Non-Hierarchic Document Clustering Using the ICL Distributed Array Processor

Edie M. Rasmussen and Peter Willett*
Dept. of Information Studies
University of Sheffield,
Western Bank, Sheffield S10 2TN, U.K.

Abstract This paper considers the suitability and efficiency of a highly parallel computer, the ICL Distributed Array Processor (DAP), for document clustering. Algorithms are described for the implementation of the single-pass and reallocation clustering methods on the DAP and on a conventional mainframe computer. These methods are used to classify the Cranfield, Vaswani and UKCIS document test collections. The results suggest that the parallel architecture of the DAP is not well suited to the variable-length records which characterise bibliographic data.

* to whom all correspondence should be addressed

1. INTRODUCTION

Cluster analysis, or automatic classification, is the name which is given to a range of techniques for the classification of datasets. By looking at similarities between objects in the dataset, it is possible to identify groups, or clusters, of objects which are alike in some way. The applications of cluster analysis are widespread (Jardine and Sibson, 1971; Anderberg, 1973; Dubes and Jain, 1980). In the information retrieval context, cluster analysis has been used for the clustering of documents, where the similarities are derived from the numbers of terms in common between documents, and for the clustering of terms, where the similarities are derived from the numbers of documents in which terms co-occur (Salton and McGill, 1983). Interest in document clustering for information retrieval arises from its potential for improving the efficiency and effectiveness of retrieval, as well as providing an alternative to the conventional Boolean search strategies (van Rijsbergen, 1979; Salton and McGill, 1983; Vorhees, 1985).

Clustering methods can be broadly divided into two types: hierarchic methods and non-hierarchic, or partitioning, methods. The hierarchic methods use an NxN similarity matrix, containing the pairwise similarities in a dataset of size N objects, to create a nested set of clusters. The non-hierarchic methods divide a dataset into a single level partition, with or without overlap between the clusters; the same method can then be applied to the resulting partition to produce a hierarchy of partitions. Both types of method have been used for document collections. The extensive clustering experiments carried out in the SMART Project (Salton, 1971) used a variety of non-hierarchic methods; more recent work (Croft, 1977; Griffiths et al., 1984; Vorhees, 1985) has used the hierarchic methods. These have been shown to be substantially more effective for retrieval than the non-hierarchic methods; however, they are substantially less efficient in operation since the non-hierarchic methods do not involve the calculation and processing of the inter-document similarity matrix (which has a time requirement of at least order $O(N^2)$ for a collection of N documents).

Clustering methods require the calculation of the similarity between items; for this, some similarity or distance function is required, and a number of similarity functions have been applied to the clustering of document collections (Salton and McGill, 1983). The one used in the work reported here, the Dice Coefficient, is typical; for two objects, D_I and D_J, the coefficient is given by

$$S(D_I,D_J) = 2* \Sigma d_{IK}*d_{JK}/(\Sigma d_{IK} + \Sigma d_{JK})$$

where $D_I = (d_{I1},d_{I2},...,d_{IT})$, T is the number of terms in the indexing vocabulary, d_{IK} is the weight of the K'term in the I'th object, and where the summations are from K=1 to K=T. In information retrieval applications, many of the d_{IK} elements will be zero-valued, and it is often the case that only the non-zero elements are stored. In a typical document clustering routine, the objects are either documents or clusters, and this function must be evaluated for each and every document-document or document-cluster pair, making clustering a computationally demanding process. There has thus been considerable interest in the development of algorithms which can allow these similarities to be calculated as efficiently as

possible (Croft, 1977; Willett, 1981; Vorhees, 1985). An alternative way of increasing clustering efficiency is by the use of parallel computer hardware, which allows some or many similarities to be calculated simultaneously. Many types of parallel machine have been developed, and these have been comprehensively surveyed by Hwang and Briggs (1984). In Flynn's typology (1966), conventional serial machines are designated as SISD - single instruction stream, single data stream; that is, instructions are executed sequentially on a serial stream of data items. Parallel machines are either single instruction stream, multiple data stream (SIMD) or multiple instruction stream, multiple data stream (MIMD), according to whether the processors act synchronously or independently. Parallel algorithms may be inherently suitable for SIMD or MIMD machines according to their requirements for communication, control and processor geometry (Kung, 1976).

