Processor Utilization in Multiprogramming Systems via Diffusion Approximations

Donald P. Gaver

Naval Postgraduate School, Monterey, California

and

Gerald S. Shedler

IBM Research Laboratory, San Jose, California

(Received December 16, 1971)

Cyclic queuing systems have been proposed by several authors in the study of the behavior of multiprogrammed computer systems. Programs in the system wait for service at the central processor unit (CPU); then, after page fault or input-output request at a data transmission unit (DTU), the process repeats until the program completes. Semi-Markov analysis of such systems, based on the apparently plausible assumption of independently but exponentially distributed CPU burst time, and independent, but nearly constant DTU time may be conducted. This paper presents some very simple approximations based on a continuous-state approximation—the simple diffusion with two reflecting barriers—to describe the CPU utilization. Computational experience from which the quality of the approximations can be assessed is reported.

SEVERAL AUTHORS have recognized that interesting characteristics of multiprogramming computer systems may be obtained by analysis of the cyclic queues occurring at the CPU (central processor unit) and DTU (data transmission unit); *see* GAVER,[3] LEWIS AND SHEDLER,[5] Shedler,[8,9] and others. A system feature that complicates the probability analysis is the non-Markovian nature of DTU service: while it may be roughly plausible to assume CPU service times (e.g., times to page fault) to be independently and exponentially distributed, it is apparently much less reasonable to make such a distributional assumption about DTU service times. Although straightforward attacks on this problem have been successfully conducted (*see* references 5 and 8), it may be useful to attempt a simpler approximate approach—the objective of this paper.

The results obtained here have the virtue of a refreshing mathematical simplicity, but are, of course, only in approximate agreement with results such as those obtained in Shedler.[8] However, when we reflect that our present models for multiprogramming are quite simplistic in the light of current knowledge of (a) program behavior at the CPU, (b) information accessing at the DTU, and (c) representation of storage hierarchies, approximations of this type may be quite adequate for providing insights. We envision using results of this kind to design computer systems with acceptable cost-performance characteristics. Such results are useful, for example, in suggesting performance characteristics of devices that will result in a balanced system. In addition, by taking cost characteristics into account,

one can begin to evaluate technologies from a total-system point of view, and thus be guided to select appropriate devices. With respect to such questions, we can consider utilizing our results to obtain an initial idea of the region of parameter space to explore more extensively by means of simulation or by improved analytical approximations, several of which are currently under development.

THE DIFFUSION APPROXIMATION

WE CONSIDER A cyclic queuing system (*see* Fig. 1) that consists of two sequential stages. The system is assumed to serve a constant number J of programs ($J \geqq 2$), each of which goes through both stages in sequence and then returns to the first stage, this process being repeated continuously. It is assumed that, after completion of CPU service, a program moves instantaneously from stage 1 to the tail of

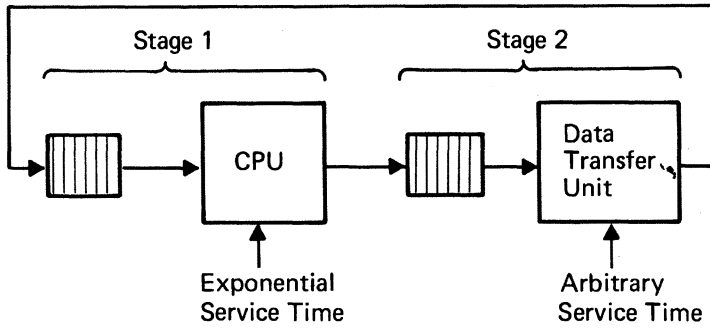

Fig. 1. The cyclic-queue model.

the queue in stage 2, and after DTU service at that stage, back to the tail of the queue in stage 1. We shall suppose in addition that the queue in front of the CPU and the queue in front of the DTU are served according to a first-in–first-out (FIFO) discipline. The assumption that J is a constant is an approximation that is justified by the common practice of operating such a system in a saturated mode.

In addition, we assume that the service times at the CPU and the DTU are mutually independent, and that CPU service times and DTU service times, respectively, are identically distributed.

