Analytic Modeling of Processor Utilization in Multithreaded Processor Architectures

Mark S. Squillante
IBM Research Division
T. J. Watson Research Center
Yorktown Heights, NY 10598
mss@watson.ibm.com

April 1994

Abstract

In this paper, we develop an analytic model of processor utilization in multithreaded processor architectures that supports both serial and parallel processing of memory requests. The system is modeled as a finite, continuous-time Markov chain whose solution can be obtained efficiently. Although it applies more generally, our modeling approach supports an important class of probability distributions that can be used to approximate the distributions of interest with sufficient accuracy in most practical cases. This results in an efficient and accurate model across a wide variety of system environments.

1 Introduction

Multithreaded processor architectures have received considerable attention in recent years, as evidenced by a number of important research papers on the subject; e.g., [16, 2, 11, 4, 1, 15]. These systems attempt to increase processor utilization by switching to a new *thread* of execution while other threads are waiting for previously issued memory requests to be satisfied, thus overlapping memory latency with useful computation.

Analytic modeling has played a significant role in providing a quantitative understanding of the tradeoffs involved in multithreaded processor architectures; e.g., [11, 1]. In this paper, we develop a new analytic model of processor utilization for such architectures, which supports both serial and parallel satisfying of memory requests. The system is modeled as a finite, continuous-time Markov chain whose solution can be obtained efficiently. Our modeling approach supports an important class of probability distributions that can be used to approximate the distributions of interest with sufficient accuracy in most practical cases. This results in an efficient and accurate model across a wide variety of system environments. We note, however, that our approach is not limited to this restriction and thus applies more generally.

We first present our Markov chain model of multithreaded processor architectures and an upper bound on processor utilization based on this model. We then derive an approximation for effective processor utilization in multithreaded systems based on a simple estimate of the proportion of processor busy time that reflects useful processor cycles. A particular instance of our model is used to demonstrate that an intuitively appealing lower bound on processor utilization, namely ignoring the context switching overhead, is in fact not a lower bound at all. We also show that using such a model to approximate processor utilization (when the context switching time is relatively small) can yield fairly significant errors in comparison to the model developed in this paper for most of the cases considered.

Throughout this paper we assume that the reader is familiar with multithreaded processor architectures, as well as the various issues involved in multithreaded processors and in the modeling of processor utilization for such architectures. We refer the interested reader to [16, 2, 11, 4, 1, 15] and the references therein for more details.

2 Analytic Model

We consider a multithreaded processor architecture with p hardware contexts, i.e., at most p threads are simultaneously resident on a processor. The processor executes a thread until one of its memory requests "misses", at which point the processor switches to the execution of another thread. For the purposes of our modeling study, the actual cause of the miss is not especially important; e.g., context switches may occur whenever a memory reference is not resident in a cache (L1, L2, etc) or the local memory module. Our only assumption in this regard is that the time between such misses is not very small. We refer to each thread that is waiting on a memory reference (i.e., its miss has not yet been satisfied) as being *blocked.* If all of the p threads are blocked, the processor remains idle until one of the outstanding memory requests is satisfied. The individual threads are assumed to be statistically identical.

We model this system as a time-homogeneous, continuous-time Markov chain $\{X(t), t \geq 0\}$ defined over a finite state space $\mathcal{S}$ [6, 10]. Throughout this section we assume the process to be irreducible and ergodic. The states of the Markov chain can be arbitrarily complex, including many levels of system detail. To simplify the model presentation, however, we restrict our subsequent discussion to a specific class of probability distributions. In particular, the times to satisfy outstanding 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 memory request misses as *i.i.d.* random variables with a phase-type distribution $B(\cdot)$ and finite mean m_B. Since certain steady-state averages often depend upon the first few moments of the distributions (and not on their detailed forms) [12, 13], phase-type distributions can be used to approximate the distributions of interest with sufficient accuracy in most practical cases. Furthermore, an important subclass of phase-type distributions can be fitted to empirical data [3]. We note, however, that our modeling approach is not limited to this restriction and thus applies more generally.

It follows from the above assumptions that the state representation of the Markov chain is given by $X(t) \equiv (I(t), J(t), K(t))$, $t \geq 0$, where $I(t)$, $0 \leq I(t) \leq p$, denotes the number of non-blocked threads at time t waiting or receiving service at the processor, $J(t)$, $J(t) \in \{1, ..., n_A + 1\}$, denotes the state of the phase variable at time t for the next memory request to be satisfied, and $K(t)$, $K(t) \in \{1, ..., n_B + 1\}$, denotes the state of the phase variable at time t for the next miss occurrence. While this representation assumes that outstanding memory requests are satisfied serially, the state space can be augmented straightforwardly to handle the processing of misses in a parallel fashion. Moreover, state-dependent transition rates can be used to handle the satisfying of memory requests in a pipelined manner, as well as other system behavior.

