Analytic Modeling of Burstiness and Synchronization Using Approximate MVA

Derek L. Eager
Daniel J. Sorin
Mary K. Vernon

Technical Report #1391

December 1998

Analytic Modeling of Burstiness and Synchronization Using Approximate MVA *

Derek L. Eager

Department of Computer Science
University of Saskatchewan
eager@cs.usask.ca

Daniel J. Sorin Mary K. Vernon

Computer Sciences Department
University of Wisconsin - Madison
{sorin,vernon}@cs.wisc.edu

Abstract

Motivated by a recent analytical model for evaluating the performance of shared memory systems with complex modern processors, this paper develops and validates new approximate mean value analysis (AMVA) techniques for (1) estimating mean residence time at a FCFS queue with high variance in service times, (2) modeling bursty arrivals and computing mean waiting time at "downstream" queues in the system, and (3) computing mean delays for software locks that are acquired while occupying the processor and held while queueing for the processor and memory system resources. These techniques build on, and are compared against, previous analytic techniques for FCFS servers that have high service time variance and for systems with simultaneous resource possession. The new techniques greatly increase the accuracy, robustness, and range of application of the shared memory multiprocesssor model. The techniques increase the model complexity, but do not appreciably degrade model solution time. The improved analytic model still computes performance estimates several orders of magnitude faster than detailed simulation.

1 Introduction

A recent paper [16] develops an AMVA-based analytic model for evaluating shared memory systems containing complex modern processors. These processors have interesting features such as parallel, out-of-order, and speculative instruction execution. The analytic model, referred to as the "ISCA98 model" in this paper, provides estimates of application execution time, or system throughput measured in instructions per cycle, that are in fairly close agreement with detailed simulation results for the Splash applications that were used to validate the model.

The ISCA98 model input parameters are derived from a simulation of the application running on a given parallel architecture, but the analytic model can then predict, as accurately as the detailed simulator, the performance impact of modifying various memory system components such as the number of memory modules per node or the speed of the memory bus. Thus, the analytic model might be used to quickly cull the memory system design space, thereby greatly reducing the size of the design space that needs to be explored using simulation. Moreover, the paper gives examples that illustrate how the measured input parameters provide insight into the behavior of applications with respect to the memory system architecture. The insight derived from the measured input parameters might, by itself, be used to tune the applications, the compiler, or the architecture.

In this paper, we consider two important modeling issues that were not fully addressed in the ISCA98 model. First, we consider the question of how to accurately model the highly bursty memory requests that are issued from the processors to the memory system. Second, we consider how to compute mean lock access delays using fundamental input parameters that are insensitive to changes in the memory system architecture. In addressing these issues, we make the following contributions:

- We develop new robust approximate mean value analysis (AMVA) techniques for estimating the mean residence time and mean residual life at a FCFS queue that has a high coefficient of variation (CV) in service time.
- We develop a new AMVA technique for modeling the bursty arrivals and estimating the mean waiting time at the resources that are "downstream" from the high-CV FCFS server, such as the memory system resources (e.g., memory bus and directory) that are local to the processor in the ISCA98 model.

*This research is supported in part by DARPA/ITO under Contract N66001-97-C-8533, the National Science Foundation under Grants CDA-9623632, MIP-9625558, and ACI 9619019, and by the Natural Sciences and Engineering Research Council of Canada under Grant OGP-0000264.

- We propose and evaluate a "method of complementary service time inflation" to model lock access contention in addition to memory system (hardware) contention.

The new technique for estimating mean residence time at the high-CV server builds on the decomposition approach to modeling high service time variability proposed by Zahorjan et al. [17]. The new AMVA interpolation for estimating the mean residual life is superior to the interpolation used in the ISCA98 model. The new techniques are not specific to the ISCA98 model, and thus might be employed in other systems that have a server with non-preemptive scheduling and highly variable service times. We show that the mean residence time technique, together with the new technique for modeling bursty arrivals at the downstream servers, greatly improves the accuracy and robustness of AMVA models for such systems.

The technique for modeling lock contention is contrasted with previous approaches to modeling simultaneous resource possession [5, 12]. As with the method of surrogate delays [5] that inspired the approach, the method of complementary service time inflation is a general AMVA technique that might be applied to other systems where customers can queue for one resource while holding or occupying another resource.

Accurately modeling the system features described above increases the robustness and applicability of the ISCA98 model. While the techniques for analyzing burstiness and lock contention increase the model complexity, the number of additional model input parameters is small, and the extensions do not appreciably degrade the model solution time, which is still on the order of a couple of seconds.

The rest of this paper is organized as follows. Section 2 provides some background on the architecture and the ISCA98 model that motivates the new AMVA techniques. Section 3 discusses the new MVA approximations for estimating mean residence times at FCFS queues that have high coefficient of variation in service times. Section 4 presents the new AMVA technique for modeling bursty arrivals at the downstream queues. Section 5 discusses the method of complementary service time inflation for computing lock access delays. Finally, Section 6 summarizes the new techniques developed in the paper and future research directions.

2 Background

Below we describe the system architecture models that motivate the modeling techniques developed in later sections of this paper. A reader who is not interested in these details can skip this section.

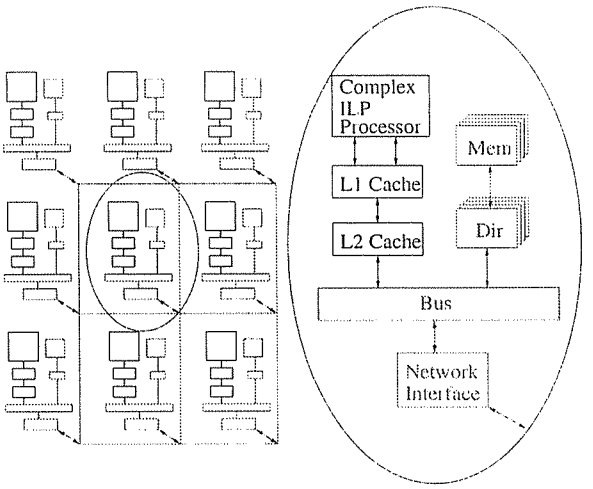

Figure 1. ISCA98 System Architecture

2.1 System Architecture

The architecture modeled in the ISCA98 model is the cache-coherent, shared-memory multiprocessor architecture, shown in Figure 1, that is modeled in the detailed RSIM simulator [10]. Cache coherence is maintained by a standard three-state (MSI) directory-based invalidation protocol.

The MIPS R10000-like processor, modeled in RSIM, exploits instruction level parallelism (ILP). Instructions are fetched into and retired from the instruction window in program order, and are issued to the functional units when their input data dependences are satisfied. Speculative execution is used for (temporarily) unresolved control dependences for up to four branch instructions. Parallel instruction execution, combined with non-blocking caches, allows the processors to have multiple outstanding requests to the memory system. The caches use miss status holding registers (MSHRs) to track the status of each outstanding miss [6].

The RSIM architecture is release consistent, which means that store instructions can be retired from the instruction window before the data returns from memory. That is, memory write or upgrade requests are asynchronous. Correctly speculated memory reads are synchronous, meaning that the load instruction cannot be retired from the top of the instruction window until the data returns from memory.

