compute node. Once the assignment of a given task is finalized, its computed completion time distribution will be used for future assignment assessments.

2.3. Bootstrap Approximation

This subsection presents an alternative method of evaluating $\mathbb{P}[\beta_{min} \leq \psi_j \leq \beta_{max}]$ known in the literature as the bootstrap method [25]. In contrast to convolution that is applicable only when $\psi_j = \sum_{i=1}^{n_j} T_{ij}$, the bootstrap procedure

can be applied to *various* forms of functional dependence between local uncertainty parameters T_{ij} and the local performance characteristic ψ_j, making it very useful in practical implementations. For example, processing of queries by a Web server is typically done in a parallel multitasking environment, and there exists a complex functional dependence [3] between the time required to process the query and a number of currently executing threads, amount of data cached, types of requests, etc.

Suppose that for each T_{ij}, there are $\underline{k}$ sample observations obtained as a result of past executions of application i on compute node j. As k grows, new sample observations are added, and the sample pmf $\underline{\widehat{f}_{(k)T_{ij}}(t_i)}$, constructed from these observations, converges in probability to $f_{T_{ij}}(t_i)$, i.e., $\widehat{f}_{(k)T_{ij}}(t_i) \xrightarrow{\mathbb{P}} f_{T_{ij}}(t_i)$. Let $\underline{\widehat{T}^*_{(k)ij}}$ denote one draw from the sample distribution $\widehat{f}_{(k)T_{ij}}(t_i)$. Let $\underline{\widehat{\psi}^*_{(k)j}}$ be a bootstrap replication whose computation is based on a known functional dependence between the set of drawn $\widehat{T}^*_{ij}$ and ψ_j, i.e., $\widehat{\psi}^*_{(k)j} = g(\widehat{T}^*_{(k)1j}, ..., \widehat{T}^*_{(k)n_jj})$. In the bootstrap simulation step [25], $\underline{B}$ bootstrap replications of $\widehat{\psi}^*_{(k)j}$ are computed: $\widehat{\psi}^*_{(k)j,1}, ..., \widehat{\psi}^*_{(k)j,B}$. If $\underline{\widehat{F}_{(B)\psi_j}(t)}$ represents a sample cumulative density function (cdf) of ψ_j derived from these bootstrap replications, then the probability for the local characteristic function ψ_j can be approximated as

$$\mathbb{P}[\beta_{min} \leq \psi_j \leq \beta_{max}] \approx \widehat{F}_{(B)\psi_j}(\beta_{max}) - \widehat{F}_{(B)\psi_j}(\beta_{min}). \quad (6)$$

Equation (6) assumes the existence of a monotone normalizing transformation for the ψ_j distribution, and it is based on a proof of bootstrap percentile confidence interval [25]. An exact normalizing transformation will rarely exist, but approximate normalizing transformations may exist—the latter causes the probability that ψ_j is in the interval $[\beta_{min}, \beta_{max}]$ to be not exactly $\widehat{F}_{(B)\psi_j}(\beta_{max}) - \widehat{F}_{(B)\psi_j}(\beta_{min})$.

The pseudocode for the bootstrap analysis is as follows:

1. $B \leftarrow$ number of bootstrap replications
2. Vboot $\leftarrow$ vector of length B
3. Vsample $\leftarrow$ vector of length n_j
4. **for**(b in $1 : B$) **{**
5. **for**(i in $1 : n_j$) **{**
6. Vsample $\leftarrow$ sample with replacement $\widehat{f}_{(b)T_{ij}}(t_i)$
7. **}**
8. Vboot $\leftarrow g$(Vsample)
9. nullify Vsample
10. **}**
11. construct $\widehat{F}_{(B)\psi_j}(t)$ from Vboot
12. Nsamples $\leftarrow$ number of samples in Vboot $\in [\beta_{min}, \beta_{max}]$
13. $\mathbb{P}[\beta_{min} \leq \psi_j \leq \beta_{max}] \approx$ Nsamples$/B$

Table 2 presents the empirical data for an experiment conducted to illustrate the accuracy of the bootstrap approximation for the case where the functional dependence between T_{ij} and ψ_j was a summation. Table 2 captures the percent error of the achieved approximations based on equation 6 with respect to the exact convolution results. In the experiment, β_{min} was set to 0, β_{max} was set to the mean value of t from $\widehat{F}_{(B)\psi_j}$, and all T_{ij} distributions were modeled by randomly assigning a probability associated with each of Ω data points with final normalizations. Each value in Table 2 represents the average across 100 different trials. As the results show, (1) relative accuracy remains insensitive to the number of applications assigned to compute node j, (2) tighter approximations were obtained by increasing the number of bootstrap replications. If distributions of uncertainty parameters were closer to normal—which occurs often in practice—the resultant bootstrap approximations would be more precise as described in the proof of equation 6 [25]. There are other bootstrap approximations that may be more accurate, especially when the nature of the expected cdf of the performance metric is known. However, some bootstrap methods require a significant amount of computation and might be prohibitively expensive in certain distributed systems.

