A Probabilistic Approach to Measuring Robustness in Computing Systems

Behdis Eslamnour and Shoukat Ali

University of Missouri-Rolla
Dept. of Electrical and Computer Engineering
Rolla, MO 65409-0040 USA
{ben88, shoukat}@umr.edu

Abstract

System builders are becoming increasingly interested in robust design. We believe that a methodology for generating robustness metrics will help the robust design research efforts and, in general, is an important step in the efforts to create robust computing systems. The purpose of the research in this paper is to quantify the robustness of a resource allocation, with the eventual objective of setting a standard that could easily be instantiated for a particular computing system to generate a robustness metric. We present our theoretical foundation for a robustness metric and give its instantiation for a particular system.

1 Introduction

System builders are becoming increasingly interested in robust design. The following are some examples of this interest. The goal of the *DOE's SciDAC (Scientific Discovery through Advanced Computing) program's Scalable Systems Software project* is "to fundamentally change the way future high-end systems software is developed to make it more cost effective and robust" [17]. One goal of DARPA's "High Productivity Computing Systems" program is to substantially improve robustness and reliability of computing systems [1]. 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 (including IBM and Raytheon). *We believe that a methodology for **generating robustness metrics** will help all of the research efforts given above, and, in general, is an important step in the efforts to create robust computing and communications systems.*

To date, no major research effort has addressed the science of design of a robust resource management system in a comprehensive, systematic manner. We believe that it is not sufficient to speak of developing yet another resource management system. Instead, such systems need to be "designed" for robustness. This research contributes to the science underlying the design of a robust resource management system.

A resource allocation is defined to be robust with respect to specified system performance features against perturbations in specified system parameters if degradation in these features is limited when the perturbations occur within a certain range. The high link and node counts of some high-end systems (many with more than 1000 processors) introduce a complexity that makes robustness of these systems a critical and challenging issue. This research will be directly helpful to such programs by providing theoretical underpinnings for robustness, and by developing a methodology for generating robustness metrics for a variety of systems.

The purpose of the research in this paper is to quantify the robustness of a resource allocation, with the eventual objective of setting a standard that could easily be instantiated for a particular computing system to generate a robustness metric.

The rest of the paper is organized as follows. A summary of some related work is given in Section 2. Section 3 presents our theoretical foundation for a robustness metric, and a formulation of the theoretical framework for a particular example of heterogeneous computing system. Some simulation experiments and their results are discussed in Section 4. Section 5 concludes this paper.

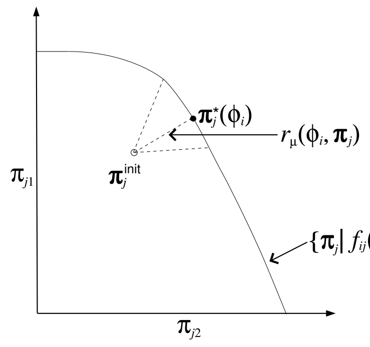
2 Related Work

Reliability, dependability and fault-tolerance of distributed systems have been addressed in several works [5, 6, 7, 10, 11, 13, 14, 15, 19, 20]. [5] presents a bi-criteria scheduling heuristic for distributed systems according to two criteria, minimization of the schedule length and maximization of the system reliability. [6] gives a definition of reliable broadcast and presents a hybrid model for asynchronous distributed systems with Byzantine faults, crashes and recoveries. [7] proposes a methodology for quantifying the effect of denial of service (DoS) attacks on a distributed system. In [10], an accrual failure detector is presented which assigns a real value (a suspicion level of failure) to each application, instead of the traditional binary information (trust vs. suspect). As defined in this paper, the suspicion level of a "faulty" process monotonically increases, while the suspicion level of a "correct" process is bounded. Accrual failure detectors can serve multiple applications with different quality of service requirements. In [11], given some distribution constraints, some indications on application execution times and link communication times, a number of failures that the system must tolerate, and some real-time constraints, a fault-tolerant distributed static scheduling heuristic is produced that schedules a source algorithm on the target architecture. [13] introduces a resource location and discovery algorithm, MPIL, for distributed systems which is both perturbation-resistant and overlay-independent. Probabilistic QoS guarantees for supercomputing systems are proposed in [14]. The proposed system enables the system and users to negotiate a mutually desirable risk strategy, and makes probabilistic guarantees on QoS such that "job j can be completed by deadline d with probability p." A metric for QoS is presented in this paper, and using this metric, the scheduler tries to maximize the QoS. In [15], the reliability of embedded computer-based systems is addressed by presenting a modular technique for evaluating sensitivity for dynamic fault trees. To improve dependability in distributed embedded systems, [19] proposes the alternative functionality mechanism, in which a lost feature is replaced with another existing function that can substitute for the lost service. [20] proposes a fault detecting protocol for duplicated processes and enhances a roll forward recovery scheme.

