PERFORMANCE EVALUATION OF SPIN LOCKS IN MULTI-PROCESSORS

Subir Varma

VTAM Performance
Research Triangle Park, N.C.

SECTION 1. INTRODUCTION

Consider a multi-processor system with K processors. Assuming that the system is tightly coupled so that the operating system maintains a single dispatching queue for all the processors, the system can be modeled by a G/G/K queue. Assuming that all the tasks that arrive into the system can be dispatched in parallel, there are two sources of performance degradation which affect the system:

- If the tasks are subject to precedence constraints of the fork-join type, then they undergo some additional synchronization delay.
- If the tasks need to access some common data structures while executing, they have to hold a lock before they can do so. Hence contention for the locks can lead to additional delays.

Both these effects can make the analysis of the queuing model extremely difficult, since it no longer satisfies product form assumptions. In the past few years, researchers (see [NeToTa] and [VaMa]) have obtained a number of results regarding systems with synchronization constraints. However, they ignore the second effect in their studies, i.e., they assume that the tasks can execute independently of one another. The principal aim of this work is to attack the second problem,i.e., obtain approximations for queueing models in which the contention for lock delay is taken into account. The main source of difficulty in solving this system is the fact that the executing jobs are no longer independent, and hence may influence each other.

There are two kinds of locks that are used in computer systems, i.e., spin locks and suspend locks. When a task is unable to obtain a spin lock, it does not release the processor, but keeps checking for the availability of the lock at regular intervals until the lock is is available. When a task is unable to obtain a suspend lock, it frees up its processor and has to be redispatched. In the present work we deal with spin locks. In multi-programmed uni-processor systems all locks were of the suspend type for obvious reasons. However as the number of processors in the system increases, spin locks are becoming more popular since thay do not incur the overhead of re-dispatching a suspended task. Some other references that deal with the locking problem are [AgTr] , [AgBu] , [HoSc] , [JaLa] , [SiMu] and [Th]

The rest of this report is organized as follows: In Section 2 we describe the system under consideration in greater detail, and provide an algorithm for the performance evaluation of spin locks. This algorithm is applied to a sample system in Section 3, and its predictions are compared with simulations. Section 4 consists of conclusions and suggestions for further research.

SECTION 2. THE SPIN LOCK ALGORITHM

As mentioned in the first section, we shall assume that the multiprocessing system can be modeled by a multi-server queue with M servers (see Figure 1).

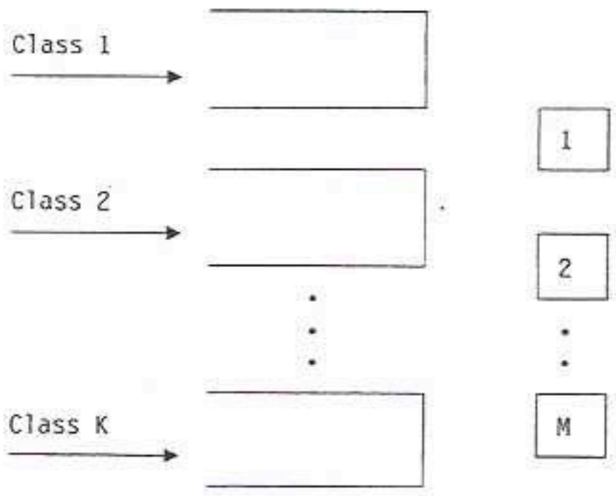

Figure 1. The hardware model

Assume that there are K classes of tasks arriving into the system, such that class k arrives according to a Poisson process with rate λ_k . We assume that all the tasks are served in FCFS order, i.e., no priorities are involved. Furthermore assume that the number of instructions to be executed by a task from the k^{th} class is L_k. If there is no contention for the locks, the service time of the k^{th} class is assumed to be exponentially distributed with rate $\frac{L_k}{R}$, where R is the MIPS rate of the processors.

