NORTHWESTERN
UNIVERSITY

Computer Science Department

Technical Report
NWU-CS-99-2
February 28, 1999

A Practical Methodology for Defining Histograms for Predictions and Scheduling

Jennifer M. Schopf

Abstract

Current distributed parallel platforms can provide the resources required to execute a scientific application efficiently. However, when these platforms are shared by multiple users, the performance of the applications using the system may be impacted in dynamic and often unpredictable ways. Performance prediction becomes increasingly difficult due to this dynamic behavior. Even performance modeling techniques that are built specifically for distributed parallel systems often require parameterization by single (point) values. However, in shared environments, point values may provide an inaccurate representation of application behavior due to variations in resource performance.

This paper addresses the use of **practical histogram stochastic values** to parameterize performance models. Whereas a point value provides a single value representation of a quantity, a stochastic value provides a set of possible values to represent a range of likely behavior. In previous work we investigated using either normal distributions or an upper and lower bound to represent stochastic data. In this work we examine using a combination of those methods, namely histograms, to represent this data. We define a practical approach to using histograms in a production setting, and then give experimental results for a set of applications under different load conditions.

A Practical Methodology for Defining Histograms for Predictions and Scheduling

Jennifer M. Schopf
Computer Science Department
Northwestern University
jms@cs.nwu.edu

Abstract

Current distributed parallel platforms can provide the resources required to execute a scientific application efficiently. However, when these platforms are shared by multiple users, the performance of the applications using the system may be impacted in dynamic and often unpredictable ways. Performance prediction becomes increasingly difficult due to this dynamic behavior. Even performance modeling techniques that are built specifically for distributed parallel systems often require parameterization by single (point) values. However, in shared environments, point values may provide an inaccurate representation of application behavior due to variations in resource performance.

This paper address the use of **practical histogram stochastic values** *to parameterize performance models. Whereas a point value provides a single value representation of a quantity, a stochastic value provides a set of possible values to represent a range of likely behavior. In previous work we investigated using either normal distributions or an upper and lower bound to represent stochastic data. In this work we examine using a combination of those methods, namely histograms, to represent this data. We define a practical approach to using histograms in a production setting, and then give experimental results for a set of applications under different load conditions.*

1 Motivation

Current distributed parallel platforms can provide the resources required to execute a scientific application efficiently. However, when these platforms are shared by multiple users, the performance of the applications using the system may be impacted in dynamic and often unpredictable ways. In order to obtain good performance, accurate performance prediction models for distributed parallel systems are needed. Most performance prediction models use parameters to describe system and application characteristics such as bandwidth, available CPU, message size, operation counts, etc. Model parameters are generally represented as a single likely value, which we refer to as a **point value**. For example, a point value for bandwidth might be 7 Mbits/second.

In practice, point values are often a best guess, an estimate under ideal circumstances, or a value that is accurate only for a given timeframe. In some situations it may be more accurate to represent system and application characteristics as a *range* of possible values; for example, bandwidth might be reported as varying between 6 and 8 Mbits/second. We refer to such values as **stochastic values**. Whereas a point value gives a single value for a quantity, a stochastic value gives a set of values, possibly weighted by probabilities, to represent a range of likely behavior [TK93].

One way to represent stochastic values is by using **histograms**. Histograms, also called VU-lists [Lue98] or frequency distributions [Bla80], consist of a set of intervals with associated probabilities. In this paper we present a practical definition for histograms and demonstrate their use when predicting the performance of parallel distributed applications.

This paper is organized as follows: Section 2 gives a brief overview of a modeling technique called structural modeling that can be parameterized by stochastic information. Section 3 details a practical definition for histograms. We describe the arithmetic needed to combine histograms in Section 4. Section 5 presents experimental results, and summary.

2 Overview of Structural Modeling

In order to predict an application's behavior on shared resources, we need both good models of the implementation and accurate information about the application and system with which to parameterize the models. In previous work [Sch98] we developed a technique called **structural modeling** that uses the functional structure of the application to define a set of equations that reflect the time-dependent, dynamic mix of application tasks occurring during execution in a distributed parallel environment. A structural model consists of a top-level model, component models, and input parameters. A top-level model is a **performance equation** consisting of component models and composition operators. Each component model is also defined to be a performance equation, parameterized by platform and application characteristics, such as benchmarks, dynamic system measurements, problem size, or the number of iterations to be executed. The output of a structural model is a predicted execution time. Structural models can also be developed for alternative performance metrics, such as speedup or response time.