We have also modified the ISCA98 model so that it can be applied to the architecture that is modeled by the SimOS [13] simulator. The SimOS architecture is similar to the Stanford FLASH [7] and the SGI Origin [8] architectures. There are two key differences between the SimOS and RSIM architectures that required structural changes to the analytic model equations. First, the SimOS architecture is sequentially consistent, and thus memory write requests

are synchronous. Second, the memory system resources are organized somewhat differently. In particular, the directory controller (DC) (with an attached memory module) is placed structurally where the network interface is placed in the RSIM architecture, and the DC performs the functions of both the directory and the network interface in the RSIM architecture.

Table 1 defines the RSIM and SimOS system architecture parameters used in the simulations that are used in this paper. Latencies are in units of nanoseconds. Note that S_{mem} is set to zero because we assume that memory access is completely overlapped with directory lookup in both architectures. Furthermore, the S_{bus} and S_{net} parameters for SimOS are fixed delays that each include an average queueing delay estimate, whereas RSIM explicitly models queueing time at the bus and at each switch in the interconnection network.

2.2 The ISCA98 and SimOS Architecture Models

The validated ISCA98 model [16] views the memory system architecture below the level two caches as a system of queues and delay centers. The queues include the split-transaction memory buses, directories and associated memory modules, and network interfaces. The interconnection network is modeled by a delay center with mean service time equal to the average network transit time including an estimate of the average queueing delay at the switches. A set of customized AMVA equations that make use of well-known approximate MVA techniques is used to compute the mean total residence time for memory requests in the memory system. The details of those equations are given in [15].

The processor and two-level cache subsystem are modeled as a FCFS queue (i.e., a "black box") that, when not blocked, issues requests to the memory system with a given mean and coefficient of variation in interrequest time (τ, CV_τ). The measured coefficient of variation is on the order of 10-12 for several of the applications evaluated in [16], indicating that parallel instruction execution can lead to highly bursty memory interrequest times. Sorin et al. observe that the standard AMVA estimate of the mean residual life of a customer in service at a FCFS center with high service time variance [11] is inaccurate for the processor queue, because the average latency for a memory transaction is often less than this estimated mean residual life, and thus customers do not arrive back at the processor at random arrival instants. [1] The ISCA98 model therefore uses a simple interpolation between this estimated mean residual life and τ to achieve the model accuracy reported in [16]. High burstiness in request arrivals at the downstream memory system resources (local to the processor), which may cause higher average queue residence times, is not modeled.

In the ISCA98 model, the input measure of mean memory interrequest time, τ, includes the average time spent waiting to acquire locks. In Section 5, we evaluate a new approach for directly computing mean lock access delays.

Each processor has a closed class of customers with population equal to a fixed number of outstanding memory requests that will be issued before the processor blocks. Letting M be the number of outstanding read requests when the processor blocks due to a correctly-speculated load request that cannot be retired, the model is solved for each possible value of M, and then the overall system throughput is computed as a weighted sum of the throughputs for each value of M. The weights are computed from the measured distribution of M, which is a model input.[2]

In the ISCA98 model, for a given value of M, two different versions of the model (i.e., two submodels) are used to compute (1) blocking due to a correctly-speculated load request that cannot be retired, and (2) additional blocking that is due to MSHRs being fully occupied by write requests, respectively. In each submodel, the processor service time is inflated to account for the processor blocking (or idle) time computed in the other submodel. The iterative solution of the two (sub)models will be referred to as the ISCA98 "architecture model" in the remainder of this paper.

To create the SimOS architecture model, we have modified the customized AMVA equations to model the different memory system organization in the SimOS architecture.[3] Furthermore, since write requests are synchronous in the SimOS architecture, iteration between two architecture submodels is not required. Instead, M is the measured number of read, write, or upgrade requests that are outstanding when the processor blocks. One model, solved for each possible value of M, accounts for blocking that is due to a correctly speculated load or store instruction that cannot be retired. All blocking that occurs when MSHRs are full is accounted for in this case.

The issues of computing mean residence time at the high-CV processor queues, modeling bursty request arrivals at the local memory system resources, and directly computing mean lock access delays, are relevant for the SimOS architecture model as well as for the ISCA98 model. Techniques for these extensions are discussed in the next three sections.

[1] The standard equation for mean residual life is accurate at all other resources in the model since the variance in service time is low at the memory system resources.

[2] Sorin et al. observe that the model input parameters are sensitive to instruction window size, organization of the processor cache hierarchy, and various aspects of the application code and compiler, but is relatively insensitive to memory system latencies beyond the level two cache [16].

[3] For example, since SimOS models the split-transaction memory bus as a delay center with average delay that includes bus queueing time as well as transfer time, the customized AMVA equations for the SimOS architecture were modified to use a delay center for the bus.

parameter	description	SimOS	RSIM
N	number of nodes		
m	number of directories per node	1	4
M_{hw}	number of MSHRs	8	8
S_{bus}	bus latency (incl. avg. queueing delay for SimOS)	75	32
S_{NI}	network interface (RSIM) or DC (SimOS) latency for message handling	20	16
S_{DC}	directory or DC latency for directory access	100	80
S_{mem}	non-overlapped memory access time	0	0
S_{net}	network latency or switch latency	120	16

Table 1. SimOS and RSIM System Architecture Parameters

3 FCFS Centers with High Variance Service Times

The standard AMVA approach for handling high (or low) variance in service times at FCFS service centers [11] has been shown to be accurate for many systems in spite of the fact that this feature violates the assumptions required for the MVA equations to hold. However, in the ILP multiprocessor model, this approximation was not sufficient to account for the burstiness of memory requests that are issued by the processors [16]. Specifically, for this model, the standard approximation overestimates mean queueing time at the processor node and underestimates mean waiting time at the local memory system resources. In this section we address the former problem by developing a new and significantly more robust AMVA approximation for the mean residence time at a high-variance FCFS center. In Section 4 we address the latter problem by developing a new AMVA approximation that captures the impact of bursty departures from the high-variance FCFS center on the mean waiting time at a "downstream" center. The new AMVA approximations may be of general use for more accurately modeling systems that have FCFS centers with high variance, or coefficient of variation (CV), in service times.

The ISCA98 model addressed the accurate estimation of mean residence time at a high-CV FCFS server by using the standard AMVA approximation for the customers found in the queue, together with a new AMVA interpolation for the mean residual life of the customer found in service, by an arrival to the server. Instead, we derive a new estimate of the mean residence time at the high-CV FCFS server by adapting the decomposition-based approximate solution technique for modeling high service time variability at FCFS centers developed by Zahorjan et al. [17], to the AMVA framework. We then provide the new interpolation used in the ISCA98 model and develop an improved interpolation. Finally we compare the accuracy of the various AMVA techniques as well as the decomposition technique by Zahorjan et al. for a variety of systems with high-CV FCFS centers, including the ILP multiprocessor system.

3.1 New AMVA Mean Residence Time Estimate

Consider first a system in which there is one FCFS center with high variance service times, and suppose that the service time distribution can be modelled with a 2-stage hyperexponential distribution. That is, with probability p a given customer has a "small" mean service time, τ_a, and with probability $1-p$ the customer has a "large" mean service time, τ_b, $\tau_a < \tau_b$ and $\tau = p\tau_a + (1-p)\tau_b$. In this case, the decomposition solution technique developed by Zahorjan et al. entails computing estimated system performance measures using weighted averages of the performance measures of two simpler models. In one of these models, customers have mean service time τ_a at the FCFS center, while in the other they have mean service time τ_b. An estimate for the overall mean system residence time, for example, is given by p times the mean system residence time in the first model, plus $1-p$ times that in the second.

