Analytic Performance Model for Speculative, Synchronous, Discrete-Event Simulation

Bradley L. Noble
Roger D. Chamberlain

Bradley L. Noble and Roger D. Chamberlain, “Analytic Performance Model for Speculative, Synchronous, Discrete-Event Simulation,” in *Proc. of 14th Workshop on Parallel and Distributed Simulation*, June 2000, pp. 35-44.

Southern Illinois University Edwardsville
and
Washington University

Analytic Performance Model for Speculative, Synchronous, Discrete-Event Simulation

Bradley L. Noble
Dept. of Electrical and Computer Engineering
Southern Illinois University Edwardsville
Edwardsville, IL
bnoble@siue.edu

Roger D. Chamberlain
Computer and Communications Research Center
Department of Electrical Engineering
Washington University, St. Louis, MO
roger@ccrc.wustl.edu

Abstract

Performance models exist that reliably describe the execution time and efficiency of parallel discrete-event simulations executed in a synchronous iterative fashion. These performance models incorporate the effects of processor heterogeneity, other processor load due to shared computational resources, application workload imbalance, and the use of speculative computation. This includes modeling the effects of predictive optimism, a technique for improving the accuracy of speculative assumptions. We extend these models to incorporate correlated workloads across the set of processors and validate the models with two different applications.

1. Introduction

Speculative computation has received a great deal of attention in the parallel computing community as a technique for balancing computational load and masking latencies in interprocessor communications [8]. It has even found its way into processor design at the instruction level, with proposals for *value prediction*, speculating the results of individual instructions or basic code blocks [11]. In discrete-event simulation, parallel algorithms that perform computation in a speculative manner are generally referred to as *optimistic* algorithms.

While both synchronous and asynchronous optimistic algorithms exist, our interest is in synchronous algorithms. This is due to a desire to avoid the inconsistent (and sometimes inexplicable) performance associated with many asynchronous protocols. Lin and Lazowska [13] coined the term "S phenomenon" to describe the observation that speedup curves for an optimistic asynchronous algorithm often have several local minima and maxima. This observation was made over a large set of different simulation applications [4, 10, 12, 22]. In addition, synchronous algorithms have an inherent simplicity and ease of implementation that is not present in asynchronous techniques.

As with many other algorithms, there is a tradeoff between simplicity and performance; the simplicity of the synchronous algorithm comes with a potential cost in performance. If frequent synchronizations are required, the algorithm becomes more fine grained. Since the critical path lies with the slowest processor at each iteration, idle time can accumulate at the other processors and the total execution time is lower bounded by the execution time of the slowest processor in each iteration. In an attempt to alleviate these performance concerns for synchronous discrete-event simulation, techniques used in asynchronous simulation algorithms (e.g., speculative computation) have been applied to the synchronous algorithm, while retaining the iterative nature of the algorithm.

In [18], we described a performance model for synchronous iterative algorithms that incorporates the effects of speculative computation. Included in this model is the degree to which speculative computations are correct (i.e., what is the impact of predictive optimism). This model assumed that the computational workload is relatively independent across the processors. However, there are many simulation applications (e.g., VLSI logic simulation) for which the workload is highly correlated. Here, we present extensions to this model that enable accurate performance prediction for correlated workloads. This new model is validated using two applications, queueing network simulation and VLSI logic simulation.

2. Speculative Computation and Predictive Optimism

The model developed here is not restricted to discrete-event simulation applications, but can be applied to any synchronous iterative algorithm. Synchronous iterative al-

gorithms include many of the compute intensive numerical methods used in science and engineering applications [1]. Figure 1 illustrates a typical set of iterations of a synchronous iterative algorithm executing on four processors (labeled 1 through 4). An iteration can be seen as consisting of 3 phases:

1. Computation – performing the computational tasks associated with the application.
2. Idle – time between first and last processor to complete work in an iteration.
3. Synchronization – time to complete the barrier synchronization operation.

Computation starts on all processors immediately following the barrier synchronization. During this phase, each processor executes all the tasks assigned to it that iteration. For discrete-event simulation, this consists of processing simulation events. Interprocessor data communication may be concurrent with computation. At the end of the computation phase, each processor enters a barrier and waits for its completion. The *idle* phase is a result of variation in computation times between processors due to imbalances in workload as the algorithm progresses, multitasking other unrelated processes (background load), or processor heterogeneity. *Synchronization* time is determined by the communication performance of the parallel platform in completing the barrier synchronization. After the barrier synchronization completes, the processors proceed to the next iteration, repeating the cycle until the algorithm completes. In the performance model for synchronous iterative algorithms described in [20], the quantity of computation to be performed on each processor during each iteration is modeled by a random variable with a known stationary distribution. The mean completion time for an iteration executing on P processors can then be modeled as the expectation of the maximum of P instances of the random variable.

Speculative computation utilizes the idle phase of the above algorithm by allowing processing to proceed into future iterations. While waiting for the barrier synchronization to complete, computation progresses speculatively, with the hope that a message arrival from a remote processor does not subsequently invalidate the computation. Once the barrier synchronization is complete, the speculated computation is tested for correctness and either committed or discarded.

