1975

An $L=S$ Criterion for Optimal Multiprogramming

Peter J. Denning

Kevin C. Kahn

Report Number:
75-162
AN L=S CRITERION FOR OPTIMAL MULTIPROGRAMMING

Peter J. Denning
Kevin C. Kean

Abstract: Balancing interpagefault lifetime (L) against page
swap time (S) has always been a performance criterion of great
intuitive appeal. This paper shows that, under normal conditions,
controlling the memory policy parameter to enforce the constraint
L \geq S, and allowing the multiprogramming load to rise as high as
demand warrants without violating this constraint, will produce a
load slightly higher than optimum. Equivalently, using the criterion
L = uS for some u slightly larger than 1 will approximate the
optimal load. Using simulations, this criterion is compared with
two others reported in the literature ("operate with L at the knee
of the lifetime curve," and "operate with the paging device at
50% utilization") and shown to be more robust except in systems
which are both execution-bound and I/O-bound.

TR-162
PD75.17

1Work reported herein supported in part by NSF Grant GJ-41289.
2Addresses: Computer Science Department, Purdue University,
West Lafayette, Indiana 47907. [317-494-8566]
INTRODUCTION

A question of perennial interest for designers of multiprogrammed systems using virtual memory is the optimal degree of multiprogramming. The optimum is characterized by maximal system service rate, which implies maximal processor utilization and minimal response time [Bra74, BBG74, BGL75, Den68b, DeG75, MuW74, RoD72, RoD73, We069]. The difficult problem is not demonstrating that an optimum exists theoretically, but rather finding some practical method of controlling the system stably near its optimum. Directly controlling the degree of multiprogramming in response to measured changes in the utilization of the processor (or indeed any other single device in the system) is likely to be unstable because of local fluctuations in the estimate of utilization. (However, controls based on aggregates of device utilizations may be more stable [BGL75].)

Let $L$ denote the system paging lifetime, i.e., the execution interval between page faults averaged over all programs, and let $S$ denote the mean paging service time of the system. We consider a class of scheduling policies with this property: the load (degree of multiprogramming) is allowed to rise as high as the demand warrants, as long as $L \geq uS$ for some constant $u$. Typically the demand is sufficient to make $L = uS$ with such a policy in operation. We shall show that, as long as the system is neither execution-bound nor I/O-bound, a policy using some small constant $u>1$ will cause the load to be approximately optimal. For simplicity, we refer to these as $L \leq S$ policies.

The successful implementation of an $L \leq S$ policy relies on a well-behaved single-parameter memory management policy for controlling lifetime. The memory policy parameter is adjusted (dynamically if necessary) so that an estimate of $L$ stays higher than the target $S$. Any working-set based
Figure 1. System configuration.
mean service time at station $i$. In particular, the page service time is

$$S = 1/b_1.$$  

For the moment, we assume that the parameters $b_1, \ldots, b_m$ are independent of $n$ (i.e., the stations contain just one server each), but $b_0$ may change with $n$. Except that no server is idle when a task is in its queue, no other assumptions are made about the service distributions or the stations.

The system lifetime $L(n)$ is the mean virtual time between paging requests when the load is $n$. Specifically, let $L_j$ denote the length of the interfault interval in the program that caused the $j$th system page fault while the load is $n$; for a sequence of $k$ such faults,

$$L(n) = \frac{1}{k} \sum_{j=1}^{k} L_j.$$  

It is important to note that a program need not have executed continuously for a time $L_j$ prior to generating the $j$th system page fault, for it may have stopped for some I/O operation; $L_j$ is thus the aggregated processor time of a program since its prior page fault.

We suppose that the main memory capacity is fixed at $M$ page frames. Under this assumption it is almost always true that $L(n)$ is strictly decreasing in $n$, since most memory policies will decrease the average space allocated to all old programs in order to increase $n$.

