Measuring Robustness for Distributed Computing Systems

Shoukat Ali[†], Anthony A. Maciejewski[‡], and Howard Jay Siegel[‡§]

[†]University of Missouri-Rolla
Department of Electrical and Computer Engineering
Rolla, MO 65409-0040 USA
shoukat@umr.edu

Colorado State University
[‡]Department of Electrical and Computer Engineering
[§]Department of Computer Science
Fort Collins, CO 80523-1373 USA
{hj, aam}@colostate.edu

Abstract—
Performing computing and communication tasks on parallel and distributed systems may involve the coordinated use of different types of machines, networks, interfaces, and other resources. All of these resources should be allocated in a way that maximizes some system performance measure. However, allocation decisions and performance prediction are often based on "nominal" values of application and system parameters. The actual values of these parameters may differ from the nominal ones, e.g., because of inaccuracies in the initial estimation or because of changes over time caused by an unpredictable system environment.

An important question then arises: given a system design, what extent of departure from the assumed circumstances will cause the performance to be unacceptably degraded? That is, how robust is the system? To address this issue, one needs to derive a design methodology for deriving the degree of robustness of a resource allocation – the maximum amount of collective uncertainty in application and system parameters within which a user-specified level of performance can be guaranteed. Our procedure for this is presented in this paper.

The main contributions of this research are (1) a mathematical description of a metric for the robustness of a resource allocation with respect to desired system performance features against multiple perturbations in multiple system and environmental conditions, (2) a procedure for deriving a robustness metric for an arbitrary system, and (3) example applications of this procedure to several different systems.

This paper is based on the robustness research presented by the authors in [Ali03,AlM03].

I. Introduction

The robust design of computing and communication systems is becoming an increasingly important issue [AlC01], [BoM02], [CaD02], [DaG01], [DeK02], [Dev01], [HaR98], [HaW02], [Iri01], [Jen01c], [LeD94], [RoL02], [SeS02], [YeZ03]. There is a need for research that addresses the issues of developing a generalized robustness metric and deriving robust resource allocations in a parallel and distributed system. Our formulation of a standard generalized robustness metric for resource allocation is an important step towards ongoing efforts to create robust designs.

The motivation for this research was provided by research supported by the DARPA's ITO Quorum program, under the project called "Management System for Heterogeneous Networks." The research involved the design and analysis of heuristics for robust resource allocation in different types of heterogeneous computing environments including the HiPer-D (High Performance Distributed

Computing Program). A typical HiPer-D computing system consists of a set of dedicated machines interconnected by high-speed communication links. A set of sensors (radars, sonars, etc.) sends streams of data sets to a set of communicating, continuously running applications that process these data sets and send their outputs to other applications or actuators.

A HiPer-D system is required to satisfy a set of throughput and latency constraints. Any allocation of the resources must enforce these quality of service (QoS) constraints by ensuring that the computation and communication times are within certain limits. When the system is first configured, it is assumed to operate under certain estimated values of the initial sensor loads (i.e., outputs from sensors). Such an initial resource allocation ensures that all throughput and latency constraints are met when the ship is first deployed. However, the system is expected to operate in a dynamic environment, where the sensor loads are expected to change unpredictably. Increases in sensor loads cause increases in the computation and communication times, which in turn may cause throughput and latency violations. Therefore, the initial resource allocation might be rendered invalid soon after the operation begins.

One way of handling the unpredictable load increases is to design a resource allocation that will tolerate as much sensor load increase as possible before a QoS violation occurs. Two questions need to be answered:

- Given a set of resource allocations, how does one determine which resource allocation tolerates the largest load increase? This task necessitates the formulation of an appropriate metric.
- How does one develop methods that can derive such a resource allocation?

For the first item, one needs a general approach because the sensor loads might not be the only uncertainties in a HiPer-D system. Two other examples are: (a) inaccurate models for computation/communication times, and (b) sudden machine or link failures.

A general approach is necessary also because for systems other than HiPer-D, there might be other uncertainties. Typically, the resource allocation decisions and the performance prediction are based on estimated/initial values of application and system parameters. However, complex computing and communication systems typically operate in an unpredictable environment where the actual values of these parameters may differ from the estimates due to a variety of reasons. As a result, the "real" system performance may degrade. An important question then arises. *Given a resource allocation, what is the maximum departure from the expected conditions that the system can tolerate and still deliver the promised performance?* That is, how *robust* is the system?