Most parallel computers have been developed for the study of scientific problems which involve extensive numeric computations; non-numeric applications of parallel processors are less well developed, with most attention being given to the development of hardware support for database management operations (Hsaio, 1983). There have been several studies of the use of parallel processors for clustering, but this has principally been for pattern recognition and image segmentation applications (Jackson et al., 1982; Ni, 1985; White, 1985). However, the nature of document clustering suggests that it is a suitable candidate for parallel processing since, as Salton and Bergmark (1981) note, there is a high degree of parallelism in the calculation of a set of inter-document similarity coefficients: the same operation must be repeated many times on different data, and the results of the individual calculations are not interdependent. This type of parallelism is well suited to an SIMD processor, for example an array processor (Hwang and Briggs, 1984), which can evaluate large numbers of similarity coefficients simultaneously by providing each processing element in the array with the data required for a single calculation, and matching a specified document against all of the elements in parallel. If there are N processors, then the time complexity of computing the elements of an NxN similarity matrix reduces from order $O(N^2)$ to order O(N) if there are no serial bottlenecks or inter-processor communications costs which interfere significantly with the parallelism. Some initial studies of the use of a parallel array processor for calculating query-document similarity coefficients have been reported by Pogue and Willett (1984) who compared serial nearest neighbour document retrieval on such a processor with that on a conventional mainframe computer. Repeated nearest neighbour calculations lie at the heart of most clustering methods and an array processor should thus also be appropriate to this type of processing.

The work described in this paper is an examination of the suitability of a parallel processor, the ICL Distributed Array Processor (DAP), for clustering large document collections using the single-pass and reallocation clustering methods. In particular, the efficiency of the DAP for document clustering will be compared with that of a large serial mainframe, the IBM 3083 BX. After a brief description of the DAP and of the document collections which were used, Section 3 discusses the two clustering methods and Section 4 the algorithms which were used to implement them on a DAP and on an IBM 3083 BX. The experimental results are presented and discussed in Section 5. The paper closes with our conclusions and suggestions for future work.

2. EXPERIMENTAL DETAILS

2.1 The ICL Distributed Array Processor

The DAP is a highly parallel, SIMD machine with large numbers of processing elements, or PEs, linked in a 2-D array; the DAP used in our experiments contained a total of 4096 PEs. The PE's are arranged in a 64x64 matrix and receive instructions broadcast from a Master Control Unit (MCU) so that processing activity is identical in all PEs (though for a particular operation it is possible to designate a PE as active or inactive through the use of a logical mask). Processing within each PE is bit-serial in nature and relatively slow; however, the great number of PEs means that very high processing rates can be obtained, given an appropriate algorithm which can map the problem under investigation onto the processor array. Each PE has an associated 16 Kbits of storage, for a total of 8 Mbytes, with all data input and output being carried out via a host ICL 2980 mainframe. The programming language used is DAP FORTRAN, a version of FORTRAN with additional parallel functions and operations. More detailed descriptions of the DAP hardware and software are provided by Parkinson (1982), Hunt and Reddaway (1983) and by Carroll et al. (1987).

2.2 Performance measurement

The performance of a parallel computer is normally expressed in terms of the speed-up; the speed-up with P processors, S(P), is defined by S(P)=T(P)/T(1) where T(1) and T(P) are the run-times to perform some operation on 1 and P processors respectively. Parkinson and Liddell (1983) suggest that this definition is not applicable to an array processor like the DAP, where T(1) has no real meaning. Moreover, the algorithms which are best suited to a parallel processor are often not those most appropriate on a serial processor which further complicates the comparison. The approach taken here is to note the time required by a serial processor running a range of algorithms for the clustering task, and then to compare this with the time required for the same task by a parallel algorithm running on the DAP.

Three document collections were used as detailed in Table 1. In each collection, the keywords representing a document were stemmed using an automatic suffix stripping routine; the documents were then represented by lists of stem numbers.