Table 2. Percent error achieved with bootstrap approximations.

n_j	number of bootstrap replications		
	100	1000	10000
10	5.63	5.61	2.16
100	8.35	3.23	2.13
1000	6.52	2.84	1.04

3. Example Application of Stochastic Robustness

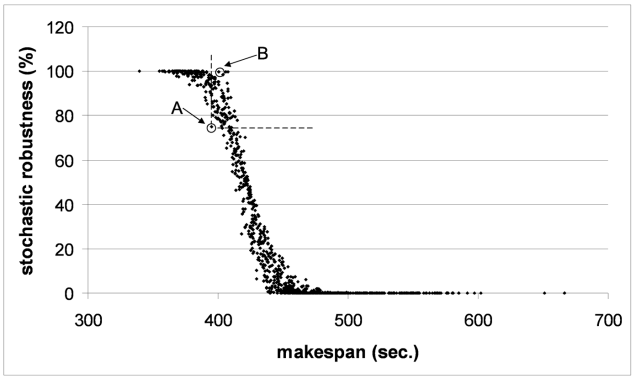
The experiments in this section seek to establish the utility of the stochastic robustness metric in distinguishing between resource allocations that perform similarly in terms

of the deterministic robustness metric from [2] and a commonly used metric, such as makespan. The simulation of the system outlined in the ARMS example of Section 1 included 1000 randomly generated resource allocations where 128 independent applications ($N = 128$) were allocated to eight machines ($M = 8$). Each of the application execution time distributions, specific to each application-machine pair, was modeled with a discrete pdf randomly constructed on the range $[0, 40]$ seconds, inclusive. To construct each discrete pdf, ten randomly selected values spread across the range of the distribution were assigned probabilities sampled uniformly on the range $(0, 1)$. All application execution time distributions were subsequently normalized. Let $mean_{av}$ be the average value computed across the means of all constructed application execution time distributions. In the conducted simulation, the QoS constraint Λ was set as follows $\Lambda = 1.5 \times N \times mean_{av}/M$. Recall, for the ARMS example Λ is a QoS constraint on system processing time that is used in the definition of the stochastic robustness metric given in equation (3). In Fig. 3, the "stochastic robustness" vertical axes correspond to the probability that the makespan will be $\leq \Lambda$. In this simulation, the deterministic robustness metric and makespan were calculated using the mean of the execution time distribution for each application-machine pair in the given allocation.

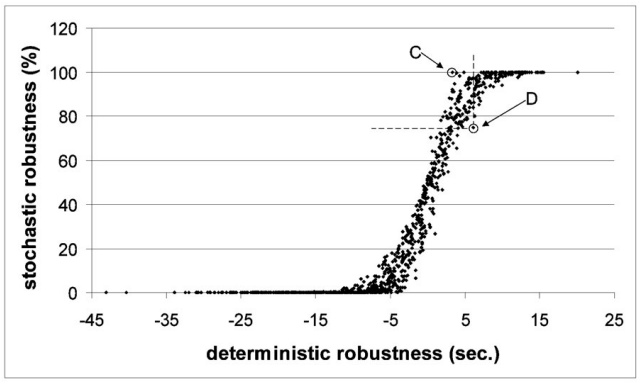
In Fig. 3(a), a comparison between the stochastic robustness metric and makespan is presented for 1000 randomly generated resource allocations. Fig. 3(a) demonstrates that, as can be expected, resource allocations that produce a very large makespan also tend to have a very small stochastic robustness metric value. However, as can also be seen in the figure, there can be a large discrepancy between the predicted performance as found using the expected makespan and the predicted performance using the stochastic robustness metric. For example, in the figure, compare the two resource allocations labeled A and B. If the comparison of these two resource allocations is made using the expected makespan, allocation A appears to be slightly superior to allocation B. However, resource allocation B presents a 99.8% probability of meeting the imposed QoS constraints, where as allocation A has only a 75% probability of meeting it. In this case, using only the expected makespan to compare the two resource allocations leads to a sizable increase in risk for a modest ($\approx 5\%$) improvement in the expected makespan. Any of the approximately 100 resource allocations above and to the right of allocation A, delineated by the dashed lines in the figure, will have a higher robustness value yet higher (worse) makespan value than A.

