W. G. Tuel, Jr.

An Analysis of Buffer Paging in Virtual Storage Systems

Abstract: Storage buffers are often used to hold temporary results or data that may be re-referenced in the near future. If these buffers are pagable, searching the buffer may cause a high number of page exceptions. A model of this phenomenon is postulated, and compared with experimental data.

Introduction

Storage buffers are often used to hold temporary results or data that may be re-referenced in the near future. In data base systems such as IBM's Information Management System [1], the buffers contain the most recently used blocks of data retrieved from auxiliary storage. Subsequent requests for data are answered by first searching the buffer, returning the requested data if therein contained; if the data are not in the buffer, the record or block is retrieved directly from the auxiliary storage device. The hope is that data will be found in the buffer fairly often, and that the time taken to search the buffer is more than offset by the time saved by not having to retrieve the data from auxiliary storage.

Buffers of the type described are often rather large-ranging from 20000 to 150000 bytes or more; the upper limit is usually set by explicit constraints on the real storage available, especially in systems without virtual storage management. In systems with virtual storage support, there is a temptation to increase the size of the buffer, exploiting the large address space provided. However, this sets up a double paging environment [2], and is self defeating because of the buffer paging induced by the search process. In the next two sections, a model for this phenomenon is postulated, and an analysis of the total paging and data read I/O activity is presented. The section entitled Empirical confirmation presents the results of a set of experiments demonstrating the phenomenon.

Buffer Management Model

Consider a buffer of N pages which is to be searched whenever a request for data is made. If the requested item is not in the buffer, an explicit data read is performed to retrieve the record. Assume further that M pages of main memory are available for the buffer.

Let $p(i)$ be the probability that the requested item is located in the ith buffer page searched, $i = 1, \cdots, N$. (Independent, identically distributed data requests are assumed. It is also assumed that these probabilities are independent of the buffer size N, and independent of the identity of the page searched.) Let $Q(N)$ be the probability that the data are not found in the buffer, i.e., that there is a "miss" to the buffer. Clearly,

$$Q(N) = 1 - \sum_{i=1}^{N} p(i). \tag{1}$$

Finally, assume that the probability s that buffer page i is in main memory is independent of i and is given by

$$s = 1 \text{ for } N \leq M$$

$$= M/N \text{ for } N > M.$$

This latter assumption states that buffer pages are assigned randomly in main memory.

It might be argued that these two assumptions are incompatible, because most virtual storage systems would allocate main memory pages to the first portion of the buffer pool, thus making the page fault probability nonuniform. While this may be true to a certain extent, the true situation is not simple to analyze, for a number of reasons: The VS/2 paging supervisor uses an available page list, rather than the LRU replacement algorithm; the buffer pool page stack is updated by scanning and altering the stack in software, rather than by hardware-use bits; and the actual unit of transfer from the data base disk to the buffer pool is variable length, thus requiring buffer pool garbage collection and compaction techniques which are not explicitly modelled. Because the effect of these perturbations is unknown, uniform probability was used.

Give the above assumptions, the expected number of page faults, given that K buffer pages are searched, is

$$f(K) = K(1 - s) \tag{2}$$

and the expected number of page faults per search is

$$F = \sum_{k=1}^{N} f(k)p(k) + f(N)Q(N)$$
$$= (1-s)\left[\sum_{k=1}^{N} kp(k) + NQ(N)\right],$$

where the second term above is due to searching the buffer and *not* finding the requested data. This expression may be further simplified using (1) to give

$$F = (1-s)\sum_{k=0}^{N-1} Q(k). \tag{3}$$

The expected number of explicit data reads per search is given by $R = Q(N)$. Thus, the total I/O activity

$$T(N) = R + F$$
$$= Q(N) + (1-s)\sum_{k=0}^{N-1} Q(k) = Q(N) \text{ for } N \le M,$$
$$= Q(N) + [(N-M)/N]\sum_{k=0}^{N-1} Q(k) \text{ for } N > M. \tag{4}$$

