Pergamon

Information Systems Vol. 23, No. 7, pp. 489–508, 1998

Printed in Great Britain
PII: S0306-4379(98)00024-6
0306-4379/98 $19.00 + 0.00

DYNAMIC ALLOCATION OF SIGNATURE FILES ON PARALLEL DEVICES†

MAN-KWAN SHAN and SUH-YIN LEE

Institute of Computer Science and Information Engineering National Chiao Tung University, HsinChu, Taiwan, ROC

(Received 7 November 1996; in final revised form 16 July 1998)

Abstract—Signature file is one of the efficient access methods for retrieval of text database. In a large database server, parallel device is utilized to achieve concurrent access. Efficient allocation of signature file on parallel device minimizes the query response time and is important in the design of large databases. In this paper, we investigate the design of parallel signature file. We propose a new dynamic allocation technique to distribute the signature file on parallel device. It is an improvement of previous approach of Fragmented Signature File. While Fragmented Signature File uses Quick Filter to distribute the partitioned frame signature file, the proposed Parallel Signature File uses a declustering technique. The proposed Parallel Signature File has some advantages. First, the qualified frame signature blocks are distributed more uniformly than Fragmented Signature Files. Second, the proposed scheme can also be used in dynamic environment. Performance analysis shows that performance of the proposed approach outperforms that of Fragmented Signature File and is not far from theoretical optimal response time. © 1998 Published by Elsevier Science Ltd.

Key words: Signature Files, Text Databases, Information Retrieval, Disk Allocation, Access Methods

1. INTRODUCTION

Signature file is one of the efficient access methods for text retrieval [4, 6]. It acts as a search filter to reduce the search space. For text retrieval, although signature file is slower than inverted file, it takes less storage overhead. Besides, signature file is also used in partial match retrieval of traditional formatted databases [19], clause indexing of Prolog databases [18], subpicture query of image databases [16, 21], text retrieval of large Chinese text databases [1].

However, in large database servers, secondary storage access becomes the bottleneck. Parallel secondary storage devices in which data can be accessed concurrently are utilized. The distribution of the signature file on parallel device may improve performance significantly. The query response time of content-based retrieval of large databases systems can be reduced by efficient distribution of the signatures on parallel device.

One of the important design goals of distribution of signature files is to maximize the throughput. Two criteria to achieve the goal are intra-query parallelism and inter-query parallelism. Intra-query parallelism must try to distribute the signatures pertinent to the query as uniformly as possible. Inter-query parallelism must try to activate as few disks as possible for small query and to process as many queries concurrently as possible.

Few researches have studied the design of signature file on the parallel machine architecture. Stanfill et al. proposed a parallel signature file developed in Thinking Machines Corporation for high-speed interactive querying of text databases on a SIMD computer, the Connection Machine [24]. However, it was based on the assumption that main memory was sufficient to hold all the text signatures. To deal with this problem, Panagopoulos et al. proposed a parallel bit-sliced signature file method on a SIMD machine [15]. A partial fetch slice-swapping algorithm is used when the size of the signature file exceeds the available memory. The partitioned signature file approach proposed by Lee et al. can also be implemented in a parallel environment by assigning each partition to a separate processor [11]. Only some partitions are searched concurrently for a query. The non-activated processors are available for inter-query parallelism. The concurrent frame signature file, *CAT* [14], is an extension of frame sliced signature file for current access. It is suitable for processing queries with few query terms. Hamming Filter, proposed

†Recommended by Stavros Christodoulakis

by Zezula et al., is a dynamic signature file for multiple disks [26]. It is an integration of the dynamic partitioning scheme Quick Filter and the static declustering strategy based on the linear error correcting codes. Each signature is allocated using the error correcting code. Then the allocated signatures in the same disk are clustered by Quick Filter. With suitable choice of code parameters, it is guaranteed that Hamming Filter achieves optimal performance. However, in dynamic environment where the size of signature file grows, the declustering effect of Hamming Filter declines. Hamming+ Filter is a generalization of Hamming Filter suitable to manage highly dynamic files. It uses a dynamic declustering, obtained through a sequence of codes, and organizes a smooth migration of signatures between disks [2]. But, Hamming$^+$ Filter can not deal with the case when the number of disks is not a power of two. The *Fragmented Signature File (FSF)*, proposed by Grandi et al. [10], is a frame-sliced partitioned parallel signature file approach on Shared-Nothing architecture of multiprocessor database computers. FSF is based on two signature file structures, the frame-slice approach and the dynamic partitioning scheme Quick Filter. It achieves both the intra-query and inter-query parallelism.

In this paper, we investigate the dynamic allocation of signature files on parallel device. We propose a new allocation scheme *Parallel Signature File (PSF)*. It is an integration of striping, dynamic partitioning and dynamic declustering technique for efficient dynamic organization of signature file. While Hamming Filter uses a static declustering strategy, the proposed PSF uses a dynamic declustering technique with little sacrifice of performance. Although Hamming$^+$ Filter uses the dynamic declustering technique, the proposed PSF is superior to deal with the case when the number of disks is not a power of two. Furthermore, though both PSF and FSF use the dynamic declustering strategy, the declustering effect and the performance of the proposed PSF is superior to that of FSF.

The remainder of this paper is organized as follows. Section 2 reviews the signature file approaches including the Fragmented Signature File. Section 3 describes the proposed dynamic allocation technique. The performance evaluation is described in Section 4. The conclusions and future works are described in Section 5.

2. SIGNATURE FILES

The Signature file access method is widely used in text retrieval. It acts as a search filter to prune most of the unqualified text. In general, the signature file access method can be considered for partial match retrieval whenever the object is characterized by a set of terms. The object may be a document, an image. Partial match retrieval retrieves the objects which contain all the queried terms.

In signature file access method, each object is associated with an object signature. An object signature is produced from the transformation of associated terms of an object. A collection of the object signatures is called a signature file. Query described by a set of specified terms is also transformed to query signature by the same method of object signature generation. After evaluating the query signature against the object signatures in a signature file, most of the impossibly qualified objects are pruned out and the objects corresponding to the qualified signatures are evaluated further. The object whose signature seems to be qualified but actually unqualified is called *false drop* [5]. The process of further evaluation of objects with qualified signatures is called *false drop resolution*. Two main design issues of signature file access method are the signature extraction method and the signature storage structure. Signature extraction method deals with the reduction of false drop probability while signature storage structure deals with the reduction of the number of physical pages that need to be accessed for evaluating the query signature.

	Book0		Book1		Book2	
	Keywords	Term Sig.	Keywords	Term Sig.	Keywords	Term Sig.
	Indexing	100 001	Indexing	100 001	Database	001 001
	Database	001 001	File	100 010	Query	010 001
	Model	010 010	Query	010 001	Security	001 100
Object Signature		111 011		110 011		011 101

Fig. 1: Illustration of Superimposed Coding

The basic types of signature extraction methods include *Word Signature*, *Superimposed Coding*, *Bit-block Compression* and *Run Length Compression* [7]. Above all, Superimposed Coding is the most popular and is the focus of this paper. In Superimposed Coding, each term is hashed into a binary coded word of size F in which m bits have value "1" while others have value "0". These binary coded words are OR-ed together to form the object signature. The number of bits set to "1" in the binary coded word is called the signature weight. If an object signature contains 1s in the same bit positions as the query signature does, then the object signature qualifies for the query signature. Figure 1 illustrates an example of Superimposed Coding applied in the searching of books in the library. In the library, each book is associated with a set of keywords. Users wish to search the book which contains the keywords specified by users. In this example, the signature size F is six bits, term signature weight m is two bits. If the user wishes to retrieve the book which contains keywords "Indexing" and "Query", then the query signature is generated by <100001> OR <010001> which is <110001>. Evaluating the query signature against the three object signatures, we get the qualified signatures, object signatures of Book0 and Book1. After false drop resolution, only Book1 is actually qualified while Book0 is a false drop.

Approaches for the storage structure of signature file includes sequential, *Bit-sliced*, *Frame-sliced*, *S-Tree*, *Quick Filter*, etc. To describe the storage structure of signature file, a signature file of sn-number of F-bits object signature can be regarded as an $sn*F$ bit matrix. The storage structure of signatures can be classified into the following [9].

(1) Sequential Signature File stores the signature matrix sequentially row by row.

(2) Vertical fragmentation stores the signature matrix column-wise and improves the response time. There are Bit-sliced approach and Frame-sliced approach.

(3) Horizontal fragmentation groups similar signatures together and provides an index on the signature matrix. There are three approaches, 2-level signature file, S-Tree and partitioned approaches. Partitioned approaches include static partitioning [12] and dynamic partitioning, *Quick Filter* [25].

Above all, Quick Filter is economical in space and is very efficient in dealing with large files of dynamic data and high weight query signature [17, 25]. Quick Filter uses linear hashing to group the signatures into pages. Signatures with the same suffix will be grouped together. Only the pages of signatures with qualified common suffix are retrieved. Therefore, the search space can be reduced. The length of the suffix is determined by current level of hashing. By the property of linear hashing, Quick Filter can dynamically organize the signatures in dynamic environment. The common suffix of each signature page may be regarded as the key of the signatures in each page. In the following, when the term "*signature key*" is mentioned, it denotes the common suffix of the signatures stored in a partitioned signature page.

Figure 2 shows the result after partitioning six signatures by Quick Filter. In this example, the capacity of a page is assumed to be two signatures. All signatures in a page have the same suffix, in this case the size of two bits. Given the query signature <010001>, only pages with the common suffix <01> and <11> are retrieved. This is done by evaluating the 2-bit suffix of query signature against that of signature pages. The pages with suffix <00>, <10> can not possibly contain the qualified signatures.