A good performance model, like a good scientific theory, is able to explain available observations and predict future events, while abstracting away unimportant details. Structural models are represented by equations that allow us to abstract away details into the top level models and the parameters, but still accurately model the performance by showing the inter-relation of application and system factors through the component models and their parameterizations. We parameterize these models with stochastic values represented as histograms, which requires not only compute-efficient definitions of histograms but the needed arithmetic as well. These are discussed in the following sections.

3 Defining Histograms

A stochastic value X may be specified using a one-dimensional histogram $H(X)$ as follows:

$$\begin{aligned} &H(X) = X_1, X_2, ...X_m;\ \ with \\ &X_1 = [\underline{x_1}, \overline{x_1}] : p_1; \quad X_2 = [\underline{x_2}, \overline{x_2}] : p_2; \quad \ldots; \quad X_m = [\underline{x_m}, \overline{x_m}] : p_m \\ &where \sum_{i=1}^{m} p_i = 1 \end{aligned} \tag{1}$$

Each X_i entry in the definition of $H(X)$ is a pair consisting of an interval $[\underline{x_i}, \overline{x_i}]$, and an associated probability, p_i. One of the primary difficulties in defining histograms for dynamic system values is the determination of the appropriate intervals and probabilities for each histogram. A tradeoff exists between adding detail to the histogram description, which may be more accurate, but increases the computational complexity. As will be seen in the next section, arithmetic over extended histograms involves combining all possible sets of probabilistic choices, which can be quite large.

3.1 Previous Histogram Definitions and Related Work

There are a number of guidelines for defining histograms in the literature [MSW81, Bla80, Cra68, Bai72, Stu26, Sco79]. These "rules of thumb" are used by researchers for defining histograms, although in most cases these concentrate on ways to *display* the information, as opposed to *using* the information as a summary technique in a practical setting, and do not provide a rigorous approach to choosing the number of intervals in a histogram. The collective experience of these researchers indicates that the "best" number of intervals to select is data-dependent, application-dependent and system-dependent, as well as dependent on how the histogram will be used and the granularity of measurement [LMH96].

In the most closely related work to our research [LH95], histograms are defined initially based on the number of intervals provided by the user. The intervals are then split, using interval splitting [MR95, Ske74] and a user supplied procedure to produce more sub-intervals for intervals with high probabilities and fewer for intervals of the same width but lower probability. This procedure is iterated for each defined histogram until splitting the interval further does not change the resulting predicted value. For their purposes, "Computational complexity is reasonable given that the number of parameters specified as histograms is not too high" [LMH96]. However, this is not practical for our setting as we do not have a good splitting routine or the compute time to spend on an iterative method.

3.2 Practical Histograms

When defining histograms for use in making predictions to determine schedules at run-time, we need an approach that uses few, if any, user supplied parameters, is computationally efficient, and yet can adequately capture the behavior of the system being described. Furthermore, since the number of partial results for arithmetic functions over histograms can be quite large (as described in the next section) we need to limit this factor as well.

In our approach we used a two-pass method and limited the number of intervals in the histogram as well as the partial results. The first pass over the data was used to evaluate an upper bound and a lower bound, and to determine the size of the individual interval bins. The second pass was used to sort the data into the interval bins, and define the histogram for the stochastic data.

To select the number of intervals, we experimentally evaluated a range of intervals up to 25 on a preliminary set of data. The primary source of varying behavior for this data was the available CPU values for the contended system. This data was supplied by the Network Weather Service (NWS) [Wol97], an online tool that provides a time series of data for CPU availability, bandwidth and latency. In previous experiments [Sch98] we had determined the amount of data from the time series to use in our predictions, namely a 5 minute window. This had been shown to be no worse than using larger or smaller windows of data.

We experimentally examined the accuracy of the histogram for a range of interval sizes and measured both the time to compute the prediction, which included the time to determine the histograms for the input parameters, and the accuracy of the prediction. For load characteristics commonly seen on networks of workstations [Sch98], we determined that the predictive accurate of more than five intervals did not increase significantly, although the computational expense grew quite large at that point. Therefore, in the following, we used five intervals divided evenly over the range unless stated otherwise.

