MODELING SPIN LOCKS WITH QUEUING NETWORKS

D. C. Gilbert
International Business Machines
Poughkeepsie, New York

I. Introduction

Modern multiprogramming, multiprocessing operating systems encounter a common problem: multiple parallel processes (perhaps being executed on multiple processors) share common data and control blocks, and must use some form of process synchronization to ensure data integrity. This problem has been addressed in many places in the literature (for instance, in Dijkstra (1965) and Dijkstra (1968)) and a clear distinction has been drawn between synchronization primitives which employ busy versus non-busy waiting (Dijkstra (1971)). Though the non-busy wait method (used by Dijkstra's semaphores, operated upon by P and V) has received wide acclaim, there is a place for the busy wait. King, et al (1974) analyze the effects of busy and non-busy wait strategies for the idealized case where there is no overhead for the process switch associated with the non-busy wait, and find that the idealized non-busy wait strategy should be preferred. Fuller, et al (1976) compare the performance of processes using busy versus non-busy waiting running on an actual multiprocessing system (C.mmp). There are criteria, however, for legitimately using the busy wait. When the projected wait time is less than the time to perform a process switch, or when the processor cannot run any other work due to the critical nature of the current work, (dispatching, virtual storage management, etc.) then a busy wait can be used. In the MVS operating system, described in Arnold, et al (1974), both types of process synchronization primitives are used: "disabled spin locks" (busy wait) and "enabled suspend locks" (non-busy wait). Analysis of MVS disabled spin locks is the subject of this paper.

System performance is greatly affected by spin locks, since in a multiprocessing (MP) environment some time is wasted waiting on other processors to finish using shared data. What is needed is a way to relate lock behavior to system performance, so that proposed lock structure changes or proposed system configuration changes (such as additional processors) may be evaluated before being implemented. Since spin locks are in fact First-Come-First-Served (FCFS) queues, queuing theory may be applied to their analysis. And since in MVS processors contend for a number of spin locks, it is possible to construct a queueing network of spin locks in which processors circulate. This sort of model satisfies the needs mentioned above: the lock structure and the number of processors can be changed at will to evaluate the effect of such changes on the system.

II. Model Specification

MVS spin locks are described in Arnold, et al (1974) and work as follows. A given lock synchronizes the use of selected data areas, by convention. There are two operations which may be performed on a lock: obtain and release. If no processor currently holds a lock for which obtain is requested, then the lock is marked held and is given to that processor. If another processor already holds the lock, then the requesting processor spins (executes a busy wait) until the processor which holds the lock releases it. More than one lock may be held by a processor simultaneously ("nested" locks), but the hold time of the nested locks is not generally very long.

A queuing network model of spin locks is shown in Figure 1. Processors circulate from a "no-lock" state (representing the system state in which a processor is executing, but holding no lock) to a state where processor requests, potentially waits for, then holds one of the set of locks. Nesting of locks is not represented in this model, but the omission is not thought to be serious. The lock queues are FCFS, single server, and the no-lock queue has an infinite number of servers so that no processor ever waits going into the no-lock state (this is just a convenience; the number of servers at the no-lock state must be at least equal to the number of processors in the network). Each queue has independent identically (exponentially) distributed service times. The routing probabilities and the mean service times for the various queues are fixed and determined from measured data. This model is then of the type of queuing network proposed by Jackson (1963) and Gordon and Newell (1967), and is therefore solvable analytically. Efficient computational schemes for solving such a model are included in the QNET4 package (Reiser (1976)).

Measured data from an existing system is necessary to drive the model. This data required is as follows.

- number of processors
- total elapsed run time
- average processor utilization
- number of lock requests for lock_i
- total hold time for lock_i
- total spin time for lock_i

The parameters necessary to specify the model are then derived as follows:

A. P_i, $i = 2, j + 1$: the probability of going to a given lock queue when leaving the no-lock state

$$P_i = \frac{\text{number of lock requests for lock}_i}{\text{total lock requests}}$$

B. $1/\mu_i$, $i = 2, j + 1$: the mean service time for each lock_i; i.e., the lock hold time

$$\frac{1}{\mu_i} = \frac{\text{total hold time for lock}_i}{\text{number of lock requests for lock}_i}$$