Frame-sliced signature file is an approach for extracting and organizing signatures [13]. In this approach, the signature file is divided into fn frames of consecutive bits. Unlike the Superimposed Coding, the bit positions to be set by each term are restricted. Each term of the object is hashed into one of the frames. Another hashing function sets some bits of the specified frame to 1. This produces the penalty of higher false drop probability than that of Superimposed Coding. Therefore, the false drop probability is expected to grow with increasing number of frames. The signatures are stored frame-wise, using fn frame files. For a query with a single term, only one frame file needs to be searched. And with TQ number of query terms, at most TQ frame files are searched. Given fixed TQ, the disk access time of signatures increases with increasing number of frames. Therefore, there is a tradeoff for selecting design parameter fn. Figure 3 gives the Frame-sliced file approach of the example in Figure 1. The number of frames fn is 2 and the frame size is 3 bits. If the queried keywords are "Indexing" and "Query", then the query signature is generated by <101 000> OR <110 000> which is <111 000>. Because keywords "Indexing" and "Query" are hashed into the first frame, only the first frame signature file R_0 needs to be accessed for query signature evaluation.

Note that there is another type of frame-sliced approach, in which the bit positions to be set are not restricted. It is a generalization of Bit-sliced approach while each slice contains more number of bits. Grandi et. al. called the frames of the Frame-sliced file approach *clustered frames* while that of the generalization of Bit-sliced approach *unclustered frames* [10].

111100	010001 000101	011110 110110	000011
00	01	10	11

Fig. 2: Clustering of 6 Signatures by Quick Filter

	Book0		Book1		Book2	
	Keyword	Term Sig.	Keyword	Term Sig.	Keyword	Term Sig.
	Indexing	101 000	Indexing	101 000	Database	000 101
	Database	000 101	File	000 110	Query	110 000
	Model	011 000	Query	110 000	Security	000 011
Object Signature		111 101		111 110		110 111
		111 111 110		101 110 111		
		Frame R_0		Frame R_1		

Fig. 3: Illustration of Frame-Siced Signature File

Fragmented Signature File (FSF) is a mixed fragmentation scheme produced by double horizontal fragmentation of vertical partitioned fragments [10]. Fragmented Signature File is based on the Shared-Nothing parallel architecture of multiprocessor database computers. It can also be implemented on other parallel database computer architecture. Fragemented Signature File is generated as follows:

(Step 1) *Vertical fragmentation*: all the object signatures in a signature file is fragmented vertically into fn frame signature files. R_i (i = 0, 1, 2, ..., fn-1) is the i-th frame file and is served by a cluster of pn_i processing units.

(Step 2) *First horizontal fragmentation*: each frame file R_i is fragmented horizontally into pn_i partitions by Quick Filter. $P_{i,j}$ ($0 \le j \le pn_i - 1$) is the j-th partitions of the i-th frame R_i. Each partition is served by a specific processing unit. Therefore, from now on, the terms 'partition' and 'processing unit' are used interchangeably. Without loss of generality, we assume that $pn_i = 2^{li}$. With Quick Filter, frame signatures with the same li-bits prefix are grouped in partition $P_{i,j}$ and allocated in the same processing unit.

(Step 3) *Second horizontal fragmentation*: each partition $P_{i,j}$ is fragmented horizontally further into $bn_{i,j}$ blocks $B_{i,j,k}$ by Quick Filter ($0 \le k \le bn_{i,j} - 1$), where $2^{hi-1} < bn_{i,j} \le 2^{hi}$, for some integer hi. The value of hi is the level of Quick Filter. $B_{i,j,k}$ is the k-th block of partition $P_{i,j}$. Each block is allocated a disk page and is also the access unit of FSF. So the terms "block" and "page" are used interchangeably. With Quick Filter, the frame signatures of frame file R_i in the same partition $P_{i,j}$ with the same (hi-1) or hi-bits suffix are grouped into block $B_{i,j,k}$.

Figure 4 demonstrates an example of Fragmented Signature File. In this example, the signature file is fragmented vertically into 3 frame signature files R_0, R_1, R_2 (fn = 3). Each frame signature file R_i is fragmented horizontally to 2 partitions $P_{i,0}$, $P_{i,1}$ (pn_i = 2, li = Log_2 pn_i = 1). Each partition $P_{i,j}$ is fragmented horizontally to 4 blocks $B_{i,j,0}$, $B_{i,j,1}$, $B_{i,j,2}$, $B_{i,j,3}$ ($bn_{i,j}$ = 4, hi = Log_2 $bn_{i,j}$ = 2). The binary code word shown in each box represents the li-bits common prefix and hi-bits common suffix of signatures stored in the block. For example, 0___10 represents the common prefix <0> and common suffix <10> of the signatures shown in blocks $B_{0,0,2}$, $B_{1,0,2}$, $B_{2,0,2}$. Given a signature <1011 0101 1010>, frame signatures <1011>, <0101>, <1010> are allocated in $B_{0,1,3}$ of $P_{0,1}$, $B_{1,0,1}$ of $P_{1,0}$, $B_{2,1,2}$ of $P_{2,1}$, respectively.

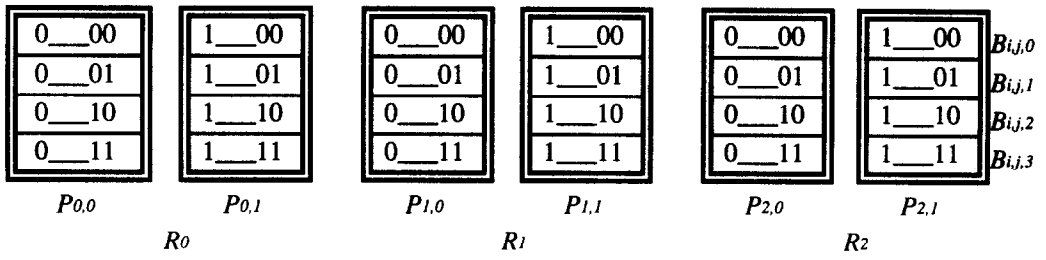

Fig. 4: An Allocation Example of FSF

In general, the advantages in FSF are stated as follows:

(1) Given the same number of processing units for each frame file and the same frame size, the load is balanced among clusters of processing units.

(2) In query processing, the first horizontal fragmentation discards the unqualified partitions and achieves inter-query parallelism. For the example in Figure 4, given query signature <1011 0101 1010>, only partitions $P_{0,1}$, $P_{1,0}$, $P_{1,1}$, $P_{2,1}$ need to be activated, i.e. there are no qualified frame signatures in other partitions. These non-activated partitions are available to serve other queries and the inter-query parallelism can be achieved. Furthermore, if the signatures are generated as frame-sliced signatures, the degree of inter-query parallelism can be promoted. In this case, only the processing unit allocating hit frames are activated.

(3) The second horizontal fragmentation discards the unqualified blocks in each activated partition and reduces the search space. For the same example in the above paragraph, in $P_{0,1}$, only block $B_{0,1,3}$ needs to be accessed for query evaluation. In $P_{1,0}$, blocks $B_{1,0,1}$ and $B_{1,0,3}$, in $P_{1,1}$, blocks $B_{1,1,1}$ and $B_{1,1,3}$, in $P_{2,1}$, blocks $B_{2,1,2}$ and $B_{2,1,3}$ are accessed.

3. DYNAMIC ALLOCATION SCEME

3.1. Proposed Parallel Signature File

Disk striping and declustering are two popular techniques for distribution of data in parallel device. Disk striping is a general purpose facility for achieving parallel data I/O and for achieving fault tolerance [20]. Disk striping tries to store each accessed object as a one-dimensional stream among the disks. Declustering technique tries to distribute the qualified objects among the disks.

The Fragmented Signature File is an integration of striping and partitioning technique. However, the performance of FSF can be improved further by increasing the degree of intra-query parallelism. Figure 5 shows an allocation example of a frame signature file R_0 after vertical fragmentation in FSF. In this example, the frame signature file R_0 is partitioned horizontally into 4 partitions. Frame signature with the same 2-bit prefix are allocated in the same processing unit. Each partition is partitioned horizontally into 4 blocks. Frame signatures in a processing unit with the same suffix are allocated in the same disk block. Given the frame signature of query signature with prefix <01> and suffix <10>, then only partitions $P_{0,1}$, $P_{0,3}$ are activated. In each of these two partitions, there are two blocks activated. The activated blocks are blocks $B_{0,1,2}$ and $B_{0,1,3}$ in partition $P_{0,1}$, blocks $B_{0,3,2}$ and $B_{0,3,3}$ in partition $P_{0,3}$. These accessed blocks are shadowed in Figure 5. The response time of the query frame signature is 2 units of disk block access time.

If the allocation example of Figure 5 is replaced by that shown in Figure 6, it is obvious that the query response time is one unit of disk block access time and is less than that of Fragmented Signature File. The qualified blocks are $B_{0,0,1}$ in $P_{0,0}$, $B_{0,1,3}$ in $P_{0,1}$, $B_{0,2,3}$ in $P_{0,2}$, $B_{0,3,1}$ in $P_{0,3}$. They are distributed evenly in the processing units allocated for this frame file.

Before describing the proposed Parallel Signature File (PSF), the definitions of symbols used in PSF and FSF are listed in Table 1 for convenience.

