R.S. Gaines Editor

VMIN—An Optimal Variable-Space Page Replacement Algorithm

Barton G. Prieve
Bell Laboratories, Naperville
R. S. Fabry
University of California, Berkeley

A criterion for comparing variable space page replacement algorithms is presented. An optimum page replacement algorithm, called VMIN, is described and shown to be optimum with respect to this criterion. The results of simulating VMIN, Denning's working set, and the page partitioning replacement algorithms on five virtual memory programs are presented to demonstrate the improvement possible over the known realizable variable space algorithms.

Key Words and Phrases: demand paging, performance measurement, multilevel memory systems, virtual memory, working set, page replacement algorithms, optimal page replacement

CR Categories: 4.30, 4.32

A version of this paper was presented at the Fifth ACM Symposium on Operating Systems Principles, The University of Texas at Austin, November 19–21, 1975.

Authors' addresses: B.G. Prieve, Bell Laboratories, Naperville, IL 60540; R.S. Fabry, Computer Science Division, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

Introduction

One of the earliest papers on virtual memory computer systems [1] contained a description of an optimum page replacement algorithm called MIN. MIN causes the fewest possible page faults for a program when executing in a *fixed* number of main memory page frames. Although MIN is unrealizable (since it requires knowledge of the future portion of the page trace), it has been valuable as a benchmark for realizable fixed-space algorithms [1–4]. For page replacement algorithms such as Denning's working set (DWS) [5] and page partitioning [6] which are variable-space replacement algorithms, MIN does not represent an adequate optimum algorithm for comparisons. In fact, in [7], both of these variable-space algorithms are shown to outperform MIN. Thus MIN is *not* optimal among variable space algorithms.

Comparing Variable-Space Algorithms

Fixed-space replacement algorithms can be compared in terms of their page fault rates at equal memory sizes. For comparing variable-space algorithms, we choose to compare page fault rates at equal *average* memory sizes. In order to avoid results which depend on the time it takes to process a page fault, we compute the average memory size in what has been called *process* or *virtual* time, which is done by using the memory size at each page reference time. We generate *performance points* for an algorithm in the average-memory-size/page-fault-rate plane and connect them to get a *performance curve* for the algorithm applied to a particular page trace. As an illustration, the performance points for DWS can be generated by simulating that algorithm on a page trace with several values of the working set window size. Clearly, an algorithm whose performance curve lies nearer the origin than another's is the better performer.

It should be noted that this criterion, which uses a combination of memory size and page faults, was selected because of its ease of calculation and because it leads to a straightforward optimum variable space algorithm. One might consider using other statistics of the memory size, such as its median or maximum. The time-space product criterion (in which time is taken to be real time) could also be used [8, 9]. An efficient way to determine the replacement decisions of an optimum replacement algorithm given the time-space product as a criterion is not as yet known. However, given two mild assumptions it can be shown that our criterion is equivalent to the time-space criterion in the following sense. If two replacement algorithms run on a page trace produce the same average memory demand and the page fault rate of the first is greater than the page fault rate of the second, then the time-space product of the first will also be greater than the time-space product of the second, and conversely. Two assumptions which are sufficient to establish this equivalence are:

1. The real time to process a page fault is independent of the memory allocation and has the same mean for both algorithms.
2. For the two algorithms that are being compared, the average memory size computed in virtual time is equal to the memory size averaged at page fault times.

Most studies using the time-space criterion assume constant processing time for page faults. The main advantage of the time-space product over the criterion we have chosen is that it recognizes that page faults become more expensive as the amount of main memory devoted to a job increases. As important as this advantage may appear, our criterion seems to approximate time-space closely enough to provide much interesting information about the relative performance of replacement algorithms.

A Variable-Space Optimum Replacement Algorithm—VMIN

We now describe a replacement algorithm called VMIN which has the property that no other page replacement algorithm can outperform it according to the performance criterion we have chosen. After each page reference, VMIN removes the page just referenced if and only if the page will not be referenced again for more than R/U time units, where R is the cost of a page fault and U is the cost of keeping one page in main memory for one reference time.[1] Thus, R/U is the cost of encountering a page fault relative to the cost of keeping a page in main memory for one reference time, and is assumed to be greater than zero. R and U were introduced in connection with the page partition replacement algorithm [6, 7]. By varying R/U, a performance curve for VMIN can be generated.

