Analytic Performance Modeling for a Spectrum of Multithreaded Processor Architectures

Pradeep K. Dubey, Arvind Krishna and Mark S. Squillante
T. J. Watson Research Center
IBM Research Division
P. O. Box 704
Yorktown Heights, NY 10598
{pradeep,krishna,mss}@watson.ibm.com

Abstract

The throughput of pipelined processors suffers from delays associated with instruction dependencies and memory latencies. Multithreaded architectures attempt to tolerate such delays by sharing the processor with independent instruction streams, or threads. In this paper we develop a comprehensive analytic framework to quantitately evaluate the performance of a wide spectrum of multithreaded machines, ranging from those that are capable of switching threads every cycle to those that switch threads only on long inter-instruction latencies. The paper also compares the modeling accuracy of discrete-time and continuous-time models in such a context. Phase-type distributions which in addition to their modeling simplicity, also accurately capture many real-life input distributions, have been used for modeling miss latencies. The model outputs are compared against previously published simulation and modeling results under similar input conditions. In certain cases, closed-form expressions for processor utilization are derived. The models developed are also used to assess the performance potential of multithreading in the current state of processor technology. The single-thread performance can be significantly boosted with a second thread, under almost all input conditions of interest. The model further indicates that if the context-switch overhead does not exceed two cycles, a set of around five threads is enough for doubling the single-thread performance even for small run-lengths, high cache-miss latencies, and quite high cache-interference between the threads.

Keywords: Multithreading, Computer architecture, Analytic modeling, Performance models, Parallel systems, Memory hierarchies and network memory systems.

1 Introduction

Processor performance has been increasing at the rate of approximately 50 percent per year, while memory access times have been only improving 5 to 10 percent a year. As a result, the latency of cache misses in terms of processor cycles is increasing rapidly. Moreover, cache miss latencies are becoming a larger fraction of overall processor performance due to the increase in the data bandwidth required to support higher degrees of instruction-level parallelism.

Two alternatives have been traditionally used to deal with cache access delay. One approach is to *reduce* these delays and hence their performance impact. Strategies such as second-level caches and prefetching of data/instructions have been used to reduce the average cache miss ratio, and hence the average cache access delay. An alternative approach is to hide or *tolerate* the cache miss latencies associated with first or second level cache access. This is normally done by switching the execution to a sequence of instructions that are data-and-control-independent of the previous instruction sequence which is blocked due to the cache miss. In other words, the processor utilization is enhanced by sharing it with multiple independent instruction *threads*. In principle, this concept is similar to that of multiprogramming and time-sharing, where disk I/O or keyboard latencies were tolerated by sharing the resources among multiple programs or multiple users, respectively. This concept was extended by Flynn and Podvin [5] to a *shared resource multiprocessor* architecture proposal in which several skeleton processors share the common ALU pipeline resource. This form of multithreading where a certain resource is shared in time by multiple threads can be termed a *temporal* form of multithreading. An alternate form of multithreading simultaneously executes the independent threads on separate resources to further exploit the parallelism. This alternate form of multithreading can be termed *spatial* multithreading. The models presented here deal solely with the temporally multithreaded architectures, and the word multithreading is used in the specific context of temporal multithreading only.

Several temporally multithreaded architectures have been recently proposed. These architectures maintain hardware contexts of several independent instruction threads. The execution control switches between these threads either on every remote memory reference [2, 13], every cache miss [11, 27], or every instruction [5, 7, 8, 14, 23, 25]. Roughly speaking, such architectures can be subdivided into two categories. Architectures in the first category switch threads on every operation that causes a delay, including short data and control interlock delays. The machines in this category can be referred to as *small latency* multithreaded machines. On the other hand, architectures in the second category are characterized by the switching of processor control among independent threads only on remote memory references, whose delays are typically much longer than those experienced in small latency machines. In fact, large cache-miss delays on the order of hundreds of processor cycles are not uncommon for processors that are part of a large-scale multiprocessor system [27]. The machines in this category can be referred to as *large latency* multithreaded machines.

The class of small latency multithreaded architectures typically have multiple distinct sources of delay (such as instruction dependency, branch delay and multi-cycle execution), each with distinct probabilities and distinct corresponding durations, and negligible delays for switching from one thread to another. The machines in the large latency category, however, can often afford to incur some context-switch overhead. We also make the observation that large latency architectures often incur network delays when responding to a context-switching event, e.g., a cache miss leading to a remote memory access [27], and hence the latency can have its own architecture-dependent probability distribution.

The primary objective of this paper is to develop analytic models for a spectrum of multithreaded architectures, ranging from small latency machines to large latency machines with context-switch overhead. We formulate discrete-time and continuous-time analytic models for both multithreading classes, and we derive closed-form expressions for the model solutions that can be efficiently computed. In the simplest instances of these models, geometric and exponential distributions are used for model parameters, respectively. This is motivated by the above observations which suggest that the thread switching event for large latency machines may be characterized by a single probability or rate, e.g., the cache miss ratio. Our analysis also includes continuous-time phase-type distributions for model parameters. The use of phase-type distributions is motivated by their important mathematical properties, which can be exploited to obtain a simpler analytic model while capturing the fundamental aspects of the multithreading tradeoff, and by results showing that phase-type distributions can be used to approximate many parameter distributions of interest with sufficient accuracy in most practical cases. We then use these analytic models to explore and assess the performance potential of multithreading given the current state of processor technology.

An exact discrete-time model of small latency machines has recently appeared in the research literature [3]. The maximum interlock delay is bounded, but otherwise the model does not impose any restriction on the nature of the delay, allowing multiple distinct interlock delays and a general distribution for the delays. We note, however, that the size of the state space for this model becomes prohibitive for processors with maximum latencies greater than 15. Our discrete-time and continuous-time models for small latency architectures address the state space explosion problems of this exact model. To assess the loss of accuracy resulting from our approximations, we compare the results of our models with those reported in [3] and show that our analytic models yield accurate performance measures.

Analytic models of large latency multithreaded machines have also appeared recently in the research literature [19]. This model assumes a fixed context-switch delay, a fixed remote memory access latency and a geometric distribution for the times between remote references for each thread. The model was compared against the trace-driven simulation results reported in [27]. The constant distributions used for certain input parameters in this model significantly increase the modeling complexity. Furthermore, as previously noted, in the case of unpredictable network delays in the memory access path, the assumption of constant memory latency may also be quite unrealistic. Our discrete-time and continuous-time models are simpler and yet support parameter distributions that can be fitted to empirical data. We also show that the results of simple instances of our models compare quite favorably with those reported in [19].

Section 2 details the development of the discrete-time and continuous-time analytic models assuming no context-switch overhead. Section 3 extends our modeling analysis to include the overhead of context switching. Each model is also compared using previously published modeling or simulation results. The following section discusses some results obtained using the model. The final section concludes the paper with an assessment of the usefulness of the models in providing a quick performance estimate of multithreaded machines in different categories.

2 Analytic Models for Small Latency Multithreading

We consider a pipelined processor system in the class of small latency multithreaded architectures that switches execution among N identical instruction threads, or i-threads. One of the ready i-threads, if any, is scheduled in the first phase of every processor cycle. Single-cycle instructions can complete execution by the end of the same cycle in which they are scheduled. Hence, an i-thread solely consisting of single-cycle instructions would be ready to be scheduled at the start of every cycle. Given our focus in this section on small latency machines, we assume negligible overhead in terms of lost cycles for switching from one i-thread to another. That is, the models considered in this section have no context-switch overhead.

A discrete-time model of such a system was recently proposed and analyzed by Dubey, et. al. [3]. In their model the maximum interlock delay is bounded by $K-1$ cycles, but otherwise the model does not impose any restriction on the nature of the interlock delay, allowing multiple distinct interlock delays and a general distribution of the delays. An exact solution for processor utilization is developed as a function of the number of instruction threads being interleaved. This analytic method provides a significantly more efficient estimate of processor utilization, as compared to the simulation alternative. We note, however, that the size of the state space for this model is given by the following sum of binomial coefficients

$$D = \binom{K-1}{0} + \binom{K-1}{1} + \cdots + \binom{K-1}{N}, \tag{1}$$

and that this size becomes prohibitive for pipelined processor with maximum inter-instruction delays greater than 15 (i.e., $K \geq 16$).

In this section we develop approximate discrete-time and continuous-time analytic models for small latency multithreaded architectures that eliminate the state space explosion problems of the exact model proposed by Dubey, et. al. The discrete-time and continuous-time models are developed in turn, followed by a comparison of these approximate models with the results of the exact model presented in [3]. Since there is no restriction on the latency times, the models of this section are also applicable to multithreaded architectures in the large latency class when the context-switch overhead is negligible. Such overheads are considered within the context of our approximate models in Section 3.

2.1 Discrete-Time Model

We model the above small latency multithreaded system as a time-homogeneous, discrete-time Markov chain $\{X_n, n \geq 0\}$ defined over a finite state space $\mathcal{S}$ [9, 18]. The states of the Markov chain can be arbitrarily complex, including many levels of system detail. To simplify the model definition, however, we restrict our attention here to geometric distributions for the model parameters (somewhat more complex distributions are considered in the next section). Specifically, the times to satisfy outstanding remote memory requests are modeled by a geometric distribution with parameter $p = 1/L > 0$. Thus, upon the occurrence of a miss, an i-thread is delayed on average for L cycles. Similarly, we model the times between occurrences of remote memory requests by a geometric distribution whose mean, denoted by R, is finite. Note that the assumption of geometric distributions implies that in any clock cycle an i-thread suffers an inter-instruction delay with probability $1/R$, and that $(1-1/R)$ refers to the probability with which an i-thread continues to remain ready for scheduling.

The states of the Markov chain can be described by how many i-threads are in the *ready* (or *non-delayed*) state. Let r represent the number of ready i-threads. Note that if $r > 0$, then one of the ready i-thread is also *active*, i.e., scheduled or executing. The active i-thread enters the *delayed* state with probability $1/R$. Some or all of the remaining $N-r$ instruction threads can also become schedulable again at the end of the current execution cycle n. Since the i-threads have identical interlock delay distributions in our model, and there is no context-switch overhead, the choice of a specific i-thread from the set of ready i-threads has no performance impact.

We use Y_r to denote the system state with r ready i-threads, $0 \le r \le N$. This implies that $\mathcal{S} = \{Y_0, Y_1, Y_2, \ldots, Y_r, \ldots, Y_{N-1}, Y_N\}$. When the system is in state Y_0, then there is no i-thread ready for scheduling. As a result, a NOP is scheduled and the processor utilization is decreased.