The Zahorjan et al. solution technique has two key advantages. First, it has a firm theoretical foundation provided by the theory of near-complete decomposability [2]. Second, it has generally very high accuracy [17]. The principal disadvantage is the cost of the technique, particularly when there are multiple FCFS centers with high variance service times. For a model including H FCFS centers with service time distributions modelled by 2-stage hyperexponential distributions, 2^H simpler models need be analyzed, one for each possible combination of service stages for each of the H centers. In some contexts, such as the ILP multiprocessor modelling application, this is too expensive.

The insight that leads to adapting the decomposition approach to the AMVA context is that it may be sufficiently accurate to apply the decomposition only at the level of the individual centers at which there is high service time variability. Thus, for a FCFS center with a service time distribution modeled by the 2-stage hyperexponential distribution described above, assuming a single closed class of N cus-

tomers, we approximate the mean residence time R at that center using:

$$R = pR_a + (1-p)R_b, \quad (1)$$

where

$$R_a = \tau_a(1 + \frac{N-1}{N}Q_a) \quad (2)$$

$$R_b = \tau_b(1 + \frac{N-1}{N}Q_b) \quad (3)$$

$$Q_a = N\frac{R_a}{R_a + R_{other}} \quad (4)$$

$$Q_b = N\frac{R_b}{R_b + R_{other}} \quad (5)$$

In the above equations, R_{other} is the mean total residence time spent at the other centers in the model, which is computed iteratively together with R within the standard AMVA iterative solution framework. The extension of these equations to multiple customer classes is straightforward.

Note that the above approach assumes that the "average behavior" seen in the rest of the system is the same regardless of which stage of the center with high service time variability is active. In section 3.3 we examine the accuracy of this approximation for many networks, including systems for which such an assumption would appear to be inaccurate. The principal advantage of this approach is that there is no need to solve 2^H separate models to obtain the solution for a system that includes H FCFS centers with high service time variability. Only the one model is solved, with modified mean residence time equations at each of the H centers as given above.

The above model is easily extended to general Coxian servers [3] by combining as necessary the above method for composing the mean residence time equation, with the standard (and quite accurate) AMVA approximation for service times with low variability. However, we have not yet investigated the accuracy that is achieved with such an extension.

3.2 New AMVA Mean Residual Life Estimates

The standard AMVA approximation for the mean residence time at a FCFS center with high (or low) variance service times assumes that customers arrive at random points in time and thus computes a relatively large value for the mean residual life of a customer in service when an arrival occurs to a high-CV FCFS center.[4] If the average residence time of the customer in the rest of the system is smaller than this mean residual life, as is the case for memory system requests in the ILP multiprocessor model, the customer does not arrive back to the high-CV server at a random point in time relative to the service time at the processor. In this case, the standard approximation can greatly overestimate mean residual life.

An alternative to the AMVA mean residence time approximation developed in section 3.1 is to develop a more accurate accurate approximation for the mean residual life of a customer found in service at high-CV FCFS center, and to use this more accurate estimate together with the standard AMVA approximation for the number of customers waiting in the queue, to compute overall mean residence time in the queue.

As before, we consider a FCFS center with a service time distribution modeled by a two-stage hyperexponential with parameters τ_a, τ_b, and p, such that $\tau = p\tau_a + (1-p)\tau_b$. The mean residual life is approximated using an interpolation between τ, which is the mean residual life in the limiting case in which the time spent at the other centers in the model approaches zero, and the mean residual life as would be seen by a random arrival. Letting R_{other} be the average total residence time in the rest of the model, and L be the "standard" mean residual life (i.e., assuming random arrival instants), a simple ad hoc interpolation is given by:

$$\text{mean residual life} \approx \frac{\tau}{\tau + R_{other}}\tau + \frac{R_{other}}{\tau + R_{other}}L \quad (6)$$

This interpolation was used in [16] and yielded results that were significantly more accurate than when the standard mean residual life was used.

An improved interpolation is obtained when $T = \frac{\tau_a\tau_b}{\tau}$ replaces τ in the weighting factors, as follows:

$$\text{mean residual life} \approx \frac{T}{T + R_{other}}\tau + \frac{R_{other}}{T + R_{other}}L \quad (7)$$

This interpolation is exact when the mean delay in the rest of the model is exponentially distributed with mean R_{other}. To see this, consider a given customer, named A, that is not at the high variance center when another customer, named B, enters service at this center. If B has mean service time equal to τ_a, then with probability $\frac{\frac{1}{R_{other}}}{\frac{1}{\tau_a}+\frac{1}{R_{other}}}$, A will arrive at the high variance center before B completes service, and in this case the mean residual life is τ_a. Similarly if B has mean service time equal to τ_b, then with probability $\frac{\frac{1}{R_{other}}}{\frac{1}{\tau_b}+\frac{1}{R_{other}}}$, A will arrive at the high variance center before B completes service, and in this case the mean residual life is τ_b. Letting r denote the mean residual life of a customer in service at an arrival instant, we thus have:

$$r = p\left[\frac{\frac{1}{R_{other}}}{\frac{1}{\tau_a}+\frac{1}{R_{other}}}\tau_a + \frac{\frac{1}{\tau_a}}{\frac{1}{\tau_a}+\frac{1}{R_{other}}}r\right]$$

[4] The estimated mean residual life equals the second moment of service time divided by twice the mean service time.

$$+(1-p)\left[\frac{\frac{1}{R_{other}}}{\frac{1}{\tau_b}+\frac{1}{R_{other}}}\tau_b+\frac{\frac{1}{\tau_b}}{\frac{1}{\tau_b}+\frac{1}{R_{other}}}r\right],$$

which reduces to the second interpolation given above.

Tables 2 and 3 compare the accuracy of four techniques for estimating the mean residual life: (1) assume that service times are exponentially distributed ("CV=1"), (2) the standard AMVA approximation, (3) the simpler (first) interpolation for mean residual life given above, and (4) the new (second) interpolation given above ("New Interp."). The results in these tables are for simple two-center models with a single class of 5 customers. One center is a FCFS center with service times modeled by a 2-stage hyperexponential distribution with mean τ and varying CV_τ^2. The other center is either a delay (Table 2) or a queueing (Table 3) center with exponentially distributed service times with mean S_{delay} or $S_{queueing}$, respectively. Exact values of the mean residual life are derived from numerical solutions of the corresponding Markov chains. For Table 3, the exact value of the mean residence time at the second queueing center (R_{other}) is used in the interpolation formulas. (In Section 4, we describe an accurate AMVA approximation for this mean residence time.)

Note that for the models in which the second center is a delay center (Table 2), the new interpolation is exact, as established above. Even for the models in Table 3, the new interpolation appears to yield reliable estimates. The reliability of the new interpolation is further supported by Figure 2, which explores the accuracy of the new interpolation over a space of two-center models in which the second center is a queueing center. Each contour line in the figure corresponds to a particular absolute value of percent relative error.

In contrast, as illustrated in Tables 2 and 3 the other techniques appear to have considerably more variable accuracy. Most notably, the estimations provided by the standard AMVA approximation are highly inaccurate for these cases. Although the simple interpolation is significantly better than the standard AMVA approximation or assuming CV=1, it appears to be significantly less accurate than the new interpolation.