Assume that there are N different types of spin locks in the system. The instruction path of each class of tasks can be divided into a number of sections such that in each section either the j^{th} spin lock is required, or no lock is required. We shall assume that the instruction path of the k^{th} class of tasks can be divided into $l(k)$ sections, and we associate a number N_i^k with the i^{th} section of the k^{th} task which indicates the type of lock held while executing that section. The case $N_i^k = 0$ indicates that no lock is held, while the cases $N_i^k = j, j = 1 \ldots N$ indicates that the j^{th} lock is held during execution.

We now introduce a simple example in order to facilitate the discussion. The system under consideration has two processors and there are two classes of customers which undergo processing. Furthermore there are two kinds of spin locks in the system. Class 1 tasks invoke both type 1 as well as type 2 locks during their processing, while class 2 tasks invoke only type 2 locks. The instruction paths of these two classes (see Figure 2 on page 3) are divided into a number of sections depending upon the type of lock held.

1000	150000	1000	50000	100
N=0	N=1	N=0	N=2	N=0

Class 1, l(1)=5

40000	50000	40000
N=0	N=2	N=0

Class 2, l(2)=3

Figure 2. Example of an instruction path

Let us first consider the simple case when there are an infinite number of processors available in the system. Then there is no delay due to hardware contention, and the only delay that exists is due to software contention for the spin locks. This case can be modeled by the queueing network shown in Figure 3 on page 4.

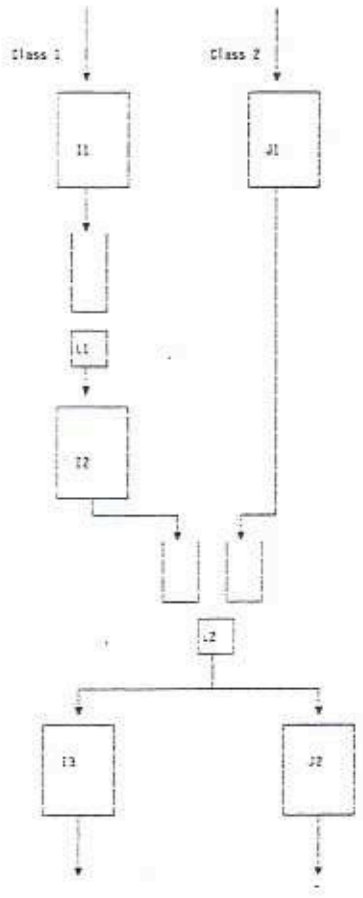

Figure 3. The case of an infinite number of processors

The case where there is no lock contention is modeled by infinite server queues I_1, I_2, I_3 and J_1, J_2. The case where there is contention for locks is modeled by single server queues L_1, L_2. Since only Class 1 tasks contend for lock 1, queue L_1 has a single input class, however since both Class 1 and Class 2 tasks contend for lock 2, queue L_2 has two input classes. The service times at each of the queues are given by execution path length of that segment (obtained from Figure 2 on page 3) divided by the MIPS rate of the processors.

The model presented above ignored the delay due to hardware contention at the processors. In order to take this effect into account, we now present a two level model in which the software contention is modeled by a closed queueing network, while the hardware contention is modeled by a multi-class multi-server queue. For the specific system under consideration, this two level model is depicted in Figure 4 on page 5 and Figure 5 on page 5.

