Proceedings of IPPS/SPDP '98

Performance Prediction in Production Environments*

Jennifer M. Schopf and Francine Berman
Computer Science and Engineering Department
University of California, San Diego
{jenny, berman}@cs.ucsd.edu

Abstract

Accurate performance predictions are difficult to achieve for parallel applications executing on production distributed systems. Conventional point-valued performance parameters and prediction models are often inaccurate since they can only represent one point in a range of possible behaviors. We address this problem by allowing characteristic application and system data to be represented by a set of possible values and their probabilities, which we call **stochastic values**.

In this paper, we give a practical methodology for using stochastic values as parameters to adaptable performance prediction models. We demonstrate their usefulness for a distributed SOR application, showing stochastic values to be more effective than single (point) values in predicting the range of application behavior that can occur during execution in production environments.

1 Introduction

Parallel and distributed production platforms provide a challenging environment in which to achieve performance. The impact of multiple users sharing a platform creates an environment where system conditions are dynamic and application performance is difficult to predict. However, prediction is critical to the achievement of performance. In particular, it is a fundamental component in both application and job scheduling, as well as resource allocation.

The problem of developing accurate parallel application performance predictions in production environments is twofold: First, the performance activities (e.g. communication and computation) represented in the performance model must reflect the dynamic changes in effective capacities (e.g. CPU load and network bandwidth) inherent in production environments. Second, the parameters used by performance models must accurately reflect the range of behavior observed in production environments.

Conventional approaches that develop fixed (unadaptable) performance models parameterized by single (point) values often provide an unacceptably inaccurate representation of application behavior in production environments. In this paper, we describe a new approach for defining performance model parameters and adaptable performance prediction models, and demonstrate that this approach can be used to predict the range of application performance in production environments more accurately than conventional models.

1.1 Point Values and Stochastic Values

Most performance prediction models use parameters to describe system and application characteristics such as bandwidth, CPU load information, message size, operation counts, etc. Model parameters are generally assumed to have a single likely value, which we refer to as a **point value**. For example, a point value for bandwidth might be 7 Mbits/second.

However, in practice, point values are often a best guess, an estimate under ideal circumstances, or a value that is accurate only for a given time frame. Frequently it is more useful to represent system and application characteristics as a *distribution* of possible values over a range; for example, bandwidth may be reported as 7 Mbits/second $\pm$ 2 Mbits/second. We refer to values represented as distributions as **stochastic values**. Whereas a point value gives a single value for a quantity, a stochastic value gives a set of possible values weighted by probabilities to represent a range of likely behavior.

Conventional performance prediction models cannot take advantage of model parameters that are stochastic values. In [Sch97] we defined a performance prediction model representation–called **structural modeling**–that reflects the time-dependent dynamic mix of application performance activities occurring during execution in a production environment. In this paper, we show how to extend structural models to allow for model parameters that are stochastic values, resulting in application performance predictions that are stochastic values as well.

Our experiments using an SOR code demonstrate that stochastic values can enhance the information conveyed by a performance prediction. Whereas point value predictions

*This research was supported in part by NASA GSRP Grant #NGT-1-52133 and DARPA Contract N66001-97-C-8531.

provide an estimate of application execution behavior for a particular system state, stochastic values predict behavior over a range of likely system states. Moreover, stochastic value predictions provide valuable additional information that can be supplied to a scheduler or to the programmer to improve the overall execution performance of distributed parallel applications.

1.2 An Example

To illustrate the role of stochastic values in parameterizing a performance model, consider a simple two-machine system (consisting of machines A and B) executing an embarrassingly parallel application with a fixed number of units of work to be completed. In order to predict the amount of work each machine should be assigned, we need an estimate of the execution time per unit of work of each machine, as shown in Table 1.

	Machine A	Machine B
Dedicated	10 sec	5 sec
Production (point)	12 sec	12 sec
Production (stochastic)	12 sec $\pm$ 5%	12 sec $\pm$ 30%

Table 1. Execution times for a unit of work in dedicated and production modes on two machines.

In a dedicated environment, the time for machine A to perform one unit of work might be 10 seconds and for machine B, 5 seconds. These can be represented as point values since typically there is negligible variance in runtimes on dedicated machines. To balance the execution times of the machines in this setting, machine B should receive twice as much work as machine A.