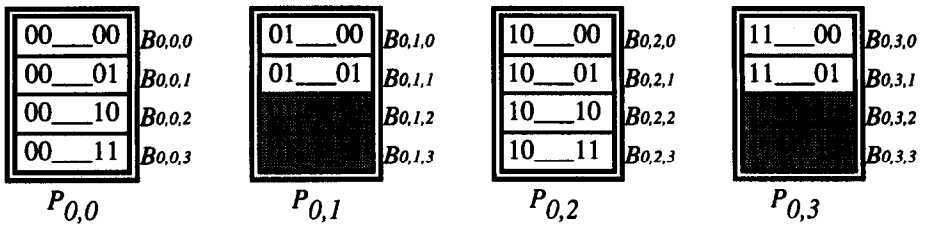

Fig. 5: An Allocation Example of Frame Signature File R_0 of FSF

Symbol	Method(s)	Definition
R_i	Both	The i-th frame signature file
$P_{i,j}$	Both	The j-th partition (processing unit) of frame R_i
$B_{i,j,k}$	Both	The k-th block of partition (or processing unit) $P_{i,j}$
fn	Both	Number of frames
pn_i	Both	Number of partitions (processing units) in the i-th frame
i	Both	Index of frame
j	Both	Index of partition (processing unit)
k	Both	Index of block
Q	Both	query signature
RQ_i	Both	The i-th frame of query signature Q
SQ_i	Both	Common suffix of Rq_i
$bn_{i,j}$	FSF	Number of blocks allocated in partition (processing unit) $P_{i,j}$
l_i, h_i	FSF	Number of bits of common prefix, suffix of signature block of frame R_i
S_i	PSF	A block of frame R_i, represented by a r_i-bits vector [s_{r_i} ... s1]
z	PSF	Index of bit position of [s_{r_i} s_{r_i-1}...sz...s1]
dn_i	PSF	Total number of blocks in the i-th frame
r_i	PSF	Number of bits of common suffix of signature block of frame R_i
$G(S_i, pn_i, r_i)$	PSF	Disk allocation function
$H(S_i, pn_i, r_i)$	PSF	Block location function
u_i	PSF	Logpn_i
sp_i	PSF	pointer of frame R_i, which designate the block to be split

Table 1: Definitions of the Symbols

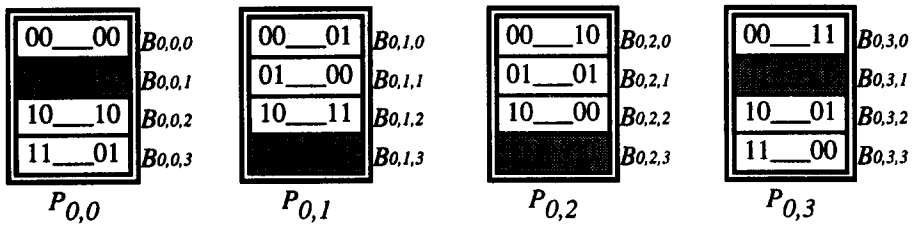

Fig. 6: An Improvement Example of FSF

The proposed Parallel Signature File (PSF), similar to Fragmented Signature File, is based on the Shared-Nothing parallel architecture of multiprocessor database computers. It can also be implemented on other parallel database computer architecture or multiple disk system such as disk array. PSF is generated as follows:

(Step 1) *Vertical fragmentation:* all the object signatures in a signature file is fragmented vertically into *fn* frame files, R_i ($i = 0, 1, 2, ...fn-1$). *fn*, the number of frames, is a design parameter. Each frame is served by a cluster of *pni* processing units.

(Step 2) *Horizontal fragmentation*: each frame file R_i is partitioned horizontally into dn_i blocks by Quick Filter ($2^{r_i-1} < dn_i \leq 2^{r_i}$). The value of r_i depends on the level of linear hashing function of Quick Filter. Each block is allocated in a disk page and is the access unit of PSF. Frame signatures with the same (r_i - 1)-bits or r_i-bits suffix are grouped together. Furthermore, we define the common key of each partitioned block as the r_i-bit common suffix of frame signatures in the same block. Each block is represented by S_i= $<s_{r_i}\, s_{r_i-1} \ldots\, s_1>$, where $s_z = 0$ or 1, $1 \leq z \leq r_i$.

(Step 3) *Declustering*: these dn_i blocks are distributed uniformly in a cluster of pn_i processing units allocated to frame R_i by a declustering technique. For a given partitioned block $<s_{r_i}\, s_{r_i-1} \ldots s_1>$, the index j of allocated processing unit $P_{i,j}$ is determined by the following disk allocation function,

$$j = G(S_i, pn_i, r_i) = (\sum_{z=1}^{r_i} s_z w_z) \text{ MOD } pn_i, \text{ where } w_z = 2^{(z-1 \text{ MOD } \log_2 pn_i)}. \quad (1)$$

Figure 7 shows an example of allocation of frame signature file R_0. The binary code word shown in each box represents the r_0-bits common suffix of signatures stored in the block. For example, ___1011 represents the 4 bits common suffix <1011> of block $B_{0,1,2}$. In this example $r_0 = 4$, $pn_0 = 4$, $\log_2 pn_0 = 2$, $w_1 = 1$, $w_2 = 2$, $w_3 = 1$, $w_4 = 2$. Note that the proposed Parallel Signature File groups frame signature R_i by r_i-bit common suffix, instead of l_i-bit prefix and h_i-bit suffix. In fact, given the same design parameters, the r_i-bit common suffix of blocks in Parallel Signature File may be regarded as the concatenation of the l_i-bit prefix and h_i-bit suffix in Fragment Signature File. In FSF, the distribution of blocks considers only the h_i-bit suffix by the nature of Quick Filter. While in PSF, all the (l_i + h_i) bits are considered.

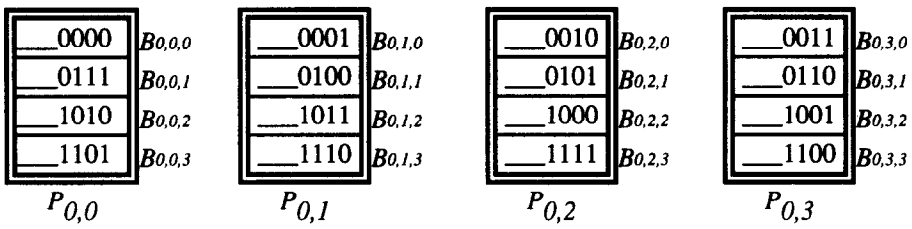

Fig. 7: An Allocation Example of Frame Signature File R_0 of PSF

When pn_i is not a power of two, $\log_2 pn_i$ in Equation (1) is replaced by $\lfloor \log_2 pn_i \rfloor$ or $\lceil \log_2 pn_i \rceil$. If $\lfloor \log_2 pn_i \rfloor$ nears $\log_2 pn_i$, then $\lfloor \log_2 pn_i \rfloor$ is chosen. Otherwise, $\lceil \log_2 pn_i \rceil$ is chosen to replace $\log_2 pn_i$.

The disk allocation function is borrowed from the declustering technique for binary Cartesian product file with a little modification [3]. This function can be understood as follows. Let u_i be equal to log pn_i. Then $w_1 = 2^0$, $w_2 = 2^1$, ...,$w_{u_i} = 2^{u_i-1}$, $w_{u_i+1} = 2^0$, $w_{u_i+2} = 2^1$, $w_{u_i+3} = 2^2$, ..., $w_{r_i} = 2^{(r_i-1 \text{ MOD } u_i)}$. It is a cyclic sequence with cycle length u_i. The disk allocation function

$$G(S_i, pn_i, r_i) = (\sum_{z=1}^{r_i} s_z w_z) \text{ MOD } pn_i = (\sum_{x=1}^{\lceil r_i / u_i \rceil} d_x) \text{ MOD } pn_i, \quad (2)$$

where $<d_{\lceil r_i/u_i \rceil} \ldots d_2\, d_1>$ is the base-2^{u_i} representation of S_i. For example, given pn_i=8, r_i=5, then u_i=3, w_1=1, w_2=2, w_3=4, w_4=1, w_5=2. The base-8 representation of <11010> is <32>. The frame signature block <11010> is allocated in processing unit (3+2) MOD 8 = 5 of the i-th frame.

Theorem 1 *Let C_n be the set, $\{z|(z-1)\ MOD\ u_i = n, z=1,2, ..., r_i\}$, where $0 \le n \le u_i-1$. For each query frame signature in which exactly u_i bits of suffix are set to 0, one from each C_n, all the qualified blocks are uniformly distributed among pn_i processing units.*

Proof. Without loss of generality, we can assume that these u_i bits are bit 1, 2, ..., u_i, then the blocks need to be accessed are the set { $<11..1s_{u_i}...s_1>$ $|s_z=0$ or 1, for z = 1, 2, ..., u_i}. Let t be equal to $\sum_{z=u_i+1}^{r_i} 1 * w_z$, which is a constant for each accessed blocks. Each of these blocks is allocated to processing unit $P_{i,j}$ where

$j = (\sum_{z=u_i+1}^{r_i} 1 * w_z + \sum_{z=1}^{u_i} s_z * w_z)\ MOD\ pn_i = (t + \sum_{z=1}^{u_i} s_z * w_z)\ MOD\ pn_i$.

Because $w_1 = 2^0$, $w_2 = 2^1$, ..., $w_{u_i} = 2^{u_i-1}$ is a cyclic sequence with cycle length u_i, the set of $\{(\sum_{z=1}^{u_i} s_z * w_z) | s_z = 0 \text{ or } 1, \text{for } z = 1,2,...u_i\}$ gives the set of pn_i numbers {0, 1, 2,..., pn_i-1}. These pn_i blocks are allocated to processing units (*t*+0) MOD pn_i, (*t* +1) MOD pn_i, ..., (*t* + pn_i -1) MOD pn_i respectively. Therefore, they are uniformly distributed among the pn_i processing units. □

To evaluate a query signature, the query signature Q is fragmented into fn frame signatures RQ_i (i = 0, 1, 2, ..., fn-1). Let SQ_i be the r_i-bits common suffix of RQ_i. Then, for each frame signature RQ_i, the frame signature blocks of frame R_i activated are the set { $S_i \square S_i \wedge SQ_i = SQ_i$ }.