We now give a brief intuitive account of our approximation. Let $N_C(t)$ denote the number of programs present at the CPU at time t, including those queued in addition to the program currently in service. Then, if $N_C(0)=0$,

$$N_C(t)=A(t)-D(t), \tag{1}$$

where $A(t)$ represents the number of arrivals at the CPU in $(0, t)$, and $D(t)$ is the number CPU departures in $(0, t)$. If we neglect boundary effects at 0 and J, $A(t)$ and $D(t)$ are independent renewal processes, so, as t becomes large, $A(t)$ and $D(t)$ are approximately normally distributed with means $t/E[D]$ and $t/E[S]$ and variances $t\text{var}[D]/(E[D])^3$ and $t\text{var}[S]/(E[S])^3$ respectively (*see* Cox,[1] page 40). It follows that $N_C(t)$ is approximately normally distributed with mean $\mu t=\{1/\text{E}[D]-$

$1/\mathrm{E}[S]\}t$ and variance $\sigma^2 t=\{\mathrm{var}[D]/(E[D])^3+\mathrm{var}[S]/(E[S])^3\}t$. We now approximate by replacing the difference of renewal processes by a diffusion (Wiener) process with drift μ and infinitesimal variance σ^2. Thus, $F(x, t)$, the distribution of $N_C(t)$, satisfies the diffusion equation

$$\partial F/\partial t=-\mu\partial F/\partial x+(\sigma^2/2)\partial^2 F/\partial x^2, \tag{2}$$

again approximately; cf. NEWELL,[7] pp. 105–107. A reflecting boundary condition at $x=0$ must be imposed, for $N_C(t)\geqq 0$, and another such boundary condition at $x=J$ constrains $N_C(t)$ to be $\leqq J$. We require the solution to (2), subject to an initial distribution, e.g.,

$$F(x, 0)=\begin{cases}1, & x\geqq x_0>0,\\ 0, & x<x_0,\end{cases} \tag{3}$$

and boundary conditions

$$F(0+, t)\geqq 0, \qquad F(J, t)=1. \tag{4}$$

For further details of the behavior of this approximation when J is infinite, *see* Gaver.[4] For a closed-form solution to (2) when J is infinite, *see* Newell.[7]

THE STATIONARY DISTRIBUTION IN THE DIFFUSION APPROXIMATION

THE STATIONARY or long-run distribution associated with our problem is $F(x)=\lim_{t\to\infty} F(x, t)$ and satisfies

$$(\sigma^2/2)d^2F/dx^2-\mu dF/dx=0$$

for $x>0$. Routine integrations lead to the solution

$$F(x)=A[1-B\exp(2\mu x/\sigma^2)]. \tag{5}$$

Invocation of the upper boundary condition provides that

$$1=A[1-B\exp(2\mu J/\sigma^2)]. \tag{6}$$

According to (5) and (6) we have

$$F(x)=[1-B\exp(2\mu x/\sigma^2)]/[1-B\exp(2\mu J/\sigma^2)]. \tag{7}$$

It remains to determine the constant B. We suggest three alternatives, and present their numerical properties.

1. *Exact fit for* $J=1$. If only one program occupies the system, then renewal theory provides that (assuming $E[D]=1$, as we shall throughout)

$$F(0+)=1/(1+E[S]). \tag{8}$$

If we set $J=1$ and $x=0$ in (7) and equate the result to (8), the value of B is determined:

$$B=E[S]/\{E[S]-\exp(2\mu/\sigma^2)\}.$$

2. *Exact fit for* $J=2$. If CPU service times are independently distributed copies of a random variable S, and likewise DTU service times are independent copies of the random variable D, then the long-run CPU utilization u, when $J=2$, is given by

$$u=1-F(0+)=E[S]/E[\max(S, D)]. \tag{9}$$

This was pointed out by J. GECSEI (private communication), and can be seen as follows. Consider the sequence of times $\{\tau_K\}$ at which either (i) the CPU is idle, a DTU service has just been completed, and the served program has moved to the CPU stage queue, or (ii) the DTU is idle, a CPU service has just been completed, and the served program has moved to the DTU-stage queue. These $\{\tau_K\}$ are regeneration points in the process, the mean regeneration time being $E[\max(S, D)]$. Since the mean amount of time that the CPU is busy between regeneration points is $E(S)$, (9) follows from a basic limit theorem for regenerative processes; *see* FELLER,[2] p. 365. Making use of (7), we obtain

$$B = E[S]/\{E[\max(S, D)][1-\exp(4\mu/\sigma^2)]+E[S]\exp(4\mu/\sigma^2)\}. \tag{10}$$