We define

$$\pi(Y_r) \ \equiv \ \lim_{n\to\infty} \Pr[\, X_n = Y_r \,], \quad 0 \le r \le N, \tag{2}$$

and

$$\boldsymbol{\pi} \ \equiv \ [\pi(Y_0),\ \pi(Y_1),\ \ldots,\ \pi(Y_N)]. \tag{3}$$

The limiting probability vector $\boldsymbol{\pi}$ is the stationary distribution for the Markov chain $\{X_n, n \ge 0\}$, and the value of each of its components $\pi(Y_r)$ represents the long-run proportion of time the process spends in state Y_r, $0 \le r \le N$. Assuming this discrete-time Markov chain to be *irreducible* (i.e., all states communicate with each other) and *ergodic* (i.e., the states are *positive recurrent* and *aperiodic*) [9, 18], the stationary distribution is obtained by solving the global balance equations

$$\boldsymbol{\pi} \ = \ \boldsymbol{\pi}\mathbf{P} \tag{4}$$

together with

$$\boldsymbol{\pi}\mathbf{e} \ = \ 1, \tag{5}$$

where $\mathbf{P}$ is the transition probability matrix for the Markov chain and $\mathbf{e}$ is a column vector of $N+1$ ones. We note that for the cases considered later in this section, the Markov chain $\{X_n, n \ge 0\}$ is irreducible and ergodic.

Let $P(Y_j, Y_k)$ denote the (j,k) element of $\mathbf{P}$, i.e., $P(Y_j, Y_k)$ is the one-step transition probability that the system enters state Y_k in the next time step given it is currently in state Y_j. The global balance equations (i.e., Equation (4)) are solved as follows. Consider a system state Y_r with $r < N$ ready i-threads. The next step is to compute the probability of entering this state from other valid system states. When $r \le N-1$, state Y_r can be entered from state Y_{r+1} if the running i-thread enters the delayed state, which occurs with probability $1/R$, and if none of the $N-r-1$ delayed i-threads enter the ready state. The number of ready i-threads would therefore be reduced by one. Note that since there can be no more than one active i-thread, the number of ready i-threads in cycle n cannot be reduced by more than one in the next cycle $n+1$. In other words, state Y_r cannot be entered from state Y_j, $j > r+1$. We then have

$$\Pr[\, X_{n+1} = Y_r \mid X_n = Y_{r+1} \,] \ = \ P(Y_{r+1}, Y_r) \ = \ (1/R)\, B(N-r-1,\, 0), \quad n \ge 0,\ 0 \le r < N, \tag{6}$$

where $B(k,l)$ refers to the probability that l of the k blocked i-threads enter the ready state. This probability can be expressed as

$$B(k,l) = \begin{cases} \dbinom{k}{l} \, (1/L)^l \, (1-1/L)^{k-l}, & l \le k, \\ \\ 0, & l > k. \end{cases} \tag{7}$$

State Y_r can also be entered from states with fewer ready i-threads, i.e., from any state Y_j such that $j \le r$. For example, the system moves from state Y_0 to state Y_r if r of the blocked i-threads become ready. The probability of this transition can be expressed as

$$\Pr[\, X_{n+1} = Y_r \mid X_n = Y_0 \,] \ = \ P(Y_0, Y_r) \ = \ B(N,\, r),\ n \ge 0. \tag{8}$$

On considering a system state Y_j, $0 < j \le r < N$, two possibilities exist:

1. If the active thread enters the delayed state, which occurs with probability $1/R$, and $r-j+1$ of the $N-j$ blocked i-threads become ready, then the system would enter state Y_r. The probability of this event is $(1/R)\, B(N-j,\, r-j+1)$, $0 < j \leq r$.

2. If the active thread continues to stay active, which occurs with probability $(1-1/R)$, and $r-j$ of the $N-j$ blocked i-threads become ready, then the system would enter state Y_r. The probability of this event is $P(Y_j, Y_r) = (1-1/R)\, B(N-j,\, r-j)$, $0 < j \leq r$.

Hence, for $n \geq 0$, $0 < j \leq r < N$,

$$\Pr[\, X_{n+1} = Y_r \mid X_n = Y_j \,] \;=\; P(Y_j, Y_r) \;=\; (1/R)\, B(N-j,\, r-j+1) + (1-1/R)\, B(N-j,\, r-j). \quad (9)$$

Combining Equations (6) through (9) above, we obtain the following equations for the steady-state probability of state Y_r:

$$\begin{aligned}\pi(Y_r) \;&=\; \pi(Y_{r+1})\,(1/R)\,B(N-r-1,\,0) + \sum_{j=1}^{r} \pi(Y_j)\,(1-1/R)\,B(N-j,\,r-j) + \\ &\quad \sum_{j=1}^{r} \pi(Y_j)\,(1/R)\,B(N-j,\,r-j+1) + \pi(Y_0)\,B(N,\,r).\end{aligned}$$

Rewriting this recursive formula to express $\pi(Y_{r+1})$ in terms of $\pi(Y_0)$, $\pi(Y_1)$, ..., $\pi(Y_r)$ yields

$$\begin{aligned}\pi(Y_{r+1})\,(1/R)\,B(N-r-1,\,0) \;&=\; \pi(Y_r) - \pi(Y_0)\,B(N,r) - \\ &\quad \sum_{j=1}^{r} \pi(Y_j)\,(1-1/R)\,B(N-j,\,r-j) - \\ &\quad \sum_{j=1}^{r} \pi(Y_j)\,(1/R)\,B(N-j,\,r-j+1). \qquad (10)\end{aligned}$$

Starting with $r = 0$, Equation (10) can be used to successively compute the steady-state probabilities $\pi(Y_1)$, $\pi(Y_2)$, $\pi(Y_3)$, ..., $\pi(Y_N)$ in terms of $\pi(Y_0)$. The stationary distribution $\boldsymbol{\pi}$ is uniquely determined by the normalization constraint, i.e., Equation (5). This steady-state probability distribution can then be used to approximate the processor utilization of a small latency multithreaded machine, which we denote by U, as follows

$$U \;=\; 1 \;-\; \pi(Y_0). \quad (11)$$

2.2 Continuous-Time Model

We can also model multithreaded architectures in the small latency class as a time-homogeneous, continuous-time Markov chain $\{X(t), t \geq 0\}$ defined over a finite state space $\mathcal{S}$ [9, 18]. The states of the Markov chain can be arbitrarily complex, including many levels of system detail. To simplify the model definition, however, we restrict our attention to a specific class of probability distributions for the model parameters. In particular, the times to satisfy outstanding remote memory requests are modeled as independent and identically distributed (*i.i.d.*) non-negative random variables with a phase-type distribution $A(\cdot)$ whose mean, denoted by m_A, is finite. Similarly, we model the times between occurrences of remote memory requests as *i.i.d.* random variables with a phase-type distribution $B(\cdot)$ and finite mean m_B. The use of phase-type distributions [15, 16, 26] is motivated in part by their important mathematical properties, which can be exploited to obtain a simpler analytic model while capturing the fundamental aspects of the multithreading tradeoff. We also note that certain steady-state averages often depend only upon the first few moments of the parameter distributions (as opposed to their detailed forms) in an important and general class of probability models [20, 21, 22], and thus phase-type distributions can be used here within the context of processor utilization to approximate parameter distributions of interest with sufficient accuracy in most practical cases [26]. Moreover, an important and rich subclass of phase-type distributions can be fitted to empirical data [4].

It follows from the above description that the state representation of the Markov chain is given by $X(t) \equiv (I(t), J_1(t), \ldots, J_{n_A}(t), K(t))$, $t \geq 0$, where $I(t) \in \{0, 1, \ldots, N\}$ denotes the number of non-blocked (or *ready*) threads at time t waiting or receiving service at the processor, $J_\ell(t) \in \{0, 1, \ldots, N - I(t)\}$, $1 \leq \ell \leq n_A$, denotes the number of blocked threads at time t whose state of the phase variable for its outstanding memory request to be satisfied is ℓ, and $K(t) \in \{1, 2, \ldots, n_B\}$ denotes the state of the phase variable at time t for the next miss occurrence. The n_A components $J_1(t), \ldots, J_{n_A}(t)$ are required in the state representation because (as throughout this paper) outstanding memory requests are processed in parallel. Note that for the states $(I(t), J_1(t), \ldots, J_{n_A}(t), K(t))$, $N - I(t)$ represents the number of outstanding remote memory requests (i.e., the number of blocked threads), and thus $\sum_{\ell=1}^{n_A} J_\ell(t) = N - I(t)$. When the system is in a state where $I(t) = 0$, then there is no i-thread ready for scheduling and the processor utilization is decreased. Since the i-threads have identical interlock delay distributions in our model, and there is no context-switch overhead, the choice of a specific i-thread from the set of ready i-threads when $I(t) > 0$ has no performance impact.

We define

$$\pi(i, j_1, \ldots, j_{n_A}, k) \; \equiv \; \lim_{t \to \infty} \Pr\left[\, I(t) = i, J_1(t) = j_1, \ldots, J_{n_A}(t) = j_{n_A}, K(t) = k \,\right], \qquad (12)$$

for $0 \leq i \leq N$, $0 \leq j_\ell \leq N - i$, $1 \leq \ell \leq n_A$, $\sum_{\ell=1}^{n_A} j_\ell = N - i$, $1 \leq k \leq n_B$, and we define

$$Z_i \; \equiv \; \left\{ \, (j_1, \ldots, j_{n_A}, k) \mid 0 \leq j_\ell \leq N - i, \; 1 \leq \ell \leq n_A, \; \sum_{\ell=1}^{n_A} j_\ell = N - i, \; 1 \leq k \leq n_B \, \right\}, \qquad (13)$$

for $0 \leq i \leq N$. Let $x_{i,z} \in Z_i$, $0 \leq z < D_i \equiv \|Z_i\|$, $0 \leq i \leq N$, be a reverse lexicographical ordering of the elements of Z_i, and let $D \equiv \sum_{i=0}^{N} D_i$. Using this we define

$$\boldsymbol{\pi} \; \equiv \; [\; \boldsymbol{\pi}_0, \; \ldots, \; \boldsymbol{\pi}_N \;] \qquad (14)$$

where

$$\boldsymbol{\pi}_i \; \equiv \; (\; \pi(i, x_{i,0}), \; \pi(i, x_{i,1}), \; \ldots, \; \pi(i, x_{i,D_i-1}) \;), \;\; 0 \leq i \leq N. \qquad (15)$$

The limiting probability vector $\boldsymbol{\pi}$ is the stationary distribution for the Markov chain $\{X(t), t \geq 0\}$, and the value of each of its components $\pi(i, j_1, \ldots, j_{n_A}, k)$ represents the long-run proportion of time the process spends in state $(i, j_1, \ldots, j_{n_A}, k)$, $i \in \{0, 1, \ldots, N\}$, $j_\ell \in \{0, 1, \ldots, N - i\}$, $1 \leq \ell \leq n_A$, $\sum_{\ell=1}^{n_A} j_\ell = N - i$, $k \in \{1, 2, \ldots, n_B\}$. Assuming this continuous-time Markov chain to be irreducible and ergodic [9, 18], the stationary distribution is obtained by solving the global balance equations

$$\boldsymbol{\pi} \mathbf{Q} \; = \; 0 \qquad (16)$$

together with

$$\boldsymbol{\pi} \mathbf{e} \; = \; 1, \qquad (17)$$

where $\mathbf{Q}$ is the infinitesimal generator matrix for the Markov chain and $\mathbf{e}$ is the column vector of D ones. We note that for the cases considered later in this section, the Markov chain $\{X(t), t \geq 0\}$ is irreducible and ergodic.

