A Customized MVA Model for Shared-Memory Systems with Heterogeneous Applications

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

Technical Report #1400

June 2001

A Customized MVA Model for Shared-Memory Systems with Heterogeneous Applications *

Daniel J. Sorin[†], Jonathan L. Lemon[†], Derek L. Eager[‡], and Mary K. Vernon[†]

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

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

Abstract

We develop and validate an analytical model for evaluating the performance of shared memory systems with ILP processors. We model heterogeneous node behavior, highly bursty memory request traffic, and lock synchronization. The new model is applicable to a broad range of architectures and applications. This technical report provides details regarding the equations that are used to model the system.

1 Introduction

Sorin et al. [7] presented an analytical model for evaluating the performance of shared memory ILP multiprocessors with homogeneous node behavior. This model is a good first step towards modeling such systems, but it has several deficiencies. To overcome its shortcomings, we have developed a new model that handles heterogeneous node behavior, high-variance in processor service times, bursty arrivals at system resources, and lock synchronization.

This technical report presents the equations used in the new model. Section 2 gives a brief outline of the system and the parameters used to describe it. Section 3 discusses the differences between the equations for the homogeneous model [6] and the heterogeneous model, and it provides the equations for computing lock synchronization.

2 System

This system of interest in this paper is the system modeled in SimOS [4], a detailed simulator developed at Stanford against which we validate our extended model. The architecture, which is similar to that of the Stanford FLASH [2] and the SGI Origin [3], is a cache-coherent, sequentially consistent shared-memory multiprocessor system, as shown in Figure 1.

Table 1 defines the system architecture parameters. Table 2 lists the application parameters. Since the model assumes heterogeneous node behavior, each of the parameters in Table 2 is per-processor.

3 Equations

The principal output measure computed by the model is the system throughput, measured in instructions retired per cycle (IPC). This throughput is computed as a function of the input parameters that characterize the workload and the memory architecture.

*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.

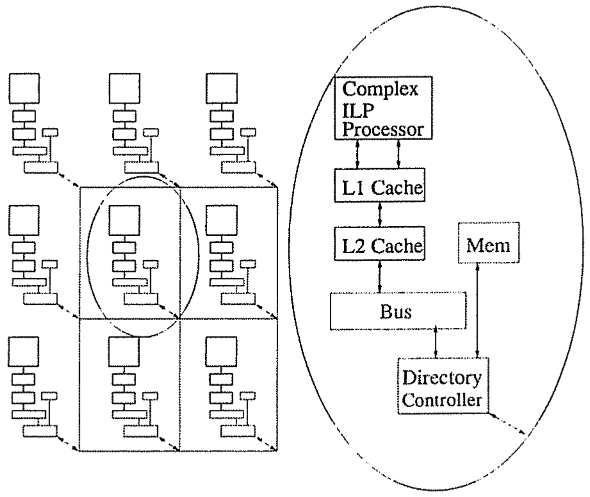

Figure 1. System Architecture

parameter	description
N	number of nodes
m	memory modules per node
M_{hw}	number of MSHRs
S_{bus}	bus latency
S_{dir}	directory latency
S_{mem}	memory/directory (DRAM) access
S_{net}	network latency

Table 1. System Architecture Parameters

3.1 Notation

In general, the heterogeneous equations have extra indices (in brackets) to indicate the node of the customer and/or the destination. Thus, a utilization term such as $U_{dir_{loc}}$ (the mean utilization of the local directory) becomes $U_{dir_{loc}}[i]$ (the mean utilization of the directory of node i by local customers) to reflect the fact that the utilization of the local directory is different for different nodes. The probabilities of the transaction types use an index in a similar fashion. For example, we now have the probability of transaction LC *at node* i, $P_{LC}[i]$. These probabilities are computed from the last six parameters in Table 2.

Since the heterogeneous model does not assume that a request that goes remote is equally likely to go to any of the other $N-1$ nodes, remote visit counts have two indices: one for the node of the customer and one for the node that the customer is visiting. For example, the term $V_{dir_{rem}}$ (visit count at a remote directory) becomes $V_{dir_{rem}}[i][j]$ (visit count of a customer of node i visiting the directory of node j). Visit counts in the homogeneous model were a function of N,m, and the transaction type. In the heterogeneous model, they depend on routing probabilities instead of N and m, such as the probability that a transaction of type y visits node j when it goes to a remote node. In the rest of this paper, we will treat the visit counts as inputs, bearing in mind that they depend on transaction types.