In a production environment, contention for the processor and memory will cause the unit execution times of the machines to vary in a machine-dependent fashion over time. If we determine unit execution time using a mean capacity measure over a 24-hour period for a fixed amount of work, it is possible that A and B will have the same unit execution time on average, say 12 seconds per unit of work. In this case, it would make sense to equally balance the work between the machines.

However, mean values are only summary statistics for a range of possible values, and may neglect critical information about the distribution of work over time. A stochastic value includes information about the distribution of values. For example, it may be that because machine B is much faster than machine A, it has more users and therefore a more dynamic load. Because of this, at any given time the unit execution time for machine B might be 12 seconds per unit of work $\pm$ 30%. That is, the time to complete a unit of work will vary over an interval from 8.4 seconds to 15.6 seconds. On the other hand, since machine A is a slower machine without as many users contending for it's use, the actual unit execution time might be 12 seconds per unit of work $\pm$ 5% (or 11.4 to 12.6 seconds per unit of work). On average both machines perform the same, however at any given time their performance may vary radically.

With additional information about the distribution of application behavior, we can develop a sophisticated scheduling strategy tuned to the user's performance metric. If the accuracy of the prediction is a priority (i.e. there is a considerable penalty for an inaccurate prediction), then more work could be assigned to the small variance machine (machine A). If there is little penalty for poor predictions, we might optimistically assign a greater portion of the work to the often faster machine B.

Note that the stochastic values as defined here are not related to Petri net models [MBC84, Mol90], also called "stochastic models", in any way except through the application of conventional statistical techniques. Some researchers, however, are beginning to use probabilistic techniques to represent data for predictions of application performance in distributed environments. Brasileiro et al. [BFM91] use probability distribution functions to calculate wait times on a token ring network. In addition, Das's group at Penn State has been studying the effect of assuming normal distributions in simulations [Das97].

Our work is also related to Karlin's work in competitive analysis [KMMO94, IKP96], business information systems [Ver], and Kesselman and Bannister's approach to Quality of Service [KB97]. Although the areas represented by these projects are dissimilar, all are beginning to address the need for additional information, as we are, to make better decisions about performance.

In this paper, we define stochastic values and show how they can be used to improve the predictions provided by structural performance models for distributed parallel applications. In Section 2, we define stochastic values and show how they can be combined in such models arithmetically. In Section 3, we demonstrate how stochastic values can be used to improve performance predictions using a distributed SOR application in several production execution environments. We conclude with a summary in Section 4.

2 Using Stochastic Values for Predictions

There are two problems that must be addressed to make it feasible to use stochastic values for performance prediction. The first problem is to determine the nature of the distribution associated with the stochastic value. In Section 2.1 we show that in many cases assuming that the distribution is normal is satisfactory. This assumption greatly simplifies the second problem, namely how to combine stochastic values to calculate performance predictions in a meaningful way. After showing how normal stochastic values can be used as prediction model parameters, we extend standard error combination rules to define the arithmetic operations needed to combine normal stochastic values in Section 2.3.

2.1 Defining Stochastic Values

Every stochastic value is associated with a distribution, that is, a function that gives the probability associated with each value in its range. General distributions are awkward to work with because they have no unifying properties. It is often appropriate to summarize or approximate a general distribution by associating it with a member of a known family of distributions that is similar to the given data.

One common family used to approximate general distributions when appropriate is the family of *normal distributions*. Normal distributions can be defined using two values: a *mean* which gives the "center" of the range of the distribution, and a *standard deviation* which gives a range around the mean. For normal distributions, a range equal to two standard deviations includes approximately 95% of the possible values of the data.

Many real phenomena generate distributions that are close to normal. For example, in-core memory benchmarks on dedicated systems may exhibit execution time values with normal distributions. Figure 1 shows a histogram of runtimes for a sample sorting code on a single workstation with no other users present and its corresponding normal distribution.

Distributions are represented graphically in two common ways: The Probability Distribution Function (PDF), as shown in Figure 1, which graphs values against their probabilities, similar to a histogram, and the Cumulative Distribution Function (CDF), as shown on the right in Figure 1, which illustrates the probability that a point in the range is less than or equal to a particular value.

Two distributions that can be successfully approximated by normal distributions are long-tailed distributions and modal distributions are described below.