As a specific example, we consider the simplest instance of this continuous-time Markov chain, namely when both model parameters have exponential distributions. That is, the times to satisfy outstanding remote memory requests and the times between remote memory requests are *i.i.d.* exponential random variables with means $m_A = L$ and $m_B = R$, respectively. Under these assumptions, our model is equivalent to an $M/M/1//N$ queueing system. The solution of Equations (16) and (17) in this case is easily shown to be [10],

$$\boldsymbol{\pi}_i \;=\; \boldsymbol{\pi}_0 \left(\frac{R}{L}\right)^i \frac{N!}{(N-i)!}, \quad 1 \leq i \leq N, \tag{18}$$

and

$$\boldsymbol{\pi}_0 \;=\; \left[\sum_{i=0}^{N} \left(\frac{R}{L}\right)^i \frac{N!}{(N-i)!}\right]^{-1}. \tag{19}$$

More generally, the elements of the generator matrix for the Markov chain $\{X(t), t \geq 0\}$ depend upon the phase-type distributions chosen, but the $\mathbf{Q}$ matrix is constructed in a relatively straightforward manner. The corresponding stationary distribution, i.e., the solution of Equations (16) and (17), can be obtained via standard numerical methods. Other techniques can be employed for very large models in which the costs of these methods prove to be prohibitive. For example, if one of the phase-type distribution functions $A(\cdot)$ or $B(\cdot)$ is an exponential distribution, then devices similar to those employed in [16] for finite versions of the $M/PH/1$ or $PH/M/1$ queues, respectively, can be used to obtain explicit formulas for the stationary distribution $\boldsymbol{\pi}$. This would lead to solutions that require $\mathcal{O}(D_i^3)$ time complexity, $0 \leq i \leq N$, instead of the $\mathcal{O}(D^3)$ complexity using standard numerical methods. Since there are no important mathematical differences between the corresponding theorems for discrete-time Markov chains, it is important to note that our approach can be applied more generally to also include discrete phase-type distributions. We refer the interested reader to [16, 17] for more technical details on $PH/PH/1$ queueing systems.

Once the stationary distribution $\boldsymbol{\pi}$ is obtained, it can be used to approximate the processor utilization as follows

$$U \;=\; 1 \;-\; \boldsymbol{\pi}_0 \mathbf{e} \tag{20}$$

where $\mathbf{e}$ is a column vector of D_0 ones. Note that we are simply summing over the steady-state probabilities for all states in which the processor is idle (i.e., $i = 0$), and then subtracting this quantity from one to obtain the processor utilization.

2.3 Validation

The discrete-time and continuous-time analytic models developed in Sections 2.1 and 2.2, respectively, address the state space explosion problems of the exact model proposed by Dubey, et. al.. To assess the loss of accuracy resulting from our approximations, in this section we compare the results of our models with the results presented in [3]. As previously noted, the exact model of Dubey, et. al. supports a general interlock delay distribution that is bounded by $K - 1$ cycles. In this paper, we approximate a general distribution by corresponding geometric and phase-type distributions in discrete and continuous-time approximations. We match parameters by equating the first moment between the general distributions and the geometric or exponential distributions.

The processor utilization of the system in these cases can be obtained from Equations (10) and (11), or Equations (18) through (20).

An idea of the accuracies of these models can be gained from Figures 1 and 2, where the processor utilizations for the discrete-time and continuous-time approximate models of this section have been compared against the model of Dubey et al. [3], assuming the input distributions of McCrackin [14]. The cases shown in these figures correspond to the cases EDF and MFD of McCrackin, both of which have an average length of 1, i.e. $R = 1$, and a maximum interlock delay of 11 cycles. The distribution for the miss latency is computed from the following information, which in turn has been computed from the information given by McCrackin.

1. Once delayed, the memory latency for the EDF case is given by: the latency is 3 cycles with a probability of 0.385, the latency is 4,5,6, or 7 cycles with a probability of 0.09625, and the latency is 11 cycles with a probability of 0.23.

2. Once delayed, the memory latency for the MFD case is given by: the latency is 7 cycles with a probability of 0.77 and the latency is 11 cycles with a probability of 0.23.

McCrackin [14] computes the performance of these distributions via simulation and Dubey et al. [3] compute the performance via an exact Markovian analysis of the same cases. We use their results directly in Figures 1 and 2. The cases enumerated above are referred to as Cases 1 and 2, respectively, in the figures.

The worst case discrepancies due to these approximations are 15 and 11 percent in Cases 1 and 2, respectively. While such inaccuracies may not be tolerable for single-pipeline CPI predictions in a small latency multithreaded pipelined processor [14], they may be acceptable in a large latency multithreaded multiprocessor environment [27]. It should also be noted that in general such inaccuracies are sensitive to the specific general distribution being approximated, and that phase-type distributions (for both continuous and discrete time) can be used to approximate such distributions with sufficient accuracy in most practical cases.

3 Analytic Models for Large Latency Multithreading

The class of large latency multithreaded architectures is characterized by the switching of processor execution control among independent instruction threads only on remote memory references, whose delays are typically much longer than those experienced in small latency machines. In fact, large cache-miss delays on the order of 32 to 128 cycles are not uncommon for processors that are part of a large-scale multiprocessor system [27]. The size of the state space under these parameter values prohibits the use of methods such as those recently proposed by Dubey, et. al. [3]. For example, the size of the state space when $K = 32$ and $N = 8$ (computed using Equation (1)) is more than 11 million states! Therefore, the exact analytic modeling approach in [3] cannot be used to model large latency multithreaded architectures. We also make the observation that machines in the large latency class often incur network delays when responding to a context-switching event, e.g., a cache miss leading to a remote memory access [27]. Thus, the probability distributions considered in the models of Section 2 could be used to obtain a more appropriate delay distribution than the constant delay assumed in previous research [19].

Another important characteristic of large latency multithreading is that such machines can often afford to incur some delay for switching from one i-thread to another. The analytic models developed thus far in this paper, however, have ignored any context-switch overhead. This may be a reasonable assumption for small latency multithreaded machines, which are attempting to hide pipeline bubbles of one or more cycles. In fact, some of the proposed implementations in the small latency category [23, 25, 14] are capable of switching contexts every cycle, at the cost of extra hardware. Conversely, for machines attempting to tolerate latencies on the order of tens or hundreds of cycles, a context-switch overhead of a few cycles can be easily justified and most large latency architectures found in practice have non-negligible context-switch overheads.

In this section we develop approximate discrete-time and continuous-time analytic models for large latency multithreaded architectures. Our approach consists of extending the models in Sections 2.1 and 2.2, respectively, to include context-switching overhead. Unlike the remote memory access delay, the delay due to a context switch in a large latency machine is quite likely to be a constant determined by the amount of machine state to be saved and retrieved. For the sake of modeling simplicity, however, in our discrete-time model we approximate the context-switch overhead by the less realistic geometric distribution. A near-deterministic context-switch overhead is easily factored into the phase-type distribution of our continuous-time model, at the expense of an increase in the size of the state space. We describe each of these modeling approaches in turn. To quantitatively assess the discrepancy due to the geometric and phase-type approximations, the utilizations predicted by our models are then compared with those of the model developed by Saavedra, et. al. [19] which assumes a constant delay distribution for the context-switch overhead.

3.1 Discrete-Time Model

We model the above large latency multithreaded system with context-switch overhead as a time-homogeneous, discrete-time Markov chain $\{X_n, n \geq 0\}$ defined over a finite state space $\mathcal{S}$. The context switch delay is encountered just before an active i-thread enters the delayed state. It represents the overhead of: (i) saving the state information of the current i-thread which just suffered an inter-instruction scheduling delay and (ii) the restoration of the state information for the ready i-thread chosen for scheduling. Thus, even though there may be ready i-threads, there is no active thread during the context-switching period. The system is therefore assumed to be in one of two types of states, depending on whether or not a context-switch is taking place. Our earlier state representation in Section 2.1 (Y_r) will now be used to refer to *non-context-switch* (or *NCS*) system states. That is, Y_r refers to a *NCS* system state with r ready i-threads, $0 \leq r \leq N$. The *context-switch* (or *CS*) system states are represented by $\widehat{Y}_r$, $0 \leq r \leq N-1$. Thus, $\widehat{Y}_r$ refers to a *CS* system state with r ready i-threads, one i-thread in the context-switching state, and the remaining $N-r-1$ i-threads in the delayed or blocked state. The set of valid *CS* states is given by $\{\widehat{Y}_0, \widehat{Y}_1, \ldots, \widehat{Y}_{N-1}\}$, whereas the *NCS* states continue to be $\{Y_0, Y_1, Y_2, \ldots, Y_r, \ldots, Y_{N-1}, Y_N\}$. The state space for the Markov chain $\{X_n, n \geq 0\}$ is given by the union of these two sets, i.e., $\mathcal{S} = \{Y_0, \widehat{Y}_0, \ldots, Y_{N-1}, \widehat{Y}_{N-1}, Y_N\}$. Recall that for $r > 0$, one of the ready i-threads is always active during the *NCS* state.

In addition to the remote memory latency and the intermiss times, the context-switch delay is assumed to have a geometric distribution, with parameter $1/C$. Thus, the average duration of a context-switch for C cycles. Recall that an active i-thread enters the *CS* state with probability $1/R$, the probability with which it suffers an inter-instruction scheduling delay. The next step is to compute the probability of entering system states Y_r and $\widehat{Y}_r$ from other valid system states.

An *NCS* state Y_r, $0 \leq r \leq N$, can be entered only through one of the following three scenarios:

1. State Y_r can be entered from a *CS* state with at most r ready i-threads, i.e., states $\widehat{Y}_j$ such that $0 \leq j \leq r \leq N-1$. If the context-switching i-thread switches to the delayed state (with

probability $1/C$), and $r-j$ of the $N-j-1$ blocked i-threads become ready, the system moves from state $\widehat{Y}_j$ to Y_r. This transition probability is given by

$$P(\widehat{Y}_j, Y_r) \; = \; (1/C)\; B(N-j-1,\, r-j),\; 0 \le j \le r \le N-1, \tag{21}$$

where $B(\cdot,\cdot)$ is as defined in Equation (7).

