Queueing Network Model of Interactive Computing Systems

PETER PIN-SHAN CHEN

Abstract—A closed queueing network model with state dependent routing probabilities is developed for the study of interactive computing systems which use swapping as a memory management strategy. An algorithm to obtain an approximate solution of the mathematical model is proposed. Based on measurements of a dual-processor PDP-10 system, the model is found to be better in predicting the system performance than the classical model without state dependent routing probabilities.

I. Introduction

There has been considerable interest in developing models for interactive computing systems. One of the earliest analyses was done by Scherr [1] for the Compatible Time-Sharing System (CTSS) at M.I.T., Cambridge, Mass. Kleinrock, Chang, Coffman, Muntz, Adiri, Avi-itzhak, and other researchers have also used queue-theoretic models to study CPU scheduling algorithms [2]-[6]. Most of these models are concerned with one system component—the CPU; thus the interrelationships between the CPU and other system components, such as disks, have been ignored.

To study an interactive computing system as a whole, a "total system model" is needed. Using Gordon and Newell's method [7], Moore [8] constructed a queueing network model for the Michigan Time-Sharing system at the University of Michigan, Ann Arbor. His model consists of peripheral devices as well as the CPU's, but the effect of memory size on system behavior is not considered. Queueing networks have also been utilized by Buzen [9] and Rice [10] to study the behavior of multiprogramming systems.

Using a different approach, Sekino [11] develops a model for MULTICS at M.I.T.. Essentially, his model is a set of hierarchically organized submodels such as the program behavior model, the secondary memory model, etc. His model includes many important system parameters but is not suitable in estimating the utilization factor and queue length of individual system resources.

Both Sekino's work and a major part of Moore's work are oriented toward interactive computing systems with virtual memory. Since a large proportion of the existing systems still use swapping as the memory management strategy [12], it is the purpose of this paper to develop a model for this type of system.

The classical approach to the modeling of program swapping behavior is to assume that each program is always swapped in at the start of the interaction, and always swapped out at the end of the interaction. Moore [8] uses this approach to model the GE-635 time-sharing system. In reality, the program swapping behavior is more complex and should depend on the main memory size, the number of jobs competing for memory, and the job sizes.

Manuscript received October 9, 1974; revised January 15, 1975.

The author was with Digital Equipment Corporation, Maynard, Mass. He is now with the Center for Information Systems Research, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Mass. 02139.

This paper proposes a new approach: the program swapping behavior is represented by state dependent routing probabilities in a closed queueing network. The probability that a program needs to be swapped in (or out) is expressed as a function of the system state and several important system parameters such as main memory size. Since the exact solution for this type of queueing network model is not available, we propose an algorithm to obtain an approximate solution and discuss its convergence. The model is compared with the classical model using measurement data obtained from a dual-processor PDP-10 system.

II. The Model

A model of an interactive computing system which consists of one CPU, one disk, one swapping drum, and a set of terminals is shown in Fig. 1. There are queues ahead of the CPU, the disk, and the swapping drum. There is no queue in front of the terminals since each terminal is dedicated to a user (i.e., a job). The mean service times of the CPU, the disk, and the swapping drum are $1/\mu_1$, $1/\mu_2$, $1/\mu_3$, respectively. The average "think time" of the users is denoted by $1/\mu_4$. All service times are considered to be exponentially distributed. The routing probability P_{ij} is the probability that a job will request the service of the jth facility after the service of the ith facility is completed. The number of jobs in the system N is assumed to be fixed during the time period we are concerned with. Thus we have a closed queueing network system.

Queueing networks are good representations of interactive systems, since each job usually goes through several service facilities in order to satisfy the user's request. The following description will make this point clear. Consider a user who sits in front of a terminal and types in a request. If his job (program) is not in main memory (this event has probability of P_{43}), the job will need to be swapped in before it is put in the CPU queue. When a job has been processed by the CPU for a while (with an average $1/\mu_1$ time units), it has four possible destinations. If the stoppage of processing is due to the fact that the allocated quantum is expired (probability P_{11}), it will be put back in the CPU queue; if the job needs information on a disk file (probability P_{12}), it will be put in the disk queue; if the main memory size is not sufficient (probability P_{13}), it will be swapped out; and finally, if all service requirements of the user's request have been satisfied, the answer to the user's request will appear on the terminal. After "thinking" for an average $1/\mu_4$ time units, the user will type in another request and begin another "interaction."