3.3 Arithmetic over Histograms

One of the primary difficulties in defining histograms for dynamic system values is the determination of the appropriate intervals and probabilities for each histogram. Since adding detail to the histogram description increases the computational complexity. Arithmetic over extended histograms involves combining all possible sets of probabilistic choices, which given the set D of stochastic values represented as histograms, each with k_D intervals, this results in $\Pi k_i * k_j$ for all $i, j \in D$. This value can be quite large, depending on how we define the number of intervals the histogram for a given stochastic value should have.

Given a set of k stochastic valued parameters, $D_1 \ldots D_k$, represented by histograms, assume that parameter D_i has m_i intervals $\{D_{i,1}, \ldots, D_{i,m_i}\}$ with $D_{i,j} = [\underline{d}_{i,j}, \overline{d}_{i,j}] : p_{i,j}$, i.e.:

$$D_i = \{D_{i,1}, D_{i,2}, \ldots, D_{i,m_i}\} with \quad D_{i,j} = [\underline{d}_{i,j}, \overline{d}_{i,j}] : p_{i,j} \quad for\ i = 1\ \ldots\ k\ and\ j = 1\ \ldots\ m_i \tag{2}$$

Since histograms are just sets of intervals, the interval arithmetic rules [Neu90] be used to combine them arithmetically. However, every interval has an associated probability. Hence, when calculating $F = D \odot E$ for some arithmetic function $\odot$ and histogram values D and E, it is necessary to calculate intermediate results for all possible combinations of intervals for each histogram. For example, given the histograms pictured in Figure 1:

$$\begin{aligned} D \rightarrow &\quad D_1 = [1,3] : 0.25; \quad D_2 = [3,5] : 0.75 \\ E \rightarrow &\quad E_1 = [2,4] : 0.6; \quad E_2 = [5,8] : 0.4 \end{aligned} \tag{3}$$

$$\begin{aligned} F = \quad D + E \rightarrow &[1+2, 3+4] : (0.25 * 0.6); \quad [1+5, 3+8] : (0.25 * 0.4); \\ &[3+2, 5+4] : (0.75 * 0.6); \quad [3+5, 5+8] : (0.75 * 0.4) \end{aligned} \tag{4}$$

This gives intermediate results (depicted in Figure 2):

$$[3,7] : 0.15; \quad [6,1] : 0.1; \quad [5,9] : 0.45; \quad [8,13] : 0.3 \tag{5}$$

Once the intermediate results have been calculated, they must be aggregated to form the result histogram, as pictured in Figure 3. A full definition of PDF's over histograms is given in [Lue98]. In this case, the resulting histogram is:

$$[3,5] : 0.15; \quad [5,6] : 0.6; \quad [6,7] : 0.7; \quad [7,8] : 0.55; \quad [8,9] : 0.85; \quad [9,13] : 0.4 \tag{6}$$

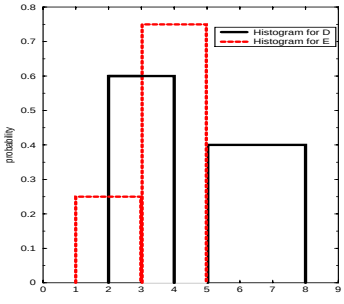

Figure 1. Histograms for sample parameters D and E

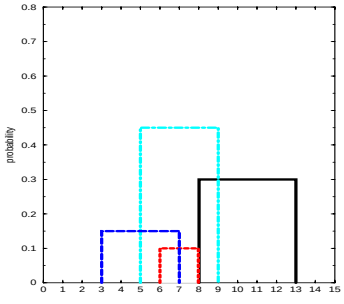

Figure 2. Histogram of intermediate results for D + E

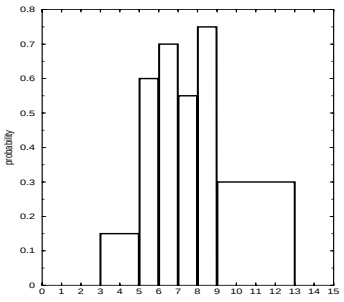

Figure 3. Result of D + E.

