may be several processes (i.e., transactions or applications) active concurrently. These properties require that reorganization be performed by a sequence of page permutations between blocks referenced by a process. The third section treats the problem of structuring the clustering algorithm as a set of independent tasks, one corresponding to each process. The section entitled "A permutation clustering algorithm" provides a specific implementation of these ideas: an algorithm which performs reorganization through a sequence of exchanges or transpositions of pages between blocks. Subsequently experiments are described which were performed on a trace taken from the AAS system. The results indicate that dynamic clustering can result in organizations which yield improved performance. This is especially interesting in view of the fact that the AAS system is one in which the application programs all have the same logical view of the data base and logically consecutive records are most often physically consecutive. That is, the application programs were designed with the knowledge of the method of organization.

Principles and requirements

Consider the process of transfers between main memory L_2 and lower levels. The unit of transfer into L_2 will be assumed to consist of a *block* of $q_2 = N$ pages which reside on contiguous space in L_3. When a processor requests data not currently in main memory, a *page fault* is said to occur. The page containing the missing information, and those other pages belonging to the same block which are not currently in L_2, are brought into main storage. The page replacement algorithm frees enough space in L_2 to hold the new pages.

The above discussion assumes the existence of a directory for the system. If pages are subject to relocation between blocks, the directory must have an entry for each page. A possible format is:

Page name	Block name	Address of block

The block containing a particular page has sometimes been referred to as the page *home address*.

When a page is requested, the directory provides the name and address of the home address block. Reorganization involves the movement of pages to different blocks. This may be done in a wide variety of ways. However, it is worth considering to what extent the clustering policy may be influenced by system considerations and a requirement for implementational simplicity. It will be seen that these do much toward determining a class of suitable techniques.

A primary requirement is that the clustering method should not require additional fetches from secondary storage. Thus

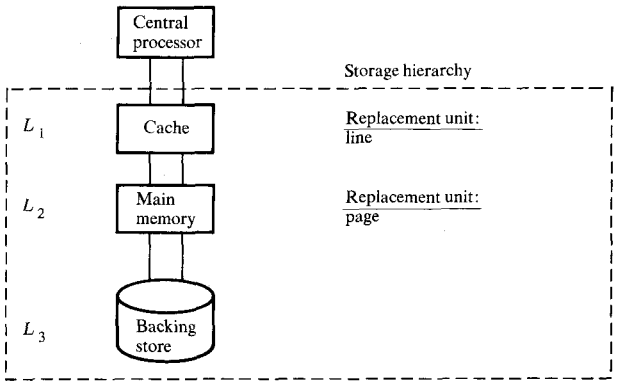

Figure 1 A three-level storage hierarchy.

1. *Only those pages currently in main memory should be considered for reorganization.*

Note that it is necessary to consider the initiation and termination of a reorganization. For example, the technique could be applied to a data base which is used only part of the time. Suppose the process starts at time t_0 and continues until t_1. A desirable property for the new organization is that it should not require an increased amount of storage on L_3. In other words,

2. *The number of blocks should not be increased.*

If empty blocks and empty page slots are regarded as always being present in main memory, then subject to block renaming, requirement 2 is equivalent to stating that reorganization must be a permutation of pages among blocks allocated to the data. Note that this includes cases where the number of such blocks decreases.

At any time t, a number of processes may be active, perhaps referencing disjoint sets of information. Requirement 2, coupled with the observation that an attempt should be made to separate sets of data never used by the same process, suggests the following condition:

3. *Pages should be permuted only between blocks referenced by the same process.*

A process may be terminated at a time which cannot be predicted in advance. When such termination occurs, it is desirable to be able to conclude the reorganization of the data it is referencing. The following restriction ensures that termination of the clustering process may be done rapidly while adhering to requirements 1-3.