In Fig. 3(b), a comparison of the stochastic robustness metric and the deterministic robustness metric is presented for 1000 randomly generated resource allocations. In Fig. 3(b), compare the two resource allocations C and D. Based on using deterministic robustness measures as in [2], allocation D (with a deterministic measure of 6.13 sec.) is preferred over C (with a deterministic measure of 3.25 sec.). However, under the new stochastic model, allocation C (with a stochastic measure of 99.9%) is preferred over D (with a stochastic measure of 75%). Thus in this case, using only the deterministic robustness metric to select a resource allocation, D appears to be more robust than C. In contrast, the stochastic robustness metric, which accounts for the distribution of makespan outcomes, shows that allocation C has a 99.9% probability of meeting the QoS constraint while allocation D has only a 75% probability of meeting the QoS constraint.

Consider the sub-region identified in Fig. 3(b) with dot-



(a)



(b)

Figure 3. A plot of stochastic robustness metric versus (a) makespan and (b) deterministic robustness, for 1000 randomly generated resource allocations. The stochastic robustness metric values for allocations A and B exemplify the conflict between the stochastic robustness metric and makespan. Similarly, the stochastic robustness metric values for allocations C and D exemplify the conflict with the deterministic robustness metric.

ted lines originating from the point D, containing all of the points above and to the left of D. Each of the identified points in the sub-region has a higher stochastic robustness metric value than D but a lower deterministic robustness metric value than D.

The results also show a number of resource allocations that have a *negative* deterministic robustness value. For the data used in this simulation study, a negative value for the deterministic robustness correlates with a low stochastic robustness value.

It is shown in [2] that the deterministic robustness metric provides better information than just makespan. However, when execution time distributions are available, the stochastic robustness metric is even better.

Differences between the stochastic robustness metric and the deterministic robustness metric can be explained by the fact that the stochastic robustness metric uses information about the distribution of outcomes for the resource allocation to determine robustness. In contrast, the deterministic robustness metric uses a scalar estimate of each application's execution time on each machine to determine a resource allocation's robustness. In this study, there were a significant number of resource allocations where the stochastic robustness metric's use of the distribution of outcomes caused the metric to produce a robustness value for the allocation that failed to correlate well with the deterministic robustness metric. Thus, if the information needed for using the stochastic model is available, or can be obtained, then a better selection among resource allocations is possible.

4. Related Work

Prior work [2] in this area has referred to a resource allocation's tolerance to uncertainty as the robustness of that resource allocation. That work also defines a set of criteria for definitively claiming that a resource allocation is robust given a deterministic estimate for each considered system parameter. This determination of robustness begins by asking the claimant to define the behavior of the system that makes it robust, i.e., differentiate between acceptable performance and unacceptable performance of the system. Given this definition of acceptable performance, the uncertainty in system parameters must be identified along with its impact on the system's ability to deliver acceptable performance.

In [2], a four-step procedure is defined for deriving a deterministic robustness metric. The authors proposed procedure was used here to motivate the derivation of a stochastic robustness metric. According to [2], the first step in defining a robustness metric requires quantitatively describing what makes the system robust. This description establishes the required QoS level that must be delivered to refer to the system as robust—essentially bounding the acceptable variation in system performance. A pair of values, β_{min} and β_{max} that bound each performance feature must be identified, quantitatively defining the tolerable variation in each of the performance features.

In the second step, all modeled system and environmental parameters that may impact the system's ability to deliver acceptable QoS are identified. These parameters are referred to as the perturbation parameters of the system. In our new stochastic approach, each perturbation parameter, or uncertainty parameter, is modeled as a random variable fully described with a pmf. In this way, all possible values of the considered perturbation parameters, and their associated probabilities, are included in the calculation of the stochastic robustness metric. Our new approach differs from that in [2], where a single deterministic estimated value for each of the identified perturbation parameters is used.