3 Probabilistic Robustness Metrics

Overview: One way of arriving at a robustness metric [2] is to start with the quantitative description of the requirement that makes the system robust. Based on this *robustness requirement*, one can determine the QoS performance features that should be limited in variation to ensure that the robustness requirement is met. Let ϕ_i be one certain performance feature, and let the bounds of the tolerable variation in ϕ_i be given by $\langle \beta_i^{\min}, \beta_i^{\max} \rangle$. Then one can identify all of the system and environment parameters (called the perturbation parameters) whose values may impact the value of the performance feature ϕ_i. Mathematically, let Π be the set of perturbation parameters. It is assumed that the elements of Π are vectors. Let $\boldsymbol{\pi}_j$ be the j-th element of Π. In the next step, one needs to determine, for every $\phi_i \in \Phi$, the relationship $\phi_i = f_{ij}(\boldsymbol{\pi}_j)$, if any, that relates ϕ_i to $\boldsymbol{\pi}_j$. In this expression, f_{ij} is a function that maps $\boldsymbol{\pi}_j$ to ϕ_i. Let $r_\mu(\phi_i,\ \boldsymbol{\pi}_j)$ be defined as the robustness of resource allocation μ with respect to the performance feature ϕ_i against the perturbation parameter $\boldsymbol{\pi}_j$, and is given as the smallest collective variation in the values of perturbation parameters that will cause the performance feature ϕ_i to violate its acceptable variation. This will be the degree of robustness of the given resource allocation. Figure 1 sums up this discussion. Mathematically, the robustness metric defined above is calculated as following [2].

$$r_\mu(\phi_i,\ \boldsymbol{\pi}_j) = \min_{\boldsymbol{\pi}_j:\,(f_{ij}(\boldsymbol{\pi}_j)=\beta_i^{\max})\vee(f_{ij}(\boldsymbol{\pi}_j)=\beta_i^{\min})} \|\boldsymbol{\pi}_j - \boldsymbol{\pi}_j^{\text{init}}\|_2. \tag{1}$$



$\boldsymbol{\pi}_j^*(\phi_i)$
$r_\mu(\phi_i, \boldsymbol{\pi}_j)$
$\boldsymbol{\pi}_j^{\text{init}}$
π_{j1}
$\{\boldsymbol{\pi}_j | f_{ij}(\boldsymbol{\pi}_j) = \beta_i^{\max}\}$
π_{j2}

Figure 1. Some possible directions of increase of the perturbation parameter π_j, and the direction of the smallest increase. The curve plots the set of points, $\{\pi_j |\ f_{ij}(\pi_j) = \beta_i^{\max}\}$. The set of boundary points, $\{\pi_j |\ f_{ij}(\pi_j) = \beta_i^{\min}\}$ is given by the points on the π_{j1}-axis and π_{j2}-axis.

With the robustness definition in Equation 1, all points with the same distance from $\boldsymbol{\pi}_j^{\text{init}}$ are located on a spheroid. This means that all components of an observation $\boldsymbol{\pi}$ contribute equally to the Euclidean distance of $\boldsymbol{\pi}$ from $\boldsymbol{\pi}_j^{\text{init}}$. In addition, it assumes that $\boldsymbol{\pi}_j$ components are not correlated. However, in real world the variables (i.e., the elements of each perturbation vector) can have different variances and be correlated. In such cases, the smallest distance to the violation boundary may not be as shown in Figure 1. Consider a point $C(x_0, y_0)$ whose distance from a boundary b_1 is required. Geometrically, the Euclidean distance form point C to boundary b_1 is the radius of the largest circle that can be drawn such that it is centered on C and just touches the boundary. All points on this circle have the same Euclidean distance from C. Now consider the case where the variables x and y have different variances; for example, variable x has a higher variance than that of y. In that case, the shortest distance to the boundary will actually have a greater component of x than that of y. This converts the circle to an ellipse, Such an ellipse is centered on C and touches the boundary, and the ratio of the principal axes of the ellipse is the same as the ratio of the standard deviations of x and y. So the robustness is now the distance from the center of the ellipse to the point where it touches the boundary instead of the Euclidean distance. Any boundary that is tangential to the ellipse is actually at the same probabilistic distance from C.

Going one step further, if correlation between x and y is considered, the ellipse mentioned above would rotate according to the correlation. In fact when x and y are positively correlated, the probabilistic distance of C from b_1 is smaller than that when they are negatively correlated.

Incorporating Parameters Variances: To account for different variances of the components, a scaling with respect to each component's standard deviation can be applied. Assume two points $\mathbf{x} = [x_1 \quad x_2 \cdots x_n]$ and $\mathbf{y} = [y_1 \quad y_2 \cdots y_n]$ in an n-dimensional space. Let $\sigma_1, \sigma_2, \cdots, \sigma_n$ be the standard deviations for the components of each dimension, respectively. The scaled vectors would be $\mathbf{u} = [\frac{x_1}{\sigma_1}, \cdots, \frac{x_n}{\sigma_n}]$ and $\mathbf{v} = [\frac{y_1}{\sigma_1}, \cdots, \frac{y_n}{\sigma_n}]$. The "variance-aware" distance between $\mathbf{x}$ and $\mathbf{y}$ would be $d_{\text{var}} = \sqrt{(\frac{x_1 - y_1}{\sigma_1})^2 + \cdots + (\frac{x_n - y_n}{\sigma_n})^2}$. The subscript "var" stands for variance. It can be noticed that d_{var} is dimensionless because x_i, y_i and σ_i have same units. In general, let scaling matrix $\mathbf{D}$ be a diagonal matrix such that $\mathbf{D} = \text{diag}(\sigma_1^2, \cdots, \sigma_n^2)$. Then, d_{var} can be rewritten as $\sqrt{(\mathbf{x}-\mathbf{y})\mathbf{D}^{-1}(\mathbf{x}-\mathbf{y})^{\text{T}}}$, where $(\mathbf{x}-\mathbf{y})^{\mathbf{T}}$ is the vector transpose of $(\mathbf{x}-\mathbf{y})$.