4. *The reorganization of a set of blocks associated with a given process should be structured as a series of page permutations, each involving as few pages as possible.*

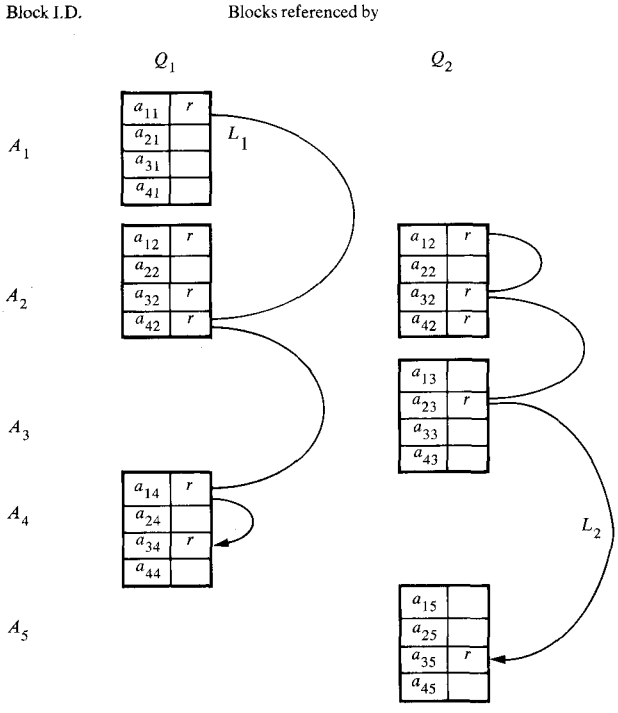

Figure 2 Contents of a main memory for the case of four pages per block.

Note that a new organization results from each such permutation, so that the clustering process may be halted without incurring much overhead. The class of algorithms discussed in the next section operate by means of a sequence of page exchanges or transpositions, each involving only two pages.

Permutation clustering by independent tasks

The discussion in the previous section suggests that reorganization should be performed by a procedure that may be termed "permutation clustering." This section considers the structure of the procedure in more detail.

A way to implement requirements 1 through 4 is to formulate the clustering as a number of tasks $\{C_k\}$, where each C_i operates on a set of blocks or pages associated with a single process Q_i. These tasks will then permute pages between blocks. But suppose the system is such that there are likely to be a number of processes referencing the same information. There then arises the possibility of interference. The tasks may be designed to cooperate; that is, each could be given information concerning which pages have been or are to be permuted by the others. However, if the number of such tasks is large, then the complexity of such cooperation may be great. One way to avoid this problem is to assign each page resident in main memory to at most one task for the purpose of reorganization. This is the approach taken below. The result is that the tasks become independent.

Consider the data to be used as a basis for the reorganization. Such data may be of two kinds: that which pertains to currently active processes, and that which represents earlier reference patterns. The procedure followed here will be to maintain data only on current processes. These data may consist of the contents of main memory and the identity of pages referenced by each process Q_k. Data provided to the set $\{C_k\}$ of clustering tasks will be further restricted so as to eliminate interaction between them. Each C_k will be provided with:

1. A list L_k of assigned pages along with their home address blocks. These are the pages eligible for permuting by C_k.
2. The identity of pages in L_k which have been referenced by Q_k.

Figure 2 illustrates the contents of main memory for a case where q_2, the number of pages per block, is four. There are two active processes Q_1 and Q_2, each with an associated clustering task. Referenced pages within each block are marked with an r. Lines connect the set of referenced pages assigned to each task. For example, page a_{42} in block A_2 was referenced by both Q_1 and Q_2 and is assigned to task C_1.

The intent of the clustering tasks is to improve the page fault performance. In order to determine what permutations tend to further this goal, it is necessary to formulate a model for the pattern of references. The following simplified model is sufficient for discussion.

Suppose that

a. There is only one active process Q_i.
b. Blocks containing pages referenced by Q_i remain resident in main memory until Q_i is terminated.
c. All reference probabilities are stationary but not necessarily independent.