A. Input Parameters

The input parameters to the model are:

1) number of jobs in the system (N)
2) main memory size (user area) (M)
3) average job size (J)
4) average CPU service time ($1/\mu_1$)

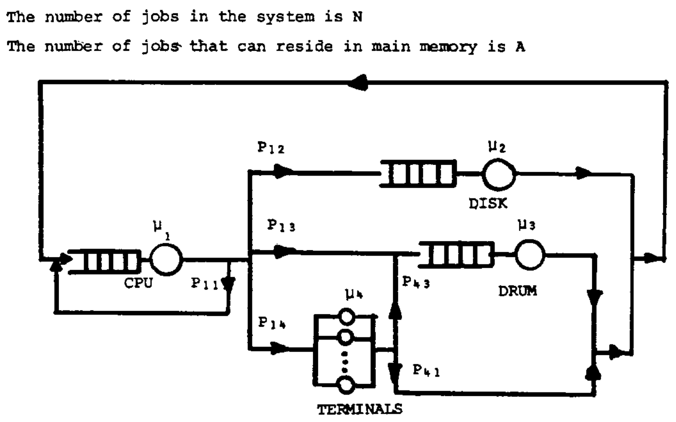

Fig. 1. A closed queueing network model for interactive systems.

5) average disk I/O time ($1/\mu_2$)
6) average job swapping time ($1/\mu_3$)
7) average user think time ($1/\mu_4$)
8) average CPU time needed per interaction (T_{CPU})
9) average number of disk I/O requests per interaction (N_{DIO}).

The values of these parameters are assumed to be available by measurements.

B. Derivation of P_{14} and P_{12}

P_{14} and P_{12} are derived from T_{CPU} and N_{DIO} as follows. Assuming that P_{14} is known, the average number of CPU requests per interaction is

$$N_{\text{CPU}} = \sum_{k=1}^{\infty} kP_{14}(1 - P_{14})^{k-1} = 1/P_{14}. \qquad (1)$$

The average CPU time needed per interaction is

$$T_{\text{CPU}} = (1/\mu_1)\,(1/P_{14}). \qquad (2)$$

Therefore, we have

$$P_{14} = 1/(\mu_1\, T_{\text{CPU}}). \qquad (3)$$

Assuming that P_{12} is known, the probability that a job makes exactly k disk I/O requests is

$$\sum_{c=k}^{\infty} \binom{C}{k} P_{12}^{k}\,(1 - P_{14} - P_{12})^{C-k} P_{14}$$
$$= P_{12}^{k} P_{14}/(P_{12} + P_{14})^{k+1}$$
$$= (P_{14}/(P_{12} + P_{14}))\,(P_{12}/(P_{12} + P_{14}))^{k}. \qquad (4)$$

The average number of disk I/O requests per interaction is

$$N_{\text{DIO}} = \sum_{k=0}^{\infty} k(P_{14}/(P_{12} + P_{14}))\,(P_{12}/(P_{12} + P_{14}))^{k} = P_{12}/P_{14}. \qquad (5)$$

Therefore, we have

$$P_{12} = N_{\text{DIO}} \cdot P_{14}. \qquad (6)$$

C. Model of Program Swapping Behavior

The program swapping behavior is represented by state dependent routing probabilities P_{43} and P_{13}. Different operating systems may have different rules for handling swapping. Therefore, the expressions for P_{43} and P_{13} may be different for different operating systems. In this paper, we derive the expressions for P_{43} and P_{13} under a set of assumptions which are reasonable in several operating systems including the TOPS-10 operating system for PDP-10.

The assumptions we made for deriving P_{43} are as follows.

1) Each job in the main memory is allocated the same amount of memory. Therefore, the maximum number of jobs that can be allocated in the main memory simultaneously is a constant and denoted by A.

2) Among all jobs in the main memory, the jobs in the "think" mode have the highest priority to be swapped out if memory space is needed for the jobs to be swapped in.

Assumption 1) is justified in systems in which the main memory is partitioned into a fixed number of parts. If the memory allocated to jobs is a random variable, we let

$$A = \lfloor M/J \rfloor \qquad (7)$$

where J is the average job size, and the resulting expression of P_{43} and P_{13} can be used as an approximation.