With this definition, perturbation parameters with higher variances receive smaller weights [21] in the robustness calculations, i.e., for the perturbation parameters with higher variances, the distance to the constraint surface would be smaller. This makes sense as less certain a measurement, the more conservative should be the estimate about how far it is from a boundary surface. The points with the same distance d_{var} from $\boldsymbol{\pi}_j^{\text{init}}$ will be located on an ellipsoid. The axes of this ellipsoid are the same as the spheroid's axes - however they are scaled by the standard deviations of the variables. Using d_{var} instead of Euclidean distance in Equation 1, the robustness $r_\mu(\phi_i, \; \boldsymbol{\pi}_j)$ would be defined as

$$\min_{\substack{\boldsymbol{\pi}_j:\,(f_{ij}(\boldsymbol{\pi}_j)=\beta_i^{\max})\vee \\ (f_{ij}(\boldsymbol{\pi}_j)=\beta_i^{\min})}} \sqrt{(\boldsymbol{\pi}_j - \boldsymbol{\pi}_j^{\text{init}})\mathbf{D}^{-1}(\boldsymbol{\pi}_j - \boldsymbol{\pi}_j^{\text{init}})^{\mathbf{T}}}. \quad (2)$$

Incorporating Parameters Correlations: Correlation means that there are associations between the variables. These associations will cause a rotation in the ellipsoid mentioned above. The more correlated the variables are, the larger will be the rotation of the ellipsoid.

We propose to use the Mahalanobis distance [21, 4] instead of Euclidean distance to take into account both the variances of variables and the correlations between variables. Let

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \cdots & \sigma_{1n} \\ \vdots & \ddots & \vdots \\ \sigma_{n1} & \cdots & \sigma_n^2 \end{bmatrix}, \quad (3)$$

where σ_{ij} is the covariance of the ith and jth variables. Then, Mahalanobis distance between two points $\mathbf{x}$ and $\mathbf{y}$, where $\boldsymbol{\Sigma}$ is the covariance matrix of the variables, equals $\sqrt{(\mathbf{x}-\mathbf{y})\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\mathbf{y})^{\mathbf{T}}}$. Using Mahalanobis distance instead of Euclidean distance in Equation 1, the robustness $r_\mu(\phi_i, \; \boldsymbol{\pi}_j)$ would be defined as

$$\min_{\substack{\boldsymbol{\pi}_j:\,(f_{ij}(\boldsymbol{\pi}_j)=\beta_i^{\max})\vee \\ (f_{ij}(\boldsymbol{\pi}_j)=\beta_i^{\min})}} \sqrt{(\boldsymbol{\pi}_j - \boldsymbol{\pi}_j^{\text{init}})\boldsymbol{\Sigma}^{-1}(\boldsymbol{\pi}_j - \boldsymbol{\pi}_j^{\text{init}})^{\mathbf{T}}}. \quad (4)$$

With this new definition of the robustness, both the variance of each perturbation parameter and the correlation between perturbation parameters are taken into account. Note that the scaling part of the probabilistic approach eliminates the dimension.

Using Mahalanobis distance instead of Euclidean distance is valid as long as the underlying parameters follow a normal distribution. According to the central limit theorem (simplified form) a random variable will be normally distributed if it is the sum of many small, independent "disturbances." This (multivariate normality) is a very good assumption in most situations, and in fact underlies the use of linear models throughout all of statistics. The assumption will not apply in cases where it is patently known to be false, e.g., categorical random variables.

Instantiation Of the Theoretical Metric: In this section, the robustness of a particular computing system is investigated, where a set of independent applications are mapped onto heterogeneous computing machines. By independent it is meant that there is no communication between the applications; yet they can be correlated. One performance measure of such a computing system is makespan, which is the completion time for the entire set of applications. Assume that in this example system, it is required that the makespan be robust against errors in task execution time estimates.

A brief description of the system model is now given. A set $\mathcal{A}$ of applications is to be mapped onto a set $\mathcal{M}$ of machines so as to minimize the makespan. Each machine executes one application at a time. Let C_i^{init} be the estimated time to compute for application a_i on the machine where it is mapped. Let C_i be equal to the actual computation time value (C_i^{init} plus the estimation error). It is assumed that C_i^{init} values are known for all i. Let $\boldsymbol{C}$ be the vector of the C_i values such that $\boldsymbol{C} = [C_1 \quad C_2 \cdots C_{|\mathcal{A}|}]$. Similarly, $\boldsymbol{C}^{\text{init}} = [C_1^{\text{init}} \quad C_2^{\text{init}} \cdots C_{|\mathcal{A}|}^{\text{init}}]$. The vector $\boldsymbol{C}$ is the perturbation parameter for this analysis. Also let F_j be the time at which m_j finishes executing all of the applications mapped on it. Now the performance feature F_j can be expressed as a function of perturbation parameters (i.e., computation times - in this example) as $F_j(\boldsymbol{C}) = \sum\limits_{i:\, a_i \text{ is mapped to } m_j} C_i$.