The expected number of page faults incurred by Q_i then becomes

$$Z = \sum_{j=1}^{M} \sum_{i=1}^{N} P(a_{ij})\,(1 - P[A_j|a_{ij}]), \tag{1}$$

where $\{A_j\}$, $j = 1, 2, \cdots, M$ is the set of blocks in the system; $\{a_{ij}\}$, $i = 1, 2, \cdots, N$ is the set of pages in block A_j; $P(a_{ij})$ is the probability that page a_{ij} is referenced by Q_i; and $P(A_j|a_{ij})$ is the probability that page a_{ij} is not the first in block A_j to be referenced by Q_i, given that a_{ij} is referenced.

The classical union bound yields

$$P(A_j|a_{ij}) \leq \sum_{k \neq i} P(a_{kj}|a_{ij}), \tag{2}$$

where $P(a_{kj}|a_{ij})$ is the probability of a prior reference to a_{kj} by Q_i, given that a_{ij} is referenced. The bound is tight

when the probability for one such a_{kj} is much larger than that for two or more.

Substitution of (2) into (1) results in

$$Z \geq \sum_{j=1}^{M} \sum_{i=1}^{N} P(a_{ij}) \left[1 - \sum_{k \neq i} P(a_{kj} | a_{ij}) \right]. \quad (3)$$

Equation (3) suggests that a goal for the clustering be the maximizing of

$$\psi = \sum_{j=1}^{M} \sum_{i=1}^{N} P(a_{ij}) \left[\sum_{k \neq i} P(a_{kj} | a_{ij}) \right]. \quad (4)$$

The significance of Eq. (4) is that it specifies a pairwise representation for the desirability of grouping a set of pages together in a block. Such a representation requires an approximation such as the above use of the union bound. In the simplest case, where page references are both stationary and independent, maximizing ψ is equivalent to placing the most frequently referenced pages together. The set $\{P(a_{kj}|a_{ij})\}$ may be regarded as entries in a pairwise page affinity matrix V_j.

For the purpose of dynamic reorganization, perhaps the most straightforward assumption, that taken here, is that pages referenced by a given process have reference probabilities and pairwise affinities sufficiently high that an effort should be made to place them together. Note that due to requirement 1 only pages currently in main memory should be considered for relocation. Thus the clustering of referenced pages must be done by a sequence of block or home address exchanges with unreferenced pages assigned to the same task C_i. Because the goal is to place referenced pages together, the unreferenced pages to be displaced by the home address exchange procedure must be adjacent. In other words, unreferenced pages must be assigned to a task C_i in groups belonging to the same block. These groups are then eligible to become *cluster points*, used to gather referenced pages.

A wide variety of clustering algorithms may be formulated by varying the method used to assign pages to the clustering tasks as well as that for choosing cluster points. Figure 3 shows a possible outcome of reorganizing the main memory contents illustrated in Fig. 2 when unreferenced pages in A_2 and A_5 are assigned as cluster points to tasks C_1 and C_2, respectively. Note that superior results would have been obtained had block A_4 been used as a cluster point for C_1. In this case, blocks A_4 and A_5 would hold all the referenced pages after reorganization.