Table 1: Characteristics of Test Collections

Collection	Cranfield	UKCIS	Vaswani
Number of documents	1400	27361	11429
Most terms per document	101	25	105
Mean terms in a document	28.7	6.7	20.3
Total postings	40241	182414	231920

All clustering runs were carried out on the 64x64 DAP at Queen Mary College, London, and on a conventional mainframe computer, this being an IBM 3083 BX at the University of Sheffield; the cycle times of these two machines are 200 and 24 nanoseconds respectively. The DAP programs were encoded in DAP-FORTRAN and the IBM programs in FORTRAN/VS using the level-3 optimising compiler. All of the data needed for each of the runs was loaded into main memory so that the recorded run-times do not include input-output operations.

3. CLUSTERING PROCEDURES

Two non-hierarchic clustering methods are studied in this report, these being the single-pass method and the reallocation method.

The single-pass method, the simplest of the partitioning methods, operates as follows (Salton, 1971; Fritsche, 1974; Willett et al., 1986):

1. Make the first document the representative for Cluster 1.
2. For the next item, calculate the similarity, S, with each existing cluster representative, using some standard similarity coefficient.
3. If the highest calculated S is greater than some specified threshold value, SMAX, add the document to the corresponding cluster and redetermine the cluster representative; otherwise, use the document to initiate a new cluster. If any documents remain to be clustered, return to Step 2.

This method requires only one pass through the dataset, and hence its name; the time requirements are typically of order O(NlogN) for order O(logN) clusters. This makes it a very efficient clustering method on a serial processor. A disadvantage is that the resulting clusters are not independent of the order in which the documents are processed, with the first clusters formed usually being larger than those created later in the clustering run. However the simplicity of the method makes it a suitable candidate for initial clustering work on the DAP, and it serves to demonstrate the link between clustering requirements and hardware restrictions. Moreover, the clusters created in the single pass method provide a useful starting position for the reallocation method.

Reallocation methods, in which items are moved between clusters until a stable arrangement is reached, are computationally more demanding than the single pass methods. It is possible, however, to overcome the order dependence of single-pass methods, without the computational investment required for hierarchical methods (Willett, 1980). Typically, a reallocation algorithm is as follows:

1. Select M cluster representatives or centroids.
2. Assign each document in the set to the most similar centroid.
3. Recalculate the cluster centroids.
4. Steps 2 and 3 constitute one pass through the file. If none, or less than some user-specified number of documents have changed cluster membership, stop; otherwise return to Step 2 for another pass through the file.

Although in theory most reallocation methods will converge (i.e., require no further changes in cluster membership) (Anderberg, 1973), there is no upper bound to the number of passes which may be required. However, in practice,four or five passes through the dataset are usually sufficient to provide clusters in which no, or relatively little, further movement is required (Willett, 1980; Willett et al., 1986).

A number of ways of specifying the initial M cluster centres have been suggested, including using M randomly selected items (Forgy, 1965), using the first M documents when ranked in order of similarity with the centroid of the entire collection (Hartigan and Wong, 1979), or using the set of centroids resulting from an initial single pass procedure (Hartigan, 1975); this last approach is well documented and is used in the work described here.

The description of the two methods illustrates the central importance of nearest neighbour searching, since both are based upon the repeated assignment of documents to the most similar cluster. Accordingly, much of our work has been devoted to considering the implementation of the nearest neighbour components of the methods. Specifically, three nearest neighbour algorithms (SA, SB and SC) were implemented on the serial IBM processor and a parallel nearest neighbour algorithm on the DAP; this work is discussed in the following section.

4. IMPLEMENTATION OF THE CLUSTERING METHODS

4.1 Serial nearest neighbour searching algorithms

Three nearest neighbour algorithms (SA, SB and SC) were implemented on the IBM 3083 processor.

Algorithm SA. This algorithm was used by Pogue and Willett (1984) in their study of best match retrieval. For every document-cluster pair which needs to be compared, pointers are maintained in both the document and cluster term lists; these pointers are moved asynchronously along the lists term by term, incrementing the similarity value as matches are found. The algorithm assumes that the

lists of document terms have previously been sorted into order. In pseudocode, the process is represented as follows, where N and M are the numbers of documents and clusters and where P and Q are the numbers of terms in the current document and in the current cluster representative:

```
DO FOR 1 TO N
  DO FOR 1 TO M
    I := 1
    J := 1
    DO WHILE (I <= P) AND (J <= Q)
      IF DocumentTerm(I) > ClusterTerm(J)
      THEN J := J+1
      ELSE IF DocumentTerm(I) < ClusterTerm(J)
           THEN I := I+1
           ELSE
             add the term weight to the
               similarity value
             I := I+1
             J := J+1
    Calculate the similarity coefficient.
```

P+Q comparisons are required for each document-cluster pair.

