To: Distribution

From: David P. Reed

Subject: A New Bound on the Performance of the Proposed New Storage System

Date: 04/19/74

In MTB-060 Steve Webber makes an argument, the essence of which is that because twice as many distinct directory pages are referenced in a given directory or segment control operation, the increase in the page fault rate on directory pages will be about doubled. In fact the situation may become a lot worse, since the new pages will displace other non-directory pages from core, reducing the effective paging pool. This note describes a technique for obtaining an estimate which explicitly accounts for this effect.

Consider the set D of directory pages, and for simplicity let us assume that there is only one type of directory operation, which in the present storage system implementation references n distinct pages in D. Further assume that this directory operation takes a short time relative to the time in between directory operations; this assumption allows us to presume that permutations of the order of page references <u>within</u> the span of the directory operation will not significantly affect the paging behavior of the system. I also assume that the number of directory pages in core at a particular instant is relatively constant over time. This latter fact has been experimentally verified by me in experiments conducted over a several hour period on a full configuration Multics system, and described later in this document.

The page reference string of the current Multics will thus have a form similar to:

$$s_i s_j s_k \cdots \underbrace{d_a s_i d_b d_b s_w d_c}_{\text{directory operation}} \cdots s_k s_l s_v$$

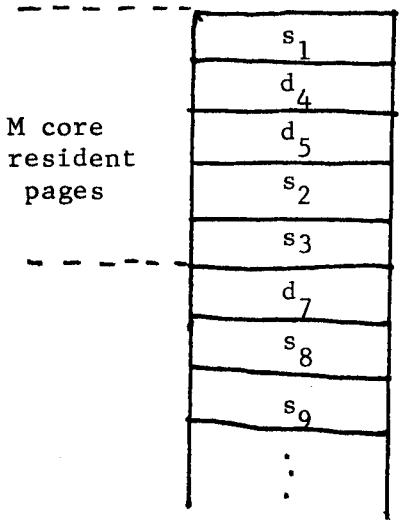
where $d_a, d_b, d_c \in D$, and $s_i, s_j, s_k \ldots \in \{\text{all non-directory pages}\}$

Directory pages are only referenced during directory operations, although non-directory pages are referenced both during directory operations and outside of the scope of directory operations.

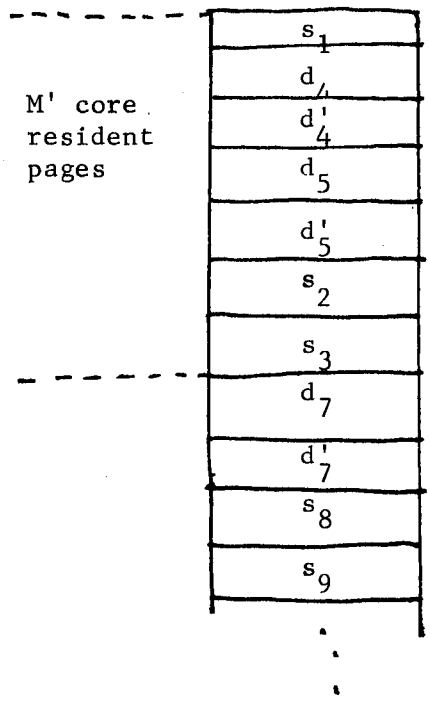
Because of our earlier assumption that permuting the order of page references within a directory operation results in no significant change in the page fault rate, we can easily produce a reference string for the proposed new storage system from the previous reference string. The change in reference pattern due to the new proposal can be approximated by assuming that a directory operation will cause a reference to twice as many distinct pages in the set D, which now has twice as many pages due to the use of two pages in the new scheme to replace one page in the old. We can thus model the new system reference string by replacing each reference to a directory page, d_a, by a pair of references to two distinct pages in D, d_a and d'_a. The new reference string corresponding to the previous example will be:

$$s_i s_j s_k \cdots \underbrace{d_a d'_a s_i d_b d'_b d_b d'_b s_w d_c d'_c}_{\text{directory operation}} \cdots s_k s_l s_v$$

Assuming core is managed by an LRU stack page replacement algorithm, the LRU stack for all pages will differ due to this change in reference string by replacing entries corresponding to a directory page by two adjacent entries in the list for the two new pages. The drawings below show sample LRU stacks for the current algorithm under a particular reference string, and the analogous LRU stack due to the transformed reference string under the proposed new storage system. To keep the analogous core boundary in both cases, I will temporarily hypothesize that the new storage system is implemented on a system with larger primary memory size M', where M' = d + M, with d being the average number of directory pages in core under the present storage system, and M being the present memory size. In this new memory size the page fault rate can easily be determined, due to the properties of the stack algorithm and the relationship of the LRU list to the LRU list in the present storage system.