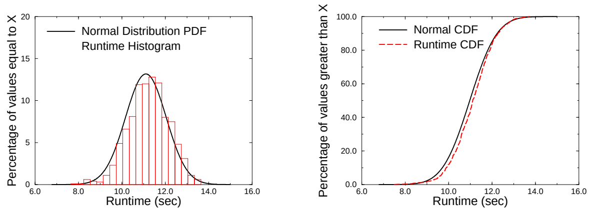

Figure 1. PDF and CDF of sample runtime with normal distribution.

2.1.1 Long-Tailed Distributions

It is often the case that characteristic system data has a threshold value, and that performance varies monotonically from that point in a long-tailed fashion, with the median several points below the threshold. For example, ethernet bandwidth between two workstations can exhibit a long-tailed distribution as shown in Figure 2.

In many cases we can adequately represent long-tailed behavior using a normal distribution. This provides a computational benefit since normal distributions are more computationally manageable than long-tailed distributions. However care must be taken so that the data points excluded in an assumption of normality do not adversely affect the quality of information resulting from the use of the distribution. In particular, we cannot assume that if we represent long-tailed data with a normal distribution that almost all of the data values will be covered within two standard deviations.

To illustrate, consider the data in Figure 2. The mean is 5.25. Assuming that this data can be approximated by a normal distribution, it's stochastic value would be reported as 5.25 ± 0.8. Since approximately 9% of the actual values are less than 4.45 or greater than 6.05, two standard deviations of the normal distribution represents 91% of the values, rather than the 95% typically assumed. In such cases, we have exchanged the efficiency of computing the distribution for the quality of its results. Normal distributions are a good substitution for long-tailed distributions only when inaccuracy in the data represented by the normal distribution can be tolerated by the scheduler, performance model, or other mechanism that will be using the data.

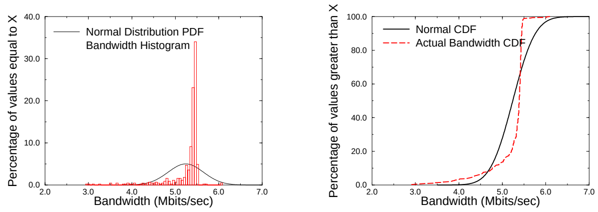

Figure 2. PDF and CDF for bandwidth with long-tailed distribution and corresponding normal distribution.

2.1.2 Modal Distributions

For some application or system characteristics, such as CPU load, the data can be viewed as several sets of data, each having its own distribution. An example of this is Figure 3, a histogram of load data for a workstation. The majority of the data lies in three distributions: a normal distribution centered at 0.94, a long-tailed distribution centered at 0.49 and another normal distribution centered at 0.33, each of which we refer to as a *mode*.

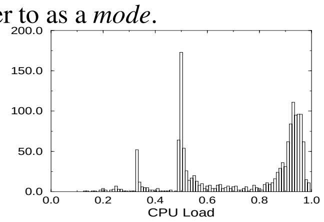

Figure 3. Load on a production workstation.

Stochastic values for modal data depend on how often the data changes mode and how unpredictably. Stochastic values to summarize multi-modal "bursty" data must be calculated differently than stochastic values derived from data that remains within a single mode. For example, consider a stochastic prediction calculated from a stochastic multi-modal value for load on a production workstation. Assume that each of the modes can be approximated by a normal distribution. If the load values remain within a single mode for the duration of the application execution time, the stochastic data can be approximated by a single normal distribution for this mode. We use this approach in Section 3.1 to parameterize a performance prediction model on workstations with production loads that remain within a single mode.

If the data changes modes frequently or unpredictably, or if the application is long-running, assuming that the data remains within a single mode is not sufficient. In this situation, we can calculate an approximate stochastic value by averaging the modal distributions based on the percentage of time the application executes in each mode. For example, if $M_1 \pm SD_1$ represents the distribution in mode 1, and $M_2 \pm SD_2$ and $M_3 \pm SD_3$ represent the distributions in modes 2 and 3 respectively, a distribution of multi-modal stochastic data can be obtained by calculating

$$P_1(M_1 \pm SD_1) + P_2(M_2 \pm SD_2) + P_3(M_3 \pm SD_3)$$

where P_i is the percentage of time the data spends in mode i. Since each mode can be thought of as having a normal distribution, so will the average stochastic value (see Section 2.3).