In the third step, the impact of the identified perturbation parameters on the system's performance features is defined. This requires identifying a function that maps a given vector of perturbation parameters to a value for the performance feature of the system. Similarly in our new stochastic environment, this involves defining the functional dependence between the input random variables and the given performance feature. However, in our new model this involves more complex computations to combine random variables.

Finally, in the fourth step, the previously identified relation is evaluated to quantify the robustness. As a measure of robustness, the authors in [2] use the "minimum robustness radius" that relies on a deterministic performance characteristic. Furthermore, it assumes there is no *a priori* information available about the relative likelihood or magnitude of change for each perturbation parameter. Thus, the minimum robustness radius is used in a deterministic worst-case analysis. In our new stochastic model, more information regarding the variation in the perturbation parameters is assumed known. Representing the uncertainty parameters of the system as stochastic variables enables the robustness metric in the stochastic model to account for all possible outcomes for the performance of the system. This added knowledge comes at a computational cost. The stochastic robustness metric requires more information and is far more complex to calculate than its deterministic counterpart. To handle the computational complexity, we considered an approximation scheme that greatly simplifies the required calculations.

In [7], the robustness of a resource allocation is defined in terms of the schedule's ability to tolerate an increase in application execution time without increasing the total execution time of the resource allocation. In this formulation, the authors define a resource allocation's robustness in terms of system slack thereby focusing their metric on a single very important uncertainty parameter, i.e., variations in

application execution times. Our stochastic robustness metric is more generally applicable, allowing for any definition of QoS and able to incorporate any identified uncertainty parameters.

Our presented methodology relies heavily on an ability to model the uncertainty parameters as stochastic variables. Several previous efforts have established a variety of techniques for modeling the stochastic behavior of application execution times [6, 10, 20]. In [6], three methods for obtaining probability distributions for task execution times are presented. The authors also present a means for combining stochastic task representations to determine task completion time distributions. Our work leverages this method of combining independent task execution time distributions and extends it by defining a means for measuring the robustness of a resource allocation against an expressed set of QoS constraints under uncertainty.

In [15], a statistical algorithm for predicting task execution times is presented. The authors present a methodology for defining data driven estimates of uncertainty parameters in a heterogeneous computing environment. In that work, the method is applied to the problem of generating an application execution time prediction given a set of observations of that application's execution times. Their model defines an application execution time random variable as the combination of two elements. The first element corresponds to a vector of known factors that have an impact on the execution time of the application and is considered to be a deterministic component of the execution time random variable. A second element accounts for all unmodeled factors that may impact the execution time of an application and represents the stochastic component of the execution time approximation. This method for predicting application execution times can be used to determine probability density functions describing the input random variables in our framework.

In [11], the authors present a derivation of the makespan problem that relies on a stochastic representation of task execution times. The authors also demonstrate that their presented stochastic approach to scheduling can significantly reduce the actual simulated system makespan as compared to some well known scheduling heuristics that are founded in a deterministic approach to modeling task execution times. The heuristics presented in that study were adapted to the stochastic model and used to minimize the expected system makespan given a stochastic model of task execution times. In our research, the emphasis is on quantitatively comparing one resource allocation to another by deriving a metric for the resource allocation's robustness, i.e., the probability to deliver on expressed QoS constraints. Thus, [11] is focused on designing a heuristic for the makespan problem in a stochastic environment, while this paper is focused on the evaluation of the robustness of a resource allocation given a modeled stochastic environment.

5. Conclusion

This paper presents a stochastic robustness metric suitable for evaluating the likelihood that a resource allocation will perform acceptably, i.e., satisfy identified QoS constraints, in an uncertain environment. In addition to the general statement of the stochastic robustness metric, the derivation, mathematical description, and computational methods of the stochastic robustness metric were also presented. The stochastic robustness metric was then applied to an example class of systems operating with periodic data sets to demonstrate its utility in evaluating the robustness of a resource allocation.

Given the raw volume of computation required to evaluate such a stochastic metric, a developed approximation scheme based on the bootstrap technique and the FFT method were tested to aid the practitioner in the actual application of the metric in different real world scenarios. A conducted simulation study demonstrates the accuracy of the bootstrap approximation and a baseline timing analysis for FFT.

There are many ways that this research on stochastic robustness may be built upon in future work. The results of this work can be leveraged to develop methods for calculating the stochastic robustness metric given system parameters that include dependencies, as was discussed earlier. Another extension of this research involves applying the stochastic robustness metric to resource allocations in a dynamic environment, where the mix of tasks to be executed is not known in advance and system feedback about completed tasks is available. This research also can be applied to the design of resource allocation techniques that utilize the stochastic robustness metric to generate allocations that are more robust [23].

