JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 15, 199-215 (1999)

Allocation of Signature File on Parallel Device For WWW Index Servers

MAN-KWAN SHAN AND SUH-YIN LEE
Institute of Computer Science and Information Engineering
National Chiao Tung University
Hsinchu, Taiwan 300, R.O.C.
E-mail: shan@cc.nctu.edu.tw

Signature files are widely used in information retrieval and database. They act as search filters for content-based retrieval. In a large database server, a parallel device is utilized to achieve concurrency access. Efficient allocation of signature files on parallel devices minimizes the query response time and is important in the design of access methods for large scale index servers. We have developed an algorithm to organize the storage of signatures in parallel secondary storage to reduce the query response time. First, signature file is clustered into signature pages. Then, the clustered signature pages are distributed among the disks using the parity check matrix of error correcting code in coding theory. Through the construction of error correcting code, the least frequently simultaneously accessed pages are allocated on the same disk. Performance analysis shows that this algorithm improves the efficiency of access.

Keywords: signature file, error correcting code, disk allocation, information retrieval

1. INTRODUCTION

Signature files are widely used in information retrieval for text databases [1, 2], partial match retrieval for traditional formatted databases, search filters for main memory databases [3], and subpicture query for image databases [4, 5]. They act as search filters to reduce the amount of data that needs to be searched. Especially, signature files are considered to be the most promising approach to retrieval of large Chinese document databases [6, 7].

However, in large database servers, disk access becomes a bottleneck. Parallel secondary storage devices in which data can be accessed concurrently are utilized. The distribution of the signature files on a parallel device may improve performance significantly. Effective distribution of signature files must allocate the signatures which are pertinent to the query as uniformly as possible. This will minimize the query response time by increasing parallelism. Improper allocation of signature files on parallel disks will put