Assuming that there is only one boundary limit, $\beta_i^{\max}$, to the makespan, the Mahalanobis robustness of F_j against $\boldsymbol{C}$ can be derived from Equation 4.

$$r_\mu(F_j,\ \boldsymbol{C}) = \min_{\boldsymbol{C}:\, F_j(\boldsymbol{C}) = \beta_i^{\max}} \sqrt{(\boldsymbol{C} - \boldsymbol{C}^{\text{init}})\boldsymbol{\Sigma}^{-1}(\boldsymbol{C} - \boldsymbol{C}^{\text{init}})^{\mathbf{T}}}. \tag{5}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of the perturbation parameter vector, $\boldsymbol{C}$.

The right hand side of Equation 5 can be interpreted as the Mahalanobis distance from the point $\boldsymbol{C}^{\text{init}}$ to the constraint plane $\mathbf{p} : F_j(\boldsymbol{C}) = \beta_i^{\max}$. Plane $\mathbf{p}$ can be expressed as vector $\mathbf{p} = [1 \ 1 \ \cdots 1 \ \beta_i^{\max}]$ in linear space. Note that vector $\mathbf{p}$ has $n+1$ elements, where n is the number of applications allocated to m_j. Mahalanobis distance from a given point $\boldsymbol{C}^{\text{init}}$ to plane $\mathbf{p}$ can be calculated as following [16]. Determine the singular value decomposition (SVD) of the symmetric covariance matrix $\boldsymbol{\Sigma}$, i.e., determine $\mathbf{R}$ and $\mathbf{D}$ such that $\boldsymbol{\Sigma} = \mathbf{R}^{\mathrm{T}}\mathbf{D}\mathbf{R}$. The matrix $\mathbf{D}$ is the component that transforms the spheroid into an ellipsoid and the matrix $\mathbf{R}$ is the component that causes the ellipsoid to rotate. Recall that $\mathbf{D} = \text{diag}(\sigma_1^2, \cdots, \sigma_n^2)$ is a diagonal matrix with $\sigma_1^2, \cdots, \sigma_n^2$ as its diagonal elements. Let $\mathbf{A}_h$ be a diagonal matrix such that $\mathbf{A}_h = \text{diag}(\sigma_1, \cdots, \sigma_n, 1)$. Also let

$$\mathbf{R}_h = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{bmatrix}, \text{and } \mathbf{T}_h = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\boldsymbol{C}^{\text{init}} & 1 \end{bmatrix},$$

where $\mathbf{0}$ is a column vector $[0 \ 0 \ 0 \cdots 0]^{\mathrm{T}}$ with a length of n.

Then, plane $\mathbf{q}$ is the transformation of plane $\mathbf{p}$ such that $\mathbf{q} = \mathbf{A}_h\mathbf{R}_h\mathbf{T}_h\mathbf{p}$. Note that in n-dimensional space both $\mathbf{p}$ and $\mathbf{q}$ have $n+1$ elements. Then the probabilistic robustness d_{M}, is the Mahalanobis distance from point $\boldsymbol{C}^{\text{init}}$ to plane $\mathbf{p}$. That is,

$$r_\mu(F_j, \boldsymbol{C}) = d_{\mathrm{M}}(\boldsymbol{C}^{\text{init}}) = \frac{q_{n+1}}{\sqrt{\sum_{i=1}^{n} q_i^2}}. \tag{6}$$

Let $r_\mu(\boldsymbol{C})$ be the overall robustness of the system. It is simply the minimum taken over all machines.

$$r_\mu(\boldsymbol{C}) = \min_j r_\mu(F_j, \boldsymbol{C}) \tag{7}$$

Note that for this formulation, the performance feature must be a linear function of the perturbation parameters. Also plane $\mathbf{q}$ is the transformation of plane $\mathbf{p}$ as described above, and q_{n+1} is the transformed $\beta_i^{\max}$.

4 Simulation Experiments

Overview: Experiments were performed for a system with five machines and 20 applications. A total of 500 resource allocations were generated by assigning a randomly chosen machine to each application, and then these resource allocations were evaluated for the makespan, load balance index, Euclidean robustness and Mahalanobis robustness.

The system was characterized by using the expected execution times of the applications on the different machines present in the system. This information is arranged in an "expected time to compute" (ETC) matrix as a model of the given system, where the entry(i, j) is the expected execution time of application i on machine j. The ETC matrix is used to express the heterogeneity among the estimated application computation times, and among the machines in the system. The variation along a row (as measured by the ratio of standard deviation to mean) is referred to as the machine heterogeneity; this is the degree to which the machine execution times vary for a given application. A system's machine heterogeneity is based on a combination of the machine heterogeneities for all tasks (rows). Similarly, the variation along a column of an ETC matrix is referred to as the task heterogeneity; this is the degree to which the application execution times vary for a given machine. A system's task heterogeneity is based on a combination of the task heterogeneities for all machines (columns).