Let $a_i$ denote the request rate for station $i$ ($i=1, \ldots, m$); that is, $1/a_i$ is the mean virtual time between two requests for station $i$. When such a request is made, the program departs the processor and joins the queue at station $i$. The average execution time of a transaction is taken as $1/a_0$, so that $a_0$ is the service completion rate. We suppose that $a_0$,
and the I/O request rates $a_2, \ldots, a_m$ are intrinsic to the programs and independent of $n$. However, the paging rate $a_1$ is a function of $n$; in fact,

$$L(n) = \frac{1}{a_1}.$$  

Under these assumptions the processor completion rate is $b_0 = a_0 + \ldots + a_m$.

The frequency of transitions from station 0 to station $i$ is

$$q_i = \frac{a_i}{a_0 + \ldots + a_m} = \frac{a_i}{b_0},$$

where $q_0$ interprets as the frequency of departures from the system.

Under the reasonable assumption that the transition rates within the system are much faster than the rates at which transactions are submitted by the source, the long run system dynamics will depend primarily on the equilibrium values attained by the system during intervals of constant load [Cou71, Cou75]. It is therefore reasonable to consider the equilibrium behavior of the system for fixed multiprogramming load $n$.

Define $\phi_i(n)$ as the equilibrium work completion rate (tasks per unit time) of station $i$, for $i=0, \ldots, m$. The system service rate is $\phi(n) = \phi_0(n)q_0$.

In equilibrium, the output rate $\phi_i(n)$ must equate the input rate at station $i$, whereupon $\phi_i(n) = \phi_j(n)q_i$. Combining these facts with equation 4,

$$\phi(n) = \begin{cases} \phi_i(n)a_i/a_1, & i=1, \ldots, m \\ \phi_0(n)a_0/b_0, & i=0 \end{cases}$$

Since $b_1$ is the maximum service rate at station $i$,

$$\phi(n) \leq \phi_0(n)b_1/a_1, \quad i=1, \ldots, m$$
Under our assumption that the I/O stations are load-independent, we can define the constant

$$I = \min \left\{ 1, \frac{b_2}{a_2}, \ldots, \frac{b_m}{a_m} \right\}$$

with which (6) reduces to

$$\phi(n) \leq a_0 \min \left\{ I, \frac{L(n)}{S} \right\}.$$  

Note that $\phi(n)$ can never exceed $a_0$, the intrinsic program completion rate.

BASIS OF THE L=S CRITERION

Figure 2 shows the intersection of the two bounds of (8) occurring at load $n_1$ (that is, $L(n_1) = S$). The L=S criterion derives from the intuition that, when $I=1$, $\phi(n)$ ought to be decreasing for $n>n_1$ -- i.e., $n_1$ is in the vicinity of and (slightly) larger than the optimum $n_0$. This intuition requires refinement according to two gross properties of the system, viz., its execution and I/O bounds.¹

¹For a simple system with $m=1$, and using $L(n) = c(M/n)^k$ for memory size $M$, Brandwajn et al. [BGL74] show that $n_0 = c(M/n)^{1/k} = e^{kS}$, where $e$ is the base of the natural logarithm. In this case $L(n_0) = e^{kS}$ and $n_1 = e^{kS}$. The policy criterion for this case is $L = e^{kS}$, and $n_1$ is not close to $n_0$. However, various practical factors will operate to force $n_1$ closer to $n_0$ than this model predicts, such as flattening of $L(n)$ for $n>n_1$ or the presence of I/O devices in the system [LeP75].
Figure 2. Bounds on system service rate.
In an execution-bound system, programs either of inherently short
execution time or subject to a maximum limit on main memory allocation,
give rise to a lifetime function which does not exceed some value \( L_{\text{max}} < \text{IS} \).
In this case, \( n_1 \) cannot be defined as a crossing point. However, it can
be defined as the largest value of \( n \) at which \( L(n) = L_{\text{max}} \). As suggested
in Figure 3, the intuition is now that this value of \( n \) will be a good
approximation to \( n_0 \).

In an I/O-bound system, the service completion rate of some I/O
station is smaller than the request rate \( b_1 < a_1 \), which implies from (7)
that \( I < 1 \). Moreover, the rate of growth of \( \phi(n) \) toward the asymptote
\( a_0 \cdot I \) will be influenced by the ratios \( b_1 / a_1 \); all other things being equal,
a reduction in some ratio will cause \( \phi(n) \) to grow more slowly in \( n \).