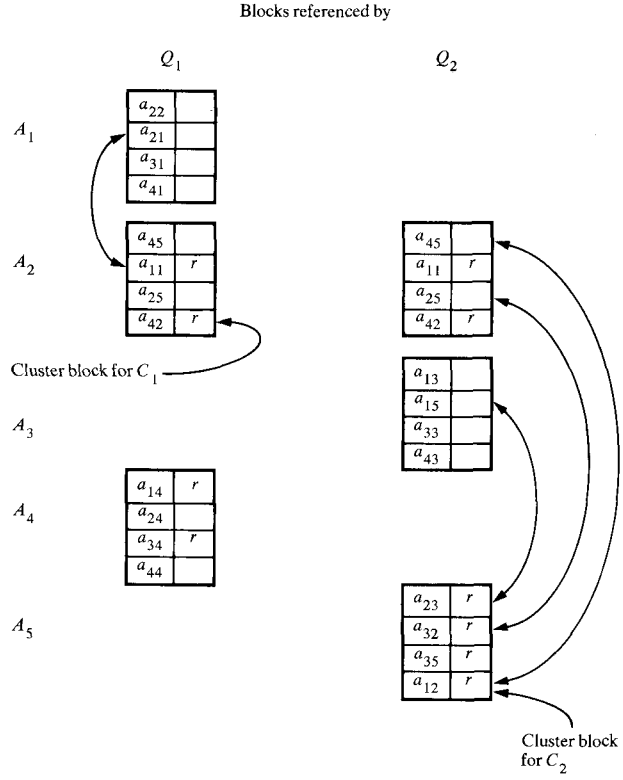

Figure 3 One way of reorganizing the main memory contents illustrated in Fig. 2.

A permutation clustering algorithm

This section describes a dynamic reorganization algorithm which is a simple implementation of the ideas developed in the previous sections. It is sufficient to indicate the procedures followed by a single clustering task C_i associated with a process Q_i.

1. A referenced page is assigned to the task C_i if Q_i was the first process to reference it during its current stay in main memory.
2. Let A_m be the block which contains the page a_{im} corresponding to the first page fault produced by Q_i. Unreferenced pages brought in as the result of the fault are assigned as a cluster point for C_i. A parameter R is set to the number of such pages. Note that this may be smaller than $N - 1$, since some pages belonging to A_m may already be resident in L_2.
3. The page a_{im} is assigned to C_i as the result of procedure 1. As other referenced pages are assigned to C_i, their names are placed on a first-in, first-out list V_{im}, and R decremented by their number. There is a separate such list for each cluster point.
4. When R reaches zero, the unreferenced pages associated with the next fault generated by Q_i become the next cluster point, and R is reset.
5. At the time that an unreferenced page from a cluster point is about to be removed from L_2, its home address is exchanged with that of a page from the associated list V_{im}.

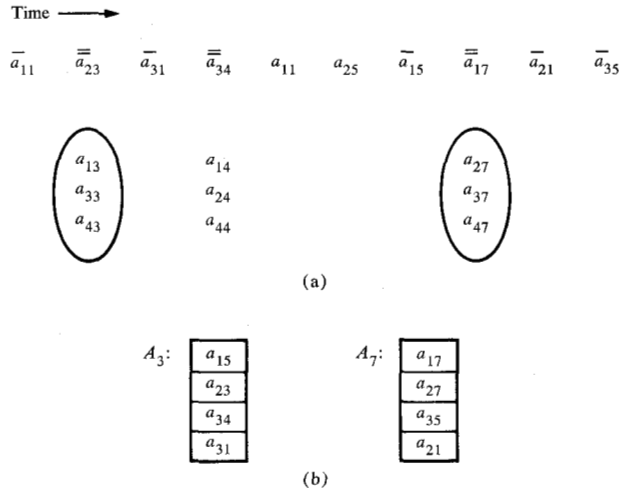

Figure 4 Operation of the cluster algorithm on a reference string. (a) The reference trace, in which faults have a double overbar, with a column of fetched pages below. First references to pages already in L_2 are indicated by a single overbar. Cluster points are encircled. (b) The resulting reorganization of A_3 and A_7, blocks containing cluster points.

6. As the last unreferenced page from a cluster point is removed from main memory, the list V_{im} associated with it is deleted. Thus pages on this list will no longer be subject to home address exchange. R is set to zero.

The above algorithm was chosen largely because of its implementational simplicity. Neither the choice of cluster points nor the assignment of referenced pages is optimal. Nevertheless, the observed performance gains, described in the following section, were substantial. Note that the algorithm is described without reference to a page replacement policy. However, it is assumed that pages referenced by a process Q_i are generally kept in main memory longer than unreferenced pages brought in by this process as the result of block fetches, a property that may be expected of most page replacement policies.

Figure 4 shows the operation of the above algorithm on the string of references generated by a process Q_i. Note that a_{11} does not get placed in a cluster point, since the first reference to it by Q_i is before a cluster point was assigned.