References

[1] S. Ali, T. D. Braun, H. J. Siegel, A. A. Maciejewski, N. Beck, L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao. Parallel, Distributed, and Pervasive Computing, chapter Characterizing resource allocation heuristics for heterogeneous computing systems, pages 91–128. Advances in Computers. Elsevier, Amsterdam, The Netherlands, 2005.

[2] 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, 15(7):630–641, July 2004.

[3] M. F. Arlitt and C. L. Williamson. Internet Web servers: Workload characterization and performance implications. IEEE/ACM Transactions on Networking, 5(5):631–645, Oct. 1997.

[4] Adaptive and Reflective Middleware Systems (ARMS). accessed Dec. 10 2005.

[5] L. A. Barroso, J. Dean, and U. Holzle. Web search for a planet: The Google cluster architecture. Micro IEEE, 23(2):22–28, Apr. 2003.

[6] G. Bernat, A. Colin, and S. M. Peters. WCET analysis of probabilistic hard real-time systems. In Proceedings 23rd IEEE Real-Time Systems Symposium (RTSS '02), 2002.

[7] L. Bölöni and D. Marinescu. Robust scheduling of metaprograms. Journal of Scheduling, 5(5):395–412, Sept. 2002.

[8] 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, 61(6):810–837, June 2001.

[9] E. G. Coffman, editor. Computer and Job-Shop Scheduling Theory. John Wiley & Sons, New York, NY, 1976.

[10] L. David and I. Puaut. Static determination of probabilistic execution times. In Proceedings 16th Euromicro Conference on Real-Time Systems (ECRTS '04), June 2004.

[11] A. Dogan and F. Ozguner. Genetic algorithm based scheduling of meta-tasks with stochastic execution times in heterogeneous computing systems. Cluster Computing, 7(2):177–190, Apr. 2004.

[12] M. M. Eshaghian, editor. Heterogeneous Computing. Artech House, Norwood, MA, 1996.

[13] D. Fernandez-Baca. Allocating modules to processors in a distributed system. IEEE Transactions on Software Engineering, 15(11):1427–1436, Nov. 1989.

[14] O. H. Ibarra and C. E. Kim. Heuristic algorithms for scheduling independent tasks on non-identical processors. Journal of the ACM, 24(2):280–289, Apr. 1977.

[15] M. A. Iverson, F. Ozguner, and L. Potter. Statistical prediction of task execution times through analytical benchmarking for scheduling in a heterogeneous environment. IEEE Transactions on Computers, 48(12):1374–1379, Dec. 1999.

[16] J.-K. Kim, D. A. Hensgen, T. Kidd, H. J. Siegel, D. S. John, C. Irvine, T. Levin, N. W. Porter, V. K. Prasanna, and R. F. Freund. A flexible multi-dimensional qos performance measure framework for distributed heterogeneous systems. Cluster Computing, 6(3), July 2006. Scheduled to appear.

[17] A. Kumar and R. Shorey. Performance analysis and scheduling of stochastic fork-join jobs in a multicomputer system. IEEE Transactions on Parallel and Distributed Systems, 4(10), Oct. 1993.

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

[19] A. Leon-Garcia. Probability & Random Processes for Electrical Engineering. Addison Wesley, Reading, MA, 1989.

[20] Y. A. Li, J. K. Antonio, H. J. Siegel, M. Tan, and D. W. Watson. Determining the execution time distribution for a data parallel program in a heterogeneous computing environment. Journal of Parallel and Distributed Computing, 44(1):35–52, July 1997.

[21] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. Journal of Parallel and Distributed Computing, 59(2):107–121, Nov. 1999.

[22] C. L. Phillips, J. M. Parr, and E. A. Riskin. Signals, Systems, and Transforms. Pearson Education, Upper Saddle River, NJ, 2003.

[23] V. Shestak, J. Smith, R. Umland, J. Hale, P. Moranville, A. A. Maciejewski, and H. J. Siegel. Greedy approaches to static stochastic robust resource allocation for periodic sensor driven distributed systems. In Proceedings the 2006 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'06), 2006. Accepted, to appear.

[24] 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, 47(1):8–22, Nov. 1997.

[25] L. Wasserman. All of Statistics: A Concise Course in Statistical Inference. Springer Science+Business Media, New York, NY, 2005.