2. State Y_r can be entered from another *NCS* state with at least one ready i-thread (i.e., an active i-thread) and at most r ready i-threads, i.e., states Y_j such that $1 \le j \le r$. If the active i-thread continues to remain active (with probability $1-1/R$) and $r-j$ of the $N-j$ blocked i-threads become ready, the system moves to state Y_r. This transition probability is expressed as

$$P(Y_j, Y_r) \; = \; (1-1/R)\; B(N-j,\, r-j),\; 1 \le j \le r. \tag{22}$$

3. Finally, state Y_r can also be entered from an *NCS* state with no active i-thread. If r of the N blocked i-threads become ready, the system moves to state Y_r. This transition probability is given by

$$P(Y_0, Y_r) \; = \; B(N,\, r). \tag{23}$$

Equations (21) through (23) can be combined to compute the steady-state probability of state Y_r as follows

$$\pi(Y_r) \;\; = \;\; \sum_{j=0}^{r<N} (1/C)\, \pi(\widehat{Y}_j)\; B(N-j-1,\, r-j) + \sum_{j=1}^{r} (1-1/R)\, \pi(Y_j)\; B(N-j,\, r-j) + \pi(Y_0)\; B(N,\, r).$$

Rewriting this recursive formula to obtain $\pi(\widehat{Y}_r)$, $0 \le r \le N-1$, in terms of $\pi(\widehat{Y}_j)$, $j \le r-1$, and $\pi(Y_j)$, $j \le r$, we obtain

$$\begin{aligned} \pi(\widehat{Y}_r)\,(1/C)\; B(N-r-1,\, 0) \;\; = \;\; & \pi(Y_r) - \sum_{j=0}^{r-1} (1/C)\, \pi(\widehat{Y}_j)\; B(N-j-1,\, r-j) - \\ & \sum_{j=1}^{r} (1-1/R)\, \pi(Y_j)\; B(N-j,\, r-j) - \pi(Y_0)\; B(N,\, r). \end{aligned} \tag{24}$$

Similarly, a *CS* state $\widehat{Y}_r$, $0 \le r < N$, can be entered only through one of the following two scenarios:

1. State $\widehat{Y}_r$ can be entered from any other *CS* state with at most r ready i-threads, i.e., states $\widehat{Y}_j$ such that $0 \le j \le r$. If the system continues to stay in the *CS* state (with probability $1-1/C$), and $r-j$ of the $N-j-1$ blocked i-threads become ready, the system moves from state $\widehat{Y}_j$ to $\widehat{Y}_r$. This transition probability is expressed as

$$P(\widehat{Y}_j, \widehat{Y}_r) \; = \; (1-1/C)\; B(N-j-1,\, r-j),\; 0 \le j \le r. \tag{25}$$

2. State $\widehat{Y}_r$ can also be entered from an *NCS* state with at least one ready i-thread (i.e., an active i-thread) and at most $r+1$ ready i-threads, i.e., states Y_j such that $1 \leq j \leq r+1$. If the active i-thread enters the CS state (with probability $1/R$) and $r-j+1$ of the $N-j$ blocked i-threads become ready, the system moves to state $\widehat{Y}_r$. This transition probability is given by

$$P(Y_j, \widehat{Y}_r) \;=\; (1/R)\, B(N-j,\, r-j+1),\; 1 \leq j \leq r+1. \tag{26}$$

Combining Equations (25) and (26), we obtain the following equations for the steady-state probability of state $\widehat{Y}_r$:

$$\pi(\widehat{Y}_r) \;=\; \sum_{j=0}^{r} (1-1/C)\, \pi(\widehat{Y}_j)\, B(N-j-1,\, r-j) + \sum_{j=1}^{r+1} (1/R)\, \pi(Y_j)\, B(N-j,\, r-j+1).$$

Rewriting this recursive formula to express $\pi(Y_{r+1})$ in terms of $\pi(\widehat{Y}_j)$ and $\pi(Y_j)$, $j \leq r$, yields

$$\begin{aligned} \pi(Y_{r+1})\, (1/R)\, B(N-r-1,\, 0) \;&=\; \pi(\widehat{Y}_r) - \sum_{j=1}^{r} (1/R)\, \pi(Y_j)\, B(N-j,\, r-j+1) - \\ &\quad \sum_{j=0}^{r} (1-1/C)\, \pi(\widehat{Y}_j)\, B(N-j-1,\, r-j). \end{aligned} \tag{27}$$

Equation (24) expresses $\pi(\widehat{Y}_0)$ in terms of $\pi(Y_0)$. Given $\pi(Y_0)$ and $\pi(\widehat{Y}_0)$, Equation (27) can be used to compute $\pi(Y_1)$ in terms of $\pi(Y_0)$. Then applications of Equations (24) and (27) yield $\pi(\widehat{Y}_1)$ and $\pi(Y_2)$, respectively. Thus, all of the steady-state probabilities for both *CS* and *NCS* system states can be successively computed in terms of $\pi(Y_0)$. Finally, the normalization constraint, i.e., $\sum_{r=0}^{N} \pi(Y_r) + \sum_{r=0}^{N-1} \pi(\widehat{Y}_r) \;=\; 1$, uniquely determines the stationary distribution $\boldsymbol{\pi}$. The processor utilization is given by

$$U \;=\; 1 \;-\; \pi(Y_0) \;-\pi(\widehat{Y}_0). \tag{28}$$

3.2 Continuous-Time Model

We can also model large latency multithreaded architectures with context-switch overhead as a time-homogeneous, continuous-time Markov chain $\{X(t), t \geq 0\}$ defined over a finite state space $\mathcal{S}$. The times between occurrences of remote memory requests continue to be modeled as *i.i.d.* random variables with a phase-type distribution $B(\cdot)$ whose mean m_B is finite (see Section 2.2). Since the processor must first context switch to an i-thread and then the i-thread executes until its next miss, the context-switch delay is a component of the times between remote memory requests. Thus, context-switching overhead is reflected in the phase-type distribution $B(\cdot)$ of our continuous-time model, and $m_B = C + R$. The times to satisfy outstanding remote memory requests continue to be modeled as (*i.i.d.*) phase-type random variables with distribution $A(\cdot)$ and mean $m_A < \infty$.

The model solution is then obtained via Equations (16) and (17). In this case, however, Equation (20) provides an upper bound on processor utilization because it represents the total proportion of time that the processor is busy, including the wasted cycles spent in context switching among threads. That is, an upper bound on utilization in large latency machines is given by

$$U_{u.b.} \;=\; 1 \;-\; \boldsymbol{\pi}_0 \mathbf{e}. \tag{29}$$

If the context-switch overhead is explicitly represented in the state space, then clearly the right-hand-side of Equation (29) can be made to also exclude the steady-state probabilities of all non-idle states that represent context switching. When this is not the case, however, we proceed by estimating the fraction of $U_{u.b.}$ that reflects useful processor cycles as follows.

Let τ be the random variable denoting the generic time between remote memory requests for each thread (i.e., the time from when the thread starts to execute until its next miss), with mean R, and let γ be the random variable denoting the generic context switching time, with mean C. As previously noted, the distribution $B(\cdot)$ (representing the total time between any pair of misses) consists of both τ and γ, and $m_B = C + R$. Since context switching is strictly overhead, and thus not an effective use of processor cycles, this factor must be excluded from the processor utilization estimate. It therefore follows that processor utilization can be approximated by

$$U \;=\; (\,1 \;-\; \boldsymbol{\pi}_0\mathbf{e}\,)\;\frac{R}{C \;+\; R}. \tag{30}$$

The distribution of γ, instead of simply its mean C, can also be used when appropriate. We note that any other causes of wasted processor cycles, e.g., thread management [1], can be excluded in a similar manner to obtain an accurate estimate of effective processor utilization.

As a specific example, we consider the simplest instance of our continuous-time Markov chain $\{X(t), t \geq 0\}$, namely when all three model parameters have exponential distributions. That is, the times to satisfy remote memory requests, the context-switching delays and the times between remote memory requests are *i.i.d.* exponential random variables with means L, C and R, respectively. Under these assumptions, our model is equivalent to an $M/GE_2/1//N$ queueing system where GE_2 denotes a two-stage generalized Erlang service distribution [6], with rates $\mu_C = 1/C$ and $\mu_R = 1/R$, and the arrival rate of each customer is given by $\lambda = 1/L$. The generator matrix $\mathbf{Q}$ for this queue, arranged in the same order as the elements of the steady-state probability vector $\boldsymbol{\pi}$ (see Equations (12) (15), has a structure given by

$$\mathbf{Q} \;=\; \begin{bmatrix} -N\lambda & N\lambda\beta & 0 & 0 & \cdots & 0 & 0 \\ S^0 & S-(N-1)\lambda I & (N-1)\lambda I & 0 & \cdots & & \\ 0 & S^0B^0 & S-(N-2)\lambda I & (N-2)\lambda I & \cdots & \vdots & \vdots \\ 0 & 0 & S^0B^0 & S-(N-3)\lambda I & \cdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & & \\ & & & & \cdots & S-\lambda I & \lambda I \\ 0 & 0 & 0 & 0 & \cdots & S^0B^0 & S \end{bmatrix}, \tag{31}$$

where $\beta = [1 \quad 0]$, $S = \begin{bmatrix} -\mu_C & \mu_C \\ 0 & -\mu_R \end{bmatrix}$, $S^0 = \begin{bmatrix} 0 \\ \mu_R \end{bmatrix}$, $B^0 = [1 \quad 0]$, and I denotes the 2×2 identity matrix.