Before answering this question one needs to clearly define robustness. Robustness has been defined in different ways by different researchers. According to [Jen01c], robustness is the degree to which a system can function correctly in the presence of inputs different from those assumed. In a more general sense, [Gri01] states that a robust system continues to operate correctly across a wide range of operational conditions. Robustness, according to [Jen01a], guarantees the maintenance of certain desired system characteristics despite fluctuations in the behavior of its component parts or its environment. The concept of robustness, as used in this research, is similar to that in [Jen01a]. Like [Jen01a], this work emphasizes that robustness should be defined for a given set of system features, with a given set of perturbations applied to the 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. For example, if a resource allocation has been declared to be robust with respect to satisfying a throughput requirement against perturbations in the system load, then the system, configured under that allocation, should continue to operate without a throughput violation when the system load increases. The immediate question is: what is the *degree* of ro-

bustness? That is, for the example given above, how much can the system load increase before a throughput violation occurs? This research addresses this question, and others related to it, by formulating the mathematical description of a metric that evaluates the robustness of a resource allocation with respect to certain system performance features against multiple perturbations in multiple system components and environmental conditions. In addition, this work outlines a procedure called FePIA (named after the four steps that constitute the procedure) for deriving a robustness metric for an arbitrary system. For illustration, the procedure is employed to derive robustness metrics for three example distributed systems. The robustness metric and the FePIA procedure for its derivation are the main contributions of this paper.

The following are the specific contributions of our research (presented in [AlM03]). For the allocation of computing and communication resources in parallel and distributed systems, this research

- gives a mathematical description of a metric for robustness,
- describes a four-step procedure, called FePIA, for deriving a robustness metric for an arbitrary system, and
- outlines example applications of this procedure to several different systems.

The rest of the paper is organized as follows. Section II describes the FePIA procedure mentioned above. It also defines a generalized robustness metric. Section III outlines some challenges in this research and possible future work. some experiments that highlight the usefulness of the robustness metric. Section IV concludes the paper.

II. Generalized Robustness Metric

This section summarizes from [AlM03] the procedure, called FePIA, for deriving a general robustness metric for any desired computing environment. The name for the above procedure stands for identifying the performance features, the perturbation parameters, the impact of perturbation parameters on performance features, and the analysis to determine the robustness. Each step of the FePIA procedure is now described.

1) Describe quantitatively the requirement that makes the system robust. Based on this *robustness requirement*, determine the QoS performance features that should be limited in variation to ensure that the robustness requirement is met. Identify the acceptable variation for these feature values as a result of uncertainties in system parameters. Consider an example where (a) the QoS performance feature is makespan (the total time it takes to complete the execution of a set of applications) for a given resource allocation, (b) the acceptable variation is up to 30% of the makespan that was calculated for the given resource allocation using estimated execution times of applications on the machines they are assigned, and (c) the uncertainties in system parameters are inaccuracies in the estimates of these execution times.

Mathematically, let Φ be the set of system performance features that should be limited in variation. For each element $\phi_i \in \Phi$, quantitatively describe the tolerable variation in ϕ_i. Let $\left\langle \beta_i^{\min}, \beta_i^{\max} \right\rangle$ be a tuple that gives the bounds of the tolerable variation in the system feature ϕ_i. For the makespan example, ϕ_i is the time the i-th machine finishes its assigned applications, and its corresponding $\left\langle \beta_i^{\min}, \beta_i^{\max} \right\rangle$ could be $\langle 0,\ 1.3 \times (\text{estimated makespan value}) \rangle$.

2) Identify all of the system and environment parameters whose values may impact the QoS performance features selected in step 1. These are called the perturbation parameters (these are similar to hazards in [BoM02]), and the performance features are required to be robust with respect to these perturbation parameters. For the makespan example above, the resource allocation (and its associated predicted makespan) was based on the estimated application execution times. It is desired that the makespan be robust (stay within 130% of its estimated value) with respect to uncertainties in these estimated execution times.

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. For the makespan example, $\boldsymbol{\pi}_j$ could be the vector composed of the actual application execution times, i.e., the i-th element of $\boldsymbol{\pi}_j$ is the actual execution time of the i-th application on the machine it was assigned. In general, representation of the perturbation parameters as separate elements of Π would be based on their nature or kind (e.g., message length variables in $\boldsymbol{\pi}_1$ and computation time variables in $\boldsymbol{\pi}_2$).