Algorithm SB. In this algorithm, the document to be clustered is converted to a logical record; it is stored as a bit string with the I'th bit set to true if the I'th term has been assigned to the document (1 <= I <= T). Thus, the similarity calculation for a document-cluster pair involves using each cluster term value as an index to the document bit string:

```
DO FOR 1 TO N
  Convert the current document to logical form
  DO FOR 1 to M
    J := 1
    DO WHILE J <= Q
      IF DocumentTerm(ClusterTerm(J)) is True
      THEN add the term weight to the
        similarity value
      J := J+1
    Calculate the similarity coefficient.
```

The number of comparisons required for each document-cluster pair is simply Q, the length of the cluster term list, and is independent of P.

Algorithm SC. Algorithms SA and SB are based on the use of the serial file organisation, in which the current document is matched against one cluster after another. Efficiencies of operation on a serial processor can be obtained by use of the inverted file organisation; this limits the number of document-cluster similarity calculations which need to be carried out. Croft (1977) has demonstrated the use of the inverted file to eliminate the calculation of zero-valued similarity coefficients, this being achieved by carrying out pairwise calculations for documents which occur in the same inverted file lists. Further improvements are possible (Willett, 1981; Perry and Willett, 1983) if terms in the cluster representatives are stored in the form of an inverted file; the document term list can then be used as indices to the inverted term-cluster lists which are needed for the similarity calculation. The algorithm is as follows:

```
DO FOR 1 to N
  I := 1
  DO WHILE I <= P
    Access the inverted file list
      corresponding to DocumentTerm(I)
    DO FOR 1 to NumberOfClusterOccurrences
      in this list
      Add the term weight to the similarity
        value corresponding to the current
        cluster
    I := I+1
  DO FOR 1 TO M
    Calculate the similarity coefficient.
```

The similarity coefficient is computed only for those clusters which have terms in common with the document. SC is the algorithm most unlike the parallel algorithm described below, since it attempts to improve efficiency by limiting the number of calculations required rather than by making the calculations faster as in algorithms SA and SB (where all of the document-cluster similarity coefficients are calculated).

4.2 Parallel nearest neighbour searching algorithm

A basis for using the parallelism in the DAP hardware is provided by the observation that it is possible to store each cluster in a separate PE. As each document is processed in turn it is compared with all of the PEs which contain a cluster, thus allowing the simultaneous matching of the current document with all of the existing clusters (for M <= 4096). Once the contribution to the similarity value of all matching terms has been tallied, the remaining calculation (dividing by the denominator in the Dice Coefficient) can also be carried out in parallel. An intrinsic DAP FORTRAN function selects the maximum similarity value from the 64x64 matrix of PEs, thus completing the comparison of a document with all clusters.

The question then arises as to how the matching operations should be implemented on the DAP; it seems that none of the three serial algorithms can be implemented in a manner which makes substantial use of the parallelism inherent in the hardware. A characteristic of the DAP is that processing must be carried out on the same relative location in each of the local storage areas associated with each PE; such a common storage area is referred to as a plane, and thus when an instruction is broadcast from the MCU, all processing must be aligned on the same plane. This means that a system of asynchronous pointers to the cluster terms (as in a parallel version of the SA algorithm) cannot be used. The SB algorithm involves an inherently serial inspection of each cluster representative in turn (for matching against the document bit string) and thus has no scope for implementing the matching operations in parallel. Finally, the SC algorithm also cannot be used owing to the Zipfian distribution of the numbers of postings in the lists which make up the inverted file; while many of the lists contain only one or two elements, some contain a very large number of postings, and these cannot be easily mapped onto the fixed, and limited, storage area associated with each PE.