We define

$$\pi(i,j,k) \equiv \lim_{t\to\infty} \mathrm{P}\left[\, I(t) = i, J(t) = j, K(t) = k \,\right], \; i = 0, ..., p, \; j = 1, ..., n_A + 1, \; k = 1, ..., n_B + 1, \quad (1)$$

*We refer the interested reader to [8] for a discussion of phase-type distributions. More formal descriptions can be found in [5, 7].

$$L \;\equiv\; \{\,(j,k)\;\mid\; j=1,...,n_A+1,\; k=1,...,n_B+1\,\}, \tag{2}$$

and we let $x_\ell \in L$, $0 \le \ell < N_L \equiv (n_A+1)(n_B+1)$, be any ordering of the elements of L. Using this we define

$$\boldsymbol{\pi} \;\equiv\; (\,\boldsymbol{\pi}_0,\; ...,\; \boldsymbol{\pi}_p\,) \tag{3}$$

where

$$\boldsymbol{\pi}_i \;\equiv\; (\,\pi(i,x_0),\; \pi(i,x_1),\; ...,\; \pi(i,x_{N_L-1})\,),\quad 0 \le i \le p. \tag{4}$$

The limiting probability vector $\boldsymbol{\pi}$ is the stationary distribution for the Markov chain $\{X(t), t \ge 0\}$. The stationary distribution is obtained by solving the global balance equations

$$\boldsymbol{\pi}\mathbf{Q} \;=\; 0 \tag{5}$$

together with

$$\boldsymbol{\pi}\mathbf{e} \;=\; 1, \tag{6}$$

where $\mathbf{Q}$ is the infinitesimal generator matrix for the Markov chain and $\mathbf{e}$ is the column vector of all ones.

Our modeling objective is to estimate processor utilization, denoted by U, in the above multi-threaded processor system. An upper bound on this measure is given by

$$U_{ub} \;=\; 1 \;-\; \boldsymbol{\pi}_0\mathbf{e}. \tag{7}$$

The right-hand-side of equation (7) provides an upper bound because it represents the total proportion of time that the processor is busy, including the wasted processor cycles spent in context switching among the threads. Our approximation of multithreaded processor utilization is based on calculating the fraction of U_{ub} that reflects useful processor cycles.

Let τ be the random variable denoting the generic time between memory request misses for each thread (i.e., the time from when the thread starts to execute until its next miss), whose mean we denote by $\bar{\tau}$, and let C be the random variable denoting the generic context switching time, whose mean we denote by $\bar{C}$. The distribution $B(\cdot)$, which represents the total time between any pair of misses, consists of both τ and C, and $m_B = \bar{C} + \bar{\tau}$. Since context switching is strictly overhead, and thus not an effective use of processor cycles, this factor should be excluded from the processor utilization estimate. It therefore follows that processor utilization in multithreaded processor architectures with p hardware contexts can be approximated as follows

$$U \;=\; (\,1 \;-\; \boldsymbol{\pi}_0\mathbf{e}\,)\;\frac{\bar{\tau}}{\bar{C} \;+\; \bar{\tau}}. \tag{8}$$

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.

Under the assumption of phase-type distributions, our analytic model is equivalent to a $PH/PH/1/p$ queueing system. The stationary distribution for this queue, i.e., the solution of equations (5) and (6), can be obtained via standard numerical methods in many instances. Other techniques can be employed for sufficiently large models in which the costs of these methods prove to be prohibitive. In particular, we note that closed-form solutions exist for finite versions of the $M/PH/1$ and $PH/M/c$ queues [8]. Since there are no important mathematical differences between the corresponding theorems for discrete parameter Markov chains, it is important to note that our approach can be applied more generally to also include discrete phase-type distributions. A subset of the routines provided by the MAGUS performance modeling tool [14] can be used to perform all such numerical computations. We refer the interested reader to [8, 9] for more technical details on $PH/PH/1$ queueing systems.