To support processing during the execution of the barrier synchronization, a fuzzy barrier implementation is used [9]. Processors signal their willingness to complete the barrier and, rather than blocking, proceed to compute speculatively. A "barrier complete" signal indicates the end of the current iteration. The execution time line, illustrated in Figure 2, shows the speculative computation occurring during the idle

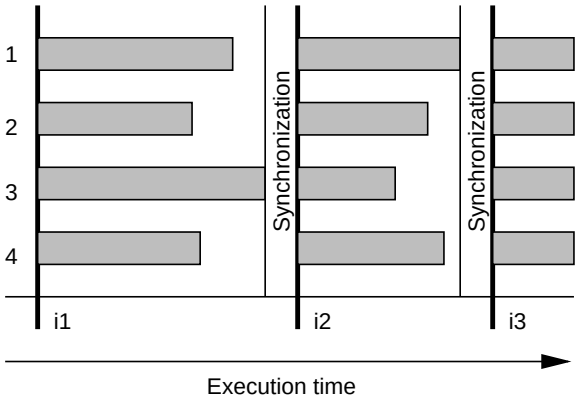

Figure 1. Synchronous iterative algorithm execution. The horizontal bars represent computation during each iteration.

times and while waiting for the barrier to complete. Note that the time required to complete iteration 2 is less than in the previous figure, since some of the computation has been completed during the otherwise idle time of iteration 1.

Mehl [14] proposed essentially this technique in the context of a conservative asynchronous algorithm, but did not report on its performance. We previously reported a set of empirical performance results in [17, 19]. Dickens et al. [3] present a performance model for a similar algorithm that predicts performance gains over a purely conservative synchronous algorithm. Steinman's Breathing Time Buckets algorithm [21] has been implemented in the SPEEDES environment and exhibits good performance on a pair of simulation models (queueing networks and proximity detection). In [18], the model of [20] is extended to incorporate the effects of speculation. A recurrence relation is developed that relates the original workload distribution, the distribution of time available for speculation, and the resulting workload distribution.

Predictive optimism is a technique for improving the accuracy of the guesses used to guide speculative computation. In traditional optimistic discrete-event simulation algorithms, the standard optimistic assumption is that if a message has not arrived on an input channel, none will arrive, and processing can continue assuming the channel is unchanged.

In predictive optimism, information theoretic techniques are used to improve the accuracy of this assumption. A predictor is placed an each input channel, and the predictor retains historical information about the messages on the channel. If a message has not arrived on a channel, the predictor can be interrogated to determine if (and when) a message is likely to arrive. If the answer is no, processing proceeds as before. If the answer is yes, speculative computation is

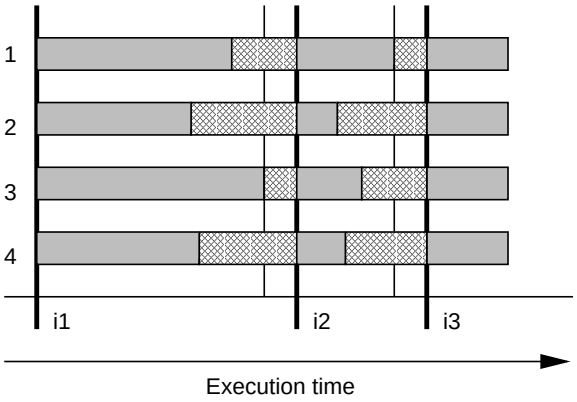

Figure 2. Speculative computation execution time line. The crosshatched areas represent speculative computing, potentially decreasing the computation needed during the subsequent iteration.

suspended, since the results are likely to be discarded, and the bookkeeping overhead of managing the speculation can be diminished.

We have previously investigated predictive optimism in the context of VLSI systems simulation [16]. In that study, the "no message arrival" assumption was compared to a first order finite state predictor and an incremental parsing predictor based on Lempel-Ziv data compression techniques [5]. Both predictors significantly decreased (or eliminated) cases where the standard assumption performed poorly (i.e., was wrong most of the time). In [6], Ferscha describes a similar mechanism, called probabilistic direct optimism control, and reports peformance for four different predictors of message interarrival time: arithmetic mean, exponential smoothing, approximated median, and an autoregressive moving average process predictor. The analytic model presented here includes mechanisms for evaluating the performance implications of variations in prediction accuracy, in order to investigate the usefulness of predictive optimism techniques.

3. Model Development

The set of variables used in the performance model is summarized in Table 1. A more complete definition of each variable is given in the text near the first use of the variable.

The execution time of a synchronous iterative algorithm requiring I iterations and running on a set of P processors can be modeled by a function consisting of three distinct terms. In any particular iteration i, there is a portion of work which is serial in nature. The first term represents the time required to complete the portion which cannot be parallelized, which we denote as $t_{s,i}$. Each processor j has some assigned work to be performed during the parallel portion of each iteration. The time for each processor to complete this work is denoted by $t_{p,i,j}$. However, the parallel portion of each iteration is not complete until the last processor completes its assigned work. This gives the second term as $\max_{1 \le j \le P}(t_{p,i,j})$. Finally, the time required for the overheads associated with the parallel algorithm itself during iteration i are denoted by $t_{ov,i}$.

Table 1. Parameters for performance model.

Param.	Definition
R_P	application run time with P processors
P	number of processors
I	number of iterations
t_s	serial computation time per iteration
t_p	parallel computation time per iteration
t_{ov}	parallelism overhead time per iteration
$w_{i,j}$	parallel work during iteration i on processor j, no speculation
W	vector-valued random process representing work each iter., no speculation
W_j	component of W representing work on processor j
$v_{i,j}$	parallel work during iteration i on processor j, with speculation
V	vector-valued random process representing work each iter., with speculation
V_j	component of V representing work on processor j
$s_{i,j}$	work available to speculate during iteration i on processor j
S	vector-valued random process representing available speculation each iter.
S_j	component of S representing speculation on processor j
r	speculation success ratio: fraction of speculative computation that is correct

Combining these terms gives a model for the execution time as

$$R_P = \sum_{i=1}^{I} \left[t_{s,i} + \max_{1 \le j \le P} t_{p,i,j} + t_{ov,i} \right] \quad (1)$$

Although this equation effectively models the execution time, it requires specific knowledge of each individual iteration. By treating each of the terms as an i.i.d.[1] random process and taking the expected value, the references to a

[1] Independent and identically distributed.

specific iteration are eliminated. A new expression for runtime is given by

$$\begin{aligned} R_P &= \sum_{i=1}^{I} \left(E\left[t_{s,i}\right] + E\left[\max_{1\le j\le P} t_{p,i,j}\right] + E\left[t_{ov,i}\right] \right) \\ &= I\left(t_s + E\left[\max_{1\le j\le P} t_{p,j}\right] + t_{ov} \right) \end{aligned} \quad (2)$$

Previous work has shown this model to be effective for estimating run time for several different types of synchronous iterative algorithms [20]. By examining the effects of speculation on the terms in this model, we incorporate both speculative computation and the impact that successful speculation has on the run time.