4 Predictions Using Histograms

Given a structural model for an implementation parameterized by point value and stochastic value system and application characteristics, we can calculate a prediction for applications running in contentious environments. This section details our experimental results, and compares using a histogram to represent stochastic values to other methods.

We examined three applications on a distributed network of Sparc workstations located in the UCSD Parallel Computation Lab. The machines are connected with a mix of 10 Mbit (slow) and 100 Mbit (fast) ethernet. Both the CPU and the network were shared with other users. For each application we ran a series of experiments under different load conditions. To show the execution time trends, we show the results of the actual execution times with the upper bound and lower bound of the histogram predictions, and in Table 5 give the percentage of values which fell into each probability interval of the histogram predictions for that graph. Figure 4 is a sample prediction for a single run of the SOR code.

In the experiments, the stochastic CPU information used in the models was supplied by the Network Weather Service [Wol97]. In categorizing the multiple background workloads seen for each experiment set, we define a machine with a **low** variation in CPU availability as having a variation of 0.25 or less, on a scale from 0 to 1, a **medium** variation when the CPU availability varied more than 0.25, but less than 0.5, and a **high** variation when the CPU values varied over half the range.

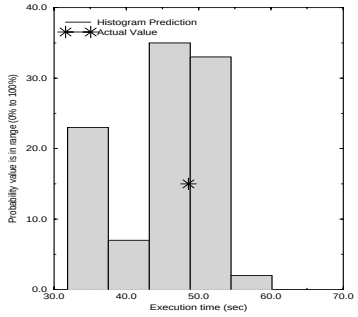

Figure 4. Sample Histogram Prediction.

Figure	Highest	2nd	3rd	4th	Lowest	Outside
SOR1 Figure 6	32	20	25	3	12	4
SOR2 Figure 7	18	23	21	13	14	11
GA 1 Figure 8	21	28	10	15	0	25
GA2 Figure 9	33	21	20	12	0	14
LU 1 Figure 10	55	26	8	7	4	0
LU2 Figure 11	42	38	0	12	8	0

Figure 5. Percentage of actual values falling in each probability interval for histogram predictions.

4.1 Application 1 - Successive Over-Relaxation Code

Successive Over-Relaxation (SOR) is a Regular SPMD code that solves Laplace's equation. Our implementation uses a red-black stencil approach where the calculation of each point in a grid at time t is dependent on the values in a stencil around it at time $t-1$. The application is divided into "red" and "black" phases, with communication and computation alternating for each.

Figure 6 shows the stochastic value predictions and the actual time of the SOR application, run when the PCL cluster had two processors with a highly variable CPU availability, one with a medium variable availability, and one with a low variable availability. A second set of experiments for the SOR code on the PCL cluster are presented in Figure 7, run when the PCL cluster had one high variable availability, two medium variable availability, and one low variable availability. These experiments show the benefits of stochastic predictions when the execution times exhibit an extreme variance, in this case variance production over 300%.

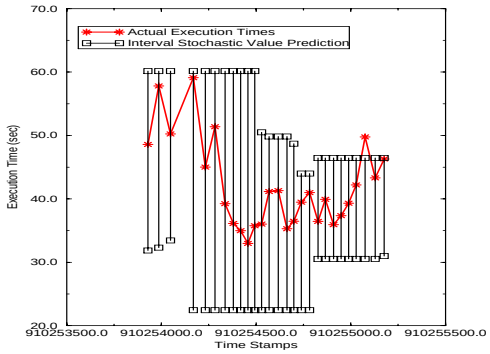

Figure 6. SOR1- stochastic value prediction for the SOR.

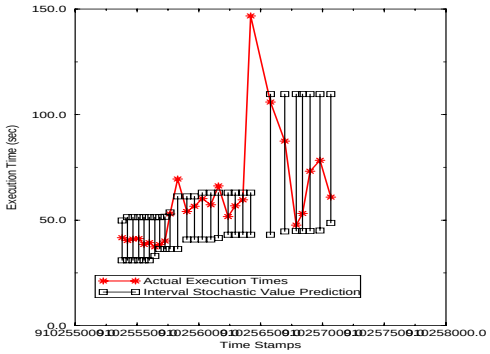

Figure 7. SOR2- stochastic value prediction for the SOR.