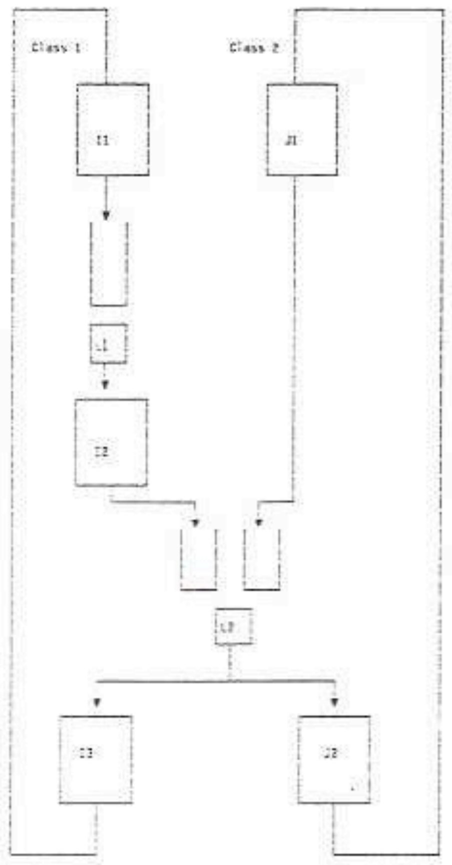

Figure 4. The software level model

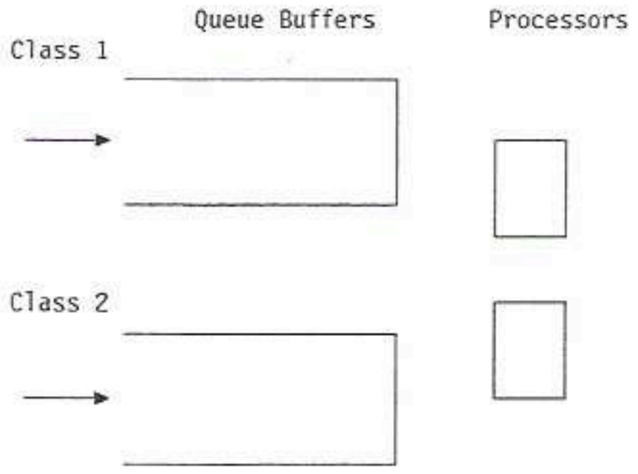

Figure 5. Hardware level model

Our basic objective is to obtain response time estimates for both classes of customers. The response time can be expressed as the sum of the waiting time in queue buffer of Fig. 5 (due to hardware contention), and the service time in the processor. However the service time is not fixed as in classical queueing systems, but depends on the level of software contention due to the presence of the spin locks. This software contention is captured by the closed queueing network in Fig. 4. Hence we propose the following two level scheme for obtaining the response time estimates.

1. Obtain estimates for the effective service times for both classes, with the help of the software level model. For example, the effective service time for a class 1 task will be the sum of the service and waiting times in chain 1 of Fig. 2, which will depend on the population of chains 1 and 2. The population m_1 of chain 1 is the number of class 1 tasks currently being executed, while the population m_2 of chain 2 is the number of class 2 tasks being executed. Since there are only two servers, the vector (m_1, m_2) can only assume the values (1,0),(0,1),(2,0),(1,1),(0,2). For each of these chain populations we obtain estimates of the chain response time by using Mean Value Analysis (MVA) [ReLa] In order to find the effective service time, we multiply the response times for each chain population by the probability of that chain population, and then sum up the numbers. The exact formula for doing so is equation (6) in the next sub-section.
2. Use these estimates of the service times to obtain estimates for the waiting times at the hardware level, with the help of multi-server queueing theory formulae.

The detailed steps required to carry out this program are now outlined for the case of a general system with M servers and K customer classes. There will be K closed chains in the software level model for this system. As illustrated in the example, segments of code that do not require locks are modeled by infinite server queues, while segments of code that require a lock are modeled by a single server queue. Each such single server queue has as many input streams as the number of times that the corresponding lock is invoked by the tasks. We now present the general algorithm for obtaining the response time.

Spin Lock Algorithm

1. The first step is to obtain an estimate for effective service rate after taking the lock contention into account. For this purpose, set value of variable $m = 1$.
2. Choose all values of $m_1 \ldots m_K$ such that

   $$m_1 + \cdots + m_K = m \qquad (1)$$

   Let m_k be population of the closed chain k.
3. Using the MVA algorithm [ReLa] , solve the system of K coupled chains and obtain the response time of each chain, given by $x_k(m_1, \ldots, m_K, m)$ for all values of $m_1 \ldots m_k$ satisfying (1).
4. If the value of m equals M, then go to the next step. Otherwise increase the value of m by one, and go back to step 2.
5. Set X_k equal to the average service time of the k^{th} class, assuming that there is no lock contention, i.e., $X_k = \frac{L_k}{R}$.
6. Let $p_i, i = 1, \ldots, M-1$ be the probability that i servers are busy in a M-server queue, with arrival rate λ given by