3.1. Speculative Workload Characterization

For the discrete-event simulation applications we are interested in, both the serial term $t_{s,i}$ and the parallel overhead term $t_{ov,i}$ do not vary significantly between iterations. As such, these terms can be treated as constants. We focus on characterizing $E\left[\max_{1\le j\le P} t_{p,i,j}\right]$ for the both the initial workload without speculative computation and for the resulting workload with speculative computation.

Let us define $w_{i,j}$ as the work to be completed on processor j during iteration i without speculative computation. Assuming the units of work are relatively constant in time (e.g., event evaluations with similar computational complexity), $t_{p,i,j}$ will be proportional to $w_{i,j}$. When speculative computation is performed, it will take work away from future iterations whenever a processor completes before the barrier synchronization. We will define this new workload as $v_{i,j}$, the work to be completed on processor j during iteration i with speculation. In this case, $t_{p,i,j}$ is proportional to $v_{i,j}$ which, by definition, must be less than or equal to $w_{i,j}$.

The development of $v_{i,j}$ from $w_{i,j}$ can be made by examining a specific iteration, i, of a synchronous algorithm that incorporates speculative computation. Examining Figure 2, we determine that the work to be completed during iteration i is $\max_{1\le j\le P} v_{i,j}$. Therefore, the amount of work that can be speculated on processor j during iteration i is given by

$$s_{i,j} = \max_{1\le j\le P}(v_{i,j}) - v_{i,j}, \quad (3)$$

provided we limit speculation to one iteration into the future. This yields a recursive formulation that relates $v_{i+1,j}$ to $v_{i,j}$:

$$v_{i+1,j} = w_{i+1,j} - r s_{i,j} \quad (4)$$

with initial condition

$$v_{0,j} = w_{0,j}. \quad (5)$$

The scalar r that is introduced in equation (4) represents the speculation success ratio, or the fraction of the speculated work that was successfully committed. Substituting (3) into (4) gives:

$$v_{i+1,j} = w_{i+1,j} - r\left(\max_{1\le j\le P}(v_{i,j}) - v_{i,j}\right) \quad (6)$$

Although the case outlined above is limited to one iteration into the future, a similiar expression can be developed relating $v_{i+2,j}$ to $v_{i+1,j}$ and $v_{i,j}$ for speculation two iterations into the future. Similiarly, this can be extended to $i+n$ iterations into the future.

These expressions can be used to empirically evaluate $v_{i,j}$ for a specific instance where $w_{i,j}$ is known, however, it does not generalize beyond the circumstances where one has knowledge of workload during individual iterations. To accomplish this, we develop a stochastic workload model.

3.2. Stochastic Workload Model

We model the workload without speculation as an i.i.d., vector-valued random process W, with component W_j representing the work to be completed on processor j. Although described in other terms, this is consistent with the model of [18]. In [18], however, the components of W were assumed to be independent. Here, we model the vector W as correlated, with a given joint density function $f_W(x_1, ..., x_P)$. Essentially, this implies that we are modeling the workload at each iteration on each processor, $w_{i,j}$, as independent in the i dimension (iterations), and correlated in the j dimension (processors).

We model the workload with speculation as an i.i.d., vector-valued random process V. As above, the component V_j represents the work to be completed on processor j. Again, the components of V are assumed to be correlated, and an important piece of the model is evaluating the unknown joint density function $f_V(x_1, ..., x_P)$.

Provided a stationary distribution for V exists, we can define the vector-valued random process S as the amount of work that can be speculated each iteration, where:

$$S = E\left[\max_{1\le j\le P}(V_j)\right]\mathbf{1} - V \quad (7)$$

and

$$V = W - rS. \quad (8)$$

Substitution of (7) into (8) yields:

$$V = W + rV - rE\left[\max_{1\le j\le P}(V_j)\right]\mathbf{1} \quad (9)$$

Although (9) gives the fixed point conditions that V must meet, it does not directly support the calculation of its distribution. However, by adding a superscript k to equation

(9), we define an iterative approach which may be used in an attempt to solve for the distribution of V.

$$V^{k+1} \quad = \quad W + rV^k - rE\left[\max_{1\leq j\leq P}(V_j^k)\right]\mathbf{1} \quad (10)$$

For the initial iteration, $V^0 = W$. Convergence is reached when the distributions of V^{k+1} and V^k are equal.

Clearly, there is no guarantee that (10) will converge to a stationary distribution for V. In some applications, the speculation process itself may be highly unstable, even oscillatory. If convergence is achieved, the result is a viable steady state distribution of the parallel workload.

4. Model Validation

The validation of the performance model is in three stages. We first discuss how to evaluate the model (i.e., compute the distribution of the random process V). We next investigate the correctness of the distribution of V. We then investigate the accuracy of the middle term in equation (2), which is represented by $E[\max_{1\leq j\leq P} W_j]$ for executions that do not exploit speculation and by $E[\max_{1\leq j\leq P} V_j]$ for executions that do employ speculative computation. To perform the validation, we will use empirical data from two discrete-event simulation applications.