If a system is both execution bound and I/O bound, the L-S criterion
may be poor. In this case, it may happen that the crossing point \( n_1 \) is
smaller than the optimum, and the policy criterion \( L = uS \) would require
\( u < 1 \). Figure 4 shows a new bound \( \phi^*(n) \), which is the system service rate
that would be observed under the (unrealizable) hypothesis that \( S = 0 \):

\[
\phi(n) \leq \min\left\{ \phi^*(n), \frac{a_0 \cdot L(n)}{S} \right\}.
\]

Let \( n_2 \) denote the crossing point of these two bounds. The combination
of the execution bound and the slow growth of \( \phi^*(n) \) caused by the I/O
bound can create a circumstance under which \( n_0 > n_1 \). Since the "plateau"
of \( \phi(n) \) is narrow, a fixed constant \( u < 1 \) is not likely to produce a robust
policy \( L = uS \) for this case.
Figure 3. Execution bound system.

Figure 4. Execution and I/O bound system.
Two independent conditions sufficient to cause $n_0 \leq n_1$ and a wide plateau in $\phi(n)$ are easily deduced. The first is that the bound $a_0 L(n)/S$ crosses $a_0 I$ with steep descent, a condition frequently encountered in practice.\(^2\) In this case $L = uS$ is a suitable policy function, and $u$ between 1 and 2 may approximate peak service rate (see below). The second is that $L_{\text{max}} > IS$ and $\phi'(n_2)$ is close to $a_0 I$. This condition is more easily achieved when there is sufficient I/O capacity, which increases the growth rate of $\phi(n)$.

The foregoing discussion may be summarized as follows. If the system is neither execution bound nor I/O bound, the criterion $L = S$ would tend to allow a load $n_1$ (slightly) larger than the optimum $n_0$. In other words, setting the memory management parameter so that $L = uS$ for some value of parameter $u > 1$, will achieve approximately maximal system service rate. If the system is I/O bound, the criterion becomes $L = uIS$. If the system is execution bound, the criterion degenerates to $L = L_{\text{max}}$. If the system is both execution bound and I/O bound, a criterion $L = uIS$ for $u < 1$ may be required, but is not likely to be robust. However, the last case is not of great practical interest anyway, since the two bounds will limit the system service rate to some maximum significantly below the best possible — the system designer has problems more important than load optimization to solve. In the cases of most interest, a simple control on the lifetime will achieve nearly the maximal throughput.

\(^2\) As an illustration, let $G(x)$ denote a program's lifetime as a function of its mean resident set size $x$. Take $L(n) = G(M/n)$ for main memory size $M$. Assuming that $n_1$ within 1 of $n_0$ is tolerable, the slope of the bound $a_0 L(n)/S$ should be at most $-1$ at $n = n_1$. Taking derivatives, this translates to the condition $G'(M/n) \leq n_1^2 S/a_0 M$. For typical values (e.g., $S = 10^4$ microseconds, $a_0 = 10^{-6}$/microsecond, $M = 100$ pages) this condition is easily satisfied by
EXPERIMENTS

Using simple queueing network models, we conducted a series of simulation experiments to explore the robustness of three optimality criteria. The first is the L-S criterion of this paper. The second is the knee criterion suggested by Denning and Graham [DeG75]; it asserts that operating a working set policy at the knee of its lifetime curve\(^3\) will define a load at which the system service rate is nearly its peak value. The third is the 50% criterion advanced by Leroudier and Potier [LeP75]; it asserts that, when the load is optimum, the paging device utilization will always be close to 50%.

Figures 5-11 are representative of a series of sixty simulations. We used a typical program lifetime function \(G(x)\), whose convex region was approximated by a curve \(cx^2\) for \(x \leq 30\), and whose knee was at \(x=38\) pages with lifetime of approximately 10000 references [DeK75]. For a memory limit \(Y\) pages per program (i.e., \(x \leq Y\)) and main memory capacity \(M\), we used for the system lifetime function

\[
L(n) = \begin{cases} 
G(M/n), & n \geq M/Y \\
L_{\text{max}} = c(M/Y), & n < M/Y 
\end{cases}
\]

We used a queueing network of one I/O station (a file disk) and Buzen’s method of computing utilizations [Buz73]. Figures 5-8 have \(\beta_1/\beta_2 = 2.0\).