C. $1/\mu_1$: the mean service time for the no-lock state

let N = number of circulating processors (the number of processors in the measured system)

L_1 = mean number of processors in the no-lock state

L_2 = mean number of processors not in the no-lock state

then $$L_1 + L_2 = N \qquad (1)$$

Let λ = rate of processors moving between the no-lock and lock states

$$= \frac{\text{total lock requests}}{\text{average processor busy time per processor}}$$

R_1 = queuing time for no-lock state = service time (since there are as many servers as processors)

R_2 = queuing time for the locks taken as a whole then by Little's Law (Allen (1975)),

$$L_1 = \lambda R_1 \tag{2}$$

$$L_2 = \lambda R_2 \tag{3}$$

from (2),

$$R_1 = \frac{L_1}{\lambda} \tag{4}$$

and from (1) and (3),

$$R_1 = \frac{N - \lambda R_2}{\lambda} \tag{5}$$

But if w_i = mean wait time for $lock_i$

$$= \frac{\text{total spin time for } lock_i}{\text{number of lock requests for } lock_i}$$

$h_i = \frac{1}{\mu_i}$ = mean hold (service) time for $lock_i$

P_i = probability of going from the no-lock state to $lock_i$

then

$$R_2 = \sum_{i=2}^{j+1} P_i(w_i + h_i) \tag{6}$$

Substituting (6) into (5) and dividing by λ, we get

$$\frac{1}{\mu_1} = \frac{N}{\lambda} - \sum_{i=2}^{j+1} P_i\ (w_i + h_i) \tag{7}$$

(7) finally expresses $1/\mu_1$ in terms of measured data values, and may be interpreted as follows. The service time for the no-lock state $(1/\mu_1)$ is found by subtracting from the mean time for a processor to complete one trip around the network $(\frac{N}{\lambda})$ the mean time spent in the lock part of the network $\sum_{i=2}^{j+1} P_i(W_i + h_i)$

Note that if N=1 in the measured system, then $w_i=0$, so the spin time for each lock need not be measured.

III. Model Usage

A Measured Performance Parameters

There are two parameters of interest which may be obtained from the measured system to characterize its lock behavior and to use to validate the model. The first is the lock utilization, which is the total time a lock is held divided by the average processor busy time per processor. Note that all times in the model are expressed in terms of processor busy time, for consistency, because spin locks are only held when a processor is actually executing instructions.

The second measured parameter relates to lock behavior to system performance. The system "spin degradation" is the percentage of processor busy time spent spinning for locks. More precisely,

$$\text{spin deg. for lock}_i = \frac{\text{total wait time for lock}_i}{\text{total system processor busy time}}$$

$$\text{spin deg. total} = \frac{\text{total wait time for all locks}}{\text{total system processor busy time}}$$

Note that spin degradation is not present when N = 1.

B. Modeled Performance Parameters

The QNET4 package supplies the lock utilizations directly, and these numbers may be checked against the measured utilizations for model validation. This check indicates how well the exponential service time distributions and no-nesting assumptions model the actual system.

Also supplied by QNET4 are the mean queue time (qt) for each lock, the mean time from arrival at the lock queue until service is completed, and the throughput for the no-lock state (λ), the rate of processors moving between the no-lock and lock states.

let qt_i = mean request rate for lock_i

$\frac{1}{u_i}$ = mean service time for lock_i

Wq_i = mean wait time in the waiting line for $lock_i$

$$\text{then} \quad Wq_i = qt_i - \frac{1}{\mu_i} \tag{8}$$

Now the spin degradation for $lock_i$ may be calculated as the mean wait time for $lock_i$ compared to the mean time around the closed loop $\frac{N}{\lambda}$:

$$\text{spin deg. for } lock_i = \frac{(P_i)\ (Wq_i)}{\frac{N}{\lambda}} \tag{9}$$

$$\text{and spin deg. total} = \sum_{i=2}^{j+1} \frac{(P_i)\ (Wq_i)}{\frac{N}{\lambda}} \tag{10}$$

The above parameters may be calculated for the model corresponding to the measured system, for validation, or for models corresponding to systems with changed lock structures or number of processors, for evaluation.

IV. Examples of Model Use

As an example of use of the spin lock model, suppose that there is a system with five spin locks, and that a run of this system yields the measured parameters shown in Figure 2, for N = 1 (one processor).