The global balance equations, i.e., Equation (16), for the $M/GE_2/1//N$ queueing system can be expressed as

$$-\,N\lambda\,\boldsymbol{\pi}_0 \;+\; \boldsymbol{\pi}_1 S^0 \;\;=\;\; 0, \tag{32}$$

$$N\lambda\,\boldsymbol{\pi}_0\,\beta \;+\; \boldsymbol{\pi}_1(S-(N-1)\lambda I) \;+\; \boldsymbol{\pi}_2 S^0B^0 \;\;=\;\; 0, \tag{33}$$

$$(N-i+1)\lambda\,\boldsymbol{\pi}_{i-1} \;+\; \boldsymbol{\pi}_i(S-(N-i)\lambda I) \;+\; \boldsymbol{\pi}_{i+1}S^0B^0 \;\;=\;\; 0, \quad 2 \leq i \leq N-1, \tag{34}$$

and

$$\lambda\,\boldsymbol{\pi}_{N-1} \;+\; \boldsymbol{\pi}_N\, S \;\;=\;\; 0. \tag{35}$$

By multiplying Equations (33) - (35) on the right with the column vector $\mathbf{e}$ and simplifying, we obtain

$$\boldsymbol{\pi}_1\, S^0 \;\;=\;\; N\lambda\,\boldsymbol{\pi}_0, \tag{36}$$

$$\boldsymbol{\pi}_{i+1}\, S^0 \;\;=\;\; (N-i)\lambda\,\boldsymbol{\pi}_i\,\mathbf{e}, \quad 1 \le i \le N-1, \tag{37}$$

and

$$\boldsymbol{\pi}_N\, S^0 \;\;=\;\; \lambda\,\boldsymbol{\pi}_{N-1}\,\mathbf{e}. \tag{38}$$

We then multiply Equation (37) on the right by B^0 and substitute the result into Equations (33) through (35), which yields

$$N\lambda\,\boldsymbol{\pi}_0\,\beta \;\;=\;\; \boldsymbol{\pi}_1(\,(N-1)\lambda I \;-\; (N-1)\lambda\mathbf{e}B^0 \;-\; S\,) \tag{39}$$

and

$$(N-i+1)\lambda\,\boldsymbol{\pi}_{i-1} \;\;=\;\; \boldsymbol{\pi}_i(\,(N-i)\lambda I \;-\; (N-i)\lambda\mathbf{e}B^0 \;-\; S\,), \quad 2 \le i \le N. \tag{40}$$

Let $\widetilde{R}(i) \equiv (N-i)\lambda I - (N-i)\lambda\mathbf{e}B^0 - S$, $1 \le i \le N$. It follows from Equations (39) and (40) that

$$\boldsymbol{\pi}_1 \;\;=\;\; N\lambda\,\boldsymbol{\pi}_0\,\beta\,\widetilde{R}(1)^{-1} \tag{41}$$

and

$$\boldsymbol{\pi}_i \;\;=\;\; (N-i+1)\lambda\,\boldsymbol{\pi}_{i-1}\,\widetilde{R}(i)^{-1}, \quad 2 \le i \le N. \tag{42}$$

One can easily show that $\widetilde{R}(i)$ is nonsingular, and thus $\widetilde{R}(i)^{-1}$ exists, $1 \le i \le N$. We define $\widehat{R}(i) \equiv \widetilde{R}(i)^{-1}$, $1 \le i \le N$, which can be expressed as

$$\widehat{R}(i) \;\;=\;\; \begin{bmatrix} \frac{(N-i)\lambda}{\mu_R\mu_C} + \frac{1}{\mu_C} & \frac{1}{\mu_R} \\ \frac{(N-i)\lambda}{\mu_R\mu_C} & \frac{1}{\mu_R} \end{bmatrix} \quad 1 \le i \le N. \tag{43}$$

The solution of the global balance equations for the $M/GE_2/1//N$ queue is therefore given by

$$\boldsymbol{\pi}_i \;\;=\;\; \frac{N!\lambda^i}{(N-i)!}\,\boldsymbol{\pi}_0\,\beta \prod_{k=1}^{i} \widehat{R}(k), \quad 0 \le i \le N. \tag{44}$$

The normalization constraint (i.e., Equation (17)) for this case can be written as

$$\sum_{i=0}^{N} \boldsymbol{\pi}_i\,\mathbf{e} \;\;=\;\; 1.$$

Substituting (44) into this equation yields the following closed-form expression

$$\boldsymbol{\pi}_0 \;\;=\;\; \left[\, \sum_{i=0}^{N} \frac{N!\lambda^i}{(N-i)!}\,\beta \prod_{k=1}^{i} \widehat{R}(k)\,\mathbf{e} \,\right]^{-1}. \tag{45}$$

We note that the above derivation can be used to obtain the corresponding results under any phase-type service distribution, where the parameters β, S, S^0 and B^0 are modified accordingly, and the matrices $\widehat{R}(i)$, $1 \le i \le N$, are obtained numerically.

The processor utilization for the $M/GE_2/1//N$ case is obtained via Equations (30) and (45). We note that this utilization result is identical to the result obtained by summing over the steady-state probabilities $\boldsymbol{\pi}_i \cdot [0 \;\; 1]$, $1 \le i \le N$.

3.3 Validation

Analogous to Section 2.3, we compare the results of our discrete-time and continuous-time models developed in Sections 3.1 and 3.2, respectively, with results that have appeared in the research literature. We use for this purpose the model developed by Saavedra et. al. [19], which was compared by the paper's authors against the simulation results reported in [27]. The model of Saavedra et. al. assumes a constant context-switch delay C, a constant remote memory access latency L, and a geometric distribution with parameter $1/R$ for the times between remote references for each thread.

As previously noted, a context-switching event in large latency multithreaded architectures is typically caused by a cache miss. We note that increasing the number of i-threads among which the multithreaded processor switches (i.e., the value of N) tends to increase the cache miss ratio. Given the inverse relationship between the cache miss ratio and the mean time between remote references for each thread, the latter is actually a monotonically nonincreasing function of N instead of the fixed value R. To be consistent with the results in [19], we use the function $R(N)$

$$R(N) = \begin{cases} R(1)N^{-\delta}\ , & if\ N \leq \lfloor R(1)^{1/\delta} \rfloor, \\ 1\ , & if\ N > \lfloor R(1)^{1/\delta} \rfloor, \end{cases} \tag{46}$$

to approximate the mean time between remote memory requests for each thread. This function is used in place of the fixed value R in each of the models considered.

Saavedra et. al. [19] consider remote memory latencies $L = 32$, 64 and 128 cycles, context-switch delays $C = 0$, 1, 4 and 16 cycles, and intermiss times for each thread given by Equation (46) where $\delta = 0.57$ and $R(1) = 16$, 32 and 64 cycles. We restate these results in Figure 3, together with the corresponding results obtained from our discrete-time (DT) and continuous-time (CT) models; in the latter case, we use the $M/GE_2/1//N$ instance of the model. Note that our DT and CT models replace the constant context-switch and cache-miss latencies of Saavedra et. al. with geometric and exponential parameter distributions, respectively.

The following inferences can be made from these results:

- There is a good correlation between the maximum utilization predictions of the two models. The average discrepancy in predicted maximums is 2.9 percent, whereas the worst-case discrepancy is 14.3 percent. Therefore the proposed approximations which yield simpler models (due to the memoryless nature of the distributions) also offer a tolerable discrepancy with respect to the constant delay distribution model of Saavedra et. al. Furthermore, as previously discussed, for many large latency multithreaded systems [27], the probability distributions considered for the memory access delay may also be more realistic than the constant delay assumption.

- For cases with zero context switch delay, the utilization increase in the proposed models with larger numbers of i-threads always offsets the loss of utilization due to increased cache-miss probability (given by the inverse of Equation (46) for the assumed value of $\delta = 0.57$). Whereas, the Saavedra et. al. model yields a non-monotonic utilization plot in these cases. Note that this discrepancy can only be due to the more realistic probability distributions for the memory access delay used in the proposed models as opposed to the constant delay assumption of the other model. In other words, all discrepancies in these zero context-switch overhead cases are unrelated to the less realistic probability distributions for the context-switch overhead in the proposed models. Note that the proposed models have also been separately validated in the previous section, assuming zero context-switch overhead.

- In the cases assuming non-zero context-switch overhead, a good correlation can also be seen in the predicted number of threads required for the maximum utilization, half of the maximum utilization and three-fourth of the maximum utilization.

4 Results

In this section, the analytic models developed in the previous sections are used to derive some results that are useful for assessing the performance potential of multithreading in the current state of processor technology. An assessement of the current commercial processor technology, as it relates to multithreading, follows. As previously mentioned, the gap between processor cycle times and memory cycle times is currently on the rise. This means that assuming constant run-length, the ratio of memory latency (L) to run-length (R) is increasing. However, new compiler techniques of aggressive prefetching of instructions and data, and memory consistency models which allow loads to be percolated arbitrarily ahead of non-conflicting stores, are enabling the compiler to reduce cache-misses, i.e., increase the run-length (R). Therefore, the ratio L/R may not increase as much as L. For uniprocessor benchmarks with relatively large working sets, such as scientific and transaction processing benchmarks with respect to L1 cache miss (i.e., L2 cache access), typical run-lengths may be as small as 10-20 cycles, whereas those with respect to memory access may be around 50-100 cycles [24]. A similar range of run-lengths is also reported by Weber and Gupta [27] using multiprocessor traces of scientific benchmarks. On the other hand, typical L2 access latencies are in the 10-20 cycles range, but memory access time is expected to grow from currently around 50 cycles to around 100-200 cycles.

The context-switch overhead (C) for large-latency machines cannot be null due to the significant amount of required state saving and restoration. If the context is being switched to a thread that has very little sharing of state with the current state (e.g., switch to a different process thread), most of the current machine state consisting of architected registers, etc. must be saved and the new state restored. However, with techniques such as multiple on-chip register banks, the context-switch may simply imply switching to an alternate bank. The latter is a more hardware-intensive solution but can be quite feasible for machines with only a few threads. Since most of the current commercial processors are only beginning to include multithreading support, the level of support would mostly be restricted to 2 4 threads. The context switch overhead (C) in such a case may be within 1 to 4 cycles.

The cache-interference factor, δ, is assumed to be 0.57, as used before in Figure 3. Although, this value of δ may seem high, it helps in restricting the reported performance advantage of multithreading to conservative levels. Using the model developed in the previous section, and keeping the above scenario in mind, a set of performance plots (Figures 4 through 6) have been generated. The context-switch overhead is assumed to be 1, 2 and 4 cycles in Figures 4, 5 and 6, respectively. The L/R ratio is varied from 1 to 1.5 to 2.0, with run-lengths of 16, 32, 64, and 128.

We make the following observations from these results:

1. The single-thread performance (as measured by processor utilization) can be significantly boosted with a second thread, even with context-switch overheads of 4 cycles and L/R ratio of 2, almost irrespective of the actual run-length.

2. If the context-switch overhead is kept within 2 cycles, a set of around 5 threads would be enough for doubling the single-thread performance (as measured by processor utilization) even for small run-lengths of 16 cycles, with an L/R ratio of 2. Around 3 to 5 threads would be sufficient to significantly improve the single-thread utilization for larger run-lengths.

3. If the context-switch overhead increases to 4 cycles, single-thread performance cannot be doubled for small run-lengths of 16 and L/R ratios of greater than one. In other words, for machines multithreaded to hide L2 access latencies, where such small run-lengths may be quite typical, limiting the context-switch overhead to around 2 cycles would be quite crucial to realizing a good performance boost.