4.2 Application 2 - Genetic Algorithm Code

We implemented a genetic algorithm (GA) for the Traveling Salesman Problem (TSP) [LLKS85]. Our distributed implementation of this GA uses a global population and synchronizes between generations [Bha96]. All of the Slaves operate on a global population (each member of the population is a solution to the TSP for a given set of cities) that is broadcast to them. Each Slave works in isolation to create a specified number of children (representing new tours), and to evaluate them. Once all the sets of children are received by the Master, they are sorted (by efficiency of the tour), some percentage are chosen to be the next generation, and the cycle repeats.

GA1, Figure 8, shows the stochastic value predictions and the actual time, run when the two machines on the cluster had a highly variable availability, and two had low variability. GA2, Figure 9, shows the stochastic value predictions and the actual time run when the PCL cluster had two highly variable machines and two low variability machines.

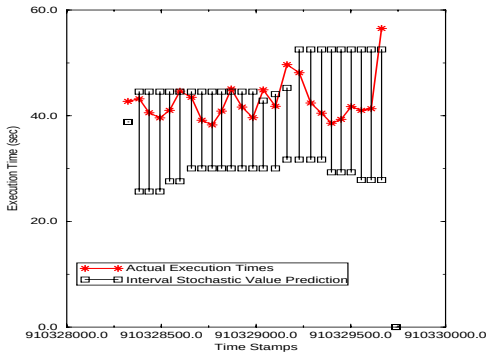

Figure 8. GA1- stochastic value prediction for the GA code.

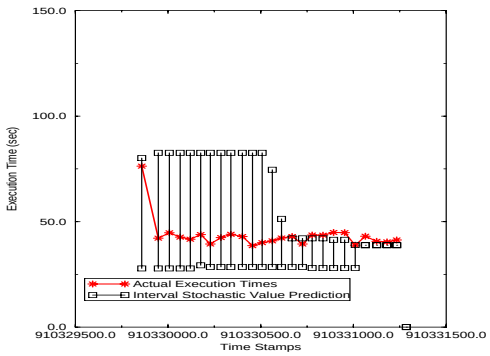

Figure 9. GA2- stochastic value prediction for the GA code.

4.3 Application 3 - LU Benchmark

The LU benchmark is a simulated CFD application that solves a block-lower-triangular/block-upper-triangular system of equations, and is one of the NAS Parallel Benchmarks (NPB) [BHS+95]. This system of equations is the result of an unfactored implicit finite-difference discretization of

the Navier-Stokes equations in three dimensions. The LU benchmark finds lower triangular and upper triangular matrixes such that $L \cdot U = A$ for an original matrix A. It has the feature that it tests the communication aspects of the system well by sending a relatively large number of 5 word messages.

LU1, Figure 10, shows the stochastic value predictions and the actual time, run when two of the machines showed a high variability in availability, and two had a low variability. LU2, Figure 11, shows the stochastic value predictions and the actual time, run when the PCL cluster one high variability machine, one medium variability machine and two low variability machines.

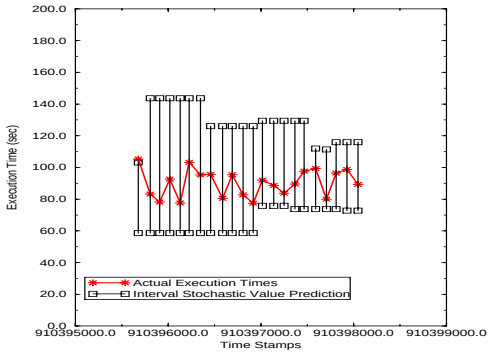

Figure 10. LU1- stochastic value prediction for LU.

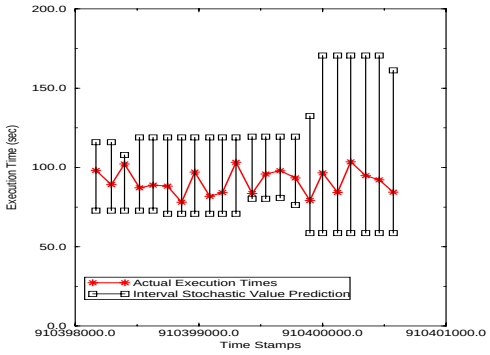

Figure 11. LU2- stochastic value prediction for LU.