The ETC values for tasks were generated by sampling a Gamma distribution [2]. For all experiments, the mean was arbitrarily set to 10. Machine heterogeneity and task heterogeneity of the system were selected to be 0.7 and 0.7, respectively. In this study, the heterogeneity of a set of numbers was measured by the ratio of standard deviation to mean. A detailed description of the method used for generating data with given values of the mean and heterogeneity can be found in [3]. Thirty different sets of ETC values were randomly generated using this method.

In all experiments, the same set of 500 resource allocations was used as well as the same value of $\beta_i^{\max}$. The value of $\beta_i^{\max}$ was set to the maximum makespan in these experiments (224 time units) found among all resource allocations in all experiments. Note that a robustness value x of a resource allocation means that the resource allocation can tolerate any combination of ETC errors without the makespan exceeding $\beta_i^{\max}$ as long as the norm (Euclidean or Mahalanobis) of the errors is less than x.

Generation of Correlation Matrices with Respect to Condition Number: To examine the Mahalanobis robustness, covariance matrix of the perturbation parameters is needed. Different methods have been developed for generating random covariance and correlation matrices (e.g., [9, 12]). We used a method based on the condition number, κ, of the correlation matrix for generating the correlation matrix. Condition number of a matrix is defined as the ratio of its maximum eigenvalue to its minimum eigenvalue [8]. That is, $\kappa = \frac{\lambda_{\max}}{\lambda_{\min}}$.

A condition number of 1 means that there is no correlation between the elements of the correlation matrix [8]. As the condition number of the correlation matrix increases, the correlation between the variables increases as well. Therefore, by choosing different values for the condition number and generating the correlation matrix based on the condition number, the range of the correlations can be controlled. To generate a correlation matrix with respect to condition number the following steps were done.
1) A set of eigenvalues λ_i's was randomly generated while satisfying the constraints $\lambda_1 + \lambda_2 + \cdots + \lambda_n = n$ and $\kappa = \frac{\lambda_n}{\lambda_1}$, where $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, and n is the number of eigenvalues.
2) Using the eigenvectors generated in the previous step and the "gallery" function of MATLAB®, a random correlation matrix R is generated.

$$\mathbf{R} = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1n} \\ \rho_{21} & 1 & \cdots & \rho_{2n} \\ \vdots & \vdots & & \vdots \\ \rho_{n1} & \rho_{n2} & \cdots & 1 \end{bmatrix}, \quad (8)$$

where ρ_{ij} is the correlation between two random variables x_i and x_j, and $\rho_{ij} = \rho_{ji}$. It must be noted that for κ equal to 1, ρ_{ij}'s would be equal to zero, and as κ increases, the absolute value of ρ_{ij}'s increases as well, and both positive and negative values of correlation coefficients can be found in the generated correlation matrix.

The next step after generating the correlation matrix was to generate the covariance matrix. Then, provided every σ_i^2 is nonzero, $\sigma_{ij} = \sigma_i \sigma_j \rho_{ij}$ [18]. The covariance matrix, $\mathbf{\Sigma}$, can be expressed as

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1^2 & \sigma_1\sigma_2\rho_{12} & \cdots & \sigma_1\sigma_n\rho_{1n} \\ \sigma_2\sigma_1\rho_{21} & \sigma_2^2 & \cdots & \sigma_2\sigma_n\rho_{2n} \\ \vdots & \vdots & & \vdots \\ \sigma_n\sigma_1\rho_{n1} & \sigma_n\sigma_2\rho_{n2} & \cdots & \sigma_2^2 \end{bmatrix}. \quad (9)$$

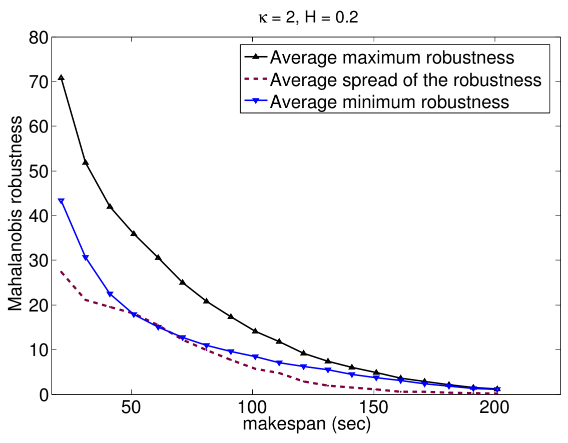
Correlation matrices were generated for two values of condition number, 2 and 500, to represent cases of low and high correlations. For each value of condition number, 30 different correlation matrices were randomly generated.

The standard deviations of the computation time values were generated by setting σ_i to HC_i^{init}, i.e., a coefficient of variation equal to H was assumed for the distribution of each C_i value on each machine. In our simulations, H was set to two values of 0.2 and 0.8 to represent low and high variation, respectively.