Besides the query evaluation, the insertion and deletion of signatures all involve the access of frame signature blocks. The location of the accessed frame signature block S_i can be derived as follows. S_i is located in block $B_{i,j,k}$ of processing unit $P_{i,j}$ of frame R_i where $j = G(S_i, pn_i, r_i)$. The index value k of $B_{i,j,k}$ can be derived from

$$k = H(S_i, pn_i, r_i) = \sum_{z=u_i+1}^{r_i} s_z \cdot 2^{z-u_i-1}, \text{where } u_i = \log pn_i. \quad (3)$$

For example, given $pn_i = 4$, $r_i = 4$, $S_i = <1101>$ is allocated to block $B_{i,0,3}$ of processing unit $P_{i,0}$ of frame R_i. The derivation of this formula comes from the dynamic feature of PSF , which is stated in the following subsection.

3.2. Dynamic Features of Parallel Signature File

In dynamic environment, signatures may be inserted or deleted frequently. Storage design of signature file must be suitable for dynamic environment. The proposed Parallel Signature File has the dynamic feature that the blocks can be expanded or contracted with frequent insertion or deletion of signatures, without reorganization of the blocks.

In the construction of PSF, the number of frames fn, is a design parameter that must be predetermined. The vertical partitioning step does not influence the dynamic organization of PSF. The horizontal fragmentation and declustering steps concern the dynamic feature of PSF. In the horizontal fragmentation step, Quick Filter partitions frame signatures by linear hashing. Linear hashing is a dynamic hashing mechanism that the range values of hashing function can change dynamically. Hashing function of linear hashing maps the key into one of the n primary pages where $2^{r-1} < n \le 2^r$. The value of r is the level of hashing. If the primary page to be stored overflows, the inserted key is stored in an overflow page linked to the inserted primary page. Besides, occurrence of this overflow triggers a page splitting. Note that the splitting page needs not be the page which overflows. Instead, pages are split in sequential order. A pointer sp is used to designate the primary page to be split next. Pointer sp is incremented by one after page splitting. When sp is equal to 2^{r-1}-1, sp is reset to zero and the value of r is incremented. This completes a run of full expansion.

The mechanism of PSF for dynamic environment consists of a pair of (sp_i, r_i) for each frame signature file R_i. sp_i is the splitting pointer while r_i is the level of hashing. sp_i is an integer value which designates

the splitting frame signature block. The block, whose integer value of the common (r_i-1) bits equals sp_i, is split next. We can easily determine the splitting block $B_{i,j,k}$ where $j = G(sp_i, pn_i, r_i\text{-}1)$, $k = H(sp_i, pn_i, r_i\text{-}1)$. Frame signatures stored in the designated block and its overflow page are distributed between the block and the newly added block. Frame signatures with suffix $<s_{r_i-1}...s_1>$ are stored in the newly added block while frame signatures with suffix $<s_{r_i-1}...s_1>$ are stored in the newly added block.

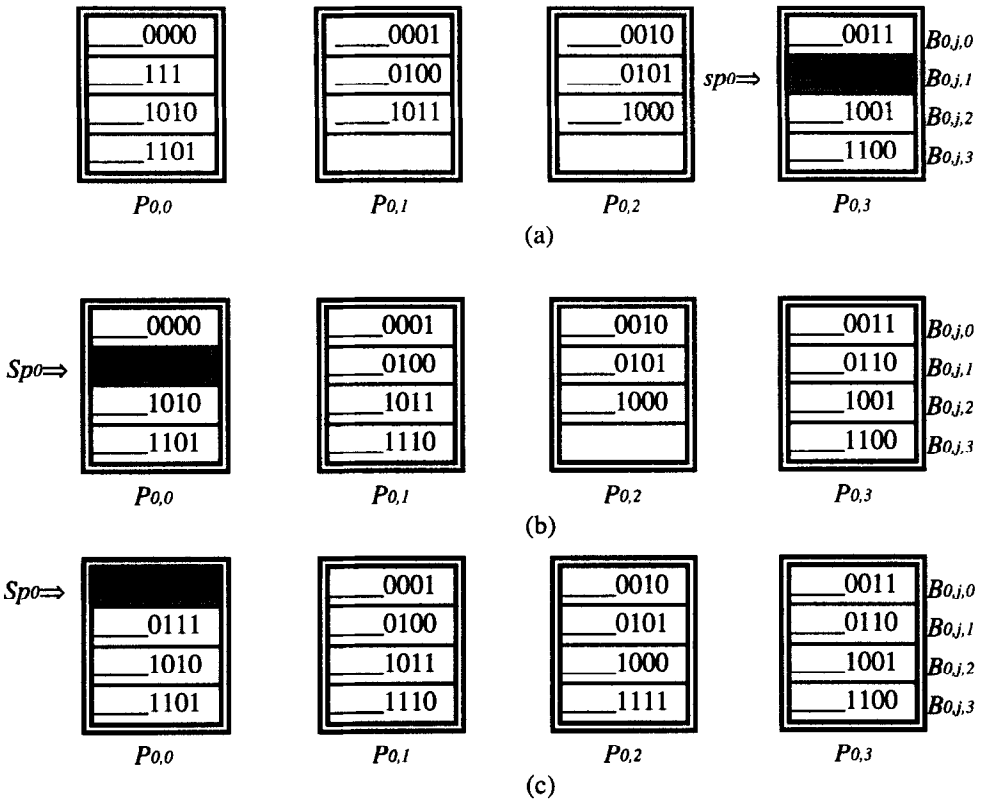

Fig. 8: Example of Dynamic Expansion of Frame Signature R_0 of PSF

Figure 8 demonstrates an example of the expansion process of frame signature file R_0 with 4 processing units. In Figure 8(a), (sp_0, r_0) = (6, 4). $G(sp_0, pn_0, r_0\text{-}1) = G(<110>, 4, 3) = 3$, $H(sp_0, pn_0, r_0\text{-}1) = 1$, block $B_{0,3,1}$ (<110>) is expanded and split into $B_{0,3,1}$ (<0110>) and $B_{0,1,3}$ (<1110>). After splitting, (sp_0, r_0) = (7, 4) which is shown in Figure 8(b). Another occurrence of overflow triggers the next expansion. $G(<111>, 4, 3) = 0$, $H(sp_0, pn_0, r_0\text{-}1) = 1$, block $B_{0,1,1}$ (<111>) is expanded and split into $B_{0,1,1}$ (<0111>) and $B_{0,1,3}$(<1111>). Because now sp_0 is equal to 2^{4-1}-1, sp_0 is reset to zero and the value of r_0 is incremented, i.e. (sp_0, r_0) = (0, 5) as shown in Figure 8(c).

Lemma 1 *It is not necessary to relocate the original designated block after block splitting.*

Proof. The reason lies in that $G(<0s_{r_i-1}...s_1>, pn_i, r_i) = (0*w_{r_i} + \sum_{z=1}^{r_i-1} s_z w_z)$ MOD pn_i, which is equal to $G(<s_{r_i-1}...s_1>, pn_i, r_i\text{-}1) = (\sum_{z=1}^{r_i-1} s_z * w_z)$ MOD pn_i. Therefore, it is not necessary to reorganize the storage structure of PSF in dynamic environment. □

Lemma 2 *Let the split block be allocated to $P_{i,t}$, (the t-th processing unit of frame signature file R_i). The newly added block is allocated to the ((t+ w_{r_i}) MOD pn_i)-th processing unit of frame signature file R_i.*

Proof. The split block $<s_{r_i-1}...s1>$ is allocated to $P_{i,t}$. Therefore, $t = G(<1s_{r_i-1}...s1>, pn_i, r_i) = (\sum_{z=1}^{r_i-1} s_z w_z)$ MOD pn_i. The newly added block is allocated to processing unit $P_{i,j}$, where $j = G(<1s_{r_i-1}...s1>, pn_i, r_i)$. Since $G(<1s_{r_i-1}...s1>, pn_i, r_i) = (1*w_{r_i} + \sum_{z=1}^{r_i-1} s_z w_z)$ MOD $pn_i = (w_{r_i} + t)$ MOD pn_i, the newly added block is allocated to $((w_{r_i} + t)$ MOD $pn_i)$-th processing unit of frame signature file R_i. □

Next, we explain the formula of block location function $H(S_i, pn_i, r_i)$.

Definition 1 Let $S_i = <s_{r_i}s_{r_i-1}...s1>$ be the common suffix of frame signature in a block. Let u_i be equal to log pn_i. The *zone* code of S_i is defined as $<s_{r_i}s_{r_i-1}...s_{u_i+1}>$.

For the example in Figure 8(c), $pn_i = 4$, $u_i = 2$, the zone code of <0111> is <01>.

Theorem 2 *For each processing unit $P_{i,j}$ of PSF ($i = 0, 1, ..., fn-1, j = 0, 1, ..., pn_i-1$), the zone codes of frame signature blocks constitute a binary code order when a run of full expansion completes.*

Proof. We prove it by induction on the level of hashing r_i.

The case of $r_i = 1, 2, ..., u_i$ is trivial. In each processung unit, the zone code of this block is null < >. The binary code values of common suffix of these blocks range from 0 to pn_i-1.

Assume that when $r_i = n$, the zone codes $<s_n s_{n-1}...s_{u_i+1}>$ of frame signature blocks $<s_n s_{n-1}...s1>$ in each processing unit $P_{i,j}$ constitute a binary code order. There are $2^n/pn_i = 2^{n-u_i}$ blocks in each processing unit.

Consider $r_i = n+1$. The newly added blocks $<1s_n s_{n-1}...s1>$ in processing unit $P_{i,j}$ are all split from the 2^{n-u_i} blocks in processing unit $P_{i,(j-w_{n+1}) \text{ MOD } pn_i}$ (according to Lemma 2). Because

(1) the zone codes of these split 2^{n-u_i} blocks constitute a binary code order.