\(^3\)A lifetime curve \(G(x)\) of a program is an increasing function specifying the mean virtual time between successive page faults for a given memory policy operating the program in mean space \(x\). For some \(x_1\), \(G(x)\) tends to be convex for \(x < x_1\) and concave for \(x > x_1\). The “knee” is a point \(x_2 > x_1\) beyond which the graph of \(G(x)\) flattens out, and is defined as the tangency point of a ray emanating from \(G(0) = 1\). See [DeK75].
so that $I=1$ (cf. (7)). Figure 8 has $S > L_{\text{max}}$ and represents an execution bound system. Figure 9 has $b_2/a_2=0.8$, giving $I=0.8$ and an I/O bound system. Figures 10 and 11 show systems which are execution bound and (almost) I/O bound.

Each figure shows the normalized service rate $\phi(n)/a_0$ (which is in this case the processor utilization), the utilization $U_1$ of the paging device, and the two bounds $I$ and $L(n)/S$ of relation (8). Each figure is marked to show the operating point generated by each control criterion. (The knee criterion point is at $n = M/38$.)

The $L=S$ criterion behaved according to expectations across the range of experiments. When the system was not execution bound, operating with $L = 1.15$ consistently defined a load whose service rate was within 2% of peak. In the execution bound system $L_{\text{max}} < S$, operating all programs at their maximum space consistently defined a load whose service rate was within 2% of peak. In the systems which were both execution and I/O bound, the criterion $L=uS$ required a value $0.6 < u < 0.9$, which was very sensitive to changes in other parameters; in these cases $\phi(n_0)/a_0$ was always significantly less than 1.

The experiments showed the knee criterion fulfilled its assertion but was less robust. As soon as $S$ exceeds the knee lifetime by a significant amount (in our case, by 40%), this criterion tends to define a load past the peak. When $S$ is significantly smaller than the knee lifetime, this criterion defines a load giving nearly the peak service rate; but this load could be much less than the maximum optimal. As expected,
LEGEND FOR FIGURES 5 TO 11

- I Bound
- L(n)/S Bound
- U (page device)
- φ(n)

Operating Points:

- □ L = S Rule
- △ Knee Rule
- ○ 50% Rule

Figure 5: M=500, L_max=12275
X_max=80, S=2000, I=1.

Figure 6: M=500, L_max=12275
X_max=80, S=2000, I=1.

Figure 7: M=500, L_max=12275
X_max=80, S=10000, I=1.
Figure 8: M=300. L_{max}=12275
\[ x_{\text{max}} = 0.5 \text{ in } \(0, 20\) \]

Figure 9: M=300. L_{max}=12275
\[ x_{\text{max}} = 0.5 \text{ in } \(0, 20\) \]

Figure 10: M=249. L_{max}=12891
\[ x_{\text{max}} = 0.5 \text{ in } \(0, 20\) \]

Figure 11: M=120. L_{max}=12891
\[ x_{\text{max}} = 0.5 \text{ in } \(0, 20\) \]
when the knee is comparable to S, the knee criterion agrees with the
criterion L=S. Interestingly, the knee criterion was the most reliable
when the system was both execution and I/O bound.

The 50% criterion appears very accurate as long as the L=S point
lies in the convex region of the system lifetime curve. However, as soon
as the L=S point falls in the concave region, the 50% rule fails. In
an empirical study of programs, Graham [Gra75] has found that 8 of 10
address traces have their lifetime function's convex regions ending
at lifetime under 3000 microseconds. Most contemporary virtual memory
systems use drums or disks for paging, for which S is at least 8000
microseconds. If Graham's programs are typical, one would not expect
to observe the 50% rule holding very often in practice. However, the
criterion could be used to test whether or not the system is operating
in its convex region without measuring the system lifetime function.

EXTENSIONS

The L=S criterion applies to a large variety of systems of the
configuration shown in Figure 1: the argument assumes only that the
rates $a_i$ and $b_i$ exist, and that all tasks submitted to a given service
station are eventually completed. Several generalizations can be made
without altering the utility of the L=S criterion.