3) Identify the impact of the perturbation parameters in step 2 on the system performance features in step 1. For the makespan example, the sum of the actual execution times for all of the applications assigned a given machine is the time when that machine completes its applications. Note that 1(b) implies that the actual time each machine finishes its applications must be within the acceptable variation.

Mathematically, for every $\phi_i \in \Phi$, determine 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. For the makespan example, ϕ_i is the finishing time for machine m_i, and f_{ij} would be the sum of execution times for applications assigned to machine m_i. The rest of this discussion will be developed assuming only one element in Π. The case where multiple perturbation parameters can affect a given ϕ_i simultaneously is discussed in [AlM03].

4) The last step is to determine the smallest collective variation in the values of perturbation parameters identified in step 2 that will cause any of the performance features identified in step 1 to violate its acceptable variation. This will be the degree of robustness of the given resource allocation. For the makespan example, this will be some quantification of the total amount of inaccuracy in the execution times estimates allowable before the actual makespan exceeds 130% of its estimated value.

Mathematically, for every $\phi_i \in \Phi$, determine the *boundary values of* $\boldsymbol{\pi}_j$, i.e., the values satisfying the boundary relationships $f_{ij}(\boldsymbol{\pi}_j) = \beta_i^{\min}$ and $f_{ij}(\boldsymbol{\pi}_j) = \beta_i^{\max}$. (If $\boldsymbol{\pi}_j$ is a discrete variable then the boundary values correspond to the closest values that bracket each boundary relationship. See [AlM03] for an example.) These relationships separate the region of robust operation from that of non-robust operation. Find the smallest perturbation in $\boldsymbol{\pi}_j$ that causes any $\phi_i \in \Phi$ to exceed the bounds $\langle \beta_i^{\min}, \beta_i^{\max} \rangle$ imposed on it by the robustness requirement.

Specifically, let $\boldsymbol{\pi}_j^{\text{orig}}$ be the value of $\boldsymbol{\pi}_j$ at which the system is originally assumed to operate. However, due to inaccuracies in the estimated parameters or changes in the environment, the value of the variable $\boldsymbol{\pi}_j$ might differ from its assumed value. This change in $\boldsymbol{\pi}_j$ can occur in different "directions" depending on the relative differences in its individual components. Assuming that no information is available about the relative differences, all values of $\boldsymbol{\pi}_j$ are possible. Figure 1 illustrates this concept for a single feature, ϕ_i, and a two-element perturbation vector $\boldsymbol{\pi}_j \in \mathbf{R}^2$. The curve shown in Figure 1 plots the set of boundary points $\{\boldsymbol{\pi}_j \mid f_{ij}(\boldsymbol{\pi}_j) = \beta_i^{\max}\}$ for a resource allocation μ. For this figure, the set of boundary points $\{\boldsymbol{\pi}_j \mid f_{ij}(\boldsymbol{\pi}_j) = \beta_i^{\min}\}$ is given by the points on the π_{j1}-axis and π_{j2}-axis.

The region enclosed by the axes and the curve gives the values of $\boldsymbol{\pi}_j$ for which the system is robust with respect to ϕ_i. For a vector $\mathbf{x} = [x_1\, x_2\, \cdots\, x_n]^{\mathrm{T}}$, let $\|\mathbf{x}\|_2$ be the ℓ_2-norm (Euclidean norm) of the vector, defined by $\sqrt{\sum_{r=1}^{n} x_r^2}$. The point on the curve marked as $\boldsymbol{\pi}_j^{\star}(\phi_i)$ has the property that the Euclidean distance from $\boldsymbol{\pi}_j^{\text{orig}}$ to $\boldsymbol{\pi}_j^{\star}(\phi_i)$, $\|\boldsymbol{\pi}_j^{\star}(\phi_i) - \boldsymbol{\pi}_j^{\text{orig}}\|_2$, is the smallest over all such distances from $\boldsymbol{\pi}_j^{\text{orig}}$ to a point on the curve. An important interpretation of $\boldsymbol{\pi}_j^{\star}(\phi_i)$ is that the value $\|\boldsymbol{\pi}_j^{\star}(\phi_i) - \boldsymbol{\pi}_j^{\text{orig}}\|_2$ gives the largest Euclidean distance that the variable $\boldsymbol{\pi}_j$ can change in *any* direction from the assumed value of $\boldsymbol{\pi}_j^{\text{orig}}$ without the performance feature ϕ_i exceeding the tolerable variation. Let the distance $\|\boldsymbol{\pi}_j^{\star}(\phi_i) - \boldsymbol{\pi}_j^{\text{orig}}\|_2$ be called the robustness radius, $r_\mu(\phi_i, \boldsymbol{\pi}_j)$, of ϕ_i against $\boldsymbol{\pi}_j$. Mathematically,