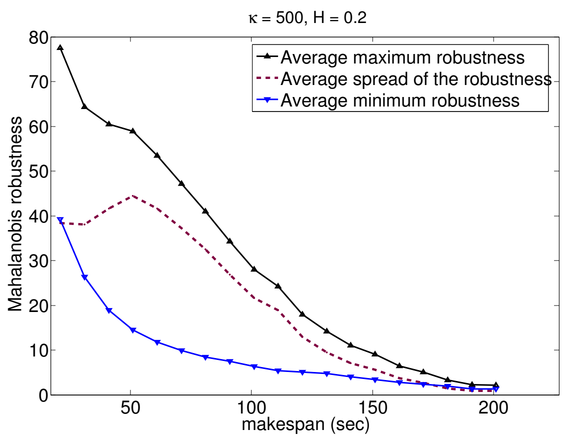
Results: Simulations were performed for two values of κ and two values of H. Within each category, 30 ETC matrices were used and for each ETC matrix, 30 different correlation matrices were generated with respect to the desired condition number. Simulations were done for each ETC-correlation matrix using the same set of 500 mappings. Therefore, 900 simulations were performed for each category and resource allocations were evaluated for their makespan, load balance index and Mahalanobis robustness. Having the results of the simulations, the makespan range was divided into 10-sec bins, then the minimum robustness, maximum robustness and the robustness spread (maximum robustness $-$ minimum robustness) within each bin was recorded for each simulation. Then the average of these three values over 900 simulations of each category were calculated for each bin. Figures 2(a) and 2(b) show the maximum robustness, minimum robustness and the spread of robustness for the 10-sec bins for four categories of κ and H.

Figures 2(a) shows that as the makespan of resource allocations decreases, Mahalanobis robustness and its spread increases. Comparing Figure 2(a) to 2(b), it can be seen that when the computation times are more correlated, the spread of robustness for the resource allocations with the same makespan is even larger. One can see that just by interpreting the makespan measure of a resource allocation, it is not possible to predict its robustness. Makespan cannot serve as a good indicator of robustness and resource allocations can be found that have the same makespan but very different robustness. Figures 2(a) and 2(b) show the results for when $H = 0.2$. Results for $H = 0.8$ were similar.

Two such resource allocations have been shown in detail in the two bar charts in Figure 3. It also shows the robustness results for one experiment. Each "*" represents one of the 500 resource allocations. Each colored box shows one application, and each vertical bar shows one machine. One can see that makespans are nearly identical but resource allocation B has a much worse robustness. Incidentally, resource allocation B also has a much worse load balance (ratio of minimum finishing time to the maximum finishing time among all machines). One could argue that resource allocation B ended up with bad robustness because it had such bad load balance. However, this research shows that even load balance and makespan taken together cannot pre-



(a)



(b)

Figure 2. Comparison of the probabilistic and non-probabilistic distances of point C from boundary b_1.

dict the robustness of a resource allocation. Figure 4 illustrates this fact.

In Figure 4 two different resource allocations, C and D, are shown. These allocations have very similar makespan ($\cong 60$) and very similar load balance values ($\cong 0.45$). However, C and D are very different in their robustness values (27.04 and 15.26, respectively). Therefore, one can see that even by considering both makespan and load balance measures of a resource allocation at the same time, one may not be able to predict its robustness. This paper contends one needs an explicit measure of robustness like the one proposed here.

Actually, using load balance and makespan to predict robustness could be quite misleading. For example, by observing the load balance charts for resource allocations E and F, shown in Figure 5, one is likely to select resource allocation F because it is definitely very well-balanced in load, and has a better makespan. However, Figure 5 also shows that resource allocation E is much better in its robustness value.

We have used Figures 3, 4, and 5 to illustrate how either makespan or load balance or both can fail to be predictors of robustness. What causes an improved robustness in each of Figures 3, 4, and 5? Let robustness machine be the machine that determines the minimum in Equation 7. Then, here are some observations.

(1). For resource allocation A in Figure 3, the robustness machine has five applications on it, with two pairs of negatively correlated applications. In contrast, for resource allocation B in Figure 3, the robustness machine has seven applications on it, and all of them are positively correlated.

(2). For resource allocation C in Figure 4, the robustness machine has six applications on it, with three pairs of negatively correlated applications. In contrast, for resource allocation D in Figure 4, the robustness machine has five applications on it, and all of them are positively correlated.

(3). For resource allocation E in Figure 5, the robustness machine has nine applications on it, with three pairs of negatively correlated applications. In contrast, for resource allocation F in Figure 5, the robustness machine has three applications on it, and all of them are positively correlated.

It can be see that for each pair of allocations in Figures 3, 4, and 5, the less robust allocation is the one that has all positively correlated applications on it. A less obvious fact is that, in the absence of correlated applications, the larger the number of applications on the robustness machine, the larger the robustness. Our intuitive explanation is that, given two resource allocations with the same makespan, if there is a larger number of applications on the last to finish machine, the elements of the diagonal matrix in Equation 2 are smaller, and therefore the robustness is larger. Recall that, in this study, the heterogeneity of execution times was given by the ratio of standard deviation to mean. So the larger the execution time of a task, the larger its contribution towards the variance. A detailed mathematical and statistical investigation of this observation will be carried out in a future research effort.