3 Example

The r-stage Erlang distribution (as well as the hyperexponential distribution) is a simple example of a phase-type distribution. We therefore consider here an $E_{10}/M/1/p$ queueing system, which reflects near-deterministic memory latencies and exponential intermiss times,† with mean "arrival" rate $\lambda = 1/m_A$ and mean "service" rate $\mu = 1/m_B = 1/(\bar{C} + \bar{\tau})$. Our choice of this model is motivated by the fact that memory latency is often very close to being a constant (recall our assumption here that memory requests are processed serially, but that our approach is not limited to this restriction – see Section 2) and by the expectation that intermiss times are much more variable. A physical interpretation of the states $(i, j, 1)$, for all $j = 1, \ldots, 10$ when $0 \le i \le p-1$ and for $j = 1$ when $i = p$, is that $p - i$, $0 \le i \le p$, represents the number of outstanding misses, i.e., the number of blocked threads. The generator matrix $\mathbf{Q}$ for this queueing system has dimension $(10p+1) \times (10p+1)$ and is given by

$$\mathbf{Q} = \begin{bmatrix} B_{0,0} & A_{0,0} & 0 & 0 & & \cdots & & 0 \\ A_{2,1} & A_{1,1} & A_{0,1} & 0 & & \cdots & & 0 \\ 0 & A_{2,2} & A_{1,2} & A_{0,2} & & \cdots & & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & & \vdots \\ 0 & 0 & & \cdots & & A_{2,p-1} & A_{1,p-1} & A_{0,p-1} \\ 0 & 0 & & \cdots & & 0 & A_{2,p} & A_{1,p} \end{bmatrix} \tag{9}$$

†Note that the r-stage Erlang distribution becomes more deterministic with increasing r, where the limit as $r \to \infty$ converges to a constant with probability 1, and that the $r = 1$ case is simply the exponential distribution. We also note that the 10-stage Erlang distribution used for the arrival process has a squared coefficient of variation equal to 0.1, whereas its value is 0 when the arrival process is deterministic.

where $A_{2,n} = \mu I$, $1 \le n \le p-1$, $A_{1,n} = -(10\lambda + \mu)I + 10\lambda J$, $1 \le n \le p-1$, $B_{0,0} = A_{1,1} + \mu I$, $A_{1,p} = -\mu$, $A_{2,p} = [\mu, Z_{1,9}]$, $A_{0,n} = \begin{bmatrix} Z_{9,9} & Z_{9,1} \\ 10\lambda & Z_{1,9} \end{bmatrix}$, $0 \le n \le p-2$, $A_{0,p-1} = \begin{bmatrix} Z_{9,1} \\ 10\lambda \end{bmatrix}$. In these matrix definitions we have used $Z_{m,n}$ to denote the $m \times n$ zero matrix, I to denote the 10×10 identity matrix, and J to denote the 10×10 matrix with ones along the diagonal directly above the main diagonal and zeros everywhere else.

The parameters m_A and $\bar{C}$ are both assumed to be independent of p, and we assign to them the values 100 and 10, respectively. The parameter $\bar{\tau}$, however, equals the inverse of the cache miss ratio, which typically decreases with increasing p (because the miss ratio tends to move in the opposite direction). We therefore consider $\bar{\tau}$ as a function of p, with the values: $\bar{\tau}(1) = 110$, $\bar{\tau}(2) = 95$, $\bar{\tau}(4) = 70$ and $\bar{\tau}(8) = 50$. These input parameters are substituted in the above A and B submatrices, recalling that $\lambda = 1/m_A$ and $\mu = 1/(\bar{C} + \bar{\tau})$, and the value of $\boldsymbol{\pi}_0$ is computed using MAGUS. The approximation for effective processor utilization (U) is obtained via equation (8), and an upper bound on this measure (U_{ub}) is given by equation (7). In Table 1 we present the corresponding results. We observe that U_{ub} is not a very tight bound on U, particularly for increasing values of p. Of course, the tightness of the bound depends upon the value of $\bar{\tau}/(\bar{C} + \bar{\tau})$.

p	m_A	$\bar{C}$	$\bar{\tau}$	U_{ub}	U	V
1	100	10	110	0.545	0.500	0.524
2	100	10	95	0.738	0.668	0.704
4	100	10	70	0.752	0.658	0.678
8	100	10	50	0.600	0.500	0.500

TABLE 1
Input Parameters and Results for an $E_{10}/M/1/p$ Queueing System

We also consider, for comparison purposes, another processor utilization approximation. Specifically, we set $\bar{C} = 0$ (hence, $\mu = 1/\bar{\tau}$) and solve the model as described above. Note that in this case equations (7) and (8) are equivalent. The corresponding results, which we denote by V, are also included in Table 1.