The parallel algorithm which was used can best be regarded as a modification of algorithm SA in which the asynchronous movement of the document and cluster pointers is replaced by a broadcasting operation in which each term in the current document is matched against each of the storage locations containing terms in the longest cluster representative; by doing this, we ensure that the document terms are also matched against each of the terms in the representatives of shorter clusters (which are filled with dummy values so that all representatives are of the same, maximum length). The algorithm is thus as follows, where QMAX is the number of terms in the longest cluster representative:

```
DO FOR 1 to N
  I := 1
  DO WHILE I <= P
    J := 1
    DO WHILE J <= QMAX in parallel
      IF DocumentTerm(I) = ClusterTerm(J)
        in any PE
      THEN add the term weight to the
         similarity value for that cluster
      J := J+1
    I := I+1
  Calculate all similarity coefficients
    in parallel.
```

In the worst case, PxQMAX comparisons are required (although this value can be reduced somewhat by determining when a match with a document term is no longer possible and then broadcasting the next term). Accordingly, the matching operations required here are inherently more time-consuming than in the SA algorithm; however, the DAP architecture ensures that these PxQMAX comparisons result in the calculation of the similarity coefficient between a document and all of the clusters, and not just one as with the serial algorithms.

Although the number of documents which can be clustered by this method is not limited, the 4096 PEs in the DAP imposes a hardware limitation on the number of clusters which can be created. In practice, a set of <= 4096 clusters was ensured by adjusting the similarity threshold, SMAX, in the single-pass method so that no overflow occurred. Since the number of clusters is not an inherent characteristic of a dataset, and since it is not at present possible to identify optimum values for any given dataset (Everitt, 1979), such an empirical solution related to the DAP's structure was considered acceptable.

The local storage available in each PE is used to hold a cluster (i.e., the terms and term weights comprising the cluster representative, together with associated characteristics such as the length of the representative and the number of documents in the cluster) as well as the results of similarity calculations and the documents themselves. It is necessary to set an upper limit on the number of documents which can join a cluster, particularly in the single-pass method, where the clusters which are formed first can become very large; the upper limit was set to 20 documents for the Cranfield and UKCIS datasets and to 10 documents for the Vaswani dataset. These restrictions were most significant in the largest (UKCIS) data set, but after reallocation processing less than 2% of the clusters were affected. These empirical decisions, though linked in this case to features of the DAP, are in fact characteristic of partition clustering methods, where a range of input parameters (threshold, number of clusters, degree of overlap etc.) normally need to be supplied. (Salton, 1971; Fritsche, 1974).

4.3 Storage and updating of cluster representatives

The parallel algorithm was used for both the single-pass and reallocation methods. The cluster representative used in our experiments was an ordered list of all the index terms appearing in the documents within a cluster, and an associated weight for each term corresponding to the number of times that the term occurred in the cluster; since the document indexing merely denoted the presence or absence of a term in a document, the weights represented the number of documents within a cluster containing the given term.

The process of updating the cluster in the single-pass method is essentially serial in nature, even on the DAP. The document which is to be clustered is compared with the representative of the cluster to which it is to be assigned, and the weights incremented for those terms in the representative which are also included within the document. In the reallocation method, where cluster updating is done at the end of each pass of the file, there is a potential for the parallel updating of all of the cluster representatives to which documents are to be assigned. However, the inability of the DAP to simultaneously access different planes makes this impossible to achieve in practice, and thus each document had to be processed in turn.

5. RESULTS AND DISCUSSION

5.1 Single-pass method

The clustering rates (in documents/second CPU time) for the single-pass method are given in Table 2. The SA algorithm proved much less efficient than SB and SC in trials on the relatively small Cranfield collection, and was therefore not used with the UKCIS and Vaswani files.

Table 2: Single Pass Cluster Rates (in Documents/Second)

	Serial A	Serial B	Serial C	Parallel
Cranfield	40.9	94.2	331.0	5.2
UKCIS	-	15.2	173.4	28.7
Vaswani	-	10.7	63.0	9.9

As expected, the inverted file SC algorithm performed better than the other serial algorithms in all of the datasets. The relatively poor performance of the parallel algorithm is due to

the need to broadcast PxQMAX index terms for each document, and to the serial component of the algorithm in the form of cluster representative update; this relatively simple part of the processing was found to require about 20% of the observed DAP times. Also, the single-pass method is not uniformly parallel, since the number of active PEs, which corresponds to the number of clusters formed, increases as the run progresses; thus, the parallel nature of the DAP only begins to play a significant role in increasing the efficiency of processing once much of the file has already been clustered. This is seen most clearly with the Cranfield dataset, where only 1400 documents, and correspondingly fewer clusters, are used thus causing most of the PEs to remain idle throughout the processing.