(2) the splitting pointer sp_i advances from 0 to 2^n-1. Block with smaller value of zone code is split before that with larger value of zone code. These split 2^{n-u_i} blocks are split in sequential.

We can conclude that the split 2^{n-u_i} blocks and the newly added 2^{n-u_i} blocks constitute a binary code order. □

With Lemma 2, it is easy to derive the block location function $H(S_i, pn_i, r_i)$ which extracts the zone code as the block location of each processing unit.

4. PERFORMANCE ANALYSIS

There exist some criteria for the performance evaluation of PSF, such as query response time and throughput. For the unclustered case of PSF, all the frame signature files need to be accessed. Besides, the number of accessed blocks in each frame signature file is the same and the accessed blocks in each frame are uniformly distributed. This reduces the query response time and, in turn, increases the throughput. Therefore, the query response time to measure the uniformity of workload is a feasible measure. For the clustered case of PSF, only some of the frame signature files need to be accessed. Those inactive processing units allocated for other frames may be served for other requests. Therefore, it is suitable to measure the performance of clustered PSF not only by query response time, but also by throughput. However, the performance of clustered frame signature file depends not only on the storage design, but also on the signature extraction design [14]. Investigation of parallel design of clustered frame signature file concerns the design of signature extraction. The focus of this paper concerns only the storage allocation of Parallel Signature File. The investigation of performance comparison between clustered and unclustered approaches for Parallel Signature File is left as the future work. Therefore, in this paper, we measure the performance by query response time. A lower bound for parallel signature allocation problem is described as follows.

Definition 2 The response time R(Q), given query signature Q, is defined as Max{$N_{P_{i,j}}(Q)$| i=0,1,2,..., fn-1, j =0,1,2,..., pn_i-1 }, where $N_{P_{i,j}}(Q)$ is the number of accessed blocks of processing unit $P_{i,j}$.

Definition 3 An allocation method is strictly optimal for a query signature Q, if the response time R(Q) is equal to $\lceil N(Q) / M \rceil$, where N(Q) is the total number of blocks accessed and M is the number of disks.

Definition 4 A parallel signature allocation method is strictly optimal, if for every query signature Q, it is strictly optimal.

This definition states that an allocation method is strictly optimal if for each query signature, the accessed pages are distributed evenly among the disks.

Symbol	Definition	Value
F	Signature size	2048 bits
Sn	Number of signatures	1024 to 524288
Bs	Block size	16384 bits (2K bytes)
M	Number of processing units	8 to 128
Fn	Number of frames	1 to 32
M	Weight of term signature	10 to 35
TQ	Number of query terms	5 to 40
w(Q)	Weight of query signature Q	
w(SQ_i)	Weight of suffix of query frame signature RQ_i	

Table 2: List of Symbols and Parameters of the Sample Signature File

Several variables may affect the query response time. The total number of signatures, the signature size and the block size determines the total number of blocks. This affects the response time. The signature size and the block size is fixed before the design of signature file. The number of processing units, of course, also affects the response time. The number of frames is another factor. Given a fixed number of processing units, the number of processing units allocated to each frame signature file is dependent on the number of frames. The number of frames affects the number of blocks in each frame signature file. The number of blocks in turn affects the number of bits considered for disk allocation. Besides, it is apparent that the weight of query signature affect the query response time. By the nature of Quick Filter, which is the horizontal partitioning mechanism of PSF, the weight of query signature determines the search space and thus affects the response time. The search space is invariant under different parallel storage structure. However, from Theorem 1, the query signature weight also affects the declustering effect. The query signature weight depends on the term signature weight, the number of query terms and the signature size. Therefore, the term signature weight, the number of query terms affect the query response time. All these parameters are listed in Table 2.

We measure the performance by mathematical analysis. Then we run an experimental simulation to verify the mathematical analysis of average response time. For mathematical analysis, consider a sample signature file with parameters listed in Table 2. We assume that each frame signature file is allocated with the same number of processing units. Besides, the space overhead taken by the OID of each frame signature is neglected.

We compare the average response time among the proposed Parallel Signature File (PSF), Fragmented Signature File (FSF) and the optimal case. The average response time is measured by

$$\sum_{\forall Q} \text{Prob}(Q) * R(Q) = \sum_{\forall Q} \text{Prob}(Q) * \max_{0 \le i \le fn-1} \{R(FQ_i)\}, \quad (4)$$

where Prob(Q) is the probability of occurrence of a query signature Q, R(Q) is the response time of query signature Q, R(FQ_i) is the response time of query frame signature FQ_i.

In general, the probability of occurrence of each query signature is assumed to be the same. The weight of signature is also assumed to be uniformly distributed among the frame signatures. The derivation of average response time of equation (4) can be simplified as

$$\sum_{\forall w} \text{Prob}(FQ^w) * R_{av}(FQ^w), \tag{5}$$

where FQ^w is the query frame signature with suffix weight w, Prob(FQ^w) is the probability of occurrence of FQ^w, $R_{av}(FQ^w)$ is the average response time of FQ^w.

We first derive the probability Prob(FQ^w). Given signature size of F bits, term signature weight of m bits and the number of query terms TQ specified in a query, then the expected weight of query signature w(Q) can be estimated as [7]:

$$w(Q) = F * \left[1 - (1 - M/F)^{TQ}\right]. \tag{6}$$

We first consider the case of unclustered frames. The weight of query signature is expected to be uniformly distributed among the frames. Given the number of frame fn, the expected weight of frame signature

$$w(FQ) = w(Q)/fn\,. \tag{7}$$

The size of frame signature is F/fn bits. Given the number of frames fn, the number of bits of common suffix r_i, the probability $\text{Prob}_{r_i, fn}(FQ_w)$, can be estimated as

$$\text{Prob}_{r_i, fn}(FQ^w) = \frac{C_w^{r_i} * C_{w(FQ)-w}^{F/fn - r_i}}{C_{w(FQ)}^{F/fn}}, \tag{8}$$

C_j^i is the combinatorial function. Equation (8) can be explained as follows. Totally, there are $C_{w(FQ)}^{F/fn}$ combinations of frame signatures with weight w(FQ). And there are $C_w^{r_i} * C_{w(FQ)-w}^{F/fn - r_i}$ combinations of frame signatures in which r_i-bits suffix has w bits set to 1.

For the case of clustered frames, the estimation of expected weight of frame signature is different from that of unclustered frames. The expected number of frames that qualify for a query composed of TQ terms can be computed as an application of Cardenas's formula in the following way [10]:

$$\text{Ht}_{fn}(Q) = fn * \left[1 - (1 - 1/fn)^{TQ}\right]. \tag{9}$$

$\text{Ht}_{fn}(Q)$ estimates the number of frames selected by TQ terms with replacement assuming that the probability of each frame being selected is the same. With the assumption that TQ query terms are uniformly distributed among the selected $\text{Ht}_{fn}(Q)$ frames, the expected frame weight can be estimated as

$$w(FQ) = \frac{F}{fn} * \left[1 - (1 - \frac{w(Q)}{F/fn})^{TQ/\text{Ht}_{fn}(Q)}\right]. \tag{10}$$

Combining equation (10) and equation (8), we can derive the probability of weight of common suffix of clustered query frame signature.

Finally, $R_{av}(FQ^w)$, the average response time of query frame signature FQ^w, is estimated by physically collect the result from the query evaluation of the sample frame signature file. The sample frame signature file contains frame signature block. Each frame signature block is represented by r_i-bits common suffix. The value r_i depends on the number of signatures, block size and the signature frame size (which in turn depends on the number of frames). The block access time of overflowing pages is ignored. For each possible query frame signature FQ^w with suffix weight w, we get the response time R(FQ^w) for PSF and FSF respectively. $R_{av}(FQ^w)$ is derived by dividing the summation of R(FQ^w) by the total number of FQ^w.

Figure 9 through Figure 12 show the performance comparison of the unclustered signature files among the proposed Parallel Signature File, Fragment Signature File and optimal response time. From the analysis, it can be seen that PSF always outperforms FSF. Besides, performance of PSF is near optimal response time. In Figure 9, we can also observe that the average response time of PSF improves with decreasing number of frames. When the number of frames is one, the response time is minimum. This is because when the number of frames is one, more bits are considered for disk allocation. Therefore, the best design parameter for the number of frames is one.

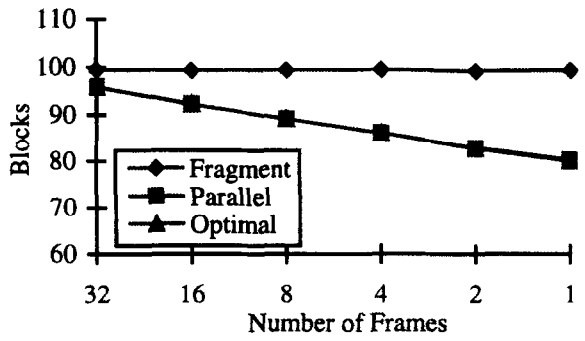

Fig. 9: Average response time versus number of frames, unclustered, signature size = 2048, term signature weight = 15, query terms = 10, number of signatures = 65536, number of processing units=64

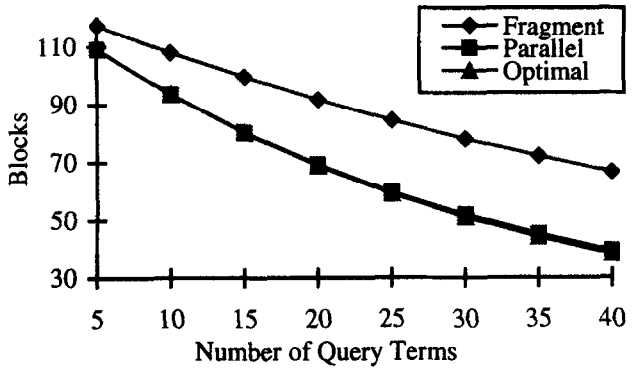

Fig. 10: Average response time versus number of query terms, unclustered, signature size = 2048, term signature weight = 10, number of frames = 1, number of signatures = 65536, number of processing units = 64

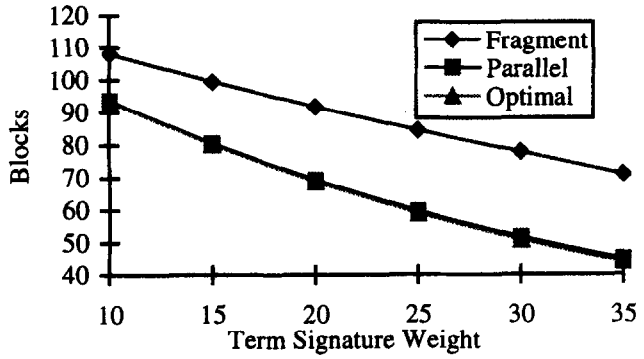

Fig. 11: Average response time versus term signature weight, unclustered, signature size = 2048, number of query terms = 10, number of frames = 1, number of signatures = 65536, number of processing units = 64

Figure 10 compares the query response time as the function of the number of query terms. Increasing number of query terms produces increasing query signature weight and, in turn, decreasing number of accessed blocks. From Figure 10, it is obvious that PSF performs well with increasing number of query terms.