Note that P_1, P_2 and P_3 change over time and are system- and application-dependent. For some production systems and applications, a static benchmark over a long period of time may be suitable to predict fixed values for P_1, P_2 and P_3. However, for other production systems and applications (such as the systems we used in our experiments), stochastic data that reflects dynamically changing modal values is required. For our experiments, we use values generated by the Network Weather Service [Wol96] at runtime for our predictions.

In this paper we assume that data can be adequately represented by a normal distribution. In the next section we show how normal stochastic data may be used to predict application performance.

2.2 Structural Modeling

In this section we define and demonstrate the use of stochastic values as inputs to performance prediction models. We focus on performance prediction models represented as **structural models** [Sch97]. Structural models provide a flexible and adaptable mechanism for predicting application performance in dynamic production systems. By parameterizing such models with stochastic values, we can calculate performance predictions that are also stochastic values. We demonstrate that stochastic values can more accurately represent execution behavior than point values in production settings in Section 3.

Structural models are composed of component models and equations representing their interactions. Component models are defined (possibly recursively) as combinations of model parameters (benchmarks, latency and bandwidth measurements, operation counts, etc.) and/or other component models. We illustrate the development of structural and component models for a distributed Red-Black Successive Over-Relaxation application (SOR) below.

Red-Black SOR is a distributed stencil application whose data resides on an NxN grid. A common data distribution for this is a strip decomposition. The application is divided into "red" and "black" phases, with communication and computation alternating for each. In our implementation, this repeats for a predefined number of iterations. We focus here on developing a structural model for Red-Black SOR targeted to a distributed system since some of the most challenging performance problems arise in this setting.

A structural model for a production SOR on a distributed system is:

$$\begin{aligned}\mathbf{ExTime} &= \sum_{i=1}^{NumIts} [Max\{\mathbf{RedComp_p}\}+ \\ &\quad Max\{\mathbf{RedComm_p}\} + Max\{\mathbf{BlackComp_p}\} \\ &\quad +Max\{\mathbf{BlackComm_p}\}\ ]\end{aligned}$$

That is, the execution time is equal to the sum of the longest running machine/data pair for each component for each iteration.

This structural model consists of four types of component models: **RedComp**$_p$, **RedComm**$_p$, **BlackComp**$_p$, and **BlackComm**$_p$, where p refers to processor p and the strip partition assigned to it. Each component model (shown in boldface) must be evaluated in order to calculate a performance prediction of application execution time. Note that most estimates of computation are based on evaluating some time per data element, and then multiplying by the number of elements being computed for the overall problem. There are two widely used approaches for this: counting the number of operations involved in computing one element, and benchmarking. In this work, the most accurate prediction model for SOR used benchmarks divided by ambient load as expressed below. This model predicts computation time accurately in a dedicated environment to within 3%.

$$\mathbf{RedComp_p} = \frac{NumElt_p * BM(Elt_p)}{load_p}$$

$$\mathbf{BlackComp_p} = \frac{NumElt_p * BM(Elt_p)}{load_p}$$

where

- $NumElt_p$: Number of elements computed by processor p,
- $BM(Elt_p)$: Benchmark time for processor p to process a single element, and
- $load_p$: Percentage of processor p available for use by this application, supplied by the Network Weather Service.

Similarly, component models were defined for **RedComm**$_p$ and **BlackComm**$_p$. Due to space constraints, we refer interested readers to [SB97] for details.

2.3 Arithmetic Operations over Stochastic Values

To calculate stochastic values for the left-hand side of structural modeling equations, the right-hand side stochastic values may need to be combined arithmetically. By assuming that the distributions associated with the stochastic values are normal, we can take advantage of the fact that normal distributions are closed under linear combinations [LM86] to define straightforward rules for combining stochastic values.

For each arithmetic operation, we define a rule for combining stochastic values based on standard statistical error propagation methods [Bar78]. In the following, we assume that point values are represented by P and all stochastic values are of the form $X_i \pm a_i$ and represent normal distributions, where X_i is the mean and a_i is two standard deviations. Note that these rules will hold true for all families of distributions closed under linear combinations, not just normal distributions.