The first application is a closed queueing network simulation of the style used in [7] and [15] to investigate the performance of asynchronous algorithms. The topology is a regular network with FCFS queueing discipline, the service requirements are exponentially distributed with a specified minimum service time, and the routing probabilities are uniformly distributed to each of the neighboring queueing stations. The second application is a VLSI logic simulation of several of the ISCAS-89 sequential benchmark circuits [2]. For each circuit, a gate-level simulation is executed using a unit delay timing model driven with random input vectors.

Our validation methodology starts with a set of trace data that directly represents $w_{i,j}, 1 \leq i \leq I, 1 \leq j \leq P$. Workload data from both the queueing network simulation and the logic simulation applications were recorded for a variety of queueing networks and logic circuits respectively. Example trace data from both applications (queuing network simulation `q8000pm3` and VLSI logic simulation `s9234`) on four processors ($P = 4$) is shown in Figures 3 and 4. The units of work shown are simulation event counts.

Figures 5 and 6 show a normalized histogram of $w_{i,j}$ for the example trace data in Figure 3 and Figure 4 respectively. These histograms will be used to approximate the joint distribution of workload without speculation, $f_W(x_1, ..., x_P)$. It is useful to note that for the queueing network simulation, the histogram workload for each processor is very similiar

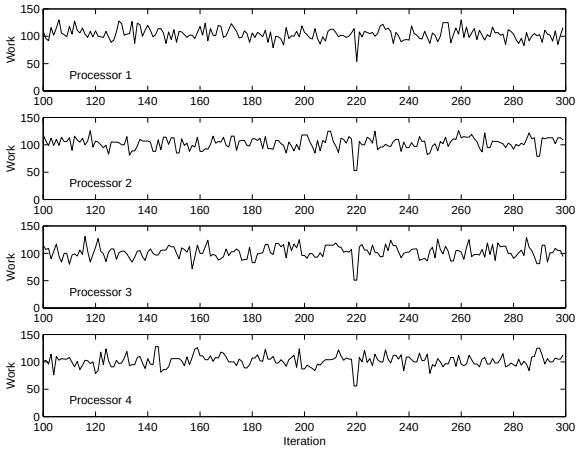

Figure 3. Example trace data for queueing network simulation application (QNS).

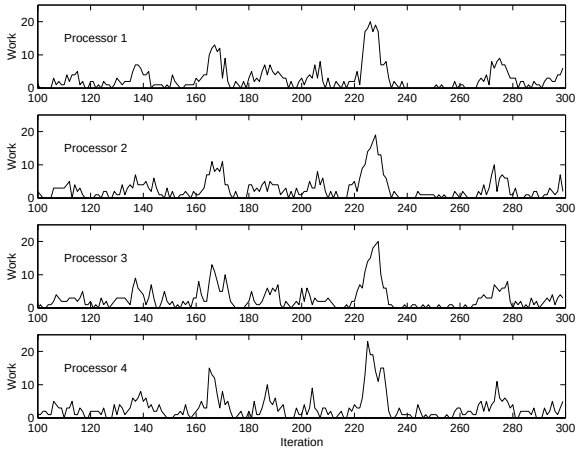

Figure 4. Example trace data for VLSI logic simulation application (VLS).

in shape. This implies a relatively uniform workload distribution across the processors. This view is also supported by a statistical analysis of the queueing networks. For example, the correlation coefficients (across j) of $w_{i,j}$ range from 0.20 to 0.24 for queueing network `q8000pm3`.

In the logic simulation case, although each histogram has the same basic shape, there are distinct differences in each distribution. The logic simulator workload, shown in Figures 4 and 6, exhibits a large degree of correlation across the processors. Again, this view is supported by a statistical analysis of the logic circuits and is to be expected since the logic circuits being simulated are governed by a clock. Iterations immediately following the clock signal have a high degree of activity which trails off in iterations later in the clock period. The correlation coefficients of $w_{i,j}$ range from 0.70 to 0.76 for circuit `s9234`.

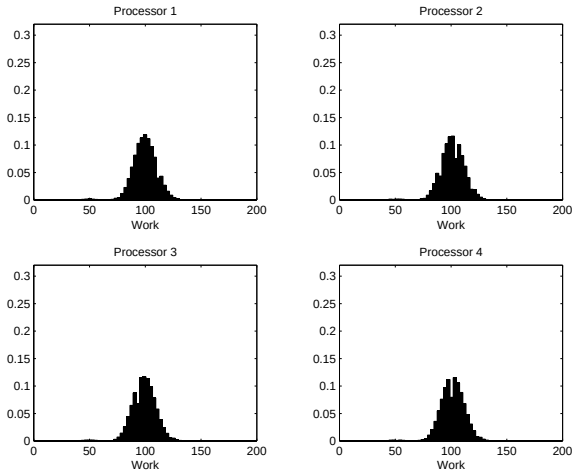

Figure 5. Normalized hist. of $w_{i,j}$ for QNS.

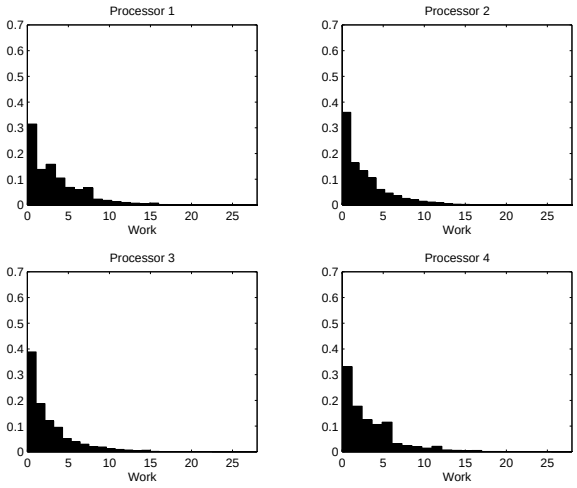

Figure 6. Normalized hist. of $w_{i,j}$ for VLS.