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

This work defines $r_\mu(\phi_i, \boldsymbol{\pi}_j)$ *to be the robust-*

ness of resource allocation μ with respect to performance feature ϕ_i against the perturbation parameter $\boldsymbol{\pi}_j$.

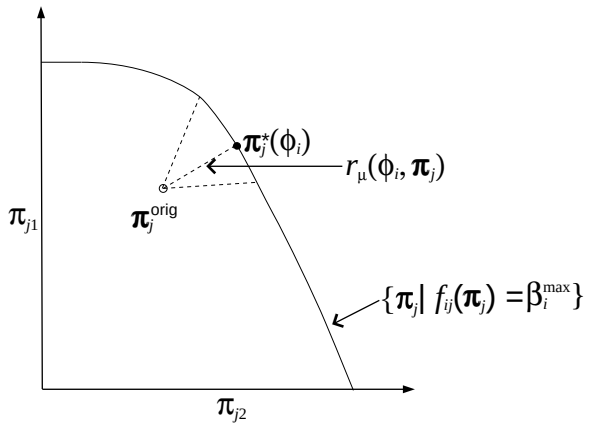

Fig. 1

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

The robustness definition can be extended easily for all $\phi_i \in \Phi$. It is simply the minimum of all robustness radii. Mathematically, let

$$\underline{\rho_\mu(\Phi, \ \boldsymbol{\pi}_j)} = \min_{\phi_i \in \ \Phi} \left(r_\mu(\phi_i, \ \boldsymbol{\pi}_j)\right). \qquad (2)$$

Then, $\rho_\mu(\Phi, \ \boldsymbol{\pi}_j)$ is the *robustness metric of resource allocation μ with respect to the performance feature set Φ against the perturbation parameter $\boldsymbol{\pi}_j$.*

Even though the ℓ_2-norm has been used for the robustness radius in this general formulation, in practice, the choice of a norm should depend on the particular environment for which a robustness measure is being sought. [AlM03] gives an example situation where the ℓ_1-norm is preferred over the ℓ_2-norm.

III. Future Work

We are interested in extending this research in the following ways:

1. Develop tractable methods for computing the robustness radius, in general. To calculate the robustness radius, one needs to solve the optimization problem posed in Equation 1. Such a computation could potentially be very expensive. One can exploit structure of this problem, along with some assumptions, to make this problem somewhat easier to solve. An optimization problem of the form $\min_{l(x)=0} f(x)$ or $\min_{c(x)\geq 0} f(x)$ could be solved very efficiently to find the global minimum if $f(x)$, $l(x)$, and $c(x)$ are convex, linear, and concave functions respectively. Some solution approaches, including the well-known interior-point methods, for such *convex optimization* problems are presented in [BoV03]. However, one needs to develop tractable methods for computing the robustness radius, for a general case.
2. Extend the application of the robustness metric and the procedure to derive it to other complex systems like computer networks, Internet environments, large-scale component-based software systems, and mobile and wireless computing systems. For example, we would like to explore the robustness of a network design/configuration with respect to some system properties against uncertainty in the networks operational environment.
3. Extend the robustness metric formulation to include combinations of perturbation parameters that (a) are measured in different units, or (b) are a mixture of continuous and discrete parameters. The research in [AlM03] does address the above issue, but more work needs to be done.
4. Extend the robustness metric formulation to consider errors in the models used for computation and communication.
5. Extend the robustness metric formulation to include probabilistic $\boldsymbol{\pi}_j$. In some situations, changes in some elements of $\boldsymbol{\pi}_j$ may be more probable than changes in other elements. In such cases, one may be able to modify the distance calculation so that the contribution from an element with a larger probability to change has a proportionally larger weight.
6. Design robust resource allocation algorithms for heterogeneous distributed computing systems, in-

cluding real-time systems and grid systems.