Similarly, the waiting time terms have two indices. The mean waiting time at a resource is equal to the sum of the mean waiting time caused by local traffic and the mean waiting time caused by remote traffic. In the homogeneous model, adding up the remote traffic could be done simply by multiplying the waiting caused by one remote node by $N-1$. In the heterogeneous model, the mean waiting due to remote traffic is computed as a sum. For example, to compute the mean waiting time of a node i customer at remote directories, we have

Parameter	Description
τ	Average time between read, write, or upgrade requests to memory, not counting the time when the processor is completely stalled or is spin-waiting on a synchronization event
CV_τ	Coefficient of Variation of τ
f_M	Fraction of processor stalls that occur with $M = 1, 2, ...$ outstanding requests in the MSHRs
$P_{read}, P_{write}, P_{upgrade}$	Probability that a memory request is a read, write, or upgrade
P_{wb}	Probability that a read or write request causes a writeback of a cache block
$P_{L\|x}$	Probability directory is local for a type x transaction; x=read, write, upgrade, write-back
$P_{M\|x,y}$	Probability home memory can supply the data for a type x, y request; x=read, write; y=local home, remote home
$P_{4hop\|x\&not-memory}$	Probability that a request of type x to a remote home is forwarded to a cache at a third node; x=read,write
X	Average number of invalidates caused by a write or upgrade to a clean line

Table 2. Application Parameters

$$W_{dir_{rem}}[i] = \sum_{\substack{j \\ j \neq i}} W_{dir_{rem}}[i][j]$$

where $W_{dir_{rem}}[i][j]$ is the mean waiting time by a customer of node i at the directory of node j.

3.2 Heterogeneous Model Equations

At the highest level, we compute the mean residence time, $R[i]$, of a given customer from node i. $R[i]$ consists of the synchronous residence times of a node i customer at all of the system resources, including the processor (pe), system bus, network, memory (mem), and directory (dir), as well as the time spent at miscellaneous resources at which there is no queueing (Z).

$$R[i] = R_{pe}^{synch}[i] + R_{bus}^{synch}[i] + R_{network}^{synch}[i] + R_{mem}^{synch}[i] + R_{dir}^{synch}[i] + Z[i]$$

Examining the directory portion of this equation highlights the differences in the equations between the heterogeneous and homogeneous models. Mean directory residence time is equal to the mean residence time at the local directory plus the mean residence time at the directories of the other nodes. The superscript of z differentiates between synchronous and asynchronous.

$$R_{dir}^{z}[i] = R_{dir_{loc}}^{z}[i] + \sum_{\substack{j \\ j \neq i}} R_{dir_{rem}}^{z}[i][j]$$

The local and remote directory mean residence times are the sums of the mean residence times over the different types of transactions, denoted by a subscript of y.

$$R^z_{dir_{loc}}[i] = \sum_y R^z_{dir_{loc,y}}[i]$$

$$R^z_{dir_{rem}}[i][j] = \sum_y R^z_{dir_{rem,y}}[i][j]$$

The mean residence times of individual transaction types are equal to the probability of the transaction type (P_y) times the visit count times the sum of the mean waiting time and the service time at the directory:

$$R^z_{dir_{loc,y}}[i] = P_y[i]V^z_{dir_{loc_y}}[i](W_{dir_{loc}}[i] + S_{dir})$$

$$R^z_{dir_{rem,y}}[i][j] = P_y[i]V^z_{dir_{rem_y}}[i][j](W_{dir_{rem}}[i][j] + S_{dir})$$

All of the terms in the above pair of equations are inputs except for the waiting times.

$$W_{dir_{loc}}[i] = W^{loc}_{dir_{loc}}[i] + W^{rem}_{dir_{loc}}[i]$$

$W^{loc}_{dir_{loc}}[i]$ is the mean waiting time of a node i customer at its directory due to local traffic (the superscript of loc denotes the cause of the waiting). $W^{rem}_{dir_{loc}}[i]$ is the mean waiting time of a node i customer at its directory due to traffic from remote nodes. Superscripts of S and A refer to synchronous and asynchronous transactions (or parts of transactions), respectively.