For $N > M$, the change in total activity as the buffer size is increased is:

$$\Delta T(N) \equiv T(N+1) - T(N)$$
$$= Q(N+1) + [M/N]\sum_{k=0}^{N-1} Q(k)$$
$$- [M/(N+1)]\sum_{k=0}^{N} Q(k)$$
$$= Q(N+1) + \left[\frac{M}{(N+1)N}\right]$$
$$\times \left\{\sum_{k=0}^{N-1} [Q(k) - Q(N)]\right\} > 0, \tag{5}$$

since $Q(k)$ is monotonically decreasing as a function of k.

Thus, the total I/O activity is an *increasing* function of the buffer size for $N > M$, and making the buffer larger than the available real storage causes an increase in total I/O work needed.

Extension for parametrized data base miss ratios

Equations (3)-(5) represent general expressions for expected number of page faults and total I/O activity. It is difficult to estimate, however, the magnitude of the effect of double paging without a more specific characterization. The empirical confirmation data in the next section suggest that $Q(N)$ may be modeled by a sum of exponentials. Let

$$Q(N) = 1 - \sum_{i=1}^{r} b(i)[1 - a(i)^N],$$

where $b(i)$ and $a(i)$, $i = 1, \cdots, r$ are parameters, $0 < a(i) < 1$. Define $Q_\infty = 1 - \Sigma_{i=1}^{r} b(i)$. Equation (4) becomes

$$F = (1-s)\left[NQ_\infty + \sum_{i=1}^{r} \frac{b(i)}{[1-a(i)]}[1 - a(i)^N]\right]$$
$$= (N-M)Q_\infty + \frac{(N-M)}{N}\sum_{i=1}^{r} \frac{b(i)}{[1-a(i)]}$$
$$\times [1 - a(i)^N] \text{ for } N > M$$
$$= 0 \text{ for } N \le M. \tag{6}$$

For large N and $Q_\infty \neq 0$, T increases linearly with N. If $Q_\infty = 0$, (in which case all references are contained in the buffer), T increases to an asymptotic value

$$T_\infty = \sum_{i=1}^{r} \frac{b(i)}{[1-a(i)]},$$

which represents entirely paging activity.

Empirical confirmation

A set of experiments was conducted to demonstrate the phenomenon described above, and to assess how well the model of the previous two sections explains the observations. The application consisted of a batch program executing a sequence of 1100 queries against a data base management system, IMS Version 2.4 [1]. IMS manages a buffer pool similar to that described above. Its size is adjustable by the user at each invocation of the application program. The queries caused a total of 9700 searches of the buffer, an average of 8.8 searches/query. The operating system environment was VS/2 Release 1.6, and the machine an IBM System/370 Model 145 with 512K bytes of main memory.

Page exceptions for the IMS functions and buffer pool searching were measured using a hardware counter gated by a signal turned on when IMS is invoked. Data on total reads to the data base were obtained from the IMS-maintained buffer statistics.

Figure 1 shows the measured average number of data base reads/search, $Q(N)$, and the measured average number of page exceptions/search, F, as a function of the specified buffer pool size. The smooth curve is the result of fitting $Q(N)$ to the sum of two exponential terms, and obtaining the corresponding values of $a(1)$, $a(2)$, and $b(1)$, $b(2)$.

Using 4096 bytes (4K) as 1 page, the empirical fit for $Q(N)$ is

$$Q(N) = 1 - b(1)[1 - a(1)^N] - b(2)[1 - a(2)^N],$$

where

$$Q_\infty = 0.1406,$$
$$b(1) = 0.67726, \quad a(1) = 0.3742,$$
$$b(2) = 0.18214, \quad a(2) = 0.9586.$$