The parallel algorithm performs better than the SB algorithm on the UKCIS file. This can be explained by the nature of the file, which contains relatively short document descriptions. Since the parallel version requires PxQMAX comparisons and since short document descriptions result in short cluster term lists, PxQMAX is not substantially larger than the QMAX or less term comparisons required for each document-cluster match in the serial procedure. Therefore, short document descriptions are processed more efficiently than long ones; this advantage is lost with the Vaswani file, which has long document descriptions, where the SB and parallel algorithms perform comparably.

A feature of both the SA and SB algorithms is their dependence on M, the number of clusters formed, while the parallel algorithm is independent of it (for M <= 4096). It would hence be expected that the parallel performance would improve relative to the others as the pass through the file progresses, and M increases; this behaviour is illustrated in Figures 1-3, which show the time taken to process the file at various points. On the Cranfield file, where the parallel algorithm performs poorly for the reasons given above, the timing values are erratic: this is due to the combined effect of the increased length of the longest cluster and of the average length of the documents in any file segment. The effect of document length is also shown clearly in Figures 2 and 3; the UKCIS and Vaswani files were processed in order of increasing document length, and discontinuities in the timing curve occur whenever there is a large increase in the maximum cluster length. This is particularly marked in the final segment of the Vaswani file.

Figure 1

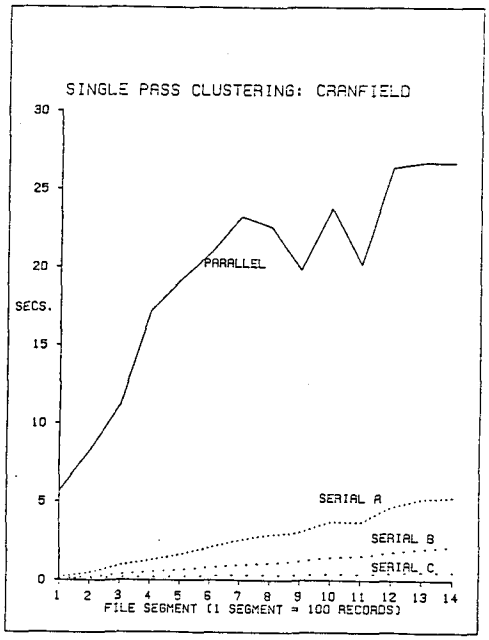


Figure 2

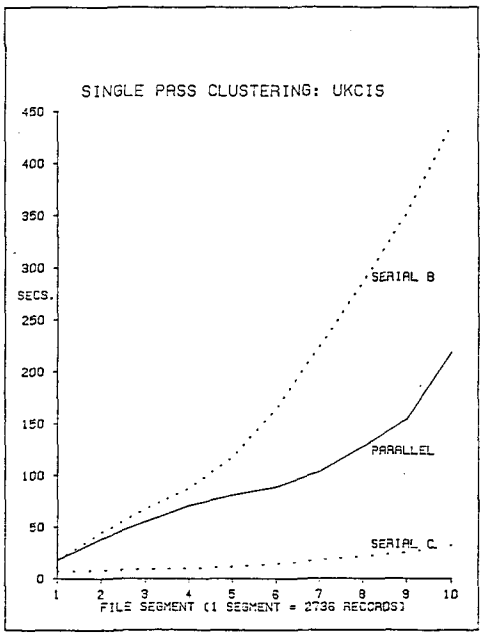
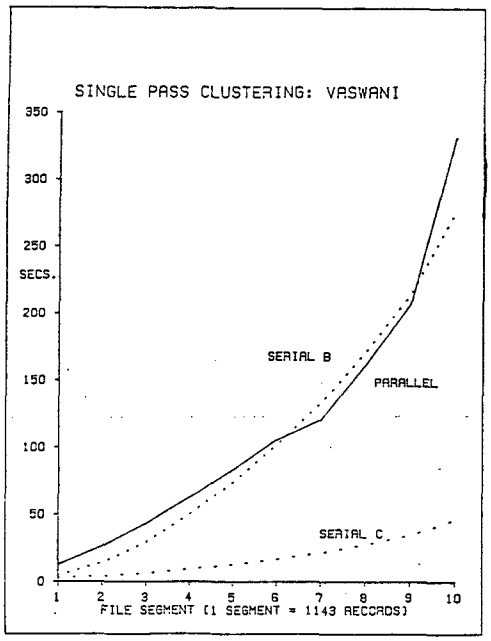