3. *Exact fit for large* J $(J\to\infty)$. Of most practical interest is the case of large J. The diffusion approximation may also be expected to work best when boundaries are visited infrequently, and this implies large J and near equality of $E[S]$ and $E[D]$.

In practice $\mu<0$, so we may allow $J\to\infty$ in (7) to discover that

$$F(x) = \lim_{J\to\infty}\{[1-B\exp(2\mu x/\sigma^2)]/[1-B\exp(2\mu J/\sigma^2)]\} = 1-B\exp(2\mu x/\sigma^2), \tag{11}$$

and hence for $J=\infty$,

$$F(0+) = 1-B. \tag{12}$$

But, for arbitrary queuing systems of the type under consideration (*see* TAKÁCS,[10] p. 142), $\lim_{t\to\infty} P\{N_C(t)=0\}=1-\rho$, if $\rho<1$, where ρ is the traffic-intensity parameter, and $\rho=E[S]/E[D]<1$ when $J=\infty$. Consequently we put $B=\rho$ and find

$$F(0+) = (1-\rho)/[1-\rho\exp(2\mu J/\sigma^2)]. \tag{13}$$

This simple expression promises to give a good approximation when J is reasonably large and μ is not far from zero but has a negative sign. The quality of the approximations may be judged by refering to the numerical examples that follow.

NUMERICAL EXAMPLES

SHEDLER[8] HAS tabulated CPU utilization, that is, the long-run probability that the CPU is busy, which depends on the DTU distribution and the degree of multiprogramming J. The numbers obtained are the result of a semi-Markov analysis, whose success seems to depend rather crucially on the assumption of an exponential service time at the CPU.

Our diffusion approximation is capable of supplying figures for CPU utilization—an explicit formula follows directly from (13) by subtraction from unity:

$$\text{CPU utilization} = \rho\{[1-\exp(2\mu J/\sigma^2)]/[1-\rho\exp(2\mu J/\sigma^2]\} \tag{14}$$

when $\rho<1$. Here we compare the 'exact' numbers obtained by Shedler[8] with those delivered by (14). Since the usual system is likely to have a relatively high degree of multiprogramming, we shall first discuss Method 3 for fitting B (exact fit for $J=\infty$). Refer to Table I. Evidently the diffusion (diff.) and semi-Markov (S-M) figures agree quite closely. The numerical comparisons suggest that the diffusion approach provides an underestimate in the case of an exponential DTU.

That this will always be the case may be shown analytically. Before doing so, we shall compare some of the results of Table I to the numbers obtained when Method 2 is used (exact fit for $J=2$). This comparison is illustrated in Table II. Apparently the procedure of fitting $J=2$ exactly does not agree as well with the S-M calculations as does the $J=\infty$ exact fit. When we recollect that explicit calcula-

TABLE I

THE CPU UTILIZATION COMPARISON BASED ON FITTING B FOR $J=\infty$

(CPU exponential mean $E[S]$; DTU service-distribution mean=1.)