Figure 11 depicts the response time as a function of weight of term signature. It is well know that giving the value of signature length F and the number of terms per object D, the optimal value of term signature weight m, that minimizes the false drop probability is derived from the following formula [7]:

$$F*\ln 2=m*D. \quad (11)$$

The values of weight in Figure 11 ranges from 10 to 35 which implies that the number of terms ranges from 40 to 140. A typical value of the number of terms is about 40 in traditional text document [5, 25]. From Figure 11, we can observe that the response time decreases with increasing weight and decreases with decreasing number of terms per object for a given value of signature length and minimum false drop probability.

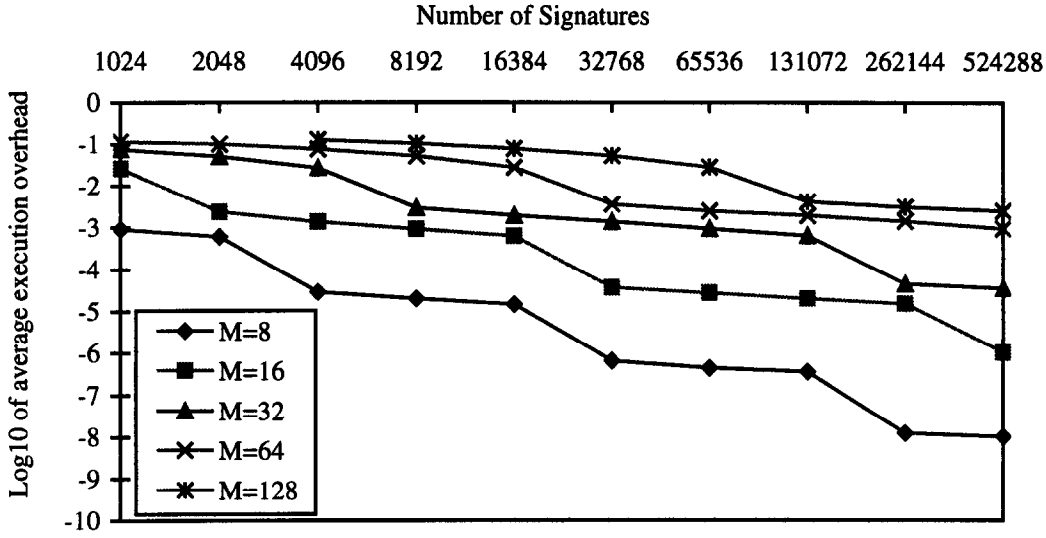

Fig 12: Average execution overhead versus number of signatures for different number of processing units, unclustered, signature size = 2048, number of query terms = 10, number of frame = 1, term signature weight = 10

Figure 12 shows the average execution overhead of query response time as the function of the number of signatures for different number of processing units. The *execution overhead* of average response time is defined as $(R_{PSF}-R_{opt})/R_{opt}$, where R_{PSF} is the average response time of PSF, R_{opt} is the average optimal response time. Note that in this figure, the value of y-axis denotes the logarithm of average execution overhead. Moreover, there exist the cases in which R_{PSF} equals R_{opt} (sn = 1028, 2048 for M = 128). Hence the average execution overhead is zero. Because it is meaningless for logarithm of zero, there are no values plot in the figure corresponding to these cases. From Figure 12, it is observed that the declustering effect improves with increasing number of signatures. This is the advantage of PSF comparing to Hamming Filter. As we have indicated earlier, although Hamming Filter obtains better declustering effect, the performance depends on suitable choice of parameters of error correcting codes. When the number of signatures grows, the performance of Hamming Filter declines. Moreover, one interesting phenomenon in Figure 12 is the curve of average execution overhead for M processing units. It is a cyclic sequence with cycle length log(M/fn) while fn, the number of frames is chosen to be one. In each cycle, the average execution overheads are almost the same. For example, in Figure 12, the curve for 16 processing units is a cyclic sequence with cycle length four. This reason lies in that, in Equation 1 (the disk allocation function), w_z, $1 \le z \le r_i$, is a cyclic sequence with cycle length log(pn_i) for the i-th frame, where pn_i is the number of processing units in the i-th frame.

Figure 13 through Figure 16 show the performance comparison for clustered signature files. Again, PSF outperforms FSF and is not far from optimal response time. Figure 13 demonstrates the effect of the number of frames on query response time. In Figure 13, in contrast to the unclustered signature file, it is observed that larger number of frames is desirable. This is because increasing number of frames produces increasing suffix weight of frame signature. Increasing suffix weight allows greater selectivity of the quick filter mechanism in the horizontal partitioning in both methods. In unclustered case, the number of frames has little effect on the suffix weight of frame signature owing to the uniform distribution of weight among frames.

However, the false drop probability of clustered frame signature file increases with increasing number of frames. The optimal number of frames must be resolved by physical analysis. Therefore, we take arbitrary number of frames, 8. Figure 14 and 15 measure the effect of the number of query terms and term signature weight. The performance of PSF also performs well with increasing number of query terms and

increasing term signature weight. Figure 16 measures the performance as a function of file size and number of processing units. We can see that even in the clustered case, the performance of PSF is not far from optimal despite of the file size or the number of processing units.

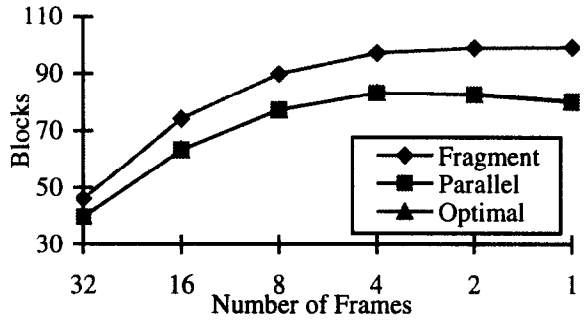

Fig. 13: Average response time, versus number of frames, clustered, signature size = 2048, term signature weight = 15, number of query terms = 10, number of signatures = 65536, number of processing units = 64

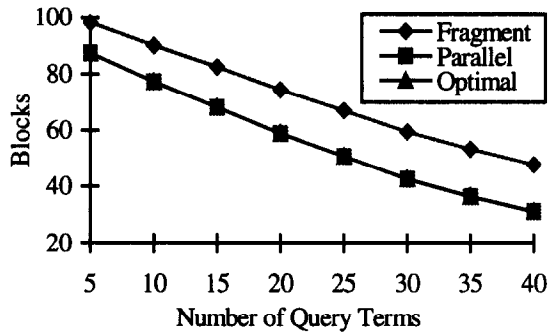

Fig. 14: Average response time versus number of query terms, clustered, signature size = 2048, term signature weight = 15, number of frames = 8, number of signatures = 65536, number of processing units = 64

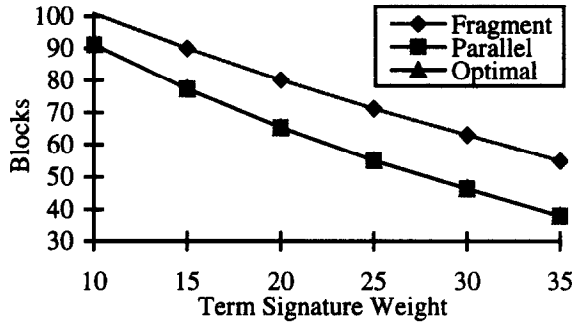

Fig. 15: Average response time versus term signature weight, clustered, signature size = 2048, number of query terms = 10, number of frames = 8, number of signatures = 65536, number of processing units = 64

Figure 16 shows the logarithm of average execution overhead of query response time as the function of the number of signature for different number of processing units. In this figure, there are no curves for 8 and 16 processing units, because the average execution overheads are all zero. From Figure 16, it is also observed that the declustering effect improves with increasing number of signatures. Similar to the unclustered case, the curve of average execution overhead for M processing units is a cyclic sequence with cycle length $\log(M/fn)$ while fn, the number of frames is chosen to be eight. For example, in Figure 16, the curve for 32 processing units is a cyclic sequence with cycle length two.