Assumption 2) is actually the memory management strategy used in many existing systems, since the jobs in the "think mode" are very unlikely to need CPU service in the immediate future (note that the average think time is in the order of seconds).

The routing probability P_{43} is derived by analyzing the status of jobs in the think mode. Let $n_i (i = 1, 2, 3, 4)$ denote the number of jobs in the ith facility. If N, the total number of jobs in the system, is not greater than A, there is no swapping activity (all jobs can reside in the main memory all the time). If N is greater than A, which is the common case, the number of jobs which cannot stay in the main memory is $N - A$. Under assumption 2), all jobs in the think mode will not be in the main memory if $n_4 \leqslant N - A$. Or, $N - A$ jobs out of n_4 jobs in the think mode are not in the main memory if $n_4 > N - A$. Therefore, the probability that a job needs to be swapped in at the beginning of the interaction, P_{43}, is equal to the probability that a job in the think mode is not in the main memory. We use the approximation

$$P_{43} = \begin{cases} (N - A)/n_4, & \text{if } n_4 \geqslant N - A \\ & \quad \text{and } N \geqslant A \\ 0, & \text{if } N \leqslant A \\ 1, & \text{if } n_4 \leqslant N - A \end{cases} \qquad (8)$$

and the probability that a job does not need to be swapped in at the beginning of the interaction is

$$P_{41} = 1 - P_{43}. \qquad (9)$$

To derive P_{13}, two more assumptions are made:

3) P_{13} is linearly proportional to $n_1 + n_2 + n_3 - A$ if $n_1 + n_2 + n_3 \geqslant A$. Otherwise, it is zero.

4) In the worst case, the maximum average number of times that a job will be swapped out before the end of the interaction is limited to one.

In assumption 3), $n_1 + n_2 + n_3 - A$ can be interpreted as the number of jobs which need space in the main memory but can not have it. When $n_1 + n_2 + n_3 > A$, some jobs have to be swapped out before the end of the interactions (note that all jobs in the think mode are already swapped out under assumption 2). When $n_1 + n_2 + n_3 \leqslant A$, P_{13} is zero since no jobs need to be swapped out except those in the think mode.

Assumption 4) represents the effect of some schemes used in the operating systems to prevent serious swapping. One such scheme is to lock the jobs which have been swapped once in the main memory until they enter the think mode.

From assumption 3), P_{13} achieves its maximum when $n_1 + n_2 + n_3$ achieves the maximum value N. Similar to (6), the maximum value of P_{13}, which is given in assumption 4), is equal to P_{14}. Following assumption 3), P_{13} varies linearly from 0 to P_{14} as $n_1 + n_2 + n_3$ varies from A to N.

Note also that the sum of P_{11}, P_{12}, P_{13}, and P_{14} is one, and P_{12} and P_{14} are determined by (3) and (6). Thus the value of P_{13} is bounded by $1 - P_{12} - P_{14}$. Therefore, we have

$$P_{13} = \begin{cases} \min\,[1 - P_{12} - P_{14}, P_{14}((n_1 + n_2 + n_3 - A)/(N - A))], \\ \qquad \text{if } n_1 + n_2 + n_3 \geqslant A \text{ and } N \geqslant A \\ 0, \quad \text{otherwise} \end{cases} \tag{10}$$

and

$$P_{11} = 1 - P_{13}. \tag{11}$$

III. Approximate Solution of the Model

From (8) and (10), we can see that P_{43} and P_{13} depend on n_4 (noting that $n_4 = N - n_1 - n_2 - n_3$). That is, we have a queueing network in which some of the routing probabilities depend on the state of the system (in this case, the number of jobs in a particular service facility).

A. Algorithm

Since the solution technique for queueing network models with state dependent routing probabilities is not available, we propose the following algorithm to obtain the approximate solution.

Step 1. First we assume an initial value for $\overline{n_4}$, the average number of jobs in the think mode.

Step 2. Use $\overline{n_4}$ to calculate P_{43} and P_{13} by (8) and (10).

Step 3. Treat P_{43} and P_{13} as fixed values and solve the model using Buzen's method [13].

Step 4. If the new value of $\overline{n_4}$, which is one of the outputs of the model, is very close to its old value, the algorithm stops. Otherwise, Steps 2, 3, and 4 are repeated.