J	$E[S]$	Erlang—1 (exponential)		Erlang—2		Erlang—3		Erlang—4		Erlang—5		Erlang—∞ (constant)	
		S-M	Diff.	S-M	Diff.	S-M	Diff.	S-M	Diff.	S-M	Diff.	S-M	Diff.
2		0.238	0.233	0.243	0.237	0.245	0.238	0.246	0.239	0.247	0.239	0.249	0.241
3		0.247	0.245	0.249	0.247	0.250	0.247	0.250	0.247	0.250	0.247	0.250	0.248
4	0.25	0.249	0.249	0.250	0.249	0.250	0.249	0.250	0.249	0.250	0.249	0.250	0.250
5		0.250	0.250	0.250	0.250	0.250	0.250	0.250	0.250	0.250	0.250	0.250	0.250
6		0.250	0.250	0.250	0.250	0.250	0.250	0.250	0.250	0.250	0.250	0.250	0.250
2		0.429	0.424	0.444	0.444	0.451	0.451	0.455	0.454	0.458	0.456	0.468	0.464
3		0.467	0.464	0.480	0.476	0.485	0.480	0.487	0.482	0.489	0.483	0.494	0.487
4	0.50	0.484	0.482	0.493	0.490	0.495	0.492	0.496	0.493	0.497	0.493	0.499	0.495
5		0.492	0.491	0.497	0.495	0.498	0.497	0.499	0.497	0.499	0.497	0.500	0.498
6		0.496	0.495	0.499	0.498	0.500	0.499	0.500	0.499	0.500	0.499	0.500	0.499
2		0.568	0.566	0.591	0.608	0.601	0.623	0.606	0.631	0.610	0.636	0.626	0.655
3		0.634	0.633	0.660	0.666	0.670	0.677	0.676	0.683	0.679	0.686	0.694	0.700
4	0.75	0.672	0.672	0.695	0.697	0.704	0.706	0.709	0.710	0.711	0.712	0.723	0.722
5		0.696	0.695	0.716	0.716	0.722	0.722	0.726	0.725	0.728	0.727	0.736	0.734
6		0.712	0.711	0.728	0.727	0.733	0.732	0.736	0.734	0.737	0.736	0.743	0.741
2		0.590	0.589	0.614	0.635	0.624	0.652	0.630	0.661	0.633	0.666	0.651	0.688
3		0.661	0.661	0.689	0.697	0.700	0.710	0.701	0.717	0.709	0.721	0.726	0.737
4	0.80	0.702	0.702	0.728	0.732	0.738	0.742	0.743	0.747	0.746	0.750	0.759	0.762
5		0.729	0.729	0.751	0.753	0.759	0.761	0.763	0.764	0.766	0.767	0.776	0.776
6		0.747	0.747	0.766	0.766	0.773	0.773	0.776	0.776	0.778	0.778	0.786	0.784
2		0.631	0.631	0.656	0.685	0.667	0.705	0.673	0.715	0.677	0.721	0.694	0.748
3		0.709	0.709	0.739	0.753	0.751	0.769	0.757	0.777	0.762	0.782	0.780	0.803
4	0.90	0.756	0.756	0.785	0.792	0.796	0.806	0.802	0.812	0.806	0.816	0.821	0.832
5		0.787	0.787	0.813	0.818	0.823	0.829	0.828	0.834	0.831	0.837	0.845	0.851
6		0.808	0.808	0.832	0.835	0.841	0.844	0.846	0.849	0.848	0.852	0.860	0.863

tion of $E[\max(S, D)]$ for general distributions of S and D is a bit troublesome, it becomes easier to recommend the method of fit obtained by Method 3.

THE DIFFUSION APPROXIMATION TO THE MARKOVIAN CYCLIC SYSTEM

IN THIS SECTION we shall show that the tendency for the diffusion (in the fit of Method 3) to underestimate CPU utilization in the exponential CPU and DTU (entirely Markov) case is no accident.

Denote the mean of S, the exponential service time at the CPU, by $E[S]=\lambda^{-1}$; again $E[D]=1$. Then explicit probability balance equations may be written:

$$\lambda p_0=p_1; \qquad (\lambda+1)p_i=\lambda p_{i-1}+p_{i+1}, \qquad 1\leqq i\leqq J-1, \qquad p_J=\lambda p_{J-1};$$

where p_j is the long-run probability of j programs at the DTU. The solution is, expressed in terms of $\rho=\lambda^{-1}$,

$$p_i=[(1-\rho)/(1-\rho^{J+1})]\rho^{J-i}.$$

TABLE II

THE CPU UTILIZATION COMPARISON

(CPU exponential mean $E[S]$; DTU service distribution mean = 1.)

