M.I.T. Laboratory for Computer Science
Computer Systems Research Division

February 13, 1976
Request for Comments No. 106

ON THE MODELING OF PAGING ALGORITHMS

by J. H. Saltzer

At the recent ACM 5th Symposium on Operating Systems Principles (November, 1975, in Austin, Texas) quite a bit of public and private discussion centered on the value of continued research in certain theoretical areas. One of the areas questioned was the mathematical modeling of page replacement algorithms. The skepticism ranged from vague uneasiness to specific complaints, but firm conclusions never quite emerged.

I would like to suggest that the problem is not that mathematical modeling is necessarily a bad idea, but rather that some modelers have been led astray by mathematical necessity coupled with lack of familiarity with typical operating conditions. The combination, as would be expected, yields results that are ignored by practical designers.

To give some specific examples:

1) In order to achieve mathematical tractability, some paging models assume that the available memory is firmly partitioned among the jobs being multiprogrammed, while others treat dynamic adjustment of partition boundaries as an explicit parameter available to the designer of the page selection algorithm. These assumptions lead to analysis of algorithms that select for removal pages belonging to the currently faulting program, or that occasionally steal pages from other programs as a way of moving the partition boundaries. In many real multiprogrammed systems that use paging

(e.g., Honeywell's Multics, BBN's TENEX, IBM's VM/370, TSS/360, and MVS, Michigan's M.T.S., M.I.T.'s I.T.S., Digital Equipment's TOPS-20, Lincoln Laboratories' APEX, and MITRE's VENUS) memory is not so explicitly partitioned. Selection of pages for removal in these systems is done across pages belonging to all programs in the memory, and if any inquiry is made as to which program owns a page, it is usually to try to avoid removing pages of the program that had the fault. The result is an environment for page selection based on interleaving of reference strings of several jobs; that interleaving depends in turn on details of the multiprogramming strategy. In some systems, the multiprogramming strategy gives preferential attention to one program at a time; from the point of view of the currently preferred program, it is almost as if it had the entire memory available without competition. On the other hand, the non-preferred programs run in an environment consisting of "leftovers" from the preferred one. The effect of these strategies on paging rates is not clear, but it seems that for modeling to be useful, the model must include the multiprogramming strategy.

2) Much modeling is based on an assumption that a program reference string and a page selection algorithm are operating in some kind of steady-state situation with a shortage of memory. The page selection algorithm tries to minimize its losses by choosing, from among pages still in use, the page not needed longest. The operating conditions inside a typical interactive time-sharing system can be very

different from that model. Many missing-page faults are caused either by the initial "swapping in" of a program, or by the processing of a stream of data by that program. Conversely, many removals are of "dead" pages, belonging to programs no longer running, or data streams that have been processed. Only sometimes (in the case of Multics, something under 5% of the time) do missing pages turn out to be re-retrievals of "live" pages recently pushed out by the current program or one of its competitors. The proprietor of a system that uses paging quickly learns to "tune" the system (perhaps by adjusting the level of multiprogramming) so that extreme scarcity of memory space is <u>not</u> the order of the day. Otherwise much of the system's resources would be used up pushing out, and then re-retrieving, still useful pages. It is this combination of phenomena that most probably accounts for the lack of a sharp "knee" in the plot of observations of mean running time between page faults, versus main memory size, reported for Multics.

3) Some models do not take into account the time spent executing the page selection algorithm itself, and they take as their measure of excellence the smallness of the number of missing page faults obtained in processing a given memory reference string. Under real operating conditions these two assumptions can be counter-productive. Real memory size is probably best chosen to be large enough that multiprogramming can just use up the capacity of the processors. At that point, the total time spent executing the page selection algorithm is typically quite a bit greater than the unusable idle time after multiprogramming; in such a situation a simpler algorithm that takes

significantly less computation time (while making a less accurate choice of pages to remove, causing more missing-page faults and increased idle time) may improve overall system performance. In systems where the time penalty for a missing page is small, as in a cache memory, a faster algorithm that causes a few more cache misses may again result in higher performance. The observation earlier that many page replacements are of "dead" pages further suggests that precise choices may not be important; the computationally fastest algorithm may do almost as well as the optimum and release some processor time to customers as well.

The ardent defender of paging algorithm modeling research will be able immediately to quote examples of data from real systems that actually do operate in the regions and under the conditions typically assumed in modeling research (small cache memories may be an example), and that defender will be correct. The issue is not whether the models ever hold, but why there is increasing uneasiness that the models are not having very wide practical impact. I think that engineers faced with practical decisions find a fairly large gap between their perception of the problems and the theoretical models, and it is for reasons like the ones listed above. Consider the likely impact of a paper in the field of management, entitled "How to minimize your losses while running an unprofitable business." The businessman hopes to operate under different, more interesting conditions, and can not be expected to pay much attention to theoretical analyses of the arrival rate of bankruptcy. Similarly, the system designer can not be expected to pay much attention to theoretical analyses of systems operating under unrealistic conditions.

Finally, one of the reasons modelers sometimes assume unrealistic conditions must be the shortage of published reports describing how real paging and multiprogramming algorithms work, and what operating conditions those algorithms are typically observed to create. As far as I have been able to determine, not one of the systems mentioned in point one, above, has had documented its paging and multiprogramming algorithms in sufficient detail, or has had reported enough statistics of typical operating conditions, to allow an isolated reader of the published literature to discover any of the three points mentioned.

My conclusion is that a prerequisite to further progress in page replacement modeling must be much more publication of observations of successful paging systems operating under realistic loads. Research-supporting agencies should note these issues and act accordingly.