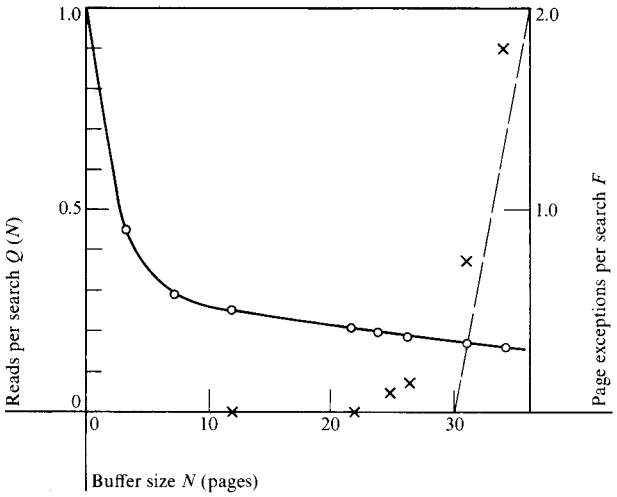

Figure 1 Average number of reads per search and average number of page exceptions per search as a function of specified buffer pool size.

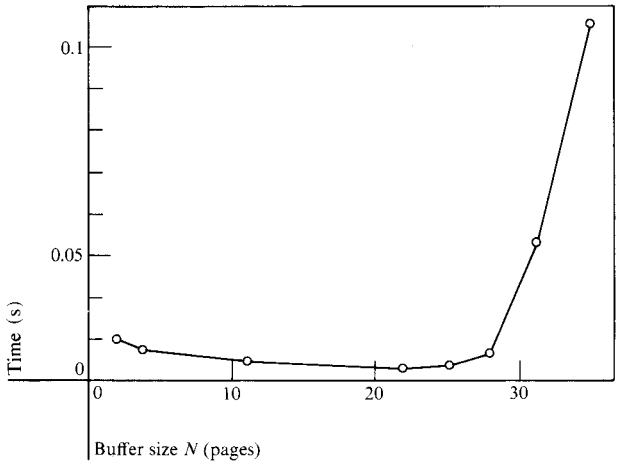

Figure 2 Average elapsed time per search as a function of buffer pool size.

Equation (6) then becomes

$$F = \{.1406 + N^{-1}[1.074(1 - 0.3742^N) + 4.41(1 - 0.9586^N)]\}(N - M) \quad \text{for } N > M.$$

This is indicated by the dashed line in Fig. 1, for a value of $M = 30$ pages.

Figure 2 presents the average elapsed time/search as a function of buffer pool size. These values reflect the total I/O activity. It is seen from both figures that the paging effect is highly significant, and that performance substantially deteriorates as the buffer pool size is increased past $N = M$.

The agreement between the calculated and observed values of page exceptions in Fig. 1 is good, allowing for the fact that the threshold value, M, is not known accurately. Note that only buffer paging is modeled and measured; total paging activity also includes paging of program code. In the experiment described, the latter adds about 30% to the total page exception count.

Fitting a functional form to $Q(k)$ is not always required. Equation (3) indicates that semi-graphical methods for summing observed values would also be appropriate.

Conclusions

A model for buffer paging in a virtual storage system has been presented. It explains the observed sudden increase in paging activity and elapsed time for a query when the specified buffer size exceeds the number of page frames available.

Analysis of the model indicates that the total I/O activity increases with increasing buffer size if the buffer is allowed to page. Thus, optimum performance is achieved by reducing the buffer size to fit the number of page frames available. This is the strategy presently used for IMS/360 running under OS/VS.

Acknowledgment

The author gratefully acknowledges the technical assistance of D. Hildebrand, R. Krampetz, and M. Smyly in providing the data base environment and measurement facilities for this study.

References

1. *Information Management System/360, Version 2, General Information Manual*, GH20-0765, IBM Corporation, White Plains, NY 1973.
2. Goldberg, R. and Hassinger, R., "The Double Paging Anomaly," *Proc. 1974 National Computer Conference*, Chicago, May 6-8, 1974.

Received February 2, 1975; revised March 4, 1975

The author is located at the IBM Research Division Laboratory, 5600 Cottle Road, San Jose, CA 95114.