J	$E[S]$	Exponential			Constant		
		S-M	Diff. (Fit $J=2$)	Diff. (Fit $J=\infty$)	S-M	Diff. (Fit $J=2$)	Diff. (Fit $J=\infty$)
2	0.25	0.238	0.238	0.233	0.249	0.249	0.241
3		0.247	0.251	0.245	0.250	0.256	0.248
4		0.249	0.254	0.249	0.250	0.258	0.250
5		0.250	0.255	0.250	0.250	0.258	0.250
6		0.250	0.256	0.250	0.250	0.259	0.250
2	0.50	0.429	0.429	0.424	0.468	0.468	0.464
3		0.467	0.468	0.464	0.494	0.492	0.487
4		0.484	0.487	0.482	0.499	0.500	0.495
5		0.492	0.495	0.491	0.500	0.503	0.498
6		0.496	0.500	0.495	0.500	0.504	0.499
2	0.75	0.568	0.568	0.566	0.626	0.626	0.655
3		0.634	0.634	0.633	0.694	0.673	0.700
4		0.672	0.673	0.672	0.723	0.696	0.722
5		0.696	0.696	0.695	0.736	0.708	0.734
6		0.712	0.712	0.711	0.743	0.716	0.741
2	0.80	0.590	0.590	0.589	0.651	0.651	0.688
3		0.661	0.661	0.661	0.726	0.702	0.737
4		0.702	0.703	0.702	0.759	0.730	0.762
5		0.729	0.729	0.729	0.776	0.745	0.776
6		0.747	0.747	0.747	0.786	0.755	0.784
2	0.90	0.631	0.631	0.631	0.694	0.694	0.748
3		0.709	0.709	0.709	0.780	0.757	0.803
4		0.756	0.756	0.756	0.821	0.791	0.832
5		0.787	0.787	0.787	0.845	0.813	0.851
6		0.808	0.808	0.808	0.860	0.828	0.863

Thus, CPU utilization $=U_e=1-p_J$, where the subscript signifies 'exact'; the corresponding figure for the diffusion is U_d, and the figures for idleness are $I_e=p_J$, and $I_d=1-U_d$, respectively. It will be shown that $I_d>I_e$, or

$$\{(1-\rho)/[1-\rho\exp(2\mu J/\sigma^2)]\}>[(1-\rho)/(1-\rho^{J+1})].$$

Now, in the present case, $2\mu/\sigma^2=2(\rho-1)/(\rho+1)$, and an easily verified inequality [ln $x<2(x-1)/(x+1)$ for $0<x<1$) shows that $\rho<\exp[2(\rho-1)/(\rho+1)]$, which

verifies the assertion. It may also be shown that the error committed by using the diffusion approximation decreases as J increases, but not to zero. The approximation improves, however, as $E[S] \to E[D]$.

DIFFUSION APPROXIMATION FOR AN EXTENSION OF THE MODEL

IN THIS SECTION we shall indicate how a diffusion approximation for CPU utilization can be obtained when the assumption of identically distributed CPU service times is relaxed to permit the J programs to have different page-fault characteristics. Specifically, we shall now assume that the CPU service times in the sequence are mutually independent, and that the CPU service times of program i are identically distributed as a random variable $S_i (1 \leqq i \leqq J)$. The diffusion approximation proposed will again be of the form (14), and we seek appropriate expressions for μ, σ^2, and ρ.

It is easily verified that the successive CPU service times S in the cyclic queuing system are a semi-Markov process, and that within this semi-Markov process $Z(t)$, the total number of page faults in time t is a *cumulative stochastic process*; *see* Cox.[1] It then follows from cumulative process theory that

$$\mu_z \equiv \lim_t [E\{Z(t)\}/t] = J/\{E[S_1] + \cdots + E[S_J]\} \tag{15}$$

$$\begin{aligned} \sigma_z^2 &\equiv \lim_{t\to\infty} [\mathrm{var}\{Z(t)\}/t] \\ &= J^2 (\mathrm{var}[S_1] + \cdots + \mathrm{var}[S_J]) / (E[S_1] + \cdots + E[S_J])^3 \\ &= \{(\mathrm{var}[S_1] + \cdots + \mathrm{var}[S_J])/J\} / \{(E[S_1] + \cdots + E[S_J])/J\}^3. \end{aligned} \tag{16}$$