Intuitively, one might think that a lower bound on multithreaded processor utilization is obtained by ignoring the context switching time, i.e., processor utilization is bounded below by V. We observe by comparing the values of U and V in Table 1 that this relation does not necessarily hold. In fact, V is always greater than U for all of the cases considered.[‡] The reason for this is that the actual intermiss times, which do include context switching overhead, are not being modeled with sufficient accuracy, thus changing the long-run behavior of the system and, in particular, the steady-state proportion of time the processor is idle. We also observe that using V to approximate multithreaded processor utilization (when the context switching overhead is small relative to the memory latencies and intermiss times) can yield fairly significant errors in comparison with the corresponding values of U, for all cases considered except those for large values of p.

4 Conclusions

In this paper, we developed a new analytic model of processor utilization in multithreaded processor architectures that supports both serial and parallel processing of memory requests. The system is modeled as a finite, time-homogeneous, continuous-time Markov chain whose solution can be obtained efficiently. Our modeling approach supports an important class of probability distributions that can be used to approximate the distributions of interest with sufficient accuracy in most practical cases. This results in an efficient and accurate model across a wide variety of system environments. We note, however, that our approach is not limited to this restriction and thus applies more generally.

We first presented our Markov chain model of multithreaded processor architectures and an upper bound on processor utilization based on this model. We then derived an approximation for effective processor utilization in multithreaded systems based on a simple estimate of the proportion of processor busy time that reflects useful processor cycles. A particular instance of our model was used to demonstrate that an intuitively appealing lower bound on processor utilization, namely ignoring the context switching overhead, is in fact not a lower bound at all. We also showed that using such a model to approximate processor utilization (when the context switching time is relatively small) can yield fairly significant errors in comparison to the model developed in this paper for most of the cases considered.

Acknowledgements

We thank Rick Eickemeyer, Ross Johnson, Hal Kossman, Steve Kunkel, Tim Mullins and Jim Rose of IBM Rochester for originally raising the problem of modeling processor utilization in multithreaded architectures. We also thank Rick Eickemeyer, Ross Johnson and Steve Kunkel for fruitful discussions regarding the research described in this paper.

[‡] Note that $U(p = 8) = 0.49988$ whereas $V(p = 8) = 0.49999$.

References

[1] A. Agarwal. Performance tradeoffs in multithreaded processors. *IEEE Transactions on Parallel and Distributed Systems*, 3(5):525–539, 1992.

[2] A. Agarwal, B.-H. Lin, D. Kranz, and J. Kubiatowicz. APRIL: A processor architecture for multiprocessing. In *Proceedings of 17th International Symposium on Computer Architecture*, May 1990.

[3] 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.*

[4] A. Gupta, J. Hennessy, K. Gharachorloo, T. Mowry, and W.-D. Weber. Comparative evaluation of latency reducing and tolerating techniques. In *Proceedings of 18th International Symposium on Computer Architecture*, May 1991.

[5] A. Jensen. *A Distribution Model Applicable to Economics.* Munksgaard, 1954.

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

[7] M. F. Neuts. Probability distributions of phase type. In *Liber Amicorum Prof. Emeritus H. Florin,* Department of Mathematics, University of Louvain, Belgium, pages 173–206, 1975.

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

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

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

[11] R. H. Saavedra-Barrera, D. E. Culler, and T. von Eicken. Analysis of multithreaded architectures for parallel computing. In *Proceedings of 2nd Annual ACM Symposium on Parallel Algorithms and Architectures*, July 1990.

[12] R. Schassberger. Insensitivity of steady-state distributions of generalized semi-Markov processes, part I. *The Annals of Probability*, 5(1):87–99, 1977.

[13] R. Schassberger. Insensitivity of steady-state distributions of generalized semi-Markov processes, part II. *The Annals of Probability*, 6(1):85–93, 1978.

[14] M. S. Squillante. MAGIC: A computer performance modeling tool based on matrix-geometric techniques. In *Proceedings of the Fifth International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 411–425, February 1991.

[15] R. Thekkath and S. J. Eggers. Impact of sharing-based thread placement on multithreaded architectures. In *Proceedings of 21st International Symposium on Computer Architecture*, pages 176–186, April 1994.

[16] W.-D. Weber and A. Gupta. Exploring the benefits of multiple hardware contexts in a mult-processor architecture: Preliminary results. In *Proceedings of 16th International Symposium on Computer Architecture*, June 1989.