Reorganization of a data base

This section describes a number of experiments on a trace tape comprising three days of transactions on the AAS data base. The transactions correspond to the processes discussed above. There are 6×10^6 data base references in the trace, an average of 38 references per transaction.

Each 1693-byte physical data record containing several AAS logical records corresponds to one page. Eight consecutively numbered physical data records, or pages, are considered as members of one block for the purpose of block fetches. That is, $q_2 = N = 8$.

Miss ratios were obtained for a number of memory sizes and a variety of memory management policies:

1. LRU (least recently used) page replacement with demand paging. Here only the faulted page is fetched.
2. Block fetching (prefetching) without reorganization. For purpose of comparison with earlier experiments, the page replacement policy used was one developed by Bennett and McKellar [3].
3. The Affinity Algorithm 1 evaluated by Bennett and McKellar [3]. This is a clustering algorithm which does not adhere to requirements 2, 3, and 4 in the section entitled "Principles and requirements."
4. Block prefetching with dynamic reorganization. In addition, for a memory size of 2048 pages,
5. Block prefetching and dynamic reorganization which is halted after operating for a variable period of time.

Figure 5 shows the miss ratios for policies 1-4. Note that for a memory size of 2048 pages, block fetching has a miss ratio that is approximately 11.5 percent lower than LRU. This indicates that references tend to be correlated, and that the linear ordering of pages into blocks is advantageous. Note that this might be due to the programmers' knowledge that the data base is linearly ordered; e.g., knowledge of the linear ordering may encourage linear searches. Application of the dynamic reorganization algorithm yields a reduction of approximately eight percent in the miss ratio when compared to block fetching.

Figure 6 compares the performance of dynamic reorganization to that of block fetching over a period of 6×10^6 references for a main memory size of 2048 pages. The figures are normalized to the performance of block fetching with the original AAS organization. That is, 100 percent denotes the miss ratio for block fetching. The lower curve gives the performance for dynamic reorganization. Departures from this curve, which result from dynamically reorganizing the data base for $1, 2, \cdots, 5$ million page references, indicate the performance of the new organization. Points on the curves are the average miss ratio for 10^5 references.

An interval of 10^5 references corresponds to approximately 30 minutes for this trace. The median time to complete a transaction is of the order of 2 minutes. The curves indicate that the organization resulting from the reordering is superior to the original for, in most cases, an hour or longer. This suggests that the "optimal" organization for AAS is time varying, and that the home address exchange algorithm tends to track it.

The new organizations obtained from the home address exchange process, which has been halted at dif-

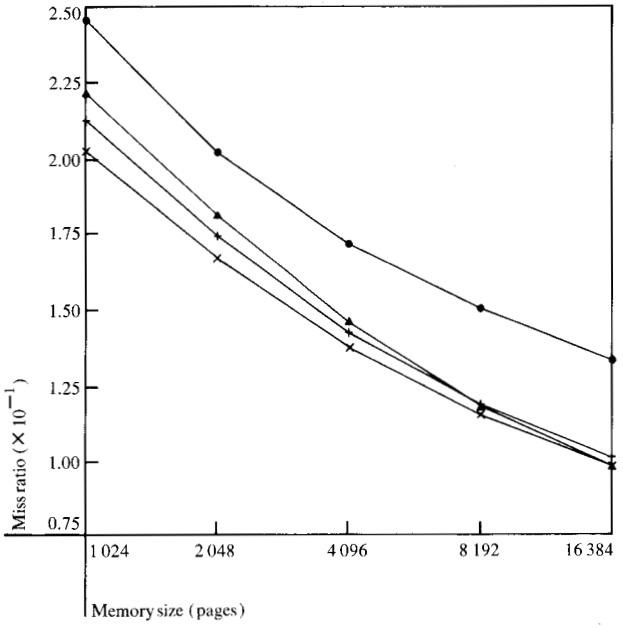

Figure 5 Miss ratios for various memory management policies for four million references: o, LRU page replacement; △, block fetching; +, affinity algorithm 1; ×, block fetching with dynamic reorganization.