We show that VMIN is optimal by exhibiting a line in the average-memory-size/page-fault-rate plane for each value of R/U below which *no* replacement algorithm can have a performance point, and upon which VMIN's performance point lies.

[1] One can view VMIN as a DWS type algorithm which uses a forward-looking window of length R/U.

Fig. 1. VMIN excluded zone.

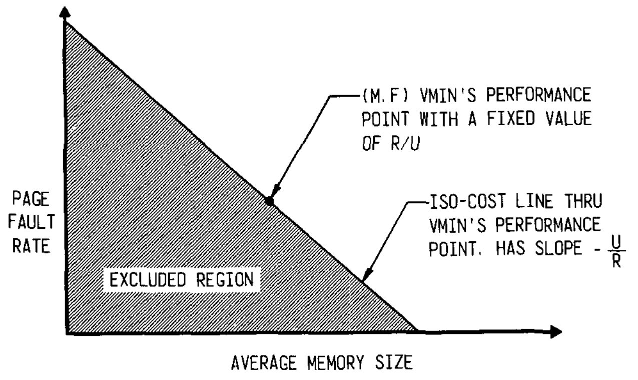
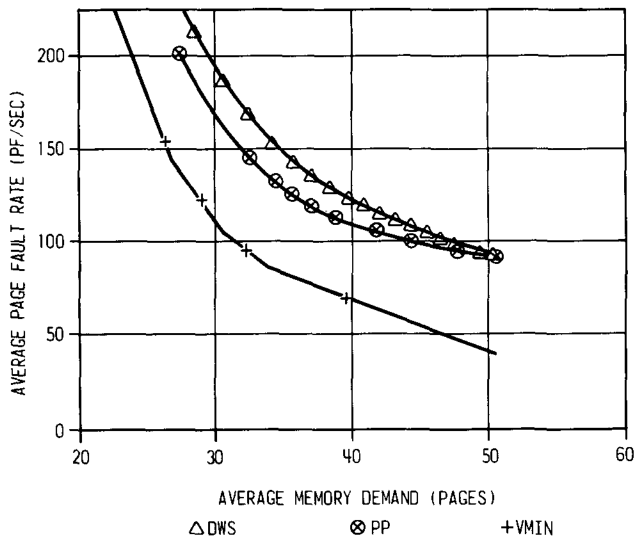

(M.F) VMIN'S PERFORMANCE POINT WITH A FIXED VALUE OF R/U
ISO-COST LINE THRU VMIN'S PERFORMANCE POINT. HAS SLOPE $-\frac{U}{R}$
PAGE FAULT RATE
EXCLUDED REGION
AVERAGE MEMORY SIZE

Fig. 2. Performance curves for VMIN, Denning working set, and page partitioning on five reference traces. (1 sec ≃ 1,000,000 memory references).



For any replacement algorithm run on a page trace, $p_1, p_2, \cdots, p_n$, define the "cost" to be:

$$C = nFR + nMU$$

where F is the page fault rate (nF is the total number of page faults), and M is the average memory size (nM is the sum of the number of pages the algorithm keeps in main memory at each reference time). This cost can be rewritten as the sum of the cost of the initial page fault for each page and the cost due to the page in the interval following each reference to it:

$$C = P \times R + \sum_{i=1}^{n} c_i$$

where the page trace contains P unique pages and c_i is defined below.

Page p_i in the page trace is either referenced again before it is removed from main memory or it is removed from main memory before it is referenced again.

If p_i is referenced next at reference j and is not removed between reference i and reference j, define the time between references $x_i = j - i$. (Thus $p_i = p_{i+x_i}$.) If p_i is removed before its next reference, let z_i be the number of references which occur after reference i before p_i is removed. (Thus p_i is removed just after the reference to p_{i+z_i}.) Using this terminology:

$c_i = U \times x_i$	If page p_i is referenced again before it is removed from memory.
$= U \times z_i$	If page p_i is never referenced again.
$= U \times z_i + R$	If page p_i is removed from memory before being referenced and p_i is referenced again.