We shall examine the problem of convergence for the algorithm. It can be shown that the output value of $\overline{n_4}$ (produced in Step 4) increases as the input value of $\overline{n_4}$ increases. Note that the output value of $\overline{n_4}$ is between 0 and N. Since a bounded monotonically increasing (or decreasing) sequence will converge, our algorithm will converge. From our computational experience, the algorithm converges to the same point, independent of its starting values.

B. Output of the Model

At the time the algorithm stops, the utilization factor U_i and average queue length $Q_i (i = 1, 2, 3, 4)$ of the ith service facility can be calculated by [13]. The average time that a request spends in the first three facilities (i.e., the waiting time plus the service time) can be derived by Little's formula [14]:

$$R_i = Q_i/(\mu_i U_i), \qquad i = 1, 2, 3. \tag{12}$$

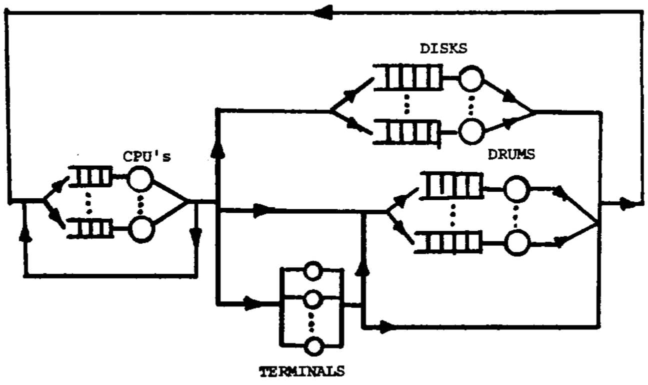

Fig. 2. Model for multiprocessor and multidevice systems.

Since each job has a dedicated terminal, we have

$$R_4 = 1/\mu_4. \tag{13}$$

The average system response time, which is the average time the system takes to serve a user's request, can be derived by considering the average number of times a job passes each facility during one interaction. The average number of "swapping in" operations at the beginning of the interaction is

$$N_{\text{BEG}} = P_{43}. \tag{14}$$

Following the same argument in the derivation for (5), the average number of swapping operations per interaction (not including the initial swapping) is

$$N_{\text{SWAP}} = P_{13}/P_{14}. \tag{15}$$

From (1), (5), (14), and (15), the average system response time is

$$\begin{aligned} R &= R_3 N_{\text{BEG}} + R_1 N_{\text{CPU}} + R_2 N_{\text{DIO}} + R_3 N_{\text{SWAP}} \\ &= R_3 P_{43} + R_1/P_{14} + R_2 P_{12}/P_{14} + R_3 P_{13}/P_{14}. \end{aligned} \tag{16}$$

Therefore, the model can be used to predict the average system response time as well as the utilization factor and the queue length of an individual system resource.

IV. Multi-CPU and Multidevice Case

In large interactive systems, there are usually many disks and sometimes more than one swapping drum and CPU. The approach we take to model this case is to treat each device or CPU as an independent server with a separate request queue. Fig. 2 illustrates a model using this approach. The solution to the extended model is a simple extension of the original model.

V. Comparison with the Classical Model

The model developed in this paper is compared with the classical model ($P_{43} = 1$ and $P_{13} = 0$) and with several sets of measured data from a real system.

The data were collected at the end of August 1974 from an in-house dual-processor PDP-10 system at Digital Equipment Corporation, Maynard, Massachusetts. Since this system is dedicated to internal uses only, the system performance may not be the same as it is in systems offered to the public.

The model we developed for the system is similar to the one shown in Fig. 2. In the actual system, one CPU is used primarily for processing short jobs and another CPU for long jobs. For simplicity, we treat them equally so that a request for CPU service has the equal probability to be served by either CPU. Although the system has two physical swapping drums, we