---

4 In studying this property, Leroudeix and Potier [LeP75] assumed that
$L(n) = c(M/n)^k$, which is everywhere convex for their choices of $k$. That
the 50% property may not hold outside the convex region does not contra-
dict their findings.
One generalization permits load-dependent service rates for the nonprocessor stations \((1, \ldots, m)\). Make the reasonable assumption that the service rates \(b_i(n)\) are nondecreasing in \(n\). Assume as before that the paging request rate \(a_1(n)\) is increasing in \(n\) while the I/O request rates \(a_2, \ldots, a_m\) are independent of \(n\). Define

\[
I(n) = \min \left\{ 1, \frac{b_2(n)}{a_2}, \ldots, \frac{b_m(n)}{a_m} \right\}
\]

and note that \(I(n)\) is increasing in \(n\). The service rate bound is now

\[
\varphi(n) \leq a_0 \min \left\{ I(n), \frac{L(n)}{S(n)} \right\}.
\]

Under the reasonable assumption that lifetime decreases more rapidly than paging service time, there will exist a unique crossing point load \(n_1\) at which

\[
L(n_1) = I(n_1) S(n_1),
\]

with the same interpretation as before. Even if \(I(n_1) = 1\), the load control in this case is more difficult to implement, since the load dependent page service rate must be used.

Another generalization assumes that the service rates \(b_i\) are constant but all the request rates \(a_i\) are load dependent. In this case we take

\[ S_i = 1/b_i \text{ and } L_i(n) = 1/a_i(n). \]

The service rate bound is

\[
\varphi(n) \leq a_0 \min \left\{ L_1(n)/S_1, \ldots, L_m(n)/S_m \right\}.
\]

Assume the system is not I/O bound, i.e., \(L_i(n) > S_i\) for small \(n\) and all \(i\). In this case \(\varphi(n)\) will attain its peak value at \(n_0 < n_1\), where \(n_1\) is the least \(n\) for which \(L_i(n) \geq S_i\) for all \(i\). In other words, the control must apply the criterion to the device \(i\) for which \(L_i = S_i\) at the smallest load; this need not be the paging device.
Both generalizations can be combined to yield an L=S criterion for cases where both service rates and request rates are load dependent.

Some systems of the form of Figure 1 are based on time slicing and swapping: that is, a complete program is swapped into main memory at the start of a quantum and out again at the end, without paging occurring in the interim. In this case the L=S criterion is interpreted to mean that the quantum should be slightly longer than the swapping time. This, however, may not be feasible since the abundance of short transactions in many such systems makes them execution bound. In such cases the L=S criterion degenerates to the policy usually implemented: grant each transaction program whatever space it requires, and fit as many such programs as possible into memory.

CONCLUSIONS

Figure 12 is a summary of the results of this paper, showing the control rule that will maximize system service rate and approximate an optimal degree of multiprogramming for various combinations of execution and I/O bounds. Except when the system is both execution and I/O bound, an L=S rule will be stable; and when no such rule is stable, the knee criterion can be used. The shaded area suggests the possibility that an L=S rule may be unstable in a system on the verge of being both execution and I/O bound.
Figure 12. Summary of control criteria.
An interesting property of systems controlled by a policy $L_uS$ is that system lifetime will be approximately $uS$ for all main memory sizes $M$ larger than that required to permit one program to achieve lifetime of $uS$. (Likewise, in the case considered in footnote (1), the lifetime at the optimum is a constant independent of memory size.) This does not contradict Saltzer's observation of linear growth in $L$ as a function of $M$ [Sal74]. In Saltzer's experiment, some of the (execution bound) programs sharing memory were inactive, belonging to terminals in "think state"; references to the paging device are thus decreased for larger $M$ because larger numbers of programs making transitions into "active state" require no paging. In our case, all the programs sharing memory are active, and increasing $M$ permits more such programs to be resident.
BIBLIOGRAPHY


[Mor72] Morris, J. B., "Demand paging through the use of working sets on the Maniac II," Comm. ACM 15, 10 (October 1972), 867-872.