4. As the number of threads keep increasing, the cache interference dominates and the performance starts to degrade, as expected. The performance in this region is almost completely determined by the context-switch overhead, C, and the run-length, R, and is independent of the miss latency, L. More specifically, the performance here is determined by the ratio R/C. This is clearly seen in Figures 4 through 6, as the curves corresponding to different values of the L/R ratio, but a common run-length and context-switch overhead ratio, merge.

We next use our analytic models to briefly consider the performance impact of the memory latency distribution and to show an important insensitivity result. Consider the continuous-time model of Section 3.2 under the assumptions that the times between occurrences of remote memory requests (including context-switching delays) have an exponential distribution $B(\cdot)$ with mean $m_B = C + R < \infty$ and the times to satisfy outstanding remote memory requests have a *general distribution* $A(\cdot)$ with mean $m_A = L < \infty$. In this case our model is equivalent to a well-known closed queueing network model, commonly called the machine repair model [12]. This model consists of N identical customers (i.e., threads) and two service centers, the first of which has a single FCFS server with exponential service (i.e., the processor) whereas the second has N servers and general service demands (i.e., the parallel memory system). This queueing network has a product-form solution in which the processor utilization (as well as certain other performance measures) depends only upon the service means at the two service centers m_A and m_B, and not on the detailed form of their service distributions $A(\cdot)$ and $B(\cdot)$. This insensitivity property, where certain invariant probabilities depend on a set of probability distributions only through their mean values, is a product-form result that follows from the quasi-reversible network of queues [26].

It therefore follows that the processor utilization for the multithreaded architecture considered above is insensitive to the probability distribution used to model remote memory access delays, i.e., the utilization is the same no matter what probability distribution is chosen. In particular, this implies that a major cause of the discrepancies in Figure (3) are due to the less variable memory latencies *and* the less variable intermiss times.

5 Conclusion

In this paper, we have presented a comprehensive analytic framework to quantitate the performance potential of a wide spectrum of multithreaded machines, ranging from those that are aimed at hiding small latencies and are capable of switching threads every cycle with negligible context-switch overheads, to those that switch threads only on long inter-instruction latencies, and have significant context-switch overhead.

The paper also compares the modeling accuracy of discrete-time and continuous-time models in such a context. The models also use phase-type distributions for modeling miss latencies. The use of phase-type distributions is motivated by their mathematical properties which yield simpler analytic model and by their ability to approximate many input parameter distributions with sufficient accuracy in most practical cases. The model outputs are compared against previously published simulation and modeling results under similar input conditions.

The models developed are also used to assess the performance potential of multithreading in the current state of processor technology. The results show that the context switching overhead greatly influences the machine performance. Both the maximum throughput and the number of threads required to achieve this maximum are sensitive to the context switch overhead, as is evident from Figures 3 through 6. Specifically, even though one might be tempted to ignore the switching overhead when it is only a small fraction (1-5%) of the average memory latency or run length, results show that it disproportionately effects the performance.

The single-thread performance can be significantly boosted with a second thread, under almost all input conditions of interest. If the context-switch overhead does not exceed two cycles, a set of around five threads is enough for doubling the single-thread performance even for small run-lengths, high cache-miss latencies, and quite high cache-interference. For machines multithreaded to hide L2 access latencies (implying small run-lengths), limiting the context-switch overhead to around two cycles would be quite crucial to good performance boost.

References

[1] A. Agarwal, "Performance Tradeoffs in Multithreaded Processors," *IEEE Trans. on Parallel and Distributed Systems*, Vol. 3, No. 5, Sep. 1992, pp. 525-539.

[2] A. Agarwal, B.-H. Lin, D. Kranz and J. Kubiatowicz, "APRIL: A Processor Architecture for Multiprocessing," *Proc. of the 17th Annual International Symposium on Computer Architecture*, May 1990, pp. 104-114.

[3] P. Dubey, A. Krishna and M. Flynn, "Analytical Modeling of Multithreaded Pipeline Performance," *Proc. of the 27th Hawaii International Conference on System Sciences*, Hawaii, Jan 1994, pp. 361-367.

[4] M. J. Faddy, "Fitting Structured Phase-Type Distributions," Technical Report, Department of Mathematics, University of Queensland, Australia, April 1994. To appear, *Applied Stochastic Models and Data Analysis.*

[5] M. J. Flynn and A. Podvin, "Shared Resource Multiprocessing," *IEEE Computer*, Vol. 5, Mar./Apr. 1972, pp. 20-28.

[6] D. Gross and C. Harris, *Fundamentals of Queueing Theory*, Second Edition, John Wiley and Sons, New York, 1985.

[7] R. H. Halstead Jr. and F. Fujita, "MASA: A Multithreaded Architecture for Parallel Symbolic Computing," *Proc. of the 15th Annual Int. Symp. on Comp. Arch.*, Honolulu, Hawaii, June 1988, pp. 443-451.

[8] H. Hirata, K. Kimura, S. Nagamine, Y. Mochizuki, A. Nishimura, Y. Nakase and T. Nishizawa, "An Elementary Processor Architecture with Simultaneous Instruction Issuing from Multiple Threads," *Proc. of the 19th Annual Int. Symp. on Comp. Arch.*, Gold Coast, Australia, May 1992, pp. 136-145.

[9] S. Karlin and H. M. Taylor, *A First Course in Stochastic Processes*, Second Edition, Academic Press, 1975.

[10] L. Kleinrock, *Queueing Systems Volume I: Theory*, John Wiley and Sons, 1975.

[11] K. Kurihara, D. Chaiken and A. Agarwal, "Latency Tolerance Through Multithreading in Large Scale Multiprocessors," *Proc. of the Second International Symposium on Shared Memory Multiprocessing*, Tokyo, Japan, April, 1991, pp. 91-101.

[12] S. S. Lavenberg, Ed., *Computer Performance Modeling Handbook*, Academic Press, 1983.

[13] T. E. Mankovich, V. Popescu and H. Sullivan, "CHoPP Principles of Operation," *Proc. of the Second International Conference on Supercomputing*, May, 1987, Vol. 1, pp. 2-10.

[14] D. C. McCrackin, "Eliminating Interlocks in Deeply Pipelined Processors by Delay Enforced Multistreaming," *IEEE Trans. on Computers*, Vol. 40, No. 10, Oct. 1991, pp. 1125-1132.

[15] M. F. Neuts, "Probability Distributions of Phase Type," In *Liber Amicorum Prof. Emeritus H. Florin*, Department of Mathematics, University of Louvain, Belgium, 1975, pp. 173-206.

[16] M. F. Neuts, *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*, The Johns Hopkins University Press, 1981.

[17] M. F. Neuts, *Structured Stochastic Matrices of M/G/1 Type and Their Applications*, Marcel Dekker, 1989.

[18] S. M. Ross, *Stochastic Processes*, John Wiley and Sons, New York, 1983.

[19] R. H. Saavedra-Barrera, R. H. Cueller and T. von Eicken, "Analysis of Multithreaded Architectures for Parallel Computing," *SPAA '90, Second Annual ACM Symposium on Parallel Algorithms and Architectures*, Crete, Greece, July, 1990, pp. 169-178.

[20] R. Schassberger, "Insensitivity of Steady-State Distributions of Generalized Semi-Markov Processes, Part I," *The Annals of Probability*, Vol. 5, No. 1, 1977, pp. 87-99.

[21] R. Schassberger, "Insensitivity of Steady-State Distributions of Generalized Semi-Markov Processes, Part II," *The Annals of Probability*, Vol. 6, No. 1, 1978, pp. 85-93.

[22] R. Schassberger, "Insensitivity of Steady-State Distributions of Generalized Semi-Markov Process with Speeds," *Advances in Applied Probability*, Vol. 10, 1978, pp. 836-851.

[23] B. J. Smith, "A Pipelined, Shared Resource MIMD Computer," *Proc. of the 1978 International Conference on Parallel Processing*, 1978, pp. 6-8.

[24] J-H. Tang, and K. So, "Performance and Design Choices of Level-two Caches," *Proc. of the 27th Hawaii International Conference on System Sciences,* Hawaii, Jan 1994, pp. 422-430.

[25] M. R. Thistle and B. J. Smith, "A Processor Architecture for Horizon," *Supercomputing '88*, Florida, Oct. 1988, pp. 35-40.

[26] J. Walrand, *An Introduction to Queueing Networks*, Prentice Hall, 1988.

[27] W. Weber and A. Gupta, "Exploring the Benefits of Multiple Hardware Contexts in a Multiprocessor Architectures: Preliminary Results," *Proc. of the 16th Annual International Symposium on Computer Architecture*, Jerusalem, Israel, June 1989, pp. 273-280.

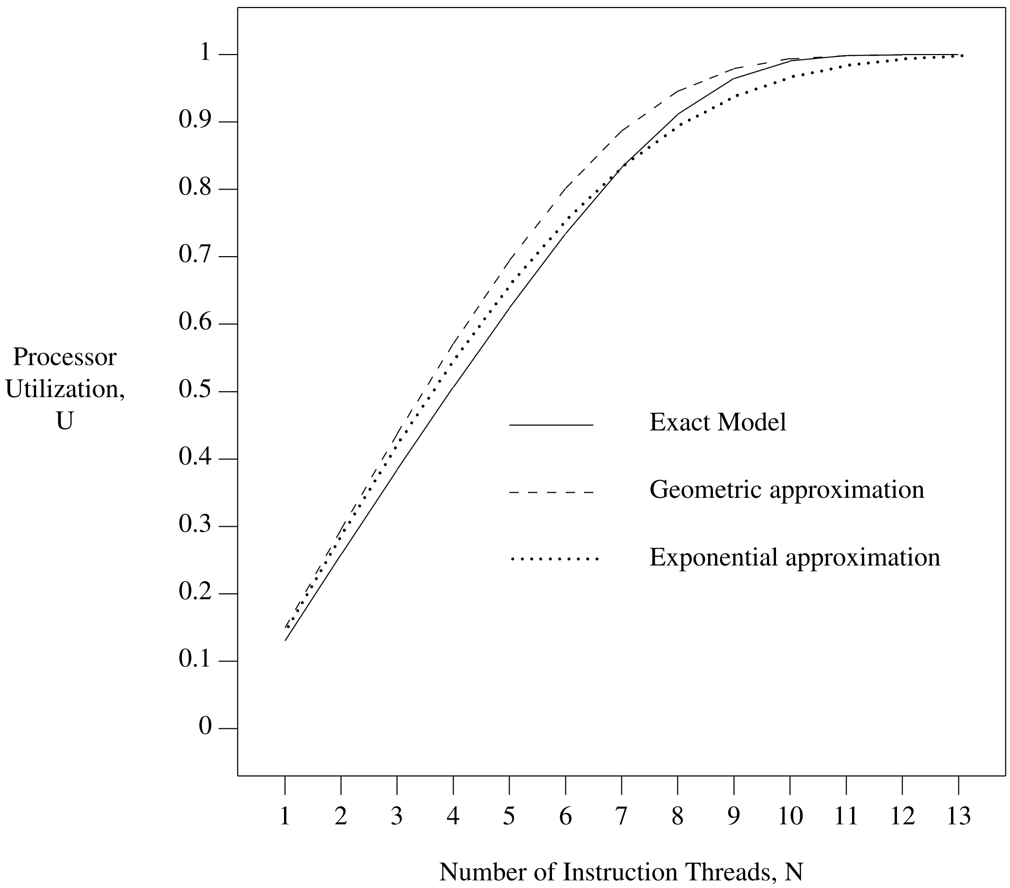

Figure 1: Comparing the exact model output with the discrete-time (geometric) approximate model and the continuous-time (exponential) approximation (M/M/1//N). Input distribution used is that corresponding to Case 1, as described in Section 2. Worst-case discrepancy in the predicted utilizations of the approximate models is 15 percent with respect to the exact analytic model.