Figure 4 summarizes the arithmetic operations between a stochastic value and a point value, two stochastic values from related distributions, and two stochastic values from unrelated distributions. These formulas are outlined below and discussed in depth in [SB97]

When combining two stochastic values, two cases must be considered: when the distributions are related and when they are unrelated. Two distributions are *related* when there is a causal connection between their values. For example, when network traffic is heavy, available bandwidth tends to be low and latency tends to be high. When network traffic is light, available bandwidth tends to be high and latency tends to be low. We say that the distributions of latency and bandwidth are related in this case. In the table, we provide "conservative" formulas for the combinations of two related stochastic values so that the data is not "over-smoothed".

When two stochastic values are unrelated, their values are independent. This may be the case when the time between measurements of a single quantity is large, or when the two stochastic values represent distinct characteristics. For arithmetic on unrelated stochastic values, we use a probability-based square root error computation.

Since normal distributions are closed under addition and subtraction, the resulting stochastic value will also have a normal distribution. The product of stochastic values with normal distributions does not itself have a normal distribution. Rather, it is long-tailed. In many circumstances, we can approximate the long-tailed distribution with a normal distribution and ignore the tail as discussed previously in Section 2.1.1.

2.3.1 Group Operations over Stochastic Values

Often, structural models will combine components using group operations such as *Max*, *Min*, or other operators. The combination of stochastic values for operations over a group must be addressed in a situation-dependent manner.

For example, consider the calculation of the *Max* operator. Depending on the penalty for an incorrect prediction, different approaches may be taken. *Max* could be calculated by choosing the largest mean of the stochastic value inputs, or by selecting the stochastic value with the largest magnitude value in its entire range. For example, to compute the *Max* of $A = 4 \pm 0.5$, $B = 3 \pm 2$ and $C = 3 \pm 1$, A has the largest mean, and B has the largest value within its range. However on average, the values of A are likely to be higher than the values of B. The usage of the value in the structural model will determine how *Max* should be calculated.

3 Experimental Verification

To demonstrate the usefulness of stochastic values as performance predictions, we derive stochastic execution time predictions for the distributed Red-Black SOR application discussed previously for two production settings. We use a stochastic value to represent CPU $load_p$, a parameter to the application structural performance model described in Section 2.2. In a dedicated setting, the structural model we used predicted application execution time to within 2%.

However, in a production environment with changing load on the machines, the dedicated model with point-valued parameters was no longer sufficient. The goal of our experiments was to explore whether stochastic values accurately predicted the range of values experienced by the SOR application in a production system.

The environment for our experiments was a production network of heterogeneous Sparc workstations connected by 10 Mbit ethernet. Workstations were shared by multiple

	Addition	Multiplication
Point Value with Stochastic Value	$(X_i \pm a_i) + P = (X_i + P) \pm a_i$	$P(X_i \pm a_i) = PX_i \pm Pa_i$
Two Stochastic Values with Related Distributions	$\sum_{i=1}^{n} (X_i \pm a_i) = \sum_{i=1}^{n} X_i \pm \sum_{i=1}^{n} a_i$	$(X_i \pm a_i)(X_j \pm a_j) = X_iX_j \pm (a_iX_j + a_jX_i + a_ia_j)$
Two Stochastic Values with Unrelated Distributions	$\sum_{i=1}^{n} (X_i \pm a_i) = \sum_{i=1}^{n} X_i \pm \sqrt{\sum_{i=1}^{n} a_i^2}$	$(X_i \pm a_i)(X_j \pm a_j) = X_iX_j \pm \left(X_iX_j\sqrt{\left(\frac{a_i}{X_i}\right)^2 + \left(\frac{a_j}{X_j}\right)^2}\right)$

Figure 4. Arithmetic Combinations of a Stochastic Values

users and exhibited diverse processor speeds, available physical memory, and CPU load. The network was also shared by other users. The dynamic load data needed for our experiments was supplied by the Network Weather Service. The Network Weather Service supplied us with accurate run-time information about the CPU load on our machines as well as the variance of those values at 5 second intervals. At times the workstations were underused, so various load generators were used to simulate other users on the system.

3.1 Platform 1

In our first set of experiments we used a production system consisting of two Sparc-2 workstations, a Sparc-5 and and Sparc-10, all connected over 10 Mbit ethernet. All resources were shared by multiple users. Analysis of the load behavior showed that the distribution of $\{load_p\}$ for the (consistently) slowest machine was tri-modal as shown in Figure 3, but values typically remained within a single mode during execution.