3.3 Accuracy of the AMVA Residence Time Estimates

Tables 4 and 5 compare the accuracy of six alternative techniques for estimating the mean residence time at a FCFS center with high variance service times: (1) assume that service times are exponentially distributed ("CV=1"), (2) the standard AMVA technique, (3) use of the simple (first) interpolation for mean residual life given in Section 3.2 together with standard AMVA for all other estimates, (4) use of the new (second) interpolation given in Section 3.2 ("New Interp.") together with standard AMVA,

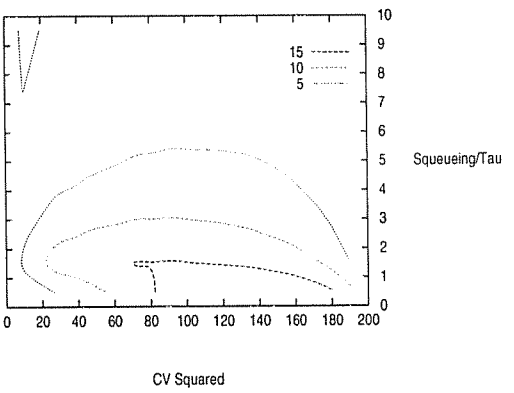

Figure 2. Accuracy of New Interpolation for Mean Residual Life (two-center models with queueing center, N=5, p=0.99, contours for absolute value of percent relative error)

(5) the new technique proposed in Section 3.1 ("AMVA-decomp"), and (6) the decomposition approach of Zahorjan et al. ("Decomp.") [17].

The techniques are compared for the same two-queue model configurations as were used in Tables 2 and 3, respectively. Exact values for the mean residence time at the FCFS center with high variance service times are derived from numerical solutions of the corresponding Markov chains. For Table 5, the exact value of R_{other} is used in the calculations for all of the techniques excepting for the decomposition approach of Zahorjan et al., in which this quantity is not used.

Note that for the models in which the "other" center is a delay center, the AMVA approximation proposed in Section 3.1 and the decomposition approach of Zahorjan et al. on which it is based, yield identical results. For the models in which the second center is a queueing center (Table 5), we might expect the AMVA-Decomp technique to be less accurate, as we would not expect the "average behavior" seen in the rest of the system (i.e., arrival queue length at the second queueing center) to be the same regardless of which stage of the center with high service time variability is active. However, for the cases considered in Table 5, the AMVA-Decomp technique appears to be quite accurate in spite of this possible issue.

The accuracy of the AMVA-Decomp approach is explored more fully for the simple two-center models in the contour plots of Figures 3 and 4. The key conclusions from Tables 4 and 5 and Figures 3 and 4, are:

- The new decomposition-based AMVA technique in

CV_τ^2	S_{delay}/τ	CV = 1	Std. AMVA	Simple Interp.	New Interp./ Actual
10	0.5	50	275	77.8	56.3
10	2	50	275	181.2	73.2
10	10	50	275	252.2	132.1
100	0.5	50	2525	87.0	108.1
100	2	50	2525	340.4	267.1
100	10	50	2525	1249.9	853.7

Table 2. Mean Residual Life Estimates (two-center models with delay center, N=5, p=0.99, τ=50)

CV_τ^2	$S_{queueing}/\tau$	CV = 1	Std. AMVA	Simple Interp.	New Interp.	Actual
10	0.5	50	275	86.6	63.0	62.0
10	2	50	275	187.4	124.6	117.8
10	10	50	275	252.2	215.9	231.0
100	0.5	50	2525	141.0	260.0	221.0
100	2	50	2525	440.6	824.0	727.4
100	10	50	2525	1269.0	1787.7	1777.3

Table 3. Mean Residual Life Estimates (two-center models with queueing center, N=5, p=0.99, τ=50)

CV_{tau}^2	S_{delay}/τ	CV = 1	Std. AMVA	Simple Interp.	New Interp.	AMVA-Decomp./ Decomp.	Actual
10	0.5	230.5	355.2	242.2	235.7	230.5	225.1
10	2	178.1	310.9	156.3	198.6	180.7	164.8
10	10	76.6	167.6	71.8	113.8	105.7	96.1
100	0.5	230.4	825.7	249.1	272.2	231.7	226.2
100	2	178.1	786.3	234.2	307.7	203.6	199.3
100	10	76.6	606.8	157.1	323.8	189.9	177.3

Table 4. Estimates of Mean Residence Time at High-CV Center (two-center models with delay center, N=5, p=0.99, τ=50)

CV_τ^2	$S_{queueing}/\tau$	$CV = 1$	Std. AMVA	Simple Interp.	New Interp.	AMVA-Decomp.	Decomp.	Actual
10	0.5	209.4	337.8	238.5	220.5	210.0	213.9	204.5
10	2	81.9	182.8	148.6	120.8	109.3	110.6	112.2
10	10	54.4	73.8	71.8	68.7	79.6	90.8	73.2
100	0.5	180.3	787.9	246.9	306.4	204.2	203.5	206.0
100	2	78.4	617.2	226.4	325.4	190.2	193.1	186.2
100	10	54.3	251.8	155.5	196.1	173.3	187.9	154.8

Table 5. Estimates of Mean Residence Time at High-CV Center (two-center models with queueing center, N=5, p=0.99, τ=50)

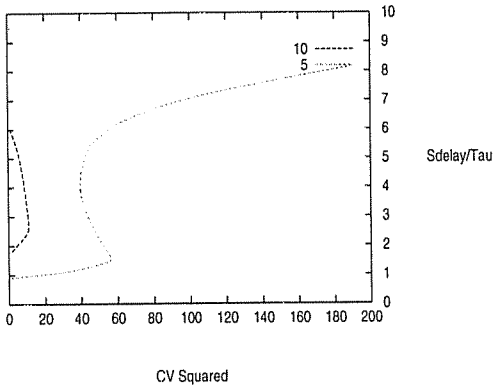

Figure 3. Accuracy of Decomposition-Based AMVA Approximation (two-center models with delay center, N=5, p=0.99, contours for absolute value of percent relative error in mean residence time at high-CV center)

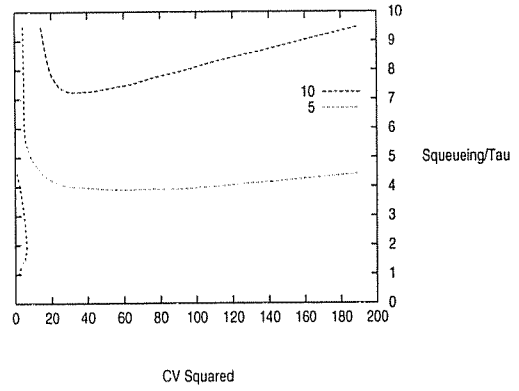

Figure 4. Accuracy of Decomposition-Based AMVA Approximation (two-center models with queueing center, N=5, p=0.99, contours for absolute value of percent relative error in mean residence time at high-CV center)

Section 3.1 yields estimates of mean residence time that have similar accuracy as those provided by the highly accurate but more costly decomposition approach. (However, we have not yet established that we can estimate the mean residence times at downstream centers as accurately as can the decomposition approach; this question is addressed in Section 4.)