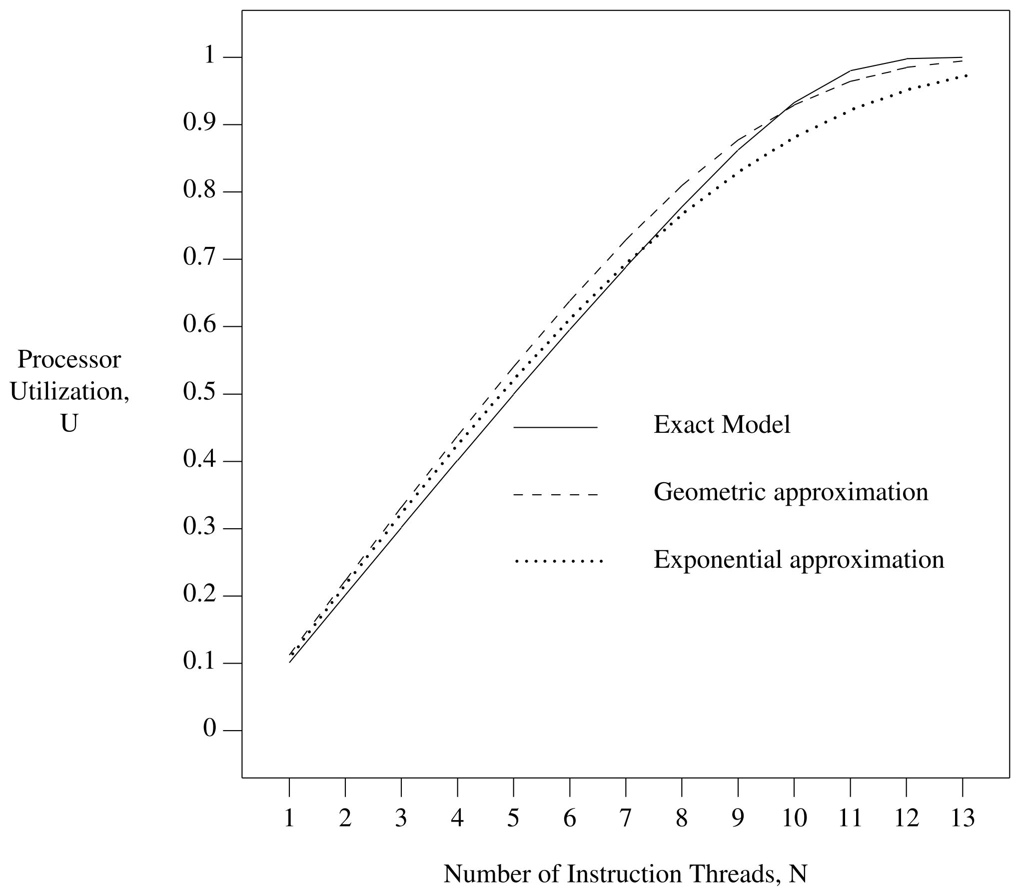

Figure 2: Comparing the exact model output with the discrete-time (geometric) approximate model and the continuous-time (exponential) approximation (M/M/1//N). Input distribution used is that corresponding to Case 2, as described in Section 2. Worst-case discrepancy in the predicted utilizations of the approximate models is 11 percent with respect to the exact analytic model.

Parameters			Max Utilization		Number of Threads for					
					Max Util		0.5(Max Util)		0.75(Max Util)	
L	C	R	Proposed DT / CT	SCvE	Proposed DT / CT	SCvE	Proposed DT / CT	SCvE	Proposed DT / CT	SCvE
32	0	16	1.0 / 1.0	0.993	- / -	16	3 / 2	2	6 / 6	6
32	0	32	1.0 / 1.0	0.991	- / -	8	1 / 1	1	3 / 2	3
32	0	64	1.0 / 1.0	0.993	- / -	5	1 / 1	1	2 / 1	2
32	1	16	0.790 / 0.786	0.803	11 / 11	10	2 / 1	2	4 / 4	4
32	1	32	0.904 / 0.902	0.908	7 / 8	7	1 / 1	1	2 / 2	2
32	1	64	0.958 / 0.957	0.959	5 / 5	5	1 / 1	1	2 / 1	1
32	4	16	0.564 / 0.561	0.594	6 / 6	5	1 / 1	1	2 / 2	2
32	4	32	0.747 / 0.746	0.759	5 / 5	5	1 / 1	1	2 / 2	1
32	4	64	0.871 / 0.871	0.864	4 / 4	4	1 / 1	1	2 / 1	1
32	16	16	0.320 / 0.319	0.348	3 / 3	3	1 / 1	1	1 / 1	1
32	16	32	0.497 / 0.496	0.528	3 / 3	2	1 / 1	1	1 / 1	1
32	16	64	0.678 / 0.677	0.705	2 / 2	2	1 / 1	1	1 / 1	1
64	0	16	1.0 / 1.0	0.990	- / -	33	7 / 7	7	18 / 18	17
64	0	32	1.0 / 1.0	0.991	- / -	17	3 / 2	3	6 / 6	6
64	0	64	1.0 / 1.0	0.990	- / -	8	1 / 1	1	3 / 2	3
64	1	16	0.712 / 0.707	0.727	23 / 24	21	4 / 3	4	9 / 8	9
64	1	32	0.868 / 0.866	0.874	14 / 14	13	2 / 2	2	5 / 4	4
64	1	64	0.945 / 0.944	0.946	9 / 9	2	1 / 1	1	3 / 2	2
64	4	16	0.477 / 0.474	0.507	11 / 11	10	2 / 2	2	4 / 4	4
64	4	32	0.683 / 0.681	0.702	8 / 8	8	2 / 1	1	3 / 3	3
64	4	64	0.839 / 0.838	0.846	6 / 6	6	1 / 1	1	2 / 2	2
64	16	16	0.263 / 0.262	0.301	5 / 5	4	1 / 1	1	2 / 2	1
64	16	32	0.437 / 0.436	0.472	4 / 4	4	1 / 1	1	2 / 1	1
64	16	64	0.628 / 0.627	0.657	4 / 4	3	1 / 1	1	2 / 1	1
128	0	16	1.0 / 1.0	0.991	- / -	61	28 / 27	26	70 / 71	40
128	0	32	1.0 / 1.0	0.991	- / -	45	7 / 7	7	18 / 18	17
128	0	64	1.0 / 1.0	0.990	- / -	17	3 / 2	3	6 / 6	6
128	1	16	0.605 / 0.600	0.706	56 / 57	42	9 / 9	13	23 / 22	29
128	1	32	0.808 / 0.805	0.815	32 / 32	30	5 / 4	5	11 / 11	11
128	1	64	0.922 / 0.921	0.924	17 / 17	16	2 / 2	2	5 / 5	5
128	4	16	0.381 / 0.379	0.406	23 / 23	21	4 / 3	4	9 / 9	10
128	4	32	0.595 / 0.593	0.615	17 / 17	15	3 / 2	3	6 / 6	6
128	4	64	0.787 / 0.786	0.797	11 / 11	10	2 / 1	2	4 / 4	4
128	16	16	0.208 / 0.207	0.236	8 / 8	7	2 / 1	1	3 / 3	4
128	16	32	0.363 / 0.362	0.396	7 / 7	6	1 / 1	1	3 / 2	3
128	16	64	0.561 / 0.561	0.589	6 / 6	6	1 / 1	1	2 / 2	2

Figure 3: Comparing outputs from the proposed analytic models with those of the model developed by Saavedra et. al., referred to as SCvE. The cache-miss probability, assuming a single thread, is given by $1/R$ and the empirically derived constant δ is assumed to be 0.57. If an i-thread is delayed due to cache-miss, it is delayed on an average for $1/L$ cycles and an average context-switch lasts $1/C$ cycles. The average discrepancy between the maximum utilization prediction of the two models is less than 3 percent and the worst-case discrepancy is 14 percent. A '-' indicates a lack of any drop in utilization. A good correlation can also be seen (in the cases assuming non-zero context-switch overhead) in the predicted number of threads required for the maximum utilization, half of the maximum utilization and three-fourth of the maximum utilization.