Under Present Storage System



Under Proposed Storage System

How can we compute the page fault rate in memory of size M' for the proposed algorithm? Let p(M) be the page fault rate of the system under the present algorithms, p'(M") be the page fault rate of the system under the proposed algorithm in the larger memory M'. We can see the page fault rate on non-directory pages in the new scheme in the larger memory will be the same by the fact that we are using a stack algorithm for page replacement, so let's define $p_{nd}(M)$ to be the page fault rate on non-directory pages in the present algorithm in memory M, and $p'_{nd}(M')$ be the analogous rate in the new system. Then,

$$p_{nd}(M) = p'_{nd}(M').$$

Since we will bring in two directory pages under the new scheme where one will have been brought in before, we can determine the paging rate on directory pages in both systems, $p_d(M)$ and $p'_d(M')$, to be related by a ratio of two:

$$p'_d(M') = 2\ p_d(M).$$

We know that the system paging rate in both cases is the sum of the appropriate directory paging rate and non-directory paging rate:

$$p(M) = p_d(M) + p_{nd}(M)$$

$$p'(M') = p'_d(M') + p'_{nd}(M')$$

So we can relate the paging rate in the new proposed system with M' blocks of memory to the paging rate in the present system as follows:

$$p'(M') = 2\ p_d(M) + p_{nd}(M)$$

or

$$p'(M') = p(M) + p_d(M).$$

This result is very similar to Webber's supposition; the problem is that we have obtained the paging rate in a larger memory. To scale down the memory size, we must use some model which relates paging rate to memory size. Saltzer's linear model seems appropriate since it has been shown to fit for the memory sizes involved. According to Saltzer's model,

$$p'(M')\ M' = p'(M)\ M,$$

or

$$p'(M) = \frac{M'}{M}\ p'(M').$$

Bringing it all together,

$$p'(M) = \frac{M + d}{M}\ (p(M) + p_d(M)).$$

In order not to have to worry about actual paging rates and memory sizes, the previous result can be normalized to express the percentage change in performance.

$$\frac{p'(M)}{p(M)} = \frac{(M + d)}{M}\ \left(1 + \frac{p_d(M)}{p(M)}\right)$$

If we let β be the proportion of page faults which are on directory pages in the current system, and γ be the proportion of pages in core which are directory pages in the current system, we get a very simple result:

$$\frac{p'(M)}{p(M)} = (1 + \gamma)(1 + \beta)$$

where

$$\gamma = \frac{d}{M}, \quad \beta = \frac{p_d(M)}{p(M)} .$$

The point of all this is that the paging behavior is the product of two terms, the first of which reflects the change in the size of the paging pool, and the second of which reflects the extra paging references. The percentage increase can be seen to be: $\beta + \gamma + \beta\gamma$. Thus for example, if 20 percent of core is taken up by directory pages and 2 percent of the paging rate is concerned with directory pages, there will be about a 22 percent change in paging rate. Thus it is obviously important to take both effects into account.

Application to the Actual Multics System Performance

Webber assumes that β is somewhere around 4 percent. I will assume this since I cannot get any useful figures on this from the system. The actual value of β might be significantly lower, but we will see that this does not matter much.

I have measured γ myself, by dumping consistent copies of the AST and analyzing them with my own AST analysis program, to see how many directory pages are present in the paging pool in core. Over a 2 hour period I obtained the result that out of about 210 pages in core available for paging, 20 of those pages were directory pages at any particular instant. This number did not vary significantly; on 45 samples I got a standard deviation of about 7 pages. Consequently, γ for that system is about 10 percent. Consequently, we can see that percentage increase in page faults is about 14 percent, and a ball park estimate of the change in system throughput in this case would be about 7 percent. This number is large compared to Webber's 2 percent.

As a result of this measurement, it would be well worth while taking a fresh look at the assumptions underlying the new storage system proposal.