   $$\lambda = \lambda_1 + \cdots + \lambda_K \qquad (2)$$

   and average service time X given by

   $$X = \sum_{k=1}^{K} \frac{\lambda_k}{\lambda} X_k \qquad (3)$$

The relevant formulae for $i = 1, \ldots, M-1$ are given by

$$p_i = \frac{(\lambda X)^i}{i!} p_0$$

where

$$p_0^{-1} = \frac{(\lambda X)^M}{M!(1-\rho)} + \sum_{k=0}^{M-1} \frac{(\lambda X)^k}{k!}$$

and ρ is the utilization for this queue given by

$$\rho = \frac{\lambda X}{M} \tag{4}$$

Also define p_M to be the probability that there are M or more tasks in the queue. This is given by the following formula.

$$p_M = \frac{p_0 (M\rho)^M}{M!(1-\rho)} \tag{5}$$

7. Define the variables ρ_k by

$$\rho_K = \lambda_K X_K$$

and the variables $q_1 \ldots q_K$ by

$$q_K = \frac{\rho_K}{\sum_{j=1}^{K} \rho_j}$$

The adjusted value of the service time $\hat{X}_k$, for a customer belonging to the k^{th} class, after taking lock contention into account is given by

$$\hat{X}_k = \sum_{m=1}^{M} \sum_{m_1} \cdots \sum_{m_K} \frac{\begin{bmatrix} m \\ m_1 \ldots m_K \end{bmatrix}}{\sum_{m=1}^{M} \sum_{m_1} \cdots \sum_{m_K} \begin{bmatrix} m \\ m_1 \ldots m_K \end{bmatrix} q_1^{m_1} \ldots q_K^{m_K} p_m} q_1^{m_1} \ldots q_K^{m_K} p_m x_K(m_1 \ldots m_K, m) \tag{6}$$

The summations over m_1 to m_k in the numerator and denominator are over the range $m_1 + \cdots + m_K = m,\ m_k \geq 1$.

8. For each k, compare the values of $\hat{X}_k$ and X_k. If there is no significant difference then put $X_k = \hat{X}_k$ and go to next step. Otherwise replace X_k by $\hat{X}_k$ and go back to step 6.
9. With λ defined as in (2), X defined as in (3) and ρ defined as in (4), (using new values for X_k in calculating X and ρ), obtain the avarage waiting time W in the multi-server queue with the formula

$$W = \frac{(\lambda X)^M}{M!} \frac{p_0 X}{M(1-\rho)^2}$$

Note that the average waiting time is the same for all classes. The average response time of the k^{th} class is given by the formula

$$R_k = W + X_k$$

Equation (6) has been obtained with the help of the following heuristic: Recall that $x_k(m_1 \dots m_k, m)$ is response time of the k^{th} chain, given that the population of the i^{th} chain is given by m_i and (1) is satisfied. In terms of the hardware model this corresponds to the case when a total of m servers (out of M) are busy, and m_i of those are serving customers belonging to class i. The probability that m servers are busy is given by p_m, and we assume that the probability that m_i servers are serving customers of type i is given by the following multinomial distribution:

$$p(m_1 \dots m_K, m) = \begin{bmatrix} m \\ m_1 \dots m_K \end{bmatrix} q_1^{m_1} \dots q_K^{m_K}$$

Finally taking into account the fact that only those chains having one or more customers of type k contribute to the effective service time of class k, we arrive at (6).

SECTION 3. COMPARISON WITH SIMULATION

In this section we compare the predictions of the algorithm with simulation results. The sample system with two classes and two servers is the one that was chosen for validation. Table 1 gives the response times of the two classes for the case when no locks are used, Table 2 gives the response time with locks obtained by using the algorithm, while Table 3 gives the response time obtained by simulation. The reader may note that the difference in percentage degradation predicted differs from the simulated value by less than 3% in light traffic to about 8% in heavy traffic.