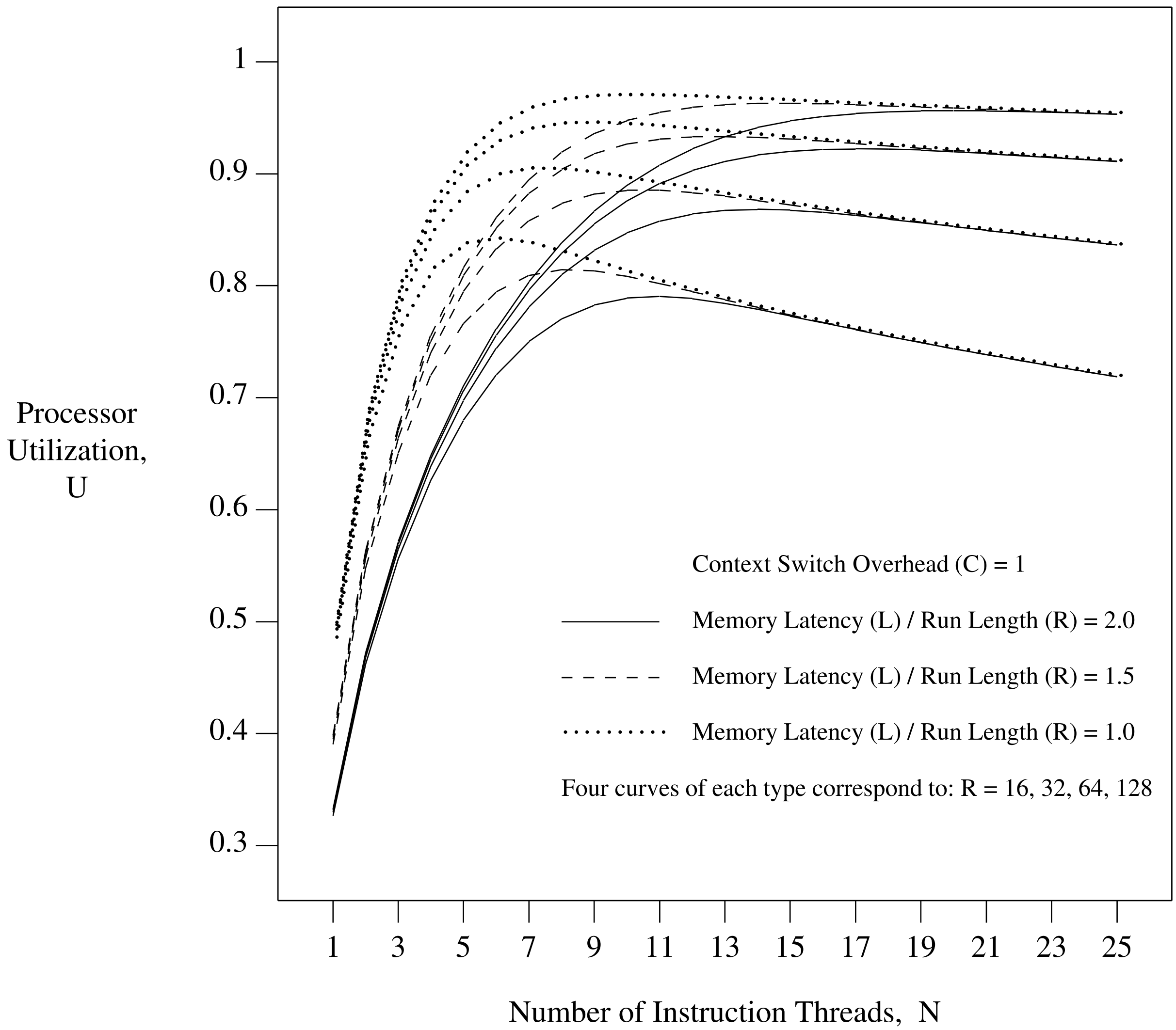

Figure 4: Processor utilization as a function of the number of instruction threads. The utilization is based on the analytic model developed in the paper. The cache interference factor, δ is assumed to be 0.57.

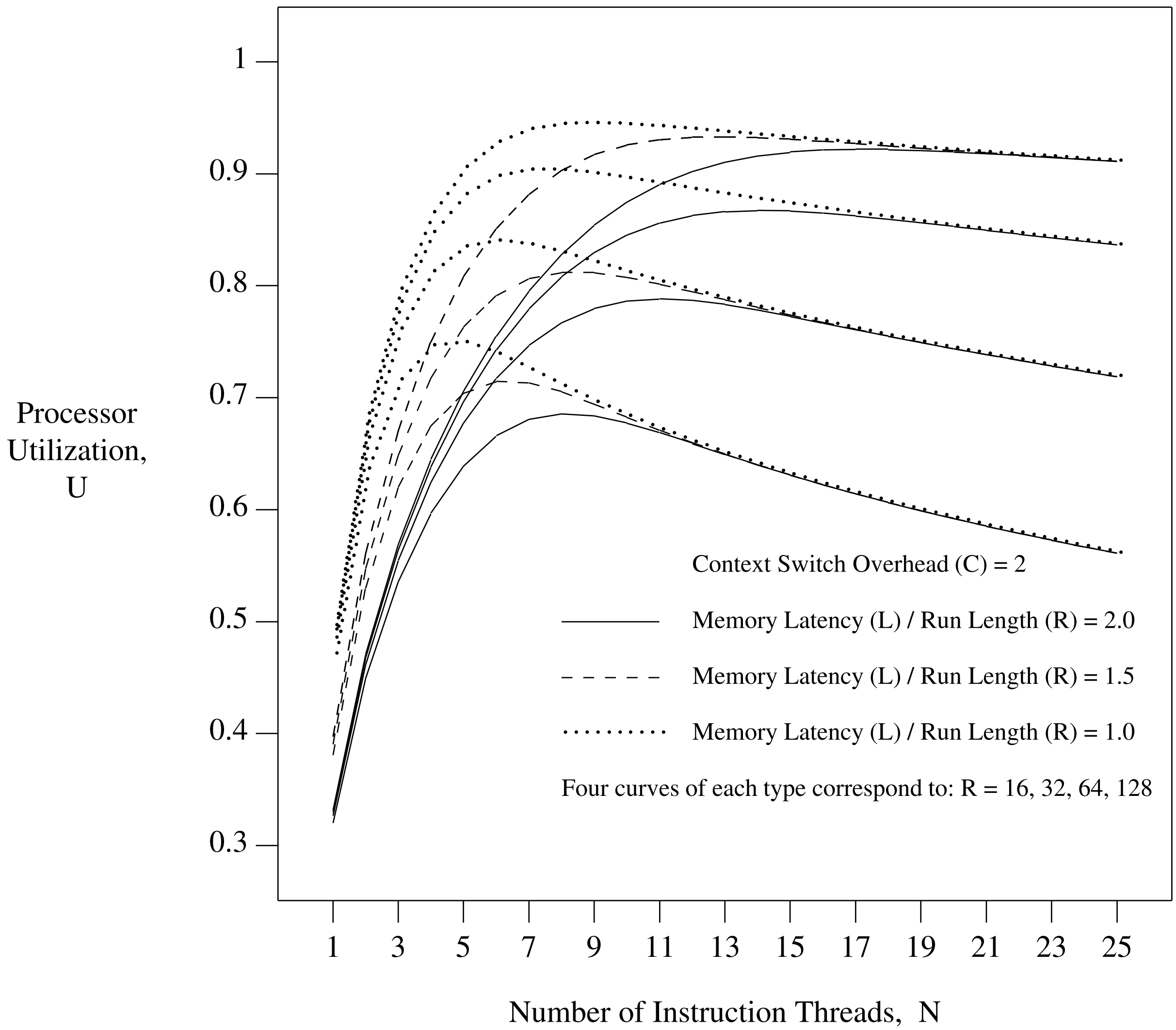

Figure 5: Processor utilization as a function of the number of instruction threads. The utilization is based on the analytic model developed in the paper. The cache interference factor, δ is assumed to be 0.57.

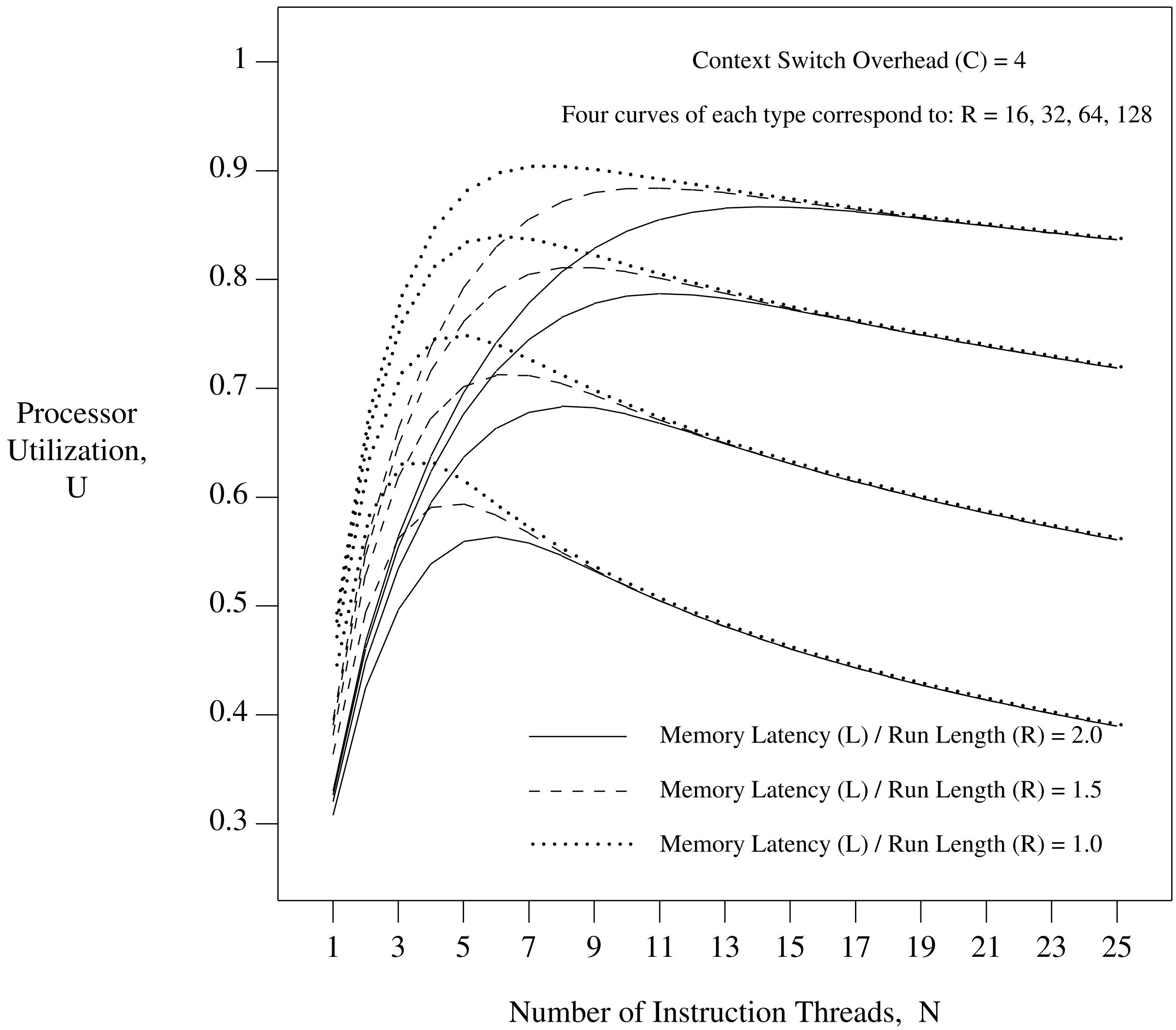

Figure 6: Processor utilization as a function of the number of instruction threads. The utilization is based on the analytic model developed in the paper. The cache interference factor, δ is assumed to be 0.57.