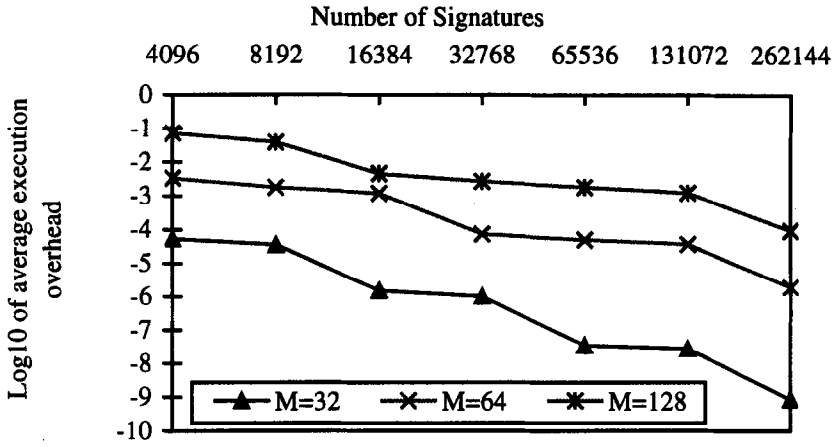

Fig. 16: Average execution overhead versus number of signatures for different number of processing units, clustered, signature size = 2048, number of query terms = 10, number of frames = 8, term signature weight = 10

In order to verify the mathematical analysis of average response time, we have implemented a simplified experimental system. In the experiment, *sn* signatures of size 2048 bits are generated by sampling 40 terms from a vocabulary of 10000 terms (key words). There are *M* disks and the disk page size is 2K bytes. The average response time is measured over a sample of 5000 queries. Each query signature is generated by uniformly choosing *TQ* terms from the vocabulary. For the unclustered case, in order to minimize the false drop probability, the term signature weight is chosen to be 35. For the clustered case, the weight of term frame signature is 10. The execution overhead of average response time is observed. Remember that the execution overhead is defined as $(R_{PSF}-R_{opt})/R_{opt}$, where R_{PSF} is the average response time of PSF, R_{opt} is the average optimal response time.

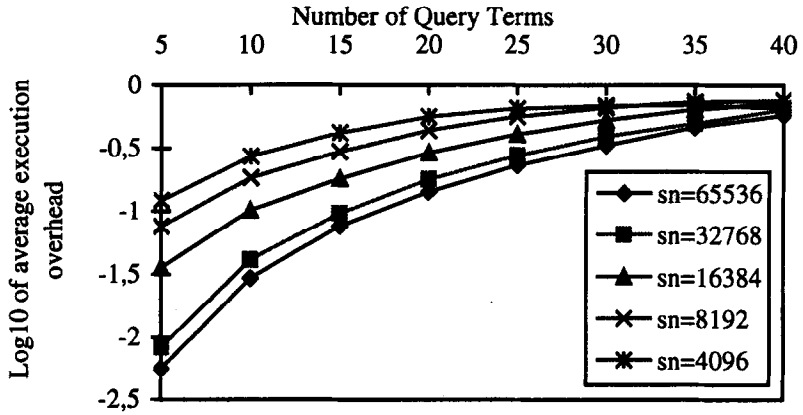

Fig. 17: Average execution overhead versus number of query terms for different number of signatures, unclustered, signature size = 2048, number of processing units = 64, term signature weight = 35

Figure 17 shows the logarithm of execution overhead of query response time as the function of the number of query terms for different number of signatures. It is observed that the execution overhead increases with increasing number of query terms. With a typical value of 5 terms query, the execution overhead is about 10^{-2}. In Figure 18, the effect of file size for the number of processing units is demonstrated. It is observed that the overhead decreases with increasing file size. This is one of the advantage of the proposed PSF. Besides, the cyclic phenomenon of the mathematical analysis shown in Figure 12 also occurs in Figure 18. The only difference lies in the scale of overhead. The reason comes from that the term signature weight of the experiment is chosen to be higher than that of mathematical analysis. Figure 18 also presents the effect of different number of processing units. The phenomenon is similar to that in Figure 12.

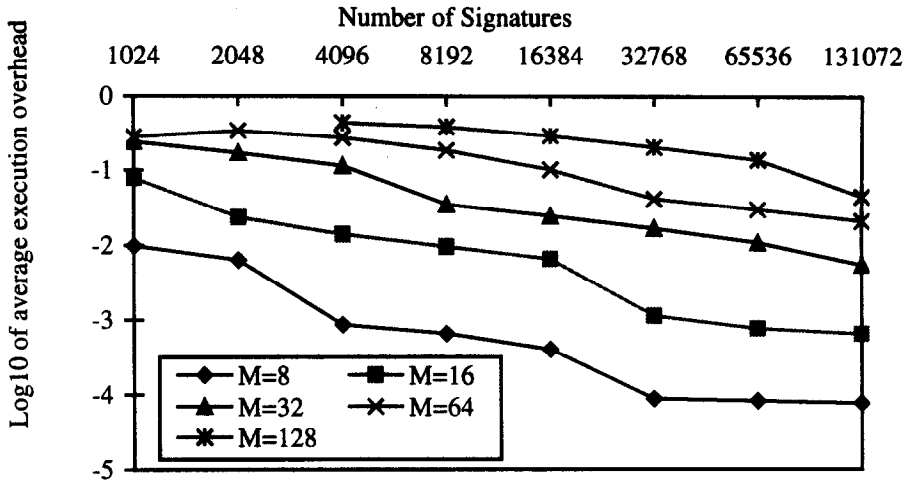

Fig. 18: Average execution overhead versus number of signatures for different number of processing units, unclustered, signature size = 2048, number of query terms = 10, term signature weight = 35

As we have presented, the advantage of proposed approach, superior to Hamming$^+$ Filter, is the ability to deal with cases when the number of processing units is not a power of two. Therefore, the result of observation for these cases is plotted in Figure 19. In Figure 19, it is observed that the peaks of the curves occur when the number of processing units is 11. This can be explained as follows. $^{\log_2 11}$ equals to 3.4597 while $^{\log_2 12}$ equals to 3.5853. Therefore, $\lfloor {}^{\log_2 11} \rfloor$ is chosen to replace $^{\log_2 11}$ in the former while $\lceil {}^{\log_2 12} \rceil$ is chosen to replace $^{\log_2 12}$ in the latter. Besides, although the overhead is higher when the number of processing units is not a power of two, the response time is also near optimal.

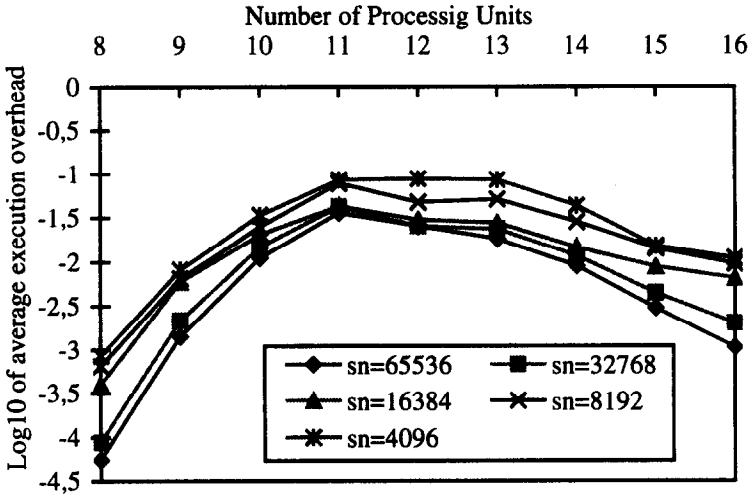

Fig. 19: Average execution overhead versus number of processing units for different number of signatures, unclustered, signature size = 2048, number of query terms = 10, term signature weight = 35

Figure 20, 21 and 22 show the execution overhead of query response time for the clustered case. Figure 20 illustrates the execution overhead as the function of the number of query terms for different number of signatures. The execution overhead increases with increasing number of query terms. Besides, in Figure 20, the curves for 4096, 8192 and 16384 signatures form one group while the other curves form the other group. This phenomenon could be explained by Figure 21. In Figure 21, the curve for 64 processing units is a cyclic curve with cycle length 3. For example, the overhead for 4096, 8192 and

16384 signatures are nearly the same. The reason why the cycle length is has been explained previously. In fact, the same phenomenon also occurs in the unclustered case except that the grouping effect is not so obvious.

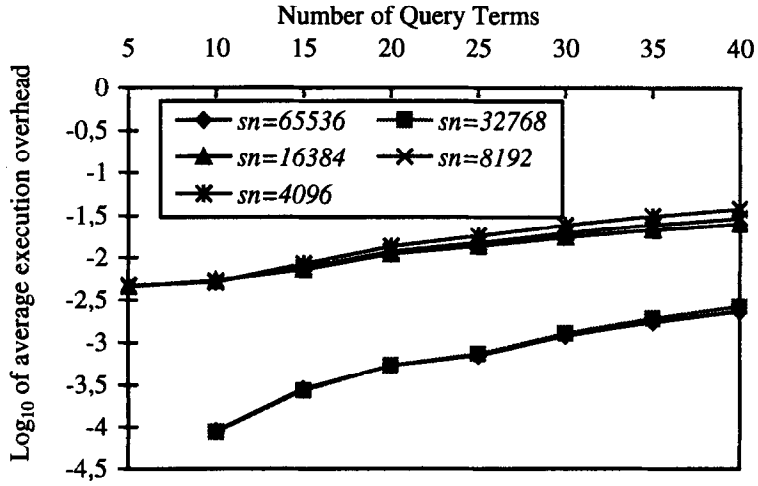

Fig. 20: Average execution overhead versus number of query terms for different number of signatures, clustered, signature size = 2048, number of processing units = 64, number of frames = 8, term signature weight = 10

Figure 21 presents the effect of number of number of signatures for different number of processing units. Similar to the result of mathematical analysis, the curve of average execution overhead for M processing units is a cyclic sequence with cycle length $\log(M/fn)$. Besides, given a specific number of signatures, the overhead increases with increasing number of processing units. Figure 22 observes the overhead when pn_i, the number of processing units allocated to each frame i, is not a power of two. Similar to the unclustered case shown in Figure 19, the peaks occur when pn_i equals 11 (88 processing units/ 8 frames).