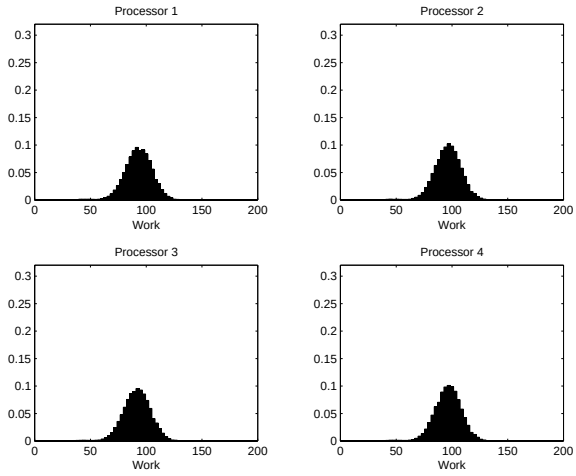

Figure 7. Dist. of V, $r = 0.5$, for QNS.

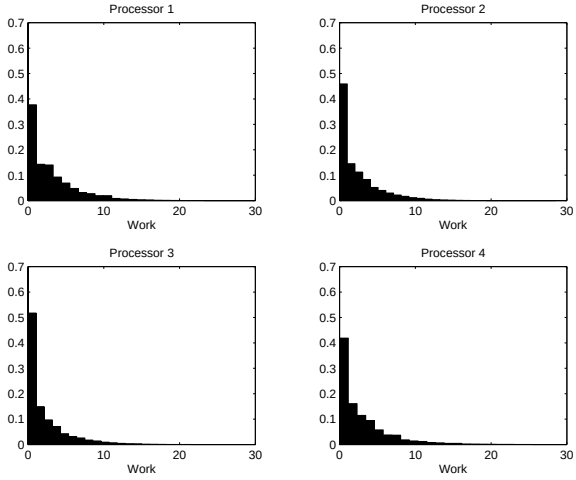

Figure 8. Dist. of V, $r = 0.5$, for VLS.

4.1. Model Evaluation

To evaluate the model, we use the empirical histogram data to represent the distribution of the random process W. However, to preserve the joint workload dependencies, this histogram must be P-dimensional. Equation (10) is then numerically evaluated using samples drawn from this approximation to the distribution of V^k to obtain a new P-dimensional histogram approximation of the distribution of V^{k+1}. This is repeated until the distributions of V^{k+1} and V^k have converged.

For the sample queueing network and logic circuit workloads of Figures 5 and 6 the resulting distribution of V is shown in Figures 7 through 10 for two different values of the speculation success ratio, r. Once $f_V(x_1, ..., x_P)$ is known, we can calculate $E[\max_{1 \le j \le P} V_j]$. These values are tabulated in Table 2.

4.2. Model Analysis

To quantify the accuracy of the model, we compare the distribution of the random process V to empirical results. The empirical data for the two applications, $w_{i,j}$, is evaluated using equation (6) to determine $v_{i,j}$. This evaluation constitutes a trace-driven simulation of the speculative execution algorithm. Histograms of the resulting workloads are shown in Figures 11 through 14 for the same two values of the success ratio, r. A good match between the histograms of Figures 7 to 10 with those in Figures 11 to 14 implies that the analytic model is effective in matching the empirical results in its estimate of $f_V(x_1, ..., x_P)$.

For the final stage of evaluating the model, we are interested in comparing $E[\max_{1 \le j \le P}(W_j)]$ and $E[\max_{1 \le j \le P}(V_j)]$ to the mean (across iterations) of $\max_{1 \le j \le P}(w_{i,j})$ and the mean (across iterations) of $\max_{1 \le j \le P}(v_{i,j})$. This comparison lets us quantify the ac-

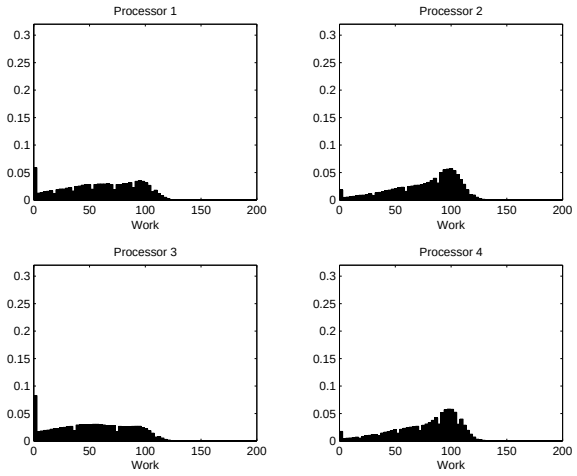

Figure 9. Dist. of V, $r = 1.0$, for QNS.

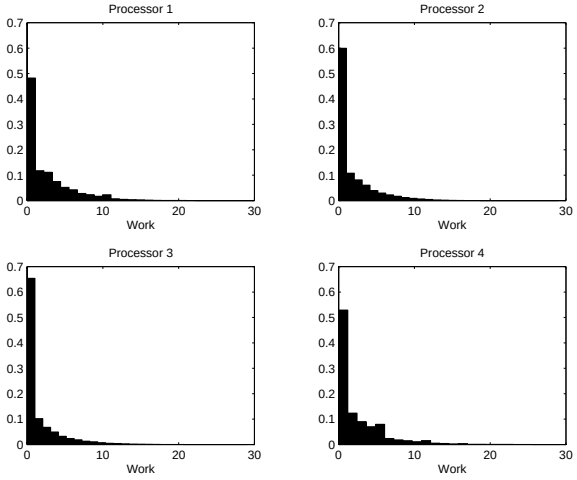

Figure 10. Dist. of V, $r = 1.0$, for VLS.

curacy of the model at characterizing the workload both with and without speculation. Tables 3 and 4 provide such a comparison for a variety of queueing networks and logic circuits with different numbers of processors, P, and values of r. Figures 15 and 16 plot these results for the case where speculation is present.