The model for this system is shown in Figure 3, with all parameters specified. The results of solving the model with N varying from 1 to 8 are shown in Figure 4. It can be seen that lock 5 is the highest contention lock, and that it accounts for most of the spin degradation.

To improve the MP performance of this hypothetical system, lock 5 may be split into more locks (if the data it synchronizes will permit). Figures 5 and 6 show the effect of splitting lock 5 into 3 and 6 locks, respectively, each having the same 1.0 msec hold time as the original lock 5. A dramatic decrease in the overall spin degradation can be seen when this breakup is performed. It should be noted, however, that such a breakup may cause additional lock requests to be added to the system, and that this new overhead subtracts from the spin degradation savings.

Other changes to the system may also be modeled, such as changing the hold times under various locks (by decreasing pathlengths in the actual system).

V. Conclusion

It has been shown that an analytic queuing network model may be constructed to represent spin lock behavior. This model may be used to predict changes in lock degradation effects as the system lock structure changes, or as the number of processors is varied.

REFERENCES:

Allen, A. O. (1975), "Elements of Queuing Theory for System Design," IBM Systems Journal 14, 2, pp. 161-187.

Arnold, J. S., Casey, D. P., and McKinstry, R. H. (1974), "Design of Tightly-coupled Multiprocessing Programming," IBM Systems Journal 13, 1, pp. 60-87.

Dijkstra, E. W. (1965), "Solution to a Problem in Concurrent Programming Control," CACM 8, 9 (Sept.), p. 569.

Dijkstra, E. W. (1968), "Cooperating Sequential Processes," in Programming Languages, F. Genuys, Ed., Academic Press, New York, pp. 43-112.

Dijkstra, E. W. (1971), "Hierarchical Ordering of Sequential Processes," Acta Informatica 1, 2, pp. 115-138.

Fuller, S. H., and Oleinick, P. N. (1976), "Initial Measurements of Parallel Programs on a Multi-Mini-Processor," Compcon 1976 (Fall), pp. 358-363.

Gordon, W. T., and Newell, G. F. (1967), "Closed Queuing Systems with Exponential Servers," Operations Research 15, 2 (April) pp. 254-265.

Jackson, J. R. (1963), "Jobshop-like Queuing Systems," Management Science 10, 1 (October), pp. 131-142.

King, W. F. III, Smith, S. E., and Wladawsky, I. (1974), "Effects of Serial Programs in Multiprocessing Systems," IBM Journal of Research and Development 18, 4 (July), pp. 303-309.

Reiser, M. (1976), "Interactive Modeling of Computer Systems," IBM Systems Journal 15, 4, pp. 309-327.