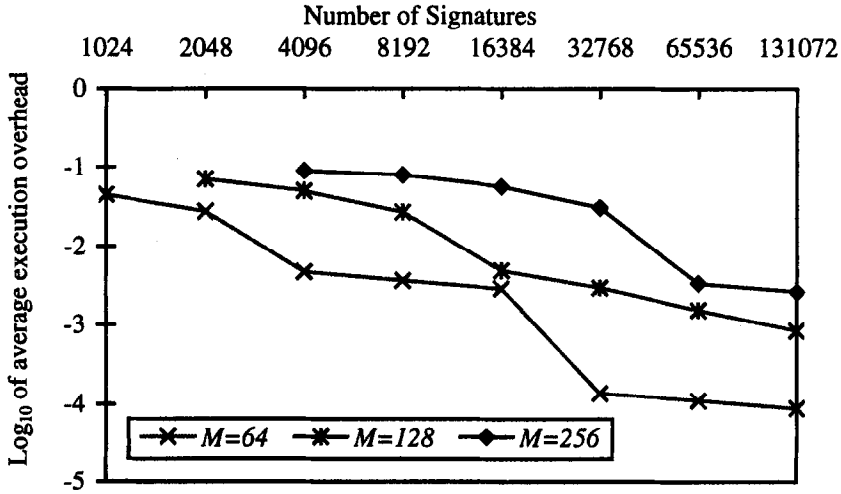

Fig. 21: Average execution overhead versus number of signatures for different number of processing units, clustered, signature size = 2048, number of query terms = 10, number of frames = 8, term signature weight = 10

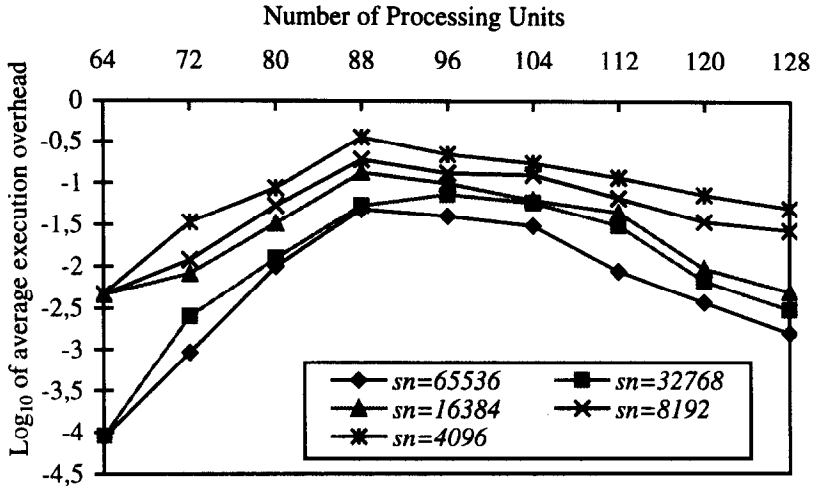

Fig. 22: Average execution overhead versus number of processing units for different number of signatures, clustered, signature size = 2048, number of query terms = 10, number of frames = 8, term signature weight = 10

5. CONCLUSIONS

Signature file access method is efficient for content-based retrieval of unformatted data, such as text, image. In large servers, parallel device can be used to meet the bandwidth requirements. Using parallel device with concurrent access, the response time of accessing signature file can be reduced and therefore the query processing time is reduced also.

Efficient placement of partitioned signature pages in parallel device will speed up query response time. We propose the dynamic storage management method, Parallel Signature File, using declustering technique. It is an improvement of previous approach of Fragmented Signature File. We have presented the disk allocation function and the block location function. The dynamic mechanism for controlling splitting sequence is also described.

The proposed allocation scheme has some advantages. The accessed blocks are distributed more uniformly than those of Fragmented Signature File. Parallel Signature File can also be used in dynamic environment. Besides, when the number of processing units is not a power of two, the proposed approach also performs well. Furthermore, the blocks in each processing unit can be clustered to minimize the disk random access time. For each processing unit, we may cluster the blocks based on the technique of Gray code [22].

The performance analysis shows that the proposed Parallel Signature File outperforms the Fragmented Signature File. Especially the improvement increases with increasing number of query terms. The performance analysis shows that in the unclustered case, the optimal number of frames is suggested to be one. That is, it is not preferable to divide the signatures into frame signatures. In the clustered case, the optimal number of frames must also consider the growing false drop probability. Analysis is required to resolve the optimal value.

Acknowledgements — We thank the anonymous referees for their valuable comments and suggestions.

REFERENCE

[1] L.F. Chien, Fast and quasi-natural language search for gigabits of Chinese texts. In *Proceedings of the 18th ACM International Conference on Research and Development in Information Retrieval SIGIR'95*, pp. 112-120, Seattle, WA (1995).

[2] P. Ciaccia, P. Tiberio and P. Zezula, Declustering of key-based partitioned signature files. *ACM Transactions on Database Systems*, **21**(3):295-338 (1996).

[3] H.C. Du. Disk allocation methods for binary Cartesian Product files. *BIT*, **26**(1):138-147 (1986).

[4] S. Christodoulakis and C. Faloutsos. Design considerations for a message file server. *IEEE Transactions on Software Engineering*, **10**(2):201-210 (1984).

[5] C. Faloutsos and S. Christodoulakis. Signature files: an access method for documents and its analytical performance evaluation. *ACM Transactions on Office Information System*, **2**(4):267-288 (1984).

[6] C. Faloutsos. Access methods for text. *ACM Computing Surveys*, **17**(1):49-74 (1985).

[7] C. Faloutsos and S. Christodoulakis. Description and performance analysis of signature file methods for office filing. *ACM Transactions on Office Information System*, **5**(3):237-257 (1987).

[8] C. Faloutsos and D. Metaxas. Declustering using error correcting codes. *In Proceedings of the 8th ACM SIGACT-SOGMOD-SIGART Symposium on Principles of Database Systems*, pp. 253-258, Philadelphia, PA (1989).

[9] C. Faloutsos. Signature-based text retrieval methods, a survey. *IEEE Computer Socity Technical Committee on Data Engineering. Special issue on document retrieval*, **13**(1):25-32 (1990).

[10]F. Grandi, P. Tibertio and P. Zezula. Frame-sliced partitioned parallel signature files. *In Proceedings of the 15th ACM International Conference on Research and Development in Information Retrieval SIGIR'92*, pp. 286-297, Denmark (1992).

[11]D.L. Lee. A word-parallel, bit-serial signature processor for superimposed coding. *In Proceedings of the 2th IEEE International Conference on Data Engineering*, pp. 352-359, Los Angeles, CA (1986).

[12]D.L. Lee and C. Leng. Partitioned signature files, design issues and performance evaluation. *ACM Transactions on Office Information Systems*, **7**(2):158-180 (1989).

[13]Z. Lin and C. Faloutsos. Frame-sliced signature files. *IEEE Transactions on Knowledge and Data Engineering*, **4**(3):281-289 (1992).

[14]Z. Lin. Concurrent frame signature files. *Distributed and Parallel Databases*, **1**(3):231-249 (1993).

[15]G. Panagopoulos and C. Faloutsos. Bit-sliced files for very large text databases on a parallel machine architecture. In *Proceedings of the EDBT'94*, pp. 379-392, Cambridge, UK (1994).

[16]S.Y. Lee and M.K. Shan. Access method of image database. *International Journal of Pattern Recognition and Artificial Intelligence*, **4**(1):27-44 (1990).

[17]F. Rabitti and P. Zezula. A dynamic signature technique for multimedia databases. *In Proceedings of the 13th ACM International Conference on Research and Development in Information Retrieval SIGIR'90*, pp. 193-210, Belgium (1990).

[18]K. Ramamohanarao and J. Shepherd, A superimposed codeword indexing scheme for very large prolog databases. In *Proceedings of the 3rd International Conference on Logic Programming*, pp. 569-576, London, UK (1986).

[19]C.S. Roberts. Partial match retrieval via the method of the superimposed codes. *Proceeding of IEEE*, **67**(12):1624-1642 (1979).

[20]K. Salem and H. Garcia-Molina. Disk stripping. In *Proceedings of the 2th IEEE Conference on Data Engineering*, pp. 336-342, Los Angeles, CA (1986).

[21]M.K. Shan and S.Y. Lee. Placement of signature file for parallel retrieval of image database. In *Proceedings of the Storage and Retrieval for Image and Video Databases II*, IS&T/SPIE Symposium on Elec. Imaging Sci. & Tech., San Jose, CA (1994).

[22]M.K. Shan and S.Y. Lee. Clustering of partitioned signature file. *Information Science-An Intrenational Journal*, **104**(3/4):321-344 (1997).

[23]M.K. Shan S.Y. Lee. Dynamic allocation of signature file for multimedia document using parallel devices. In *Proceedings of the 18th ACM International Conference on Research and Development in Information Retrieval SIGIR'95*, Poster Session, Seattle, WA (1995).

[24]S. Stanfill and B. Kahle. Parallel free-text search on the Connection machine system. *Communications of the ACM*, **29**(12): 1229-1239 (1986).

[25]P. Zezula, F. Rabitti and P. Tiberio. Dynamic partitioning of signature files. *ACM Transactions on Information Systems*, **9**(4):336-369 (1991).

[26]P. Zezula, P. Ciaccia and P. Tiberio. Hamming filter: a dynamic signature file organization for parallel stores. In *Proceedings of the 19th International Conference on Very Large Data Bases*, VLDB'93, pp. 314-326, Dublin, Ireland (1993).