The first two data columns in the tables correspond to parallel workload when no speculation is present. The first column (labeled $E[\max_{1 \le j \le P} W_j]$) corresponds to the modeled results, and the second column (labeled mean of $\max_{1 \le j \le P} w_{i,j}$) corresponds to the empirical results. To calculate the empirical results, the max operator was applied at each iteration, and the mean was taken across iterations. As seen in both tables, the model of the workload, W, strongly matches the empirical workload, $w_{i,j}$, for both the queueing network and logic simuation for a variety of topologies and circuits. This points to the fact that modeling the original workload via an i.i.d. random process is a

Table 2. Model results for $P = 4$.

	$E[\max_{1 \le j \le P} V_j]$ for `q8000pm3`	$E[\max_{1 \le j \le P} V_j]$ for `s9234`
$r = 0.5$	106.17	4.39
$r = 1.0$	102.02	4.08

reasonable approach.

The last two columns in the tables correspond to parallel workload when speculation is present. The column labeled $E[\max_{1 \le j \le P} V_j]$ corresponds to the modeled results, and the column labeled mean of $\max_{1 \le j \le P} v_{i,j}$ corresponds to the empirical results. This data is also plotted in Figures 15 and 16. In the queueing network simulation, the model of the workload with speculation, V, closely matches the empirically calculated speculative workload, $v_{i,j}$, for all queueing networks and for different values of the speculation success ratio, r. These results are what we expected for the queueing network application, since the previous model (of [18]) worked well under these circumstances.

The limitations of the previous model were apparent for the VLSI logic simulation application. The independence assumption (across processors) caused the earlier model to consistently overstate the execution time. Here, however, we have an excellent match between modeled and empirical results. Explicitly modeling the correlation in workload across the processors has enabled us to reliably predict the performance of an application that is not well characterized when an independence assumption is made.

5. Summary and Conclusions

This paper has presented and validated a performance model for synchronous iterative algorithms that include speculative computation. The model was applied to two discrete-event simulation applications: queueing network simulation and VLSI logic simulation. Unlike earlier models, the model presented here effectively handles circumstances where the parallel workload across the processors is highly correlated, such as is the case for VLSI logic simulation.

We believe the usefulness of this model will come in two forms. First, any good analytic model can be used to better understand the performance issues associated with a parallel application execution. This can point to improvements in efficiency via changes in the algorithm, execution platform modifications, or various other forms of performance tuning.

Second, since the analytic model only requires empirical data that can be collected from a serial execution of the simulation, it can be used to predict the performance of a new application prior to parallel implementation. This can

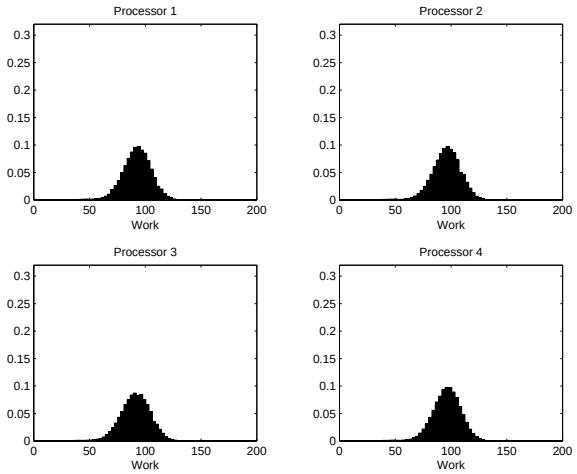

Figure 11. Hist. of $v_{i,j}$, $r = 0.5$, for QNS.

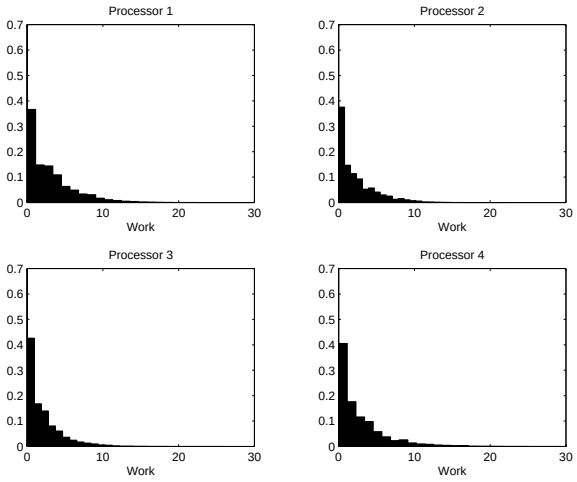

Figure 12. Hist. of $v_{i,j}$, $r = 0.5$, for VLS.

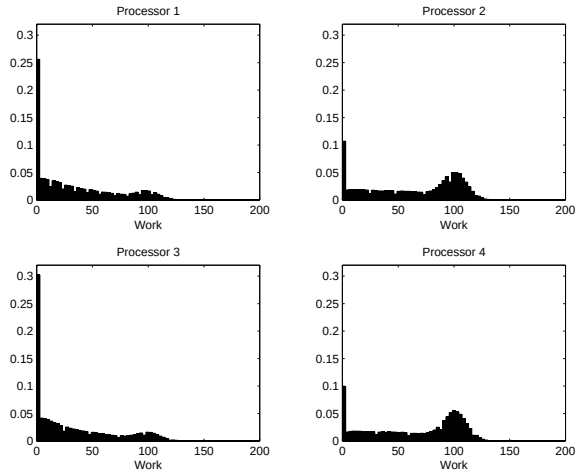

Figure 13. Hist. of $v_{i,j}$, $r = 1.0$, for QNS.