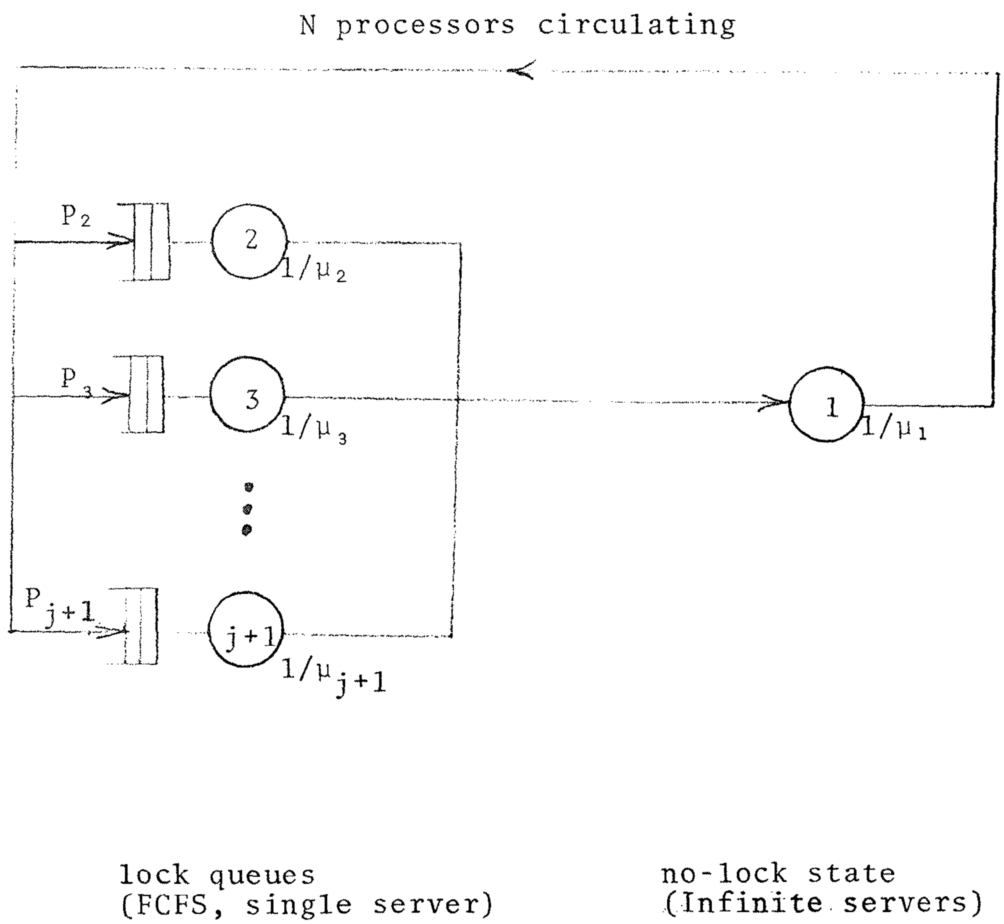

Figure 1: Spin Lock Queuing Model

Elapsed time = 100 sec

Processor Utilization = .90

	LOCKS				
	1	2	3	4	5
Total requests	625	1250	2500	5000	10000
Total hold time, sec	.04	.16	.63	2.5	10.0
Mean hold time, msec	.0625	.125	.25	.5	1.0
Utilization, %	.04	.18	.70	2.78	11.11