- The other techniques have more variable accuracy.
- Although the new interpolation technique was shown to be highly accurate for estimating mean residual life, this is not always, by itself, sufficient for accurately estimating overall mean residence time at a high-CV FCFS center.
- The standard AMVA approximation technique is again not very robust.

Although the new interpolations for estimating the mean residual life are not as robust for predicting mean residence time in the queue as the decomposition-based approaches, they are simple to implement and yield considerably improved mean residence time estimates as compared with the standard AMVA approximation. This is consistent with the observations in [16] that the simple interpolation was needed in the ISCA98 model. Figure 5 illustrates the accuracy of memory request throughputs estimated by the ISCA98 model with standard AMVA, and with the simple mean residual life interpolation, as compared with the estimates from the detailed RSIM simulator, for various Splash benchmarks. The results for Simple Interp are perhaps more accurate than one would expect from the preceding results in this section. This is due to the fact that, in cases where Simple Interp (or New Interp) overpredict mean residence time, the error is partially cancelled in the throughput estimate because the bursty arrivals at the downstream queues are not modeled.

4 Bursty Arrivals

A FCFS center with high variance service times generates bursty departures. Bursty departures may yield bursty arrivals at downstream centers, increasing queueing at these centers. In the ILP multiprocessor system, for example, there is a large coefficient of variation in the processor service time (τ) for many applications, resulting in bursty memory requests. The standard AMVA equations do not capture the bursty arrival behavior at the memory system resources, which results in an underestimate of the mean arrival queue length of requests from the processor, and thus an underestimate of mean residence time.

In this section we develop a new AMVA approximation that successfully captures the impact of bursty arrivals in

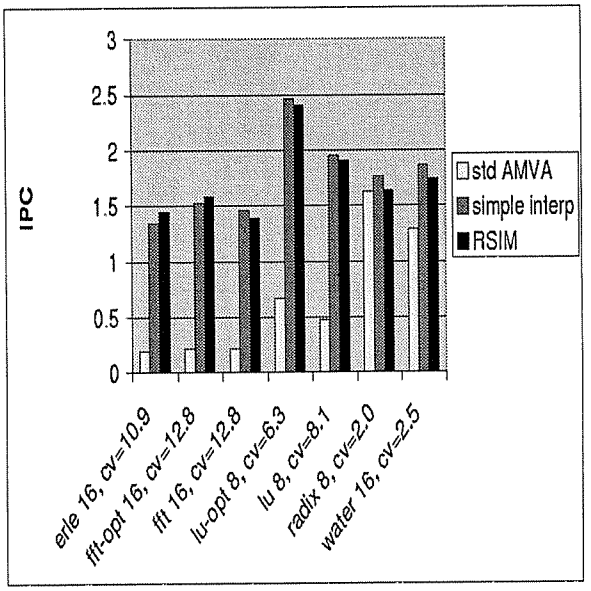

Figure 5. Accuracy of ILP Multiprocessor Throughput Estimates

this context. This new approximation is possibly useful in other contexts as well.

4.1 A Model of Bursty Arrivals

Our approach entails modeling a bursty arrival process as consisting of "bursts" containing a geometrically distributed number of requests, with exponentially distributed inter-burst times as well as interarrival times within a burst. Three parameters characterize the arrival process:
k, the average number of customer arrivals within a burst,
I, the mean interarrival time within a burst, and
L, the mean time between bursts.

The value of I is determined from the service time distribution(s) at the center(s) generating the arrivals. For example, consider the simple case where arrivals are generated by departures from a FCFS queueing center with service times modeled by a 2-stage hyperexponential distribution. In this case, I is equal to the smaller of the mean service times of the two stages.

Letting X denote the arrival rate (equal to the center throughput which is iteratively computed during the iterative AMVA solution), and CV_a^2 denote the squared coefficient of variation of interarrival times (to be computed below), k and L can be defined by the following two equations:

$$\frac{k}{(k-1)I + L} = X \tag{8}$$

$$\frac{\frac{k-1}{k}2I^2 + \frac{1}{k}2L^2}{(\frac{(k-1)I+L}{k})^2} - 1 = CV_a^2 \tag{9}$$

Thus, estimation of k and L requires estimation of CV_a^2. For this purpose, the method proposed by Sevcik et al. [14] can be employed. For example, consider again the case that a single FCFS queueing center with high variance service times (the "upstream" center), generates arrivals to a single "downstream" center. Assuming that the arrivals to the upstream center are not themselves bursty, an approximation for CV_a^2 at the downstream center is expressed in terms of the squared coefficient of variation of service times, CV_s, at the upstream center and the utilization (U) of the upstream center, as follows [14]:

$$CV_a^2 = 1 + U^2(CV_s^2 - 1) \tag{10}$$

As described below, the value of k may need to be capped to some value less than that computed from the above equations (and the value of L correspondingly modified), depending on the size of the customer population and on the mean queue length estimates produced during the iterative AMVA solution.

4.2 An Approximation for Mean Residence Time

The new approximation for the mean residence time R at a queueing center with bursty arrivals employs the above model, with parameters k, I, and L, together with the assumption that the center is never empty in the midst of a burst. This assumption is likely to be reasonable for most parameter settings of interest for the ILP multiprocessor system.

Given this assumption, and assuming a single closed class of N customers, the residence time of a customer is equal to its own service time, plus those of the customers found at the center by the "lead arrival" of the burst (less any service time already acquired by the customer in service, but for simplicity we will assume memoryless service times for this calculation), plus those of the prior customers within the same burst, minus the time from the start of the burst until the customer's arrival. Since the burst size is geometrically distributed with mean k, on average there are $k-1$ prior customers within a burst. Thus, the average time from the start of the burst until the customer's arrival is $(k-1)I$. Letting Q_{nb} be the mean queue length during the time intervals between bursts, and making the standard AMVA approximation for the arrival instant mean queue length,

$$R = S(1 + \frac{N-1}{N}Q_{nb} + (k-1)) - (k-1)I \quad (11)$$

Q_{nb} can be derived from the overall average mean queue length $Q = RX$, computed during the iterative AMVA solution, and the assumption that the center never idles in the midst of a burst, yielding:

$$Q_{nb} = Q - X\frac{I(k-1)(S-I)}{S} \quad (12)$$

Clearly, however, there cannot be more than $N-1$ customers that are either found already present at the center by the lead arrival of the burst, or that arrive prior in the burst. Thus,

$$k \leq N - \frac{N-1}{N}Q_{nb} \quad (13)$$

Ensuring that this constraint is satisfied may require reducing k to the value of the rhs of the above relation, and then recomputing L (so as to ensure that the constraint $\frac{k}{(k-1)I+L} = X$ remains satisfied), during the course of the iterative AMVA solution.

4.3 Validation Results

Tables 6 and 7 compare five approaches for handling FCFS centers with high variance service times, and the bursty arrivals that they generate at downstream centers: (1) assume that service times are exponentially distributed ("CV=1"), (2) the standard AMVA approximation, (3) the approach proposed in Section 3.1 with no attempt to model bursty arrivals ("AMVA-Decomp"), (4) the approach proposed in Section 3.1 in conjunction with the above technique for modeling bursty arrivals ("AMVA-Decomp-Bursty"), and (5) the decomposition approach of Zahorjan et al. ("Decomp.") [17]. The results in this table are for the same two-center model configurations that were used in Tables 2 and 3. Exact values for the performance metrics shown are derived from numerical solution of the corresponding Markov chains. The accuracy of the decomposition-based AMVA approach in conjuction with the proposed technique for modeling bursty arrivals is explored more fully for these simple two-center models in the contour plots shown in Figures 6 and 7.