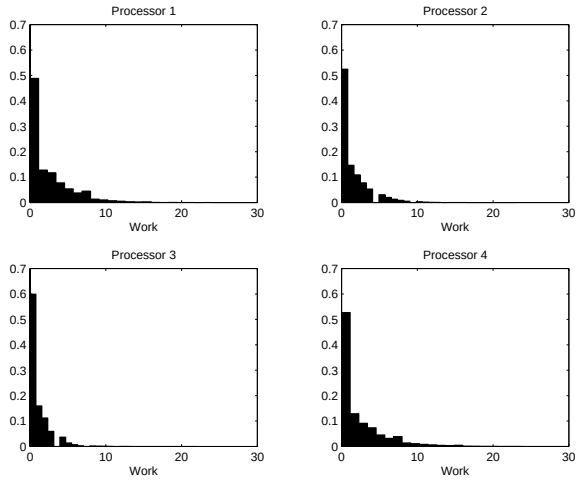

Figure 14. Hist. of $v_{i,j}$, $r = 1.0$, for VLS.

help a simulation practitioner to evaluate whether or not it is worth the effort to implement a synchronous parallel version of the application.

References

[1] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, NJ, 1989.

[2] F. Brglez, D. Bryan, and K. Kozminski. Combinational Profiles of Sequential Benchmark Circuits. In *Proc of the Int'l Symp. on Circuits and Systems*, pages 1929–1934, May 1989.

[3] P. M. Dickens, D. M. Nicol, P. F. Reynolds, Jr., and J. M. Duva. The Impact of Adding Aggressiveness to a Non-Aggressive Windowing Protocol. In *Proc. of the 1993 Winter Simulation Conf.*, pages 731–739, December 1993.

[4] M. Ebling, M. Di Loreto, M. Presley, F. Wieland, and D. Jefferson. An Ant Foraging Model Implemented on the Time Warp Operating System. In *Proc. of the SCS Multiconference on Distributed Simulation*, pages 21–28, March 1989.

[5] M. Feder, N. Merhav, and M. Gutman. Universal Prediction of Individual Sequences. *IEEE Trans. on Information Theory*, 38(4):1258–1270, July 1991.

[6] A. Ferscha. Probabilistic Adaptive Direct Optimism Control in Time Warp. In *Proc. of 9th Workshop on Parallel and Distributed Simulation*, pages 120–129, June 1995.

[7] R. M. Fujimoto. Performance Measurements of Distributed Simulation Strategies. *Trans. of Society for Computer Simulation*, 6:89–132, 1989.

[8] V. Govindan and M. Franklin. Speculative Computation: Overcoming Communication Delays. In *Proc. of*

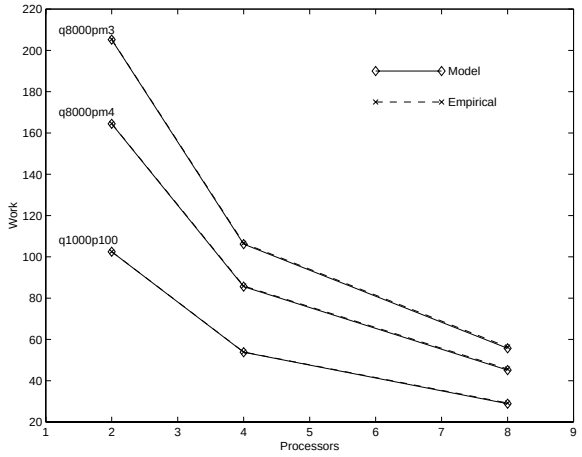

Figure 15. Speculative model validation for queueing network simulation.

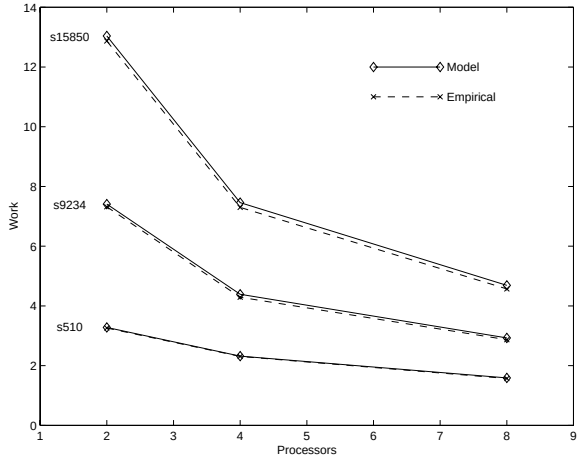

Figure 16. Speculative model validation for VLSI logic simulation.

the 1994 Int'l Conf. on Parallel Processing, volume III, pages 12–16, August 1994.

[9] R. Gupta. The Fuzzy Barrier: A Mechanism for the High Speed Synchronization of Processors. In *Proc. of the Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, pages 54–63, April 1989.

[10] P. Hontalas, B. Beckman, M. Di Loreto, L. Blume, P. Reiher, K. Sturdevant, L. Van Warren, J. Wedel, F. Wieland, and D. Jefferson. Performance of the Colliding Pucks Simulation on the Time Warp Operating System (Part 1: Asynchronous Behavior and Sectoring). In *Proc. of the SCS Multiconference on Distributed Simulation*, pages 3–7, March 1989.