We describe a representative experiment in which the load of the slowest machine in our environment was in the center mode, with a mean of 0.48. Two standard deviations of the preliminary data gave us a stochastic $load_{slowest}$ value of 0.48 $\pm$ 0.05. All other model parameters were point values.

Figure 5 shows that the execution time measurements fall entirely within the stochastic prediction of execution time for problem sizes that fit within main memory. The maximal discrepancy between the means of the modeled stochastic predictions (a reasonable point value representative of the stochastic predictions) and actual execution times is 9.7%. The discrepancy between modeled stochastic predictions and actual execution times is 0%. Our experiments indicate that within a single mode, stochastic information can be used effectively to accurately define the range of application behavior. The next set of experiments explore the regime where $load_p$ shifts between modes during execution.

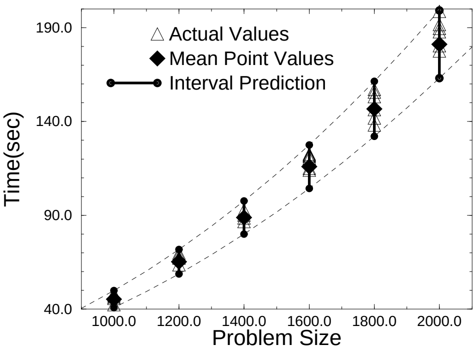

Figure 5. Execution times with predicted stochastic interval for SOR application on non-bursty platform.

3.2 Platform 2

Our second set of experiments were run on a platform consisting of a Sparc-5, a Sparc-10, and two UltraSparcs, again connected with 10 Mbit ethernet, and with both the machines and the network shared with other users.

The loads on this platform were multi-modal, and changed between modes frequently for all the machines, so the technique we used for determining $\{load_p\}$ on Platform 1 did not apply to Platform 2. To evaluate the structural model, we needed to determine an adequate stochastic value for $load_p$ on each machine. Each stochastic $load_p$ value required the determination of both an expected mean and the distribution associated with it. To determine the distribution we needed to have an acceptable time frame of past values to examine. Too small a time frame will give a limited picture of system behavior, while too large a time frame would provide little additional useful information.

To determine a time frame for the distribution, we used the following approach: The NWS supplied us with both the present predicted mean load (point value) as well as a history file of previous load measurements for each machine. These mean point values were used as parameters to the structural model, resulting in a point value estimate for the application execution time. This gave us a time frame to use for generating the associated distribution information for the stochastic $\{load_p\}$ values. Given this time frame, we determined $load_p$ for each processor using the averaging technique described in Section 2.1.2.

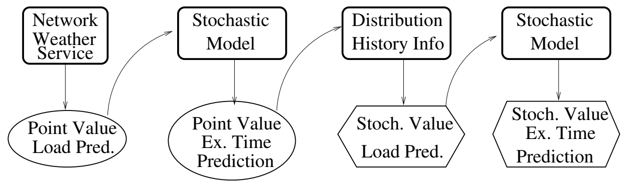

Figure 6. Determining a Stochastic Prediction for "bursty" multi-modal distributions.

Our approach (illustrated in Figure 6) is generalizable and can be used for other applications. In the future, distributional data may also be supplied by the NWS.

Figure 7 shows the execution times and predicted times for the SOR application for a moderate (1600x1600) problem size under bursty conditions, and Figure 8 shows $load_p$ one one of the machines during those runs. For example, the first point on Figure 7 show a stochastic prediction of 42.79 $\pm$ 6.35, with an actual execution time of 45.5415.

Our experiments show that stochastic values predict application behavior with small error. In particular, we capture approximately 80% of the actual execution times within the range of stochastic predictions, with a maximum error of approximately 14%, where the error between a value v not in the range of a stochastic value $X \pm a$ is the minimum distance between v and $(X - a, X + a)$. In contrast, the average difference between the actual execution times and the means of the modeled normal stochastic values exhibits a maximum error of 38.6%.