The results show that, for the cases considered, the combined use of the new decomposition-based AMVA technique and the proposed technique for modeling bursty arrivals yields estimates of mean residence time that are nearly as accurate as those provided by the more costly decomposition approach. In contract, the other techniques have much more variable accuracy.

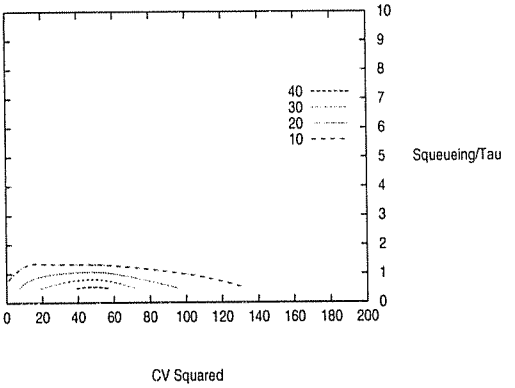

Figure 6. Accuracy of Decomposition-Based AMVA Approximation + Bursty Arrival Approximation (two-center models with queueing center, N=5, p=0.99, contours for absolute value of percent relative error in mean residence time at the center with bursty arrivals)

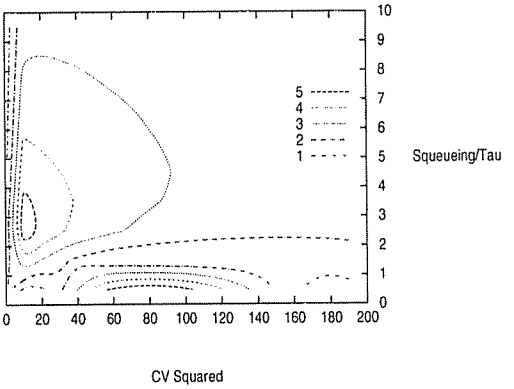

Figure 7. Accuracy of Decomposition-Based AMVA Approximation + Bursty Arrival Approximation (two-center models with queueing center, N=5, p=0.99, contours for absolute value of percent relative error in mean system residence time)

CV_τ^2	$S_{queueing}$ / τ	$CV = 1$	Std. AMVA	AMVA-Decomp.	AMVA-Decomp.-Bursty	Decomp.	Actual
10	0.5	40.7	33.8	40.7	40.7	47.8	53.4
10	2	437.2	353.8	416.3	464.6	452.2	431.5
10	10	2456.7	2441.4	2436.8	2495.0	2449.6	2439.4
100	0.5	40.7	28.3	40.4	78.6	104.1	96.4
100	2	437.2	176.4	361.3	482.1	483.0	473.2
100	10	2456.7	2293.4	2362.8	2496.2	2468.8	2450.7

Table 6. Estimates of Mean Residence Time at Queueing Center with Bursty Arrivals (two-center models with queueing center, N=5, p=0.99, τ=50)

CV_τ^2	$S_{queueing}$ / τ	$CV = 1$	Std. AMVA	AMVA-Decomp.	AMVA-Decomp.-Bursty	Decomp.	Actual
10	0.5	259.4	383.5	259.6	259.6	261.7	257.9
10	2	518.7	558.1	526.5	572.0	562.8	543.7
10	10	2510.9	2515.9	2516.3	2574.1	2540.4	2512.7
100	0.5	259.4	852.6	262.9	286.5	307.6	302.4
100	2	518.7	923.7	553.1	672.3	676.1	659.4
100	10	2510.9	2558.3	1536.8	2669.2	2656.7	2605.5

Table 7. Estimates of Mean System Residence Time (two-center models with queueing center, N=5, p=0.99, τ=50)

5 Synchronization

Two predominant synchronization primitives used by parallel applications are barriers and locks.[5] Barriers are global synchronization primitives, in that all processors must wait for all other processors to have reached the barrier before they are allowed to continue. Locks are generally used to enforce mutual exclusion to a shared software resource. Processors contend for a lock, acquire the lock, use the shared resource, and then they release the lock so that other processors can obtain access to the shared resource.

In the ISCA98 model, lock access delays and barrier delays were included in the model input parameters. This does not permit investigation of how these delays might change when varying memory system configuration. Below we propose an approach for predicting average lock contention delays (and barrier delays) from basic model input parameters that are insensitive to changes in the memory system beyond the processor cache hierarchy. The technique for estimating lock contention may be more generally applicable to other systems in which resources are symmetrically held while waiting for other resources to become available.

The specific context that we consider is one in which the application program queues for a lock while occupying (or holding) the processor. Symmetrically, while holding the lock, the program can queue for memory system resources or the processor. Conversely, the program can release the lock while still holding the processor or it can complete service at a memory system resource while still holding the lock. This "symmetric" simultaneous resource possession problem can be contrasted with the "asymmetric" problem previously considered in [5, 9], in which resource A (e.g. a tape drive) is acquired before queueing for resource B (e.g., a disk) and then A is held for the entire queueing and service time at B. The ILP multiprocessor simultaneous resource possession problem can also be contrasted with the method of layers [12], in which a client does not occupy the processor while queueing for the software resource (i.e., the server). One further important contrast between the problem considered here and the problem considered in previous work is that each processor in the architecture model has a class of customers with population equal to the number of memory requests that can be issued before the processor blocks; there may be more than one customer per processor due to ILP, yet the customers from a given processor cannot contend with each other for any of the locks. Our proposed solution differs from previous approaches primarily because we are addressing a different simultaneous resource possession problem. We comment further on this after describing

[5] Flags are not addressed in this paper.

the proposed technique.

5.1 A Method of Complementary Service Time Inflation

We will first describe the proposed technique assuming only one lock is held at a time; then we will discuss generalizations for specific types of nested lock acquisition, which occur in the Splash benchmarks in SimOS. Further generalization for simultaneous lock access is beyond the scope of this paper.

The approach is inspired by the method of surrogate delays, first proposed by Jacobson and Lazowska [5]. Like the method of complementary service time inflation used in the ISCA98 model to iterate between two architecture submodels, we iterate between the architecture model that estimates memory request throughput for each processor, and a lock contention model that computes average lock access delay (or processor spin-waiting time) due to lock contention. Customers in the architecture model represent actual or potential memory requests that can be issued by each processor. Customers in the lock contention model represent processors. During the iterative solution, specific service times in each model are inflated to reflect delays in the other model, as explained below.