Figure 2: Hypothetical Measured Parameters

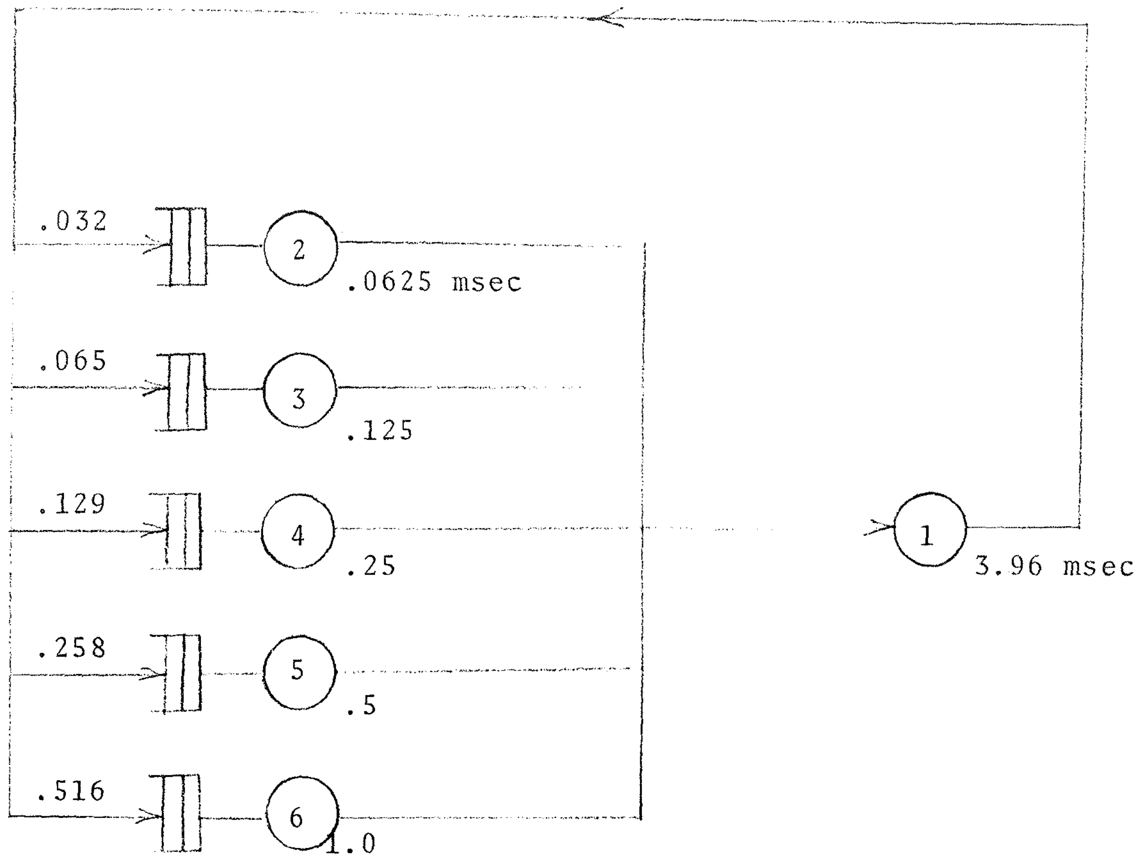

Figure 3: Model of Hypothetical System

	1	2	3	4	5	TOTAL
UTILIZATION, %						
N = 1	.0	.2	.7	2.8	11.1	
N = 2	.1	.4	1.4	5.5	21.9	
N = 3	.1	.5	2.0	8.1	32.4	
N = 4	.2	.7	2.7	10.6	42.4	
N = 5	.2	.8	3.3	13.0	51.9	
N = 6	.2	1.0	3.8	15.2	60.8	
N = 7	.3	1.1	4.3	17.2	68.9	
N = 8	.3	1.2	4.8	19.0	76.1	
SPIN DEG, %						
N = 1	.0	.0	.0	.0	.0	.0
N = 2	.0	.0	.0	.1	1.2	1.3
N = 3	.0	.0	.0	.2	2.6	2.8
N = 4	.0	.0	.0	.2	4.3	4.5
N = 5	.0	.0	.0	.3	6.2	6.5
N = 6	.0	.0	.0	.4	8.4	8.8
N = 7	.0	.0	.0	.4	11.0	11.4
N = 8	.0	.0	.0	.5	13.9	14.4

Figure 4: Results of Model

	1	2	3	4	5-7	TOTAL
UTILIZATION, %						
N = 1	.0	.2	.7	2.8	3.7	
N = 2	.1	.4	1.4	5.5	7.4	
N = 3	.1	.5	2.1	8.3	11.0	
N = 4	.2	.7	2.8	10.9	14.6	
N = 5	.2	.9	3.4	13.6	18.1	
N = 6	.3	1.0	4.1	16.2	21.6	
N = 7	.3	1.2	4.7	18.8	25.0	
N = 8	.3	1.4	5.4	21.3	28.4	
SPIN DEG, %						
N = 1	.0	.0	.0	.0	.0	.0
N = 2	.0	.0	.0	.1	.1	.5
N = 3	.0	.0	.0	.2	.3	1.0
N = 4	.0	.0	.0	.2	.4	1.6
N = 5	.0	.0	.0	.3	.6	2.1
N = 6	.0	.0	.0	.4	.8	2.7
N = 7	.0	.0	.0	.5	.9	3.3
N = 8	.0	.0	.0	.6	1.1	4.0

Figure 5: Results of Model with Lock 5 Split 3 Ways

	1	2	3	4	5-10	TOTAL
UTILIZATION, %						
N = 1	.0	.2	.7	2.8	1.9	
N = 2	.1	.4	1.4	5.5	3.7	
N = 3	.1	.5	2.1	8.3	5.5	
N = 4	.2	.7	2.8	11.0	7.4	
N = 5	.2	.9	3.5	13.7	9.2	
N = 6	.3	1.1	4.1	16.4	11.0	
N = 7	.3	1.2	4.8	19.1	12.8	
N = 8	.4	1.4	5.5	21.7	14.5	
SPIN DEG, %						
N = 1	.0	.0	.0	.0	.0	.0
N = 2	.0	.0	.0	.1	.0	.3
N = 3	.0	.0	.0	.2	.1	.6
N = 4	.0	.0	.0	.2	.1	.9
N = 5	.0	.0	.0	.3	.1	1.2
N = 6	.0	.0	.0	.4	.2	1.5
N = 7	.0	.0	.0	.5	.2	1.9
N = 8	.0	.0	.0	.6	.3	2.2

Figure 6: Results of Model with Lock 5 Split 6 Ways