5 Conclusions

Parallel and distributed heterogeneous computing and communication systems may operate in dynamic environments that undergo unpredictable changes causing certain system performance features to degrade. Such systems need robustness to guarantee limited degradation despite fluctuations in the behavior of their component parts or environment. In this paper, the research was focused on a prob-

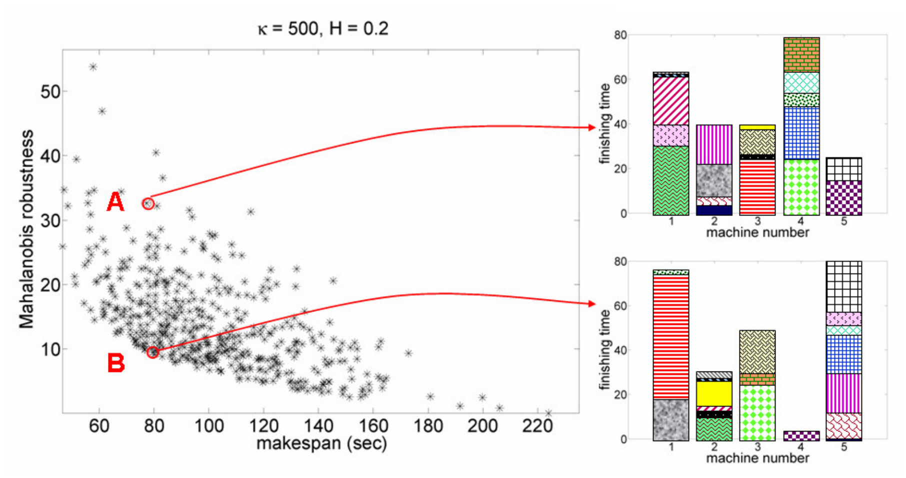


Figure 3. Mahalanobis robustness (with respect to makespan), where correlation is high (κ = 500). Each * represents one of the 500 randomly chosen resource allocations.

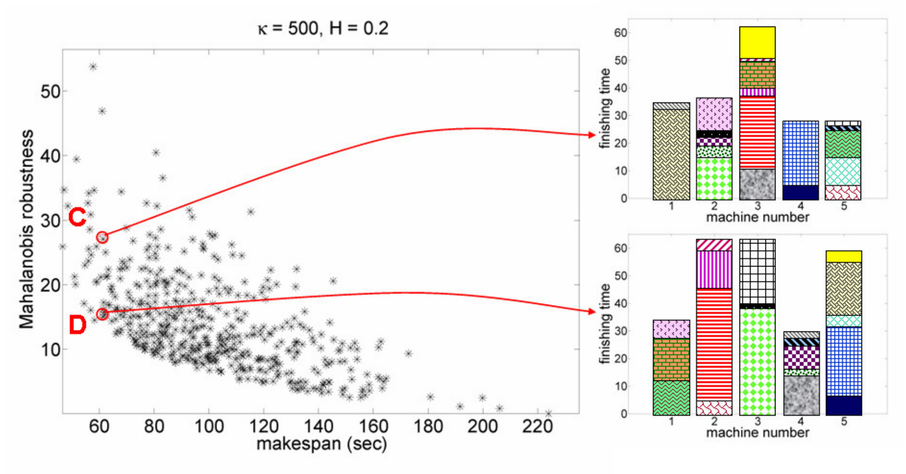


Figure 4. Mahalanobis robustness (with respect to makespan), where correlation is high (κ = 500).

abilistic approach to measuring robustness in distributed computing systems. In this approach, variance and correlation of the parameters were taken into account. Robustness evaluations for several configurations of an example system were performed, where a set of independent applications were mapped onto a set of heterogeneous computing machines. The results show that the Mahalanobis robustness measure offers more insight than the Euclidean measure, and two other popular measures, makespan and load balance index.

For systems in which the computation time values are highly correlated, resource allocations can be found which have very similar makespans and load balance values, but very different robustness values. It was also observed that for such a highly heterogeneous and correlated system, even considering both makespan and load balance index cannot predict robustness. Actually, it can be quite misleading. A more balanced resource allocation with a good makespan can have a very small robustness value. Therefore, for such a system, neither makespan nor load balance index can serve as a robustness indicator, and the need for the robustness measure proposed in this research is more acute.

Acknowledgments: The authors thank Dr. David Drain for his valuable comments.

References

[1] High Productivity Computing Systems, `http://www.darpa.mil/ipto/programs/hpcs/index.htm`.