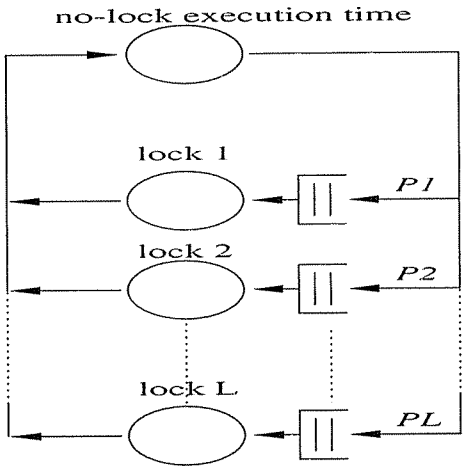

Figure 8. Lock Contention Model

The lock contention model is illustrated in Figure 8. The number of customers in the lock contention model equals the number of processors that access any of the L locks. The (fundamental) input parameters for this model are:
L: number of locks that have non-negligible contention
$r_{nolock,j}$: avg number of memory accesses by processor j between lock requests
$P_{lock_i,j}$: prob that a lock request from processor j is for lock i
$r_{lock_i,j}$: average number of memory requests by processor j while holding lock i

The model includes a FCFS queue for each of the L locks, and a delay center representing the execution time while not accessing or holding any of the L locks in the submodel. The service time at each delay center is computed as $r_{nolock,j}$ times $\frac{1}{X_{j,nolock}}$, where $X_{j,nolock}$ is the throughput of memory requests for processor j estimated by a version of the architecture model in which the service time at processor j is not inflated to include average lock spin time, but service time at each other processor is inflated with the estimates of average lock spin time from the previous iteration lock contention model. The service time at each queueing center represents lock holding time which is similarly computed as $r_{lock_i,j}$ times $\frac{1}{X_{j,nolock}}$.

Solution of the lock contention model yields the mean waiting time for each lock by each processor. This average lock spin time is added to the mean processor service time (τ) in the architecture submodel using the fraction of memory requests that are lock accesses, $\frac{1}{r_{nolock,j}+\sum_i P_{lock_i,j} * r_{Lock_i,j}}$, and the lock selection frequencies. In addition, CV_{tau} is recomputed assuming that lock spin time has a coefficient of variation (arbitrarily) equal to one. Solving the overall model involves iterating between the architecture and lock contention models until the (complementary) service time inflation factors converge.

The above iterative model is generalized for specific cases of nested lock requests by creating a separate lock contention model for the set of locks at each level of the lock hierarchy, and iterating among the lock submodels as well at the architecture model, appropriately inflating the various service times.

5.2 Validation Results

The accuracy of the proposed method of complementary service time inflation is illustrated by the sample results in Figures 9, 10, and 11. The first two figures compare the application of the method in the SimOS architecture model, compared with measurements obtained using SimOS, on an implementation of the "Sieve of Eratosthenes" algorithm for finding prime numbers. This application was selected due to its heavy lock contention, as is seen by the high mean lock waiting times reported in the table. The predicted mean waiting times and throughputs are seen to agree reasonably well with the measured values, even in this extreme case. The estimated throughput of CPU 2 is predicted to be higher than that of CPU 1; however the actual throughputs of each processor are in fact quite similar, and relatively large mean lock spin time is estimated accurately enough to obtain this qualitative result.

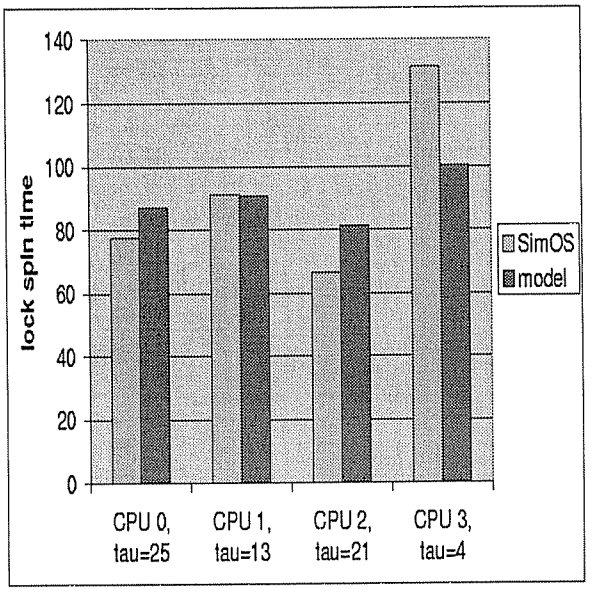

Figure 9. Accuracy of Complementary Service Time Inflation Model in Comparison to Measurements from SimOS ("Sieve of Eratosthenes" algorithm on 4 CPUs, mean lock waiting times as measured and predicted, measured value of the processor service time)

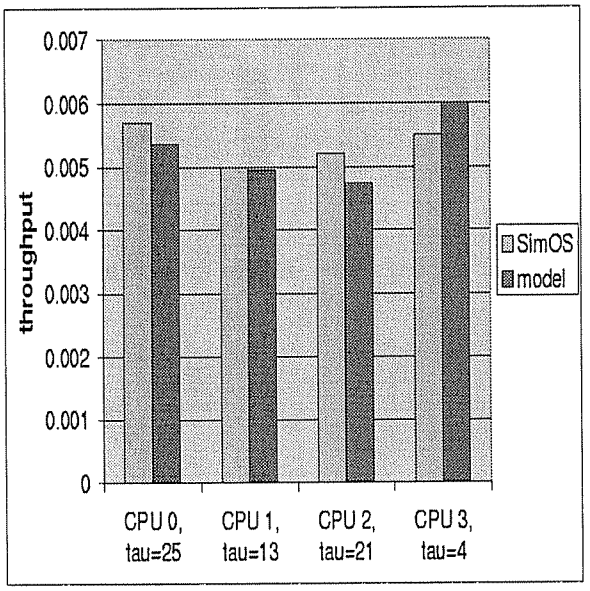

Figure 10. Accuracy of Complementary Service Time Inflation Model in Comparison to Measurements from SimOS ("Sieve of Eratosthenes" algorithm on 4 CPUs, throughput as measured and predicted, measured value of the processor service time)

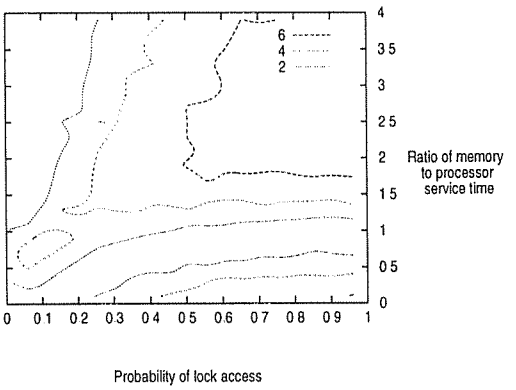

Figure 11. Accuracy of Complementary Service Time Inflation Model in Comparison with Simulation Results (4 processors, 4 memories, 5 customers per processor, 2 locks, 10 memory requests while holding lock, contours for absolute value of percent relative error in memory request throughput)

The results in Figure 11 compare analytic results to results obtained from simulation for a range of simple models.[6] These models consist of four service centers representing processors and four representing memory resources. There are five customers per processor center, representing actual or potential memory requests. Customers cycle between their processor center and a randomly selected (with equal selection probabilities) memory resource, with exponentially distributed service times at the processors and memories. A customer about to leave a processor may attempt to acquire one of two locks, with probability that is varied in the experiments. If an attempt is made to acquire a lock that is held by another processor, the requesting processor is blocked until the lock is freed. Ten memory references are made while holding a lock, implying (for these parameters) low variability in lock holding times. It is assumed that processors may request and hold only one lock at a time, and that the selection frequencies for the two locks are equal. For the cases considered, the method of complementary service time inflation is seen to have very good accuracy, particularly in the regions of most interest that correspond to low probability of lock access (per memory request).

[6]The simulation values that were used in generating the relative error results in this plot have 95% confidence intervals that are within 1%.