4.4 Summary of Experiments

In summary, for the majority of the experiments, we achieved predictions that captured the execution behaviors using histogram representations for the stochastic information. In addition, the shape and percentages of the histogram intervals were close to the actual execution values, and seemed to add valuable information to the prediction. Future work involves examining how these values affect scheduling decisions as well as adaptive methods for defining histograms.

4.5 Comparison of Stochastic Representations

This work has given a detailed approach to using histograms to represent stochastic values. In previous work, we represented stochastic values as distributions [SB97] and as intervals [SB99]. Each approach has advantages and disadvantages. Intervals are the most easily defined since the minimum and maximum value in a data set for a stochastic value are easy to determine. However, outliers in the data can affect the size of the interval, and no details about the shape of the data are included in a simple range. Histograms allow the shape of a stochastic value to be elucidated, and lend themselves to the grouping of values, but even with out approach, can be difficult to accurately define in a practical setting. Distributions can be defined using well understood metrics for the data, but in order to be tractable arithmetically, we must assume that the associated data fits a well-defined and computationally efficient family of distributions, such as the family of normal distributions, which is not always a valid assumption.

References

[Bai72] A. Baines. Introduction. In O. Davies and P. Goldsmith, editors, *Statistical Methods in Research and Production*, Published for Imperial Chemical Industries Limited by Oliver and Boyd, 1972.

[Bha96] Karan Bhatia. Personal communication, 1996.

[BHS+95] D. Bailey, T. Harris, W. Saphir, R. van der Wijngaart, A. Woo, and M. Yarrow. The nas parallel benchmarks 2.0. Technical Report Report NAS-95-020, December 1995.

[Bla80] Leland Blank. *Statistical Procedures for engineering, management, and science*. McGraw-Hill Book Company, 1980.

[Cra68] J. M. Craddock. *Statistics in the Computer Age*. The English Universities Press Limited, 1968.

[LH95] J. Lüthi and G. Haring. Mean value analysis for queuing network models with intervals as input parameters. TR-950101, Institute of applied science, University of Vienna, July 1995.

[LLKS85] Lawler, Lenstra, Kan, and Shmoys. *The Traveling Salesman Problem*. Wiley & Sons, 1985.

[LMH96] J. Lüthi, S. Majumdar, and G.Haring. Mean value analysis for computer systems with variabilities in workload. In *Proc. of IPDS '96*, 1996.

[Lue98] Johannes Luethi. Histogram-based characterization of workload parameters and its consequences on model analysis. In *Proceedings of the Workshop on Workload Characterization in High-Performance Computing Environments, jointly with MASCOTS'98*, 1998.

[MR95] S. Majumdar and R. Ramadoss. Interval-based performance analysis of computing systems. In *Model Analysis and Simulation of Computer and Telecommunication Systems*, 1995.

[MSW81] William Mendenhall, Richard L. Scheaffer, and Dennis D. Wackerly. *Mathematical Statistics with Applications*, pages 4–6. Duxbury Press, 1981.

[Neu90] Arnold Neumaier. *Interval methods for systems of equations*, Basic Properties of Interval Arithmetic. Cambridge University Press, 1990.

[SB97] Jennifer M. Schopf and Francine Berman. Performance prediction in production environments. In *Proceedings of IPPS/SPDP '98*, 1997.

[SB99] Jennifer M. Schopf and Francine Berman. Using stochastic intervals to predict application behavior on contended resources. In *Proceedings of ISPAN 99*, 1999.

[Sch98] Jennifer M. Schopf. *Performance Prediction and Scheduling for Parallel Applications on Multi-User Clusters*. PhD thesis, University of California, San Diego, 1998.

[Sco79] D.W. Scott. On optimal and data-based histograms. *Biometrika*, 66, 1979.

[Ske74] Stig Skelboe. Computation of rational integer functions. *BIT*, 14, 1974.

[Stu26] H. A. Sturges. *Journal of American Statistical Ass.*, 21, 1926.

[TK93] H. Taylor and S. Karlin. *An Introduction to Stochastic Modeling*. Academic Press, 1993.

[Wol97] R. Wolski. Dynamically forecasting network performance to support dynamic scheduling using the network weather service. In *Proc. High Performance Distributed Computing*, August 1997.