Table 1: Resp. Time w/o locks

Arrival Rate	Class 1 Resp.	Class 2 Resp.
50	6.47	5.55
100	6.89	5.97
150	7.56	6.84
200	9.44	8.52
250	13.23	12.31

Table 2: Simulated Resp. Time			Calculated Response Time			
Arrival Rate	Class 1 Resp.	Class 2 Resp.	Class 1 Resp.	% Diff.	Class 2 Resp.	% Diff.
50	7.09	5.79	6.86	3.24	5.72	1.21
100	8.22	6.59	7.80	5.11	6.45	2.12
150	10.27	8.39	9.56	6.91	8.01	4.53
200	15.00	12.80	13.74	8.4	11.98	6.41

We now use the validated model to predict the performance degradation when there are six processors instead of two. These numbers are provided in Table 3.

Table 3: W/O Locks			With Locks	
Arrival Rate	Class 1 Resp.	Class 2 Resp.	Class 1 Resp.	Class 2 Resp.
250	6.34	5.42	9.66	6.74
300	6.36	5.44	11.20	7.37
350	6.38	5.46	14.04	8.74
400	6.43	5.51	25.02	17.00

SECTION 4. CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

The results presented here can be extended in a number of directions some of which are outlined here.

- One of the restrictions in using the MVA algorithm in Step 3 is that it restricts all instruction paths under a single lock to be of the same length. This restriction can be removed by using an approximate form of the MVA given by Reiser [Re]
- If any one of the modules executed under a lock has more than one task which can execute it, then it can be modeled by a multiple server queue in the software level model. The approximate MVA algorithm by De Silva and Muntz [SiMu] is capable of handling multi-server queues.
- An extension of the ideas presented here can be used to model suspend locks. In these systems, the tasks that are suspended can be represented by another class of arrivals into system. Research is presently underway to solve this system.
- The case when processor is subject to arrivals from multiple classes with differing priorities can be analyzed with the help of the approximations in the paper by Buzen and Bondi [BuBo]. However all the classes which hold locks are restricted to be in the same priority class.

Acknowledgement The author would like to thank Mr. Srinivas Bharadwaj for developing a program to implement the spin lock algorithm.

4.1 REFERENCES

[AgrTr] J.R. Agre and S.K. Tripathi, "Modeling reeentrant and non-reentrant software". ***Performance Evaluation Review 11,*** 163-178, (Aug-Sept 1982).

[AgBu] S.C.Agrawal and J.P. Buzen, "The aggregate server method for analyzing serialization delays in computer systems", ***ACM Trans. Comput. Systems CS-1,*** 116-143, (1983).

[BuBo] J.P.Buzen and A.B. Bondi, "The response times of priority classes under preemptive resume in M/M/m queues", ***Operations Research Vol 31,*** No. 3, 456-465, (May-June 1983).

[HoSc] J.Hoffman and H. Schmutz, "Performance analysis of suspend locks in operating systems", ***IBM J. Res. Dev. 26,*** 242-259, (March 1982).

[JaLa] P.A. Jacobson and E.D. Lazowska, "A redusction technique for evaluating queueing networks with serialization delays", ***PERFORMANCE'83,*** 45-59, (1983).

[NeToTa] R. Nelson, D. Towsley and A.N. Tantawi, "Performance Analysis of parallel processing systems", ***IEEE Trans. Software Eng. 14,*** 4, 532-540, (April 1988).

[Re] M. Reiser, "A queueing network analysis of computer communication networks with window flow control", ***IEEE Trans. Commun. 27,*** 1109-1209, (1979).

[ReLa] M. Reiser and S.S. Lavenberg, "Mean value analysis of closed multi-chain queueing networks", ***J.ACM 27,*** 313-322, (1980).

[SiMu] E. de Souza e Silva and R.R. Muntz, "Approximate solutions for a class of non-product form queueing network models", ***Performance Evaluation 7,*** 221-242, (1987).

[Th] A. Thomasian, "Queueing network models to estimate serialization delays in computer systems", ***Performance'83,*** 61-81, (1983).

[VaMa] S. Varma and A.M. Makowski, "Interpolation approximations for fork-join queues", To be submitted to ***Operations Research.***