5.3 Comparison with Previous Work

The above technique differs from the method of layers in that (1) the customers are very different in our architecture model due to modeling instruction level parallelism and not client-server applications, and (2) queueing for software (lock access) is reflected in our architecture model by inflating the processor service time, whereas queueing for a server is represented by a delay center where clients are not occupying hardware resources in the method of layers.

Menasce et al. [9] use a model with service time inflation at the tape drive to represent the time spent queueing for a disk while holding the tape drive. Since all customers in the model can simultaneously compete for the tape drive, separate submodels are not needed for the system they model. The separate lock contention model is needed for the ILP multiprocessor model to prevent customers from the same processor from simultaneously competing for the locks.

Jacobson and Lazowska [5] model the same simultaneous resource possession problem as Menasce et al. using an iteration between two models, each of which explicitly models the queueing for one of the simultaneously held resources but represents the queueing for the other resource (computed by the other model) as a service time at a (surrogate) delay center. To our knowledge, the accuracy of the Menasce at al. approach has not been compared against the accuracy of this surrogate delay center approach. Another open question for future research is to compare the method of complementary service time inflation against the method of surrogate delays for the simultaneous resource possession problem considered in this earlier work. In the ILP multiprocessor model, complementary service time inflation is needed due to the different nature of the system.

5.4 Predicting Barrier Delays

The execution time between barriers can be computed separately for each barrier pair. Thus, we solve the analytic model for each era between barriers, to obtain the estimated throughput for each processor. The total execution time for each processor is computed from the estimated throughput for that processor and the measured number of memory requests generated by the processor during the era. The exectution time of the slowest node in each era is then used to compute the total execution time and throughput of the system. Previous work by Dubois and Briggs [4] computes the longest execution time in an era from both the estimated mean and variance of the execution time for each processor; however, based on results in [1], we use simply the mean execution time. We have validated that this leads to correct estimates of total execution time for the barriers in the Splash applications we have modeled.

6 Conclusions

This paper has developed and evaluated three new AMVA techniques that are motivated by behavior that occurs in shared memory systems with complex modern processors. In the case of high coefficient of variation in the time between memory requests, we've developed a new decomposition-based AMVA technique for estimating the mean residence time at a FCFS queue that has high service time variance, and a new model of bursty arrivals at the "downstream" queues. These techniques were shown to have remarkably low error over a wide range of two-queue system configurations. The decomposition-based approximation for mean residence time at the high-CV FCFS queue requires one additional input parameter, is easy to apply, and has errors under 10% over most of the parameter space examined. Similarly, the model of bursty arrivals at the downstream queues is also easy to apply, and together with the decomposition-based AMVA technique at the high-CV FCFS queue, yields errors in mean residence time at the center with bursty arrivals that are under 10% over nearly all of the parameter space examined.

We also developed an improved AMVA interpolation for estimating mean residual life of the customer in service at an arrival instant at the high-CV FCFS queue. Validation results for the two queue models show that this interpolation is highly accurate for estimating mean residual life, but mean residence time in the queue is not accurately estimated by this mean residual life estimate combined with standard AMVA techniques for estimating the rest of the average queueing delay. Furthermore, the standard AMVA technique for estimating mean residual life and mean residence time, based on the random arrival assumption, is highly inaccurate for regions of the parameter space that are of interest for the shared memory system architecture model.

In the case of lock contention, we proposed a new method of iterating between the architecture model (containing queues for the hardware resources) and a separate lock contention model, using complementary service time inflation to represent the architecture queueing delays in the lock contention model, and for representing average lock spin time in the architecture model. This approach is moderately accurate for a parallel application with high lock contention on the SimOS architecture, and has very good accuracy over a broad parameter space for a simpler architecture model.

These new techniques have greatly increased the accuracy, robustness, and range of applicability of the analytic model previously developed for evaluating shared memory architectures with complex modern processors. A companion paper [] applies the SimOS architecture model, including the extensions developed in this paper, to a vari-

ety of Splash-2 benchmarks. For each Splash-2 benchmark that has been successfully run on SimOS, there is excellent agreement between the analytic estimates and the SimOS detailed simulation estimates of per-processor throughput. Future reseach includes applying the techniques in this paper to a greater variety of Splash-2 and other parallel applications running on ILP multiprocessor architectures.

References

[1] V. Adve and M. Vernon. The Influence of Random Delays on Parallel Task Execution Times. In *Proc. ACM SIGMETRICS*, pages 61–73, May 1993.

[2] P. J. Courtois. *Decomposability: Queueing and Computer System Applications*. Academic Press, New York, 1977.

[3] D. R. Cox. The Use of Complex Probabilities in the Theory of Stochastic Processes. In *Proc. Cambridge Phil. Soc. 51*, 1955.

[4] M. Dubois and F. A. Briggs. Performance of Synchronized Iterative Processes in Multiproccesor Systems. *IEEE Transactions on Software Engineering*, SE-8(4):419–431, July 1982.

[5] P. Jacobson and E. Lazowska. Analyzing Queueing Networks with Simultaneous Resource Possession. *Communications of the ACM*, 25(2):142–151, Feb. 1982.

[6] D. Kroft. Lockup-Free Instruction Fetch/Prefetch Cache Organization. In *Proc. 8th Int'l Symp. on Computer Architecture*, pages 81–87, May 1981.

[7] J. Kuskin et al. The Stanford FLASH Multiprocessor. In *Proc. 21st Int'l Symp. on Computer Architecture*, Apr. 1994.

[8] J. Laudon and D. Lenoski. The SGI Origin: A ccNUMA Highly Scalable Server. In *Proceedings of the 24th Annual International Symposium on Computer Architecture*, June 1997.

[9] D. Menasce, O. Pentakalos, and Y. Yesha. An Analytic Model of Hierarchical Mass Storage Systems with Network-Attached Storage Devices. In *Proc. ACM SIGMETRICS*, May 1996.

[10] V. Pai, P. Ranganathan, and S. Adve. RSIM Reference Manual. Technical Report 9705, Department of Electrical and Computer Engineering, Rice University, Aug. 1997.

[11] M. Reiser and S. Lavenberg. Mean Value Analysis of Closed Multichain Queueing Networks. Report RC-7023, IBM T.J. Watson Research Center, Mar. 1978.

[12] J. Rolia and K. Sevcik. The Method of Layers. *IEEE Transactions on Software Engineering*, SE-21(8):689–700, Aug. 1995.

[13] M. Rosenblum, S. Herrod, E. Witchel, and A. Gupta. Complete Computer Simulation: The SimOS Approach. *IEEE Parallel and Distributed Technology*, 1995.

[14] K. Sevcik, A. Levy, S. Tripathi, and J. Zahorjan. Improving Approximations of Aggregated Queuing Network Subsystems. *Computer Performance*, 1977.

[15] D. Sorin et al. A Customized MVA Model for ILP Multiprocessors. Technical Report 1369, Computer Sciences Dept., Univ. of Wisconsin - Madison, Mar. 1998.

[16] D. Sorin, V. Pai, S. Adve, M. Vernon, and D. Wood. Analytic Evaluation of Shared-Memory Parallel Systems with ILP Processors. In *Proc. 25th Int'l Symp. on Computer Architecture*, June 1998.

[17] J. Zahorjan, E. Lazowska, and R. Garner. A Decomposition Approach to Modelling High Service Time Variability. *Performance Evaluation*, pages 35–54, 1983.