Figure 3



5.2 Reallocation method

The times taken to cluster the same files using the reallocation method are shown in Table 3. The SA algorithm was again elminated as being too slow; SB, though slower than SC, was retained to show the effect of calculating all the similarity coefficients.

Table 3: Reallocation Cluster Times (in Seconds)

	Serial B	Serial C	Parallel
Cranfield	121.1	22.5	1219.8
UKCIS	22352.5	1033.4	4516.6
Vaswani	12230.7	988.3	7019.6

The times shown are for four iterations. On the IBM, the time taken for one SC iteration was up to 64% longer than required in the single-pass run, since the number of clusters was fixed rather than increasing during the run. On the DAP, the time per iteration was also up to 52% longer since the number of comparisons required is dependent on the length of the longest cluster, which is stable in a reallocation iteration but gradually increasing in a single pass run. This increase in cluster length more than offsets the decrease in serial processing required during cluster representative updating.

In general, the patterns noted for the single-pass method are repeated here. The inverted file (SC) algorithm is best for all files. However, the dependence of the SB algorithm on the number of clusters, M, which is fixed at the beginning of the reallocation processing, has caused its performance to deteriorate. The parallel algorithm is now clearly superior to SB on both the UKCIS and Vaswani files.

6. CONCLUSIONS

The results obtained here demonstrate an application area where the use of parallel processing techniques does not give increases in efficiency over the use of a conventional processor using a fast serial algorithm. One obvious reason for this behaviour is that good serial algorithms have been developed over a long period, while parallel algorithms are still relatively scarce. However, a more important factor is that the choice of a good parallel algorithm may be so restricted, as it is here, by the parallel hardware, that no alternative algorithm seems to be available. In the DAP the limitations are rigorous and include a relatively slow individual processor speed, and an inability to address different planes in different PEs (which means that it is not possible to implement a parallel version of the SA algorithm). To compete effectively with the inverted file algorithm for document clustering using the partition methods discussed here, a faster, more flexible type of processor is required.

The best parallel performance found here was on a file of short document descriptions, where the number of term matching operations required is not large. When the exhaustivity of the document indexing is high, the complexity of the parallel algorithm means that serial algorithms are much to be preferred, especially the inverted file algorithm which makes use of the fact that a very sparse data matrix needs to be processed and that many of the similarities which need to be evaluated are zero valued. This is characteristic of bibliographic data, where each document is indexed by some small number of terms selected from a large indexing vocabulary. With fixed format data, each record will have all of the available fields, and there will be no advantage to be gained from the use of an inverted file algorithm. Some preliminary clustering experiments suggest that such data are much better suited to the DAP architecture than the variable length records used here, and further work is being done in this area. A study on the DAP, implementing the Jarvis-Patrick clustering method (Jarvis and Patrick, 1973) for a dataset of chemical structures, has been completed and will be reported in detail elsewhere. In brief, the chemical structures are represented as fixed format bit strings (denoting the presence or absence of predefined chemical substructures). The use of this representation with a parallel version of algorithm SB allows DAP nearest neighbour searching to be accomplished about two to three times as fast as nearest neighbour searching on the IBM. (The precise degree of speed-up depends on the number of nearest neighbours and the size

of the dataset.) Current work involves the use of the DAP for hierarchic clustering of large files of fixed format demographic data using Ward's method.

Acknowledgement We thank the staff of the DAP Support Unit, Queen Mary College for help with the development of our programs, and the British Library Research and Development Department for funding under grant number SI/G/760.

7. REFERENCES

M.R. Anderberg (1973). Cluster Analysis for Applications. New York: Academic Press.

D.M. Carroll, C. Pogue and P. Willett (1987). Bibliographic pattern matching using the ICL Distributed Array Processor. Journal of the American Society for Information Science (in press).