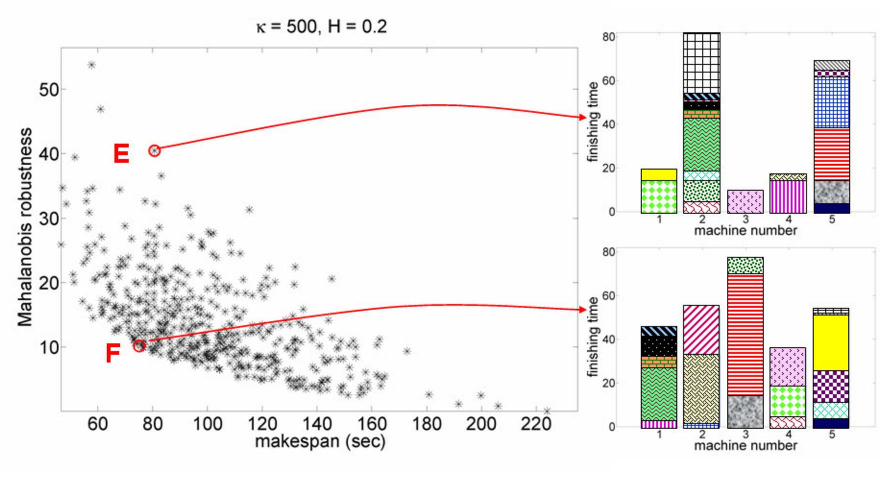


Figure 5. Mahalanobis robustness (with respect to makespan), where correlation is high (κ = 500).

[2] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim. Measuring the robustness of a resource allocation. *IEEE Trans. on Parallel and Distributed Systems*, 15(7):630–641, July 2004.

[3] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, and S. Sedigh-Ali. Representing task and machine heterogeneities for heterogeneous computing systems. *Tamkang J. of Science and Engineering*, 3(3):195–207, invited, Nov. 2000.

[4] T. Anderson. *An Introduction to Multivariate Statistical Analysis*. John Wiley & Sons, New York, NY, 1984.

[5] I. Assayad, A. Girault, and H. Kalla. A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints. In *The 2004 Int'l Conf. on Dependable Systems and Networks (DSN 2004)*, pages 347–356, July 2004.

[6] M. Backes and C. Cachin. Reliable broadcast in a computational hybrid model with Byzantine faults, crashes, and recoveries. In *The 2003 Int'l Conf. on Dependable Systems and Networks (DSN 2003)*, pages 37–46, June 2003.

[7] G. Badishi, I. Keidar, and A. Sasson. Exposing and eliminating vulnerabilities to denial of service attacks in secure gossip-based multicast. In *The 2004 Int'l Conf. on Dependable Systems and Networks (DSN 2004)*, pages 223–232, July 2004.

[8] D. A. Belsley, E. Kuh, and R. E. Welsch. *Regression Diagnostics*. John Wiley & Sons, New York, NY, 1980.

[9] P. I. Davies and N. J. Higham. Numerically stable generation of correlation matrices and their factors. *BIT Numerical Mathematics*, 40(4):640–651, Dec. 2000.

[10] X. Défago, P. Urbán, N. Hayashibara, and T. Katayama. Definition and specification of accrual failure detectors. In *The 2005 Int'l Conf. on Dependable Systems and Networks (DSN 2005)*, pages 206–215, June-July 2005.

[11] A. Girault, H. Kalla, and M. Sighireanu. An algorithm for automatically obtaining distributed and fault-tolerant static schedules. In *The 2003 Int'l Conf. on Dependable Systems and Networks (DSN 2003)*, pages 159–168, June 2003.

[12] M. Hirschberger, Y. Qi, and R. E. Steuer. Randomly generating portfolio-selection covariance matrices with specified distributional characteristics. http://www.terry.uga.edu/˜rsteuer/PDF-Links/Simulation.pdf, 2004.

[13] S. Y. Ko and I. Gupta. Perturbation-resistant and overlay-independent resource discovery. In *The 2005 Int'l Conf. on Dependable Systems and Networks (DSN 2005)*, pages 248–257, June-July 2005.

[14] A. J. Oliner, L. Rudolph, R. K. Sahoo, J. E. Moreira, and M. Gupta. Probabilistic QoS guarantees for supercomputing systems. In *The 2005 Int'l Conf. on Dependable Systems and Networks (DSN 2005)*, pages 634–643, June-July 2005.

[15] Y. Ou and J. B. Dugan. Sensitivity analysis of modular dynamic fault trees. In *The IEEE Int'l Computer Performance and Dependability Symp. (IPDS 2000)*, pages 35–43, March 2000.

[16] K. Schindler and H. Bischof. On robust regression in photogrammetric point clouds. In *25th DAGM Pattern Recognition Symp. (DAGM'03)*, 2003.

[17] SciDAC Scalable Systems Software Team. What are the system software challenges for SciDAC? White paper, available at `http://www.csm.ornl.gov/~geist/cgi-bin/enote.cgi?nb=system&action=view&page=-41`, Feb. 2005.

[18] S. R. Searle. *Matrix Algebra Useful for Statistics*. John Wiley & Sons, New York, NY, 1982.

[19] C. P. Shelton and P. Koopman. Improving system dependability with functional alternatives. In *The 2004 Int'l Conf. on Dependable Systems and Networks (DSN 2004)*, pages 295–304, July 2004.

[20] P. Sobe. Reaching efficient fault-tolerance for cooperative applications. In *The IEEE Int'l Computer Performance and Dependability Symp. (IPDS 2000)*, pages 48–57, March 2000.

[21] M. Stastny. Mahalanobis distance. `http://www.wu-wien.ac.at/usr/h99c/h9951826/distance.pdf`, 2001.