$$W^{loc}_{dir_{loc}}[i] = \sum_y \left(W^{loc,y^{synch}}_{dir_{loc}}[i] + W^{loc,y^{asynch}}_{dir_{loc}}[i]\right)$$

$$W^{rem}_{dir_{loc}}[i] = \sum_y \left(W^{rem,y^{synch}}_{dir_{loc}}[i] + W^{rem,y^{asynch}}_{dir_{loc}}[i]\right)$$

The equations for mean waiting time at remote directories are as follows:

$$W_{dir_{rem}}[i][j] = W^{others}_{dir_{rem}}[i][j] + W^{rem}_{dir_{rem}}[i][j]$$

$W^{others}_{dir_{rem}}[i][j]$ is the mean waiting time of a node i customer at the directory of node j due to traffic from all nodes other than node j. $W^{rem}_{dir_{rem}}[i][j]$ is the mean waiting time of a node i customer at the directory of node j due to traffic from node j.

$$W_{dir_{rem}}^{others}[i][j] = \sum_{y} \left(W_{dir_{rem}}^{others,y^S}[i][j] + W_{dir_{rem}}^{others,y^A}[i][j] \right)$$

$$W_{dir_{rem}}^{rem}[i][j] = \sum_{y} \left(W_{dir_{rem}}^{rem,y^S}[i][j] + W_{dir_{rem}}^{rem,y^A}[i][j] \right)$$

The following equations are for the mean waiting times of specific transaction types. Thus, $W_{dir_{loc}}^{loc,y^z}[i]$ is the mean waiting time at node i's directory due to local traffic for transactions of type y. Mean waiting time for a single other node i customer equals $\frac{R_{dir_{loc,y}}^{z}[i]}{R[i]} - U_{dir_{loc,y}}^{z}[i]$ (the probability that a customer is in the queue but not in service) times the service time, plus $U_{dir_{loc,y}}^{z}[i]$ (the probability that a customer is in service) times the mean residual life of a customer in service. Therefore, to get the total mean waiting time, we multiply by the number of other node i customers, $M[i]-1$.

$$W_{dir_{loc}}^{loc,y^z}[i] = (M[i]-1)\left[\left(\frac{R_{dir_{loc,y}}^{z}[i]}{R[i]} - U_{dir_{loc,y}}^{z}[i] \right) S_{dir} + U_{dir_{loc,y}}^{z}[i] \left(\frac{S_{dir}}{2} \right) \right]$$

$$W_{dir_{loc}}^{rem,y^z}[i] = \sum_{\substack{j \\ j \neq i}} M[j] \left[\left(\frac{R_{dir_{rem,y}}^{z}[j][i]}{R[j]} - U_{dir_{rem,y}}^{z}[j][i] \right) S_{dir} + U_{dir_{rem,y}}^{z}[j][i] \left(\frac{S_{dir}}{2} \right) \right]$$

The equation for $W_{dir_{rem}}^{others,y^z}[i][j]$ has two main terms, since it consists of waiting by a node i customer at node j due to other customers of node i, $W_{dir_{rem}}^{i,y^z}[i][j]$, and waiting due to customers that are of neither node i or node j, $W_{dir_{rem}}^{(k\neq i,j),y^z}[i][j]$. Notice that this latter term is not a function of i, except in that $k \neq i$.

$$W_{dir_{rem}}^{others,y^z}[i][j] = W_{dir_{rem}}^{i,y^z}[i][j] + \sum_{\substack{k \\ k\neq i,j}} W_{dir_{rem}}^{(k\neq i,j),y^z}[i][j]$$

$$W_{dir_{rem}}^{i,y^z}[i][j] = (M[i]-1)\left[\left(\frac{R_{dir_{rem,y}}^{z}[i][j]}{R[i]} - U_{dir_{rem,y}}^{z}[i][j] \right) S_{dir} + U_{dir_{rem,y}}^{z}[i][j] \left(\frac{S_{dir}}{2} \right) \right]$$

$$W_{dir_{rem}}^{(k\neq i,j),y^z}[i][j] = M[k]\left[\left(\frac{R_{dir_{rem,y}}^{z}[k][j]}{R[k]} - U_{dir_{rem,y}}^{z}[k][j] \right) S_{dir} + U_{dir_{rem,y}}^{z}[k][j] \left(\frac{S_{dir}}{2} \right) \right]$$

$$W_{dir_{rem}}^{rem,y^z}[i][j] = M[j]\left[\left(\frac{R_{dir_{loc,y}}^{z}[j]}{R[j]} - U_{dir_{loc,y}}^{z}[j] \right) S_{dir} + U_{dir_{loc,y}}^{z}[j] \left(\frac{S_{dir}}{2} \right) \right]$$