W.B. Croft (1977). Clustering large files of documents using the single-link method. Journal of the American Society for Information Science 28: 341-344.

R. Dubes and A.K. Jain (1980). Clustering methodologies in exploratory data analysis. Advances in Computers 19: 113-227.

B.S. Everitt (1979). Unresolved problems in cluster analysis. Biometrics 35: 169-181.

M. Flynn (1966). Very high-speed computing systems. IEEE Proceedings 54: 1901-1909.

E. Forgy (1965). Cluster analysis of multivariate data: Efficiency versus interpretability of classifications. Biometrics 21: 768.

M. Fritsche (1974). Automatic Clustering Techniques in Information Retrieval. Luxembourg: Commission of the European Communities.

A. Griffiths, L.A. Robinson and P. Willett (1984). Hierarchic agglomerative clustering methods for automatic document classification. Journal of Documentation 40: 175-205.

J. A. Hartigan (1975). Clustering Algorithms. New York: Wiley.

J.A. Hartigan and M.A. Wong (1979). A k-means clustering algorithm. Applied Statistics 28: 100-108.

D.K. Hsaio, ed. (1983). Advanced Database Machine Architecture. Englewood Cliffs, N.J.: Prentice-Hall.

D.J. Hunt and S.F. Reddaway (1983). Distributed processing power in memory. In: The Fifth Generation Computer Project. London: Pergamon Infotech. pp. 49-62.

K. Hwang and F.A. Briggs (1984). Computer Architecture and Parallel Processing. New York: McGraw-Hill.

A.A. Jackson, H.M. Sykes, and R.S. Blake (1982). Drooling - a non-parametric multidimensional clustering algorithm for distributed array processor. Computer Physics Communications 27: 351-364.

N. Jardine and R. Sibson (1971). Mathematical Taxonomy. New York: Wiley.

N. Jardine and C.J. van Rijsbergen (1971). The use of hierarchic clustering in information retrieval. Information Storage and Retrieval 7: 217-240.

R.A. Jarvis and E.A. Patrick (1973). Clustering using a similarity measure based on shared near neighbors. IEEE Transactions on Computers C-22: 1025-1034.

H.T. Kung (1976). Synchronized and asynchronous parallel algorithms for multiprocessors. In: Algorithms and Complexity: New Directions and Recent Results. New York: Academic Press. pp. 153-200.

L.M. Ni (1985). A VLSI systolic architecture for pattern clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-7: 80-89.

D. Parkinson (1982). Practical parallel processors and their uses. In: Parallel Processing Systems. Cambridge: Cambridge University. pp. 215-236.

D. Parkinson and H.M. Liddell (1983). The measurement of performance on a highly parallel system. IEEE Transactions on Computers C-32: 32-37.

S.A. Perry and P. Willett (1983). A review of the use of inverted files for best match searching in information retrieval systems. Journal of Information Science 6: 59-66.

C. Pogue and P. Willett (1984). An evaluation of document retrieval from serial files using the ICL Distributed Array Processor. Online Review 8: 569-584.

G. Salton, ed. (1971). The SMART Retrieval System: Experiments in Automatic Document Processing. Englewood Cliffs, N.J.: Prentice-Hall.

G. Salton and D. Bergmark (1981). Parallel computation in information retrieval. Lecture Notes in Computer Science 111: 328-342.

G. Salton and M.J. McGill (1983). Introduction to Modern Information Retrieval. New York: McGraw-Hill.

E. Vorhees (1985). The Effectiveness and Efficiency of Agglomerative Hierarchical Clustering in Document Retrieval. Cornell University: PhD thesis.

R.A. White (1985). Data structures for implementing the CLASSY algorithm on the MPP. In: The Massively Parallel Processor. Cambridge, Mass.: MIT. pp. 31-61.

P. Willett (1980). Document clustering using an inverted file approach. Journal of Information Science 2: 223-231.

P. Willett (1981). A fast procedure for the calculation of similarity coefficients in automatic classification. Information Processing and Management 17: 53-60.

P. Willett, V. Winterman and D. Bawden (1986). Implementation of nonhierarchic cluster analysis methods in chemical information systems: selection of compounds for biological testing and clustering of substructure search output. Journal of Chemical Information and Computer Sciences 26: 109-118.