IV. Conclusions

The formulation of a completely general robustness metric that could be applied to a variety of (possibly heterogenous) parallel and distributed systems is an important contribution of our research. Such systems, consisting of a set of machines and networks, frequently operate in uncertain or dynamic environments where the quality of service that is delivered degrades due to unpredictable circumstances, such as sudden machine failures, higher than expected system load, or inaccuracies in the estimation of system parameters. The robustness metric, determines, for a given system design, what extent of departure from the assumed circumstances will cause the quality of service to be unacceptably degraded. This paper has summarized a mathematical description of a metric for the robustness of a resource allocation with respect to desired system performance features against multiple perturbations in various system and environmental conditions. In addition, the research describes a procedure, called FePIA, to methodically derive the robustness metric for a variety of parallel and distributed resource allocation systems. Such a metric can help researchers evaluate a given resource allocation for robustness against uncertainties in specified perturbation parameters.

References

[AlC01] J. Albowicz, A. Chen, and L. Zhang, "Recursive position estimation in sensor networks," *IEEE International Conference on Network Protocols (ICNP'01)*, Nov. 2001.

[AlM03] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim, "Measuring the robustness of resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, accepted, to appear in May 2004.

[BoM02] L. Bölöni and D. C. Marinescu, "Robust scheduling of metaprograms," *Journal of Scheduling*, Vol. 5, No. 5, Sept. 2002, pp. 395–412.

[BoV03] S. Boyd and L. Vandenberghe, *Convex Optimization,* to appear in 2003, available at `http://www.stanford.edu/class/ee364/index.html`.

[CaD02] J. M. Carlson and J. Doyle, "Complexity and robustness," *Proceedings of National Academy of Science (PNAS)*, Vol. 99, No. 1, Feb. 2002, pp. 2538–2545.

[DaG01] A. J. Davenport, C. Gefflot, and J. C. Beck, "Slack-based techniques for robust schedules," *6th European Conference on Planning (ECP-2001)*, Sept. 2001, pp. 7–18.

[DeK02] J. DeVale and P. Koopman, "Robust software – no more excuses," *IEEE International Conference on Dependable Systems and Networks (DSN '02)*, June 2002, pp. 145–154.

[Dev01] J. DeVale, *High Performance Robust Computer Systems*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 2001.

[Gri01] S. D. Gribble, "Robustness in complex systems," *8th Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, May 2001, pp. 21–26.

[HaR98] E. Hart, P. M. Ross, and J. Nelson, "Producing robust schedules via an artificial immune system," *The 1998 International Conference on Evolutionary Computing*, May 1998, pp. 464–469.

[HaW02] D. E. Hastings, A. L. Weigel, and M. A. Walton, "Incorporating uncertainty into conceptual design of space system architectures," *ESD Internal Symposium, Working Paper Series*, May 2002, ESD-WP-2003-01.01.

[Iri01] IRIS: Infrastructure for Resilient Internet Systems, "IRIS ITR proposal," `http://project-iris.net/proposal.html`, 2001.

[Jen01a] E. Jen, "Stable or robust? What is the difference?," *Complexity*, to appear.

[Jen01c] M. Jensen, "Improving robustness and flexibility of tardiness and total flowtime job shops using robustness measures," *Journal of Applied Soft Computing*, Vol. 1, No. 1, June 2001, pp. 35–52.

[LeD94] V. J. Leon, S. D. Wu, and R. H. Storer, "Robustness measures and robust scheduling for job shops," *IEE Transactions*, Vol. 26, No. 5, Sept. 1994, pp. 32–43.

[RoL02] R. Rodrigues, B. Liskov, and L. Shrira, "The design of a robust peer-to-peer system," *10th ACM SIGOPS European Workshop*, Sept. 2002, pp. 22–25.

[SeS02] M. Sevaux and K. Sörensen, "Genetic algorithm for robust schedules," *8th International Workshop on Project Management and Scheduling (PMS 2002)*, Apr. 2002, pp. 330–333.

[YeZ03] F. Ye, G. Zhong, S. Lu, and L. Zhang, "A robust data delivery protocol for large scale sensor networks," *2nd International Workshop on Information Processing in Sensor Networks (IPSN'03)*, Apr. 2003.