Lastly, we have the utilization equations. The first equation is the mean utilization of node i's directory by a local customer, and the second equation is the mean utilization of node j's directory by a customer of node i.

$$U^z_{dir_{loc},y}[i] = \frac{P_y[i]}{R[i]}(V^z_{dir_{loc},y}[i]S_{dir})$$

$$U^z_{dir_{rem},y}[i][j] = \frac{P_y[i]}{R[i]}(V^z_{dir_{rem},y}[i][j]S_{dir})$$

Modeling the other resources in the system is similar to what has been shown here for the directory.

3.3 Processor Modeling

The processor is modeled differently from the other resources, due to its burstiness. Processor mean residence time is computed as follows:

$$R_{pe}[i] = \tau[i](1 + Q_{pe}[i]) + U_{pe}[i]\tau_{residual}[i]$$

where Q_{pe} and U_{pe} refer to the mean queue length and utilization, respectively, and they are computed using standard MVA equations. This equation is the same as for the heterogeneous model, except that the mean residual life is computed using a new interpolation [1]. One intermediate term in this interpolation is the standard MVA equation for mena residual life:

$$\tau_{res-std}[i] = \frac{\tau[i]}{2} + \frac{VAR(\tau[i])}{2\tau[i]}$$

Another intermediate term we use is based upon a model of the processor as a two-stage hyperexponential server, shown in Figure 2, with service times $\tau_a[i]$ and $\tau_b[i]$. These service times, as well as the probabilities of going to the servers, are chosen such that the mean service time and the variance in the service time match $\tau[i]$ and $VAR(\tau)[i]$, respectively. The problem is underconstrained, since there are three variables and two constraints. The heuristic we use for a third constraint is to choose $\tau_a[i]$ equal to the smallest value of τ that is seen frequently. The intermediate term $T[i]$ is then computed as follows:

$$T[i] = \frac{\tau_a[i]\tau_b[i]}{\tau[i]}$$

Using these terms, we have the following interpolation for mean residual life:

$$\tau_{residual} = \frac{\tau[i]T[i]}{T[i] + R[i] - R_{pe}[i]} + \frac{(R[i] - R_{pe}[i])\tau_{res-std}[i]}{T[i] + R[i] - R_{pe}[i]}$$

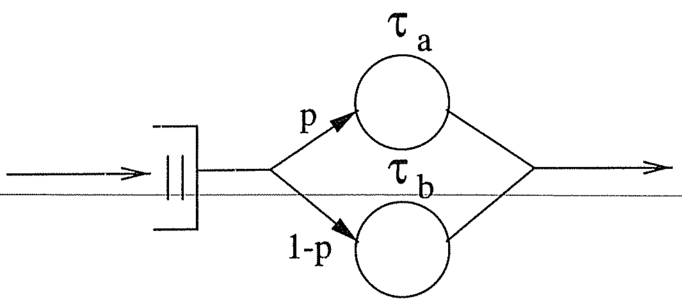

Figure 2. Hyperexponential Model of Processor

3.4 Modeling Downstream Burstiness

As explained more thoroughly in previous work [1], we model a bursty arrival process at a downstream resource (e.g, a directory) 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. A bursty process is parameterized with three parameters: K, I, and L. K is the mean number of customer arrivals within a burst, I is the mean interarrival time within a burst, and L is the mean time between bursts.

Since we assume that downstream burstiness is only caused by traffic from the local processor, the burstiness model only changes the mean waiting time equations, $W^{loc}_{dir_{loc}}[i]$ and $W^{rem}_{dir_{rem}}[i][j]$. Without loss of generality, we will only show the equations for $W^{loc}_{dir_{loc}}[i]$. First, we compute the mean queue length.

$$Q^{loc}_{dir_{loc}}[i] = \sum_{y} Q^{loc,y}_{dir_{loc}}[i]$$

$$Q^{loc,y}_{dir_{loc}}[i] = (M[i] - 1)\frac{R^{z}_{dir_{loc,y}}[i]}{R[i]}$$

Then we compute the mean queue length during the time between bursts (NB = not bursty).