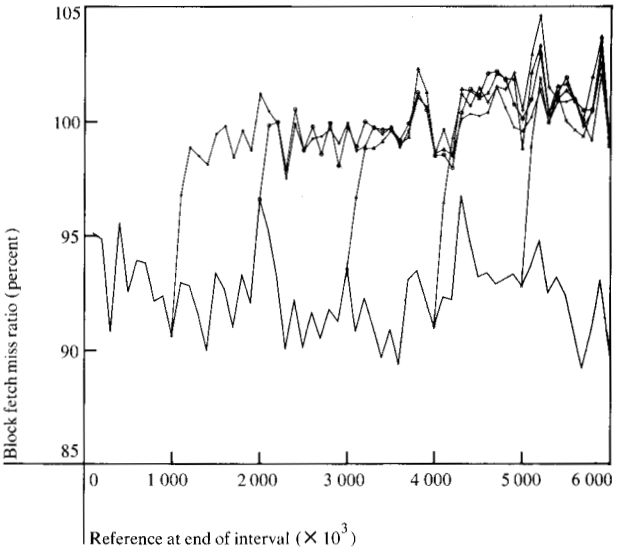

Figure 6 Normalized miss ratios for block fetching with dynamic reorganization running continuously and being halted after 1, 2, ···, 5 million references. A value of 100 percent denotes the miss ratio for block fetching without reorganization. Points on the curve are the average miss ratio for 10^5 references.

ferent times, can be expected to be dissimilar. However, Fig. 6 shows that the shapes of the performance curves exhibit a substantial similarity after a transient period of an hour or so (2×10^5 references). This suggests that there may be periods of time during which there are a significant number of sequential accesses. At other times, the reference clustering is less well ordered, and dynamic reorganization shows larger gains.

Concluding remarks

A class of dynamic reorganization algorithms has been described which exhibits a number of desirable system properties. A member of this class was then used for several experiments involving a dynamic reorganization of the AAS data base. The results indicate that the optimal organization is time varying and that dynamic techniques can indeed improve performance. When the clustering is stopped, the resulting organization is superior to the original for some time, after which there is little difference. The performance gains obtained were, while interesting, not sufficient to justify adoption for this application.

The extent to which these results would be typical for a variety of data bases is open to conjecture. The organization of the AAS data is familiar to the application developers, and reference patterns may reflect this knowledge. For systems containing data bases which are subject to accessing and modification in unpredictable ways, dynamic reorganization may be the best alternative.

The discussion has centered on properties of simple algorithms for performing the reorganization and on experiments for determining their benefits in terms of lowered miss ratios. Topics not discussed in any detail were the issues of cost, implementation, and what additional information relating to usage patterns might be used for the permutation clustering.

References

1. J. Gecsei, D. R. Slutz, I. Traiger, and R. Mattson, "Evaluation Techniques for Storage Hierarchies," *IBM Syst. J.* **9,** 78 (1970).
2. M. Joseph, "An Analysis of Paging and Program Behaviour," *Comput. J.* **13,** 48 (1970).
3. B. T. Bennett and A. C. McKellar, IBM Thomas J. Watson Research Center, Yorktown Heights, NY. Private communication.
4. *IBM System/360 Operating System Indexed Sequential Access Method*, Form No. Y28-6618, IBM Corporation, White Plains, NY.
5. D. J. Hatfield and J. Gerald, "Program Restructuring for Virtual Memory," *IBM Syst. J.* **10,** 168 (1971).
6. J. P. Considine and A. H. Weis, "Establishment and Maintenance of a Storage Hierarchy for an On-line Data Base Under TSS/360," *AFIPS Conference Proceedings*, Fall Joint Computer Conference, 1969, p. 433.
7. J. H. Wimbrow, "A Large-Scale Interactive Administrative System," *IBM Syst. J.* **10,** 260 (1971).

Received February 1, 1977

The authors are located at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York 10598.