[11] J. Huang and D. Lilja. Exploiting Basic Block Value Locality with Block Reuse. In *Proc. of 5th Int'l Symp. on High Performance Computer Architecture*, pages 106–114, January 1999.

[12] D. Jefferson et al. Distributed Simulation and the Time Warp Operating System. In *Proc. of the 11th ACM Symp. on Operating Systems Principles*, 1987.

[13] Y.-B. Lin and E. D. Lazowska. Processor Scheduling for Time Warp Parallel Simulation. In *Proc. of the SCS Multiconference on Advances in Parallel and Distributed Simulation*, pages 11–14, January 1991.

[14] H. Mehl. Speedup of Conservative Distributed Discrete Event Simulation Methods by Speculative Computing. In *Proc. of the SCS Multiconference on Advances in Parallel and Distributed Simulation*, pages 163–166, January 1991.

[15] D. M. Nicol. High Performance Parallelized Discrete-Event Simulation of Stochastic Queueing Networks. In *Proc. of the 1988 Winter Simulation Conf.*, 1988.

[16] B. L. Noble and R. D. Chamberlain. Predicting the Future: Resource Requirements and Predictive Optimism. In *Proc. of 9th Workshop on Parallel and Distributed Simulation*, pages 157–164, June 1995.

[17] B. L. Noble and R. D. Chamberlain. Performance of Speculative Computation in Synchronous Parallel Discrete-Event Simulation on Multiuser Execution Platforms. In *Proc. of the 8th IASTED Int'l Conf. on Parallel and Distributed Computing and Systems*, pages 489–494, October 1996.

[18] B. L. Noble and R. D. Chamberlain. Performance Model for Speculative Simulation Using Predictive Optimism. In *Proc. of 32nd Hawaii Int'l Conf. on System Sciences*, January 1999.

[19] B. L. Noble, G. D. Peterson and R. D. Chamberlain. Performance of Synchronous Parallel Discrete-Event Simulation. In *Proc. of 28th Hawaii Int'l Conf. on System Sciences*, Vol. II, pages 185–186, January 1995.

[20] G. D. Peterson and R. D. Chamberlain. Parallel Application Performance in a Shared Resource Environment. *Distributed Systems Engineering*, 3:9-19, 1996.

[21] J. S. Steinman. SPEEDES: A Multiple-Synchronization Environment for Parallel Discrete-Event Simulation. *Int'l Journal in Computer Simulation*, 2:251–286, 1992.

[22] F. Wieland, L. Hawley, A. Feinberg, M. Di Loreto, L. Blume, P. Reiher, B. Beckman, P. Hontalas, S. Bellenot, and D. Jefferson. Distributed Combat Simulation and Time Warp: The Model and Its Performance. In *Proc. of the SCS Multiconference on Distributed Simulation*, pages 14–20, March 1989.

Table 3. Model validation for QNS.

Queueing network	P	r	$E[\max_{1\le j\le P} W_j]$	mean of $\max_{1\le j\le P} w_{i,j}$	$E[\max_{1\le j\le P} V_j]$	mean of $\max_{1\le j\le P} v_{i,j}$
q8000pm3	2	0.5	208.42	208.38	205.19	205.35
		1.0	208.36	208.38	201.96	201.96
	4	0.5	110.45	110.43	106.17	106.56
		1.0	110.42	110.43	102.02	102.16
	8	0.5	60.06	60.05	55.70	56.29
		1.0	60.06	60.05	51.46	51.96
q8000pm4	2	0.5	167.27	167.28	164.43	164.58
		1.0	167.28	167.28	161.65	161.57
	4	0.5	89.38	89.36	85.55	85.90
		1.0	89.37	89.36	81.79	81.91
	8	0.5	49.04	49.04	45.15	45.68
		1.0	49.05	49.04	41.40	41.85
q1000p100	2	0.5	104.76	104.77	102.46	102.49
		1.0	104.78	104.77	100.16	99.76
	4	0.5	56.80	56.80	53.78	53.91
		1.0	56.80	56.80	50.74	50.59
	8	0.5	31.87	31.87	28.86	29.20
		1.0	31.86	31.87	25.98	26.25

Table 4. Model validation for VLS.

Logic circuit	P	r	$E[\max_{1\le j\le P} W_j]$	mean of $\max_{1\le j\le P} w_{i,j}$	$E[\max_{1\le j\le P} V_j]$	mean of $\max_{1\le j\le P} v_{i,j}$
s38584	2	0.5	92.41	92.30	91.32	90.79
		1.0	91.91	92.30	90.17	89.75
	4	0.5	49.30	49.51	47.84	47.48
		1.0	49.58	49.51	47.32	45.95
	8	0.5	27.49	27.50	26.08	25.17
		1.0	27.44	27.50	25.33	23.82
s15850	2	0.5	13.57	13.59	13.04	12.87
		1.0	13.57	13.59	12.73	12.41
	4	0.5	8.22	8.22	7.46	7.30
		1.0	8.20	8.22	7.05	6.58
	8	0.5	5.46	5.47	4.69	4.57
		1.0	5.46	5.47	4.33	4.05
s9234	2	0.5	7.92	7.92	7.41	7.32
		1.0	7.95	7.92	7.13	6.87
	4	0.5	4.98	4.98	4.39	4.29
		1.0	4.99	4.98	4.08	3.84
	8	0.5	3.58	3.59	2.93	2.87
		1.0	3.59	3.59	2.62	2.47
s510	2	0.5	3.53	3.53	3.28	3.26
		1.0	3.53	3.53	3.15	3.09
	4	0.5	2.70	2.70	2.32	2.31
		1.0	2.69	2.70	2.14	2.09
	8	0.5	2.01	2.01	1.59	1.57
		1.0	2.01	2.01	1.40	1.31