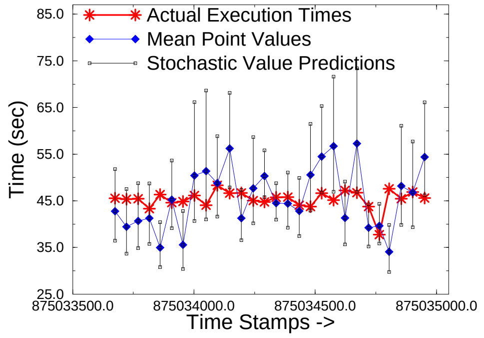

Figure 7. Execution times for 1600x1600 problem size on system with bursty load.

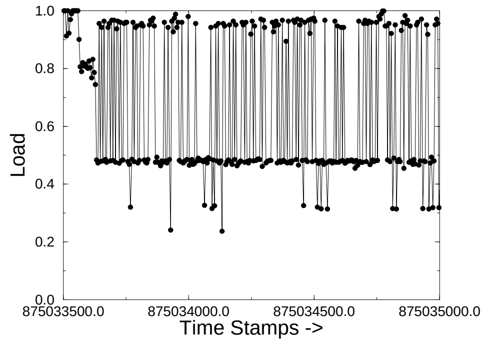

Figure 8. Load during runtimes shown in Figure 7.

These results are typical. We performed experiments for a number of problem sizes with similar results [SB97]. For all problem sizes, 74% of the actual execution times fell within the range delineated by the stochastic predictions. Moreover the error between the stochastic predictions and the actual times not within the stochastic range were typically on the order of 15%, whereas the error between the point value means predictions and the actual execution times were often as large as 50%.

4 Conclusion

In this paper, we describe a new approach to application performance prediction in production environments. We have defined **stochastic values** to reflect a likely range of behavior for model parameters, and extended the definition of **structural models** to allow for stochastic parameters as well as stochastic performance predictions. Experiments with a distributed SOR application demonstrate that in production settings, stochastic values can accurately identify the range of application execution behavior, providing more comprehensive and more accurate information about application execution than point values. We are currently exploring the way such information can be used as part of sophisticated strategies for scheduling, performance modeling, and program development for distributed parallel programs.

Acknowledgements

We are especially grateful to Neil Spring and Rich Wolski for many valuable discussions, their assistance with the experiments, and the use of the Network Weather Service. Thanks to Allen Downey and Keith Marzullo for commenting on a previous draft of this paper. Silvia Figueira supplied the original implementation of the SOR code. Further information is available on-line at `www.cs.ucsd.edu/users/jenny/research.html`.

References

[Bar78] B. Austin Barry. *Errors in Practical Measurement in Science, Engineering and Technology*. John Wilsey & Sons, 1978.

[BFM91] Marcos A. G. Brasileiro, James A. Field, and Antao B. Moura. Modelling and analysis of time critical applications on local area networks. In *Proc. of Chilean Computer Science Society*, pages 459–71, 1991.

[Das97] Chita Das. Personal communication, 1997.

[IKP96] S. Irani, A. R. Karlin, and S. Phillips. Strongly competitive algorithms for paging with locality of reference. *SIAM Journal on Computing*, 25(3):477–97, June 1996.

[KB97] C. Kesselman and J. Bannister. Qualis: Guaranteed quality of service for metacomputing, 1997. Private communication.

[KMMO94] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–71, June 1994.

[LM86] Richard J. Larsen and Morris L. Marx. *An Introduction to Mathematical Statistics and Its Applications*. Prentice-Hall, 1986.

[MBC84] M. Ajmone Marson, G. Balbo, and G. Conte. A class of generalized stochastic petri ners for the performance analysis of multiprocessor systems. *ACM TOCS*, pages 93–122, May 1984.

[Mol90] M.K. Molloy. Performance analysis using stochastic petri nets. *IEEE Transactions on Parallel and Distributed Systems*, C-31:913–7, September 1990.

[SB97] Jennifer Schopf and Fran Berman. Performance prediction in production environments. Technical Report #CS97-558, UCSD, CSE Dept., 1997.

[Sch97] Jennifer M. Schopf. Structural prediction models for high-performance distributed applications. In *CCC '97*, 1997. Also available as `www.cs.ucsd.edu/users/jenny/CCC97/index.html`.

[Ver] A. Verstraete. Business information systems. `www.smeal.psu.edu/misweb/infosys/ibismain.html`.

[Wol96] Rich Wolski. Dynamically forecasting network performance using the network weather service(to appear in the journal of cluster computing). Technical Report TR-CS96-494, UCSD, CSE Dept., 1996.