TABLE I
COMPARISON OF THE STATE DEPENDENT MODEL WITH THE CLASSICAL MODEL

Case		1	2	3	4	5
Number of Jobs		40	36	68	72	70
Average Job Size (K Words)		6.559	6.678	11.681	8.609	9.548
Main Memory Size (User Area) (K Words)		118	118	182	182	182
Averge CPU Service Time (sec.)		0.02814	0.00708	0.00568	0.01230	0.01347
Overhead in Job Switching (sec.)		0.00386	0.00196	0.00237	0.00255	0.00265
Average CPU Time needed per interaction (sec.)		0.411	0.2598	0.098	0.09015	0.0837
Average User Think Time (sec.)		7.81	6.539	5.95	2.948	5.938
Number of Disk I/O per interaction		8.94	1.904	4.583	4.003	4.183
Average System Response Time (sec.)	Measured	3.23	2.299	2.44	2.948	1.612
	Prediction by the state dependent model	3.5467	1.6327	2.2143	2.5029	1.7779
	Prediction by the Classical Model	3.5746	1.7038	2.0857	2.4022	1.9334

model them as one logical drum since that is the way the operating system treats it. The number of disks in the system is six. The disk I/O requests are also assumed to be distributed with equal probabilities over all disks. The disk service time is 80 ms, and the swapping time (the time to swap a job out and another job in) is 108 ms.

Table I contains five sets of measured data. Note that there is a parameter called overhead in job switching, which is the overhead associated with each CPU service. Therefore, in using (3) to calculate P_{14}, the value of $1/\mu_1$ should be replaced by the CPU service time minus the overhead in job switching.

These five sets of data are used as input to both our model and the classical model, and the response times predicted by two models are compared against the measured data. In four out of five cases, the values predicted by our model are closer to the measured data. In cases 1 and 5, the swapping activities are lower in our model ($P_{43} < 1$ and $P_{13} = 0$) than that in the classical model ($P_{43} = 1$ and $P_{13} = 0$). In cases 3 and 4, our model predicts more swapping activities ($P_{43} = 1$ and $P_{13} > 0$). Only in case 2 does the classical model predict better than our model, but the difference between the two predicted values is quite small.

ACKNOWLEDGMENT

The author wishes to express his thanks to R. Turner for his effort in collecting the data and his help in formulating the model for the PDP-10. The author is also indebted to J. Buzen and J. Bell for many stimulating discussions.

REFERENCES

[1] A. L. Scherr, "An analysis of time-shared computer systems," Ph.D. dissertation, M.I.T., Cambridge, Mass., Tech. Rep. MAC-TR-18, 1965.
[2] W. Chang, "A queueing model for a simple case of time-sharing," *IBM Syst. J.*, vol. 5, no. 2, pp. 115-125, 1966.
[3] L. Kleinrock, "Analysis of a time-shared processor," *Naval Res. Logist. Quart.*, vol. 11, pp. 59-73, Mar. 1964.
[4] E. G. Coffman, Jr. and L. Kleinrock, "Feedback queueing models for time-shared systems," *J. Ass. Comput. Mach.*, pp. 549-576, Oct. 1968.
[5] E. G. Coffman, Jr., R. R. Muntz, and H. Trotter, "Waiting time distributions for process-sharing systems," *J. Ass. Comput. Mach.*, vol. 17, pp. 123-130, Jan. 1970.
[6] I. Adiri and B. Avi-Itzhak, "A time-sharing queue with a finite number of customers," *J. Ass. Comput. Mach.*, vol. 2, pp. 315-323, Apr. 1969.
[7] W. J. Gordon and G. F. Newell, "Closed queueing systems with exponential servers," *Oper. Res.*, vol. 15, pp. 254-265, Mar. 1967.
[8] G. G. Moore, "Network models for large-scale time-sharing systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, 1971.
[9] J. P. Buzen, "Queueing network models of multiprogramming," Ph.D. dissertation, Div. Eng. Appl. Phys., Harvard Univ., Cambridge, Mass., 1971.
[10] D. R. Rice, "An analytical model for computer system performance analysis," Ph.D. dissertation, Univ. Florida, Gainesville, 1971.
[11] A. Sekino, "Performance evaluation of multiprogrammed time-shared computer systems," Ph.D. dissertation, Dep. Elec. Eng., M.I.T., Cambridge, Mass., Tech. Rep. MAC-TR-103, 1973.
[12] S. E. Madnick and J. J. Donovan, *Operating Systems.* New York: McGraw-Hill, 1974.
[13] J. P. Buzen, "Computational algorithms for closed queueing network with exponential servers," *Commun. Ass. Comput. Mach.*, vol. 16, pp. 527-531, Sept. 1973.
[14] J. D. C. Little, "A proof of the queueing formula: $L = \lambda W$," *Oper. Res.*, vol. 9, pp. 383-387, 1961.