Thus, the cost C of a replacement algorithm on a page trace can be determined by summing the costs of each page's inter-reference intervals, represented by the c_i's. By definition, $c_i \geq 0$. VMIN has been defined to minimize each c_i, and therefore has a cost C_{VMIN} which is less than or equal to the cost of any replacement algorithm on the same page trace. (Note that for VMIN, all z_i's occurring in the definition of c_i will be zero, since VMIN either removes page p_i *immediately* after being referenced or leaves it in main memory until its next reference. Also notice that VMIN's replacement decision after each reference is independent of any past or future decision; thus the c_i's can be minimized independently.)

If F is the page fault rate for VMIN on a page trace and M is the average memory size, then $C_{\text{VMIN}} = nFR + nMU$. The line given by:

$$C_{\text{VMIN}}/n = fR + mU$$

in the average-memory-size/page-fault-rate plane $m-f$ passes through VMIN's performance point (M, F) and defines an "isocost" line. Any replacement algorithm with its performance point on this line will have a cost equal to C_{VMIN}, while any algorithm with a performance point above this line will have a cost greater than C_{VMIN}. Similarly, any algorithm with a performance point below the isocost line would have a smaller cost than VMIN. Since no algorithm can have such a cost, no performance point can lie below the isocost line. Therefore, we have established an excluded area as illustrated in Figure 1.

By varying R/U, one obtains a number of performance points for VMIN and a number of isocost lines below which no algorithm's performance point can lie. The results of superimposing all the excluded regions results in a concave excluded region such as is illustrated by our experimental results.

Experimental Study

A previous paper [7] described comparisons of several variable- and fixed-space algorithms. The same program reference data, collected from substantial execution periods, were used in this study. Interval reference sets, a compact representation of a program's referencing patterns, were used. Five well-established and heavily used virtual memory programs running under TSS/360 were monitored. Details of this procedure are given in [6]. VMIN simulation results are illustrated in Figure 2, with DWS and page partitioning also exhibited. R/U was set at 50,000 to get the leftmost point on the VMIN curve and at 350,000 to get the rightmost point on the curve. From this and the average memory sizes at these points one can approximate the average page fault processing times required if one uses time-space product as the measure of performance. At the leftmost point a page fault ties up an average of approximately 25 pages of memory so it should be processed in about 2000 memory reference times; at the rightmost point 50 pages are tied up by a page fault so it should be processed in about 7000 memory reference times.

Conclusion

VMIN, a replacement algorithm that minimizes page faults at a given average memory size, has been derived and shown to be an optimal variable-space page-replacement algorithm for the comparison method used. A simulation based on measured program behavior compares two realizable algorithms with the theoretical optimum. For the page traces examined, the two realizable algorithms result in at least 50 percent more page faults than VMIN for a given average memory demand.

Acknowledgment. Our presentation has benefited from the comments of a number of colleagues who read drafts of this paper.

References

1. Belady, L.A. A study of replacement algorithms for a virtual-storage computer. *IBM Systems J. 5*, 2 (1966), 78–101.

2. Coffman, E.G., and Varian, L.C. Further experimental data on the behavior of programs in a paging environment. *Comm. ACM 11*, 7 (July 1968), 471–474.

3. Brawn, B.S., and Gustavson, F.G. Program behavior in a paging environment. AFIPS Conf. Proc., Vol. 33, 1968 FJCC, AFIPS Press, Montvale, N.J., 1968, pp. 1019–1032.

4. Thorington, J.M., and Irwin, J.O. An adaptive replacement algorithm for paged-memory computer systems. *IEEE Trans. Comp. C-21*, 10 (Oct. 1972), 1053–1061.

5. Denning, P.J. The working set model for program behavior. *Comm. ACM 11*, 5 (May 1968), 323–333.

6. Prieve, B.G. A page partition replacement algorithm. Ph.D. Th., U. of California, Berkeley, Dec. 1973.

7. Prieve, B.G., and Fabry, R.S. Evaluation of a page partition replacement algorithm. *Comm. ACM* (to appear).

8. Belady, L.A., and Kuehner, C.J. Dynamic space-sharing in computer systems. *Comm. ACM 12*, 5 (May 1969), 282–288.

9. Chu, W.W., and Opderbeck, H. The page fault frequency replacement algorithm. AFIPS Conf. Proc., Vol. 41, 1972 FJCC, AFIPS Press, Montvale, N.J., 1972, pp. 597–609.