$$QNB^{loc,y}_{dir_{loc}}[i] = Q^{loc}_{dir_{loc}}[i] - I[i](K[i]-1)K[i]\left(S_{dir} - \frac{I[i]}{S_{dir}(I[i](K[i]-1)+L[i])}\right)$$

The mean waiting time is equal to:

$$W^{loc}_{dir_{loc}}[i] = S_{dir}QNB^{loc,y}_{dir_{loc}}[i] + (K[i]-1)(S_{dir} - I[i])$$

Once again, we have an underconstrained problem with three variables and two constraints (discussed later). The heuristic used here is to choose $I[i]$ equal $\tau_a[i]$ (where $\tau_a[i] < \tau_b[i]$). Now we need to compute $K[i]$ and $L[i]$ based on the coefficient of variation of the departure process, $CV_d[i]^2$, and the mean throughput, $\Lambda[i]$.

$$CV_d[i]^2 = 1 + M[i]^2 U_{pe}[i]^2 (CV_r[i]^2 - 1)$$

$$\Lambda[i] = \frac{M[i]}{R[i]}$$

The equations for computing $K[i]$ and $L[i]$ were derived by solving the following two constraint equations. Note that $CV_a[i]$ is the coefficient of variation of interarrival times at node i's directory.

$$\frac{K[i]}{(K[i]-1)I[i]+L[i]} = \Lambda[i]$$

$$\frac{\frac{K[i]-1}{K[i]}2I[i]^2 + \frac{1}{K[i]}2L[i]^2}{\left(\frac{(K[i]-1)I[i]+L[i]}{K[i]}\right)^2} - 1 = CV_a[i]^2$$

The following equation computes $CV_a[i]^2$ from $CV_\tau[i]$, according to an approximation by Sevcik and Levy [5]:

$$CV_a[i]^2 = 1 + U_{pe}[i]^2(CV_\tau[i]^2 - 1)$$

Now we have the equations for $K[i]$ and $L[i]$.

$$K[i] = \frac{1}{2}\frac{-4\Lambda[i]I[i] + 1 + \Lambda[i]^2 2I[i]^2 + CV_d[i]^2}{1 + \Lambda[i]^2 I[i]^2 - 2\Lambda[i]I[i]}$$

$$L[i] = \frac{1}{2}\frac{2\Lambda[i]I[i] - 1 - CV_d[i]^2}{\Lambda[i](\Lambda[i]I[i] - 1)}$$

In practice, solving for $K[i]$ and $L[i]$ requires that we check to make sure that the values are self-consistent. For example, if $K[i]$ is less than 1, the burstiness model is unnecessary and will produce invalid results.

3.5 Modeling Lock Synchronization

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.

Like the method of complementary service time inflation used in [7] 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.

The lock contention model is illustrated in Figure 3. 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

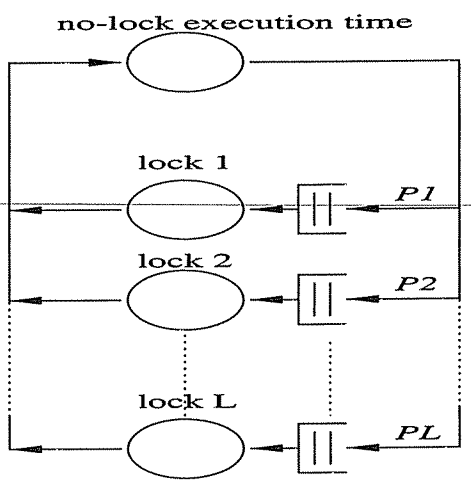

Figure 3. Lock Contention Model

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.

References

[1] D. Eager, D. Sorin, and M. Vernon. Analytic Modeling of Burstiness and Synchronization Using Approximate MVA. Technical Report 1391, Computer Sciences Dept., Univ. of Wisconsin - Madison, Dec. 1998.
[2] J. Kuskin et al. The Stanford FLASH Multiprocessor. In *Proc. 21st Int'l Symp. on Computer Architecture*, Apr. 1994.
[3] 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.
[4] M. Rosenblum, S. Herrod, E. Witchel, and A. Gupta. Complete Computer Simulation: The SimOS Approach. *IEEE Parallel and Distributed Technology*, 1995.
[5] K. Sevcik, A. Levy, S. Tripathi, and J. Zahorjan. Improving Approximations of Aggregated Queuing Network Subsystems. *Computer Performance*, 1977.

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

[7] 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.