To establish the drift and infinitesimal variance for the diffusion approximation, put

$$\mu = 1 - \mu_z = 1 - J/(E[S_1] + \cdots + E[S_J]), \tag{17}$$

$$\begin{aligned} \sigma^2 &= \mathrm{var}[D]/E[D]^3 + \sigma_z^2 = \mathrm{var}[D]/(E[D])^3 \\ &\quad + \{(\mathrm{var}[S_1] + \cdots + \mathrm{var}[S_J])/J\} / \{(E[S_1] + \cdots + E[S_J])/J\}^3, \end{aligned} \tag{18}$$

and

$$\rho = 1/\mu_z = (E[S_1] + \cdots + E[S_J])/J. \tag{19}$$

One can also assume that each program's DTU service time comes from a specific distribution, and generalize these expressions accordingly in an obvious way.

CONCLUSIONS AND DIRECTIONS FOR FURTHER WORK

ALTHOUGH CERTAIN OF the cyclic queue problems relevant to multiprogramming studies may be solved by conventional methods (semi-Markov processes, or the 'phase' approach described in MORSE,[6] the results are nearly always cumbersome and difficult to compute. The approximate approach exhibited here furnishes simple formulas that may be evaluated by hand computation. The accuracy of the approximation seems quite adequate for the cases studied. However, efforts to improve the approximation are suggested for the following reasons.

1. Data analysis of actual CPU service times indicates that they are apt to be somewhat more skewed than the exponential.

2. The cyclic queue model discussed here may also be used to represent a

reliability situation. In this interpretation components of subsystems take the place of programs. A DTU service time represents the service life of a component, and CPU service is identified with a repair; in this model there is only one repair facility. Finally, $J-1$ is the number of spare components in the system. One question of interest concerns the long-run probability that the DTU is 'busy,' equivalent to the probability that all J items in the system are not undergoing repair. This 'availability' figure is approximated by $F(J-1)$. Another question relates to the probability that, if initially (i) a new component is in operation (at the DTU), and (ii) the $J-1$ spares are queued behind the latter, then the time until the DTU becomes idle for the first time exceeds t. The latter probability may be approximated by solving a diffusion equation, this time with one reflecting and one absorbing barrier. The details remain to be worked out, and the quality of the approximation evaluated.

ACKNOWLEDGMENT

GAVER'S PORTION of this work was partially supported by the National Science Foundation. He is also a consultant to IBM.

REFERENCES

1. D. R. Cox, *Renewal Theory*, Methuen Monograph, Methuen, London, 1962.
2. W. FELLER, *An Introduction to Probability Theory and its Applications*, Vol. II, Wiley, New York, 1966.
3. D. P. GAVER, "Probability Models for Multiprogramming Computer Systems," *J. ACM* **14**, 423–438 (1967).
4. ———, "Diffusion Approximations and Models for Certain Congestion Problems," *J. Appl. Prob.* **5**, 607–623 (1968).
5. P. A. W. LEWIS AND G. S. SHEDLER, "A Cyclic-Queue Model of System Overhead in Multiprogrammed Computer Systems," *J. ACM* **18**, 199–220 (1971).
6. P. M. MORSE, *Queues, Inventories and Maintenance*, Wiley, New York, 1958.
7. G. F. NEWELL, *Applications of Queueing Theory*, Chapman and Hall, Ltd., London, 1971.
8. G. S. SHEDLER, "A Cyclic-Queue Model of a Paging Machine," IBM Research Report RC-2814, IBM Watson Research Center, Yorktown Heights, New York, March 1970.
9. ———, "A Queueing Model of a Multiprogrammed Computer with a Two-Level Storage System," *Comm. ACM* **16**, 3–10 (1973).
10. L. TAKÁCS, *Introduction to the Theory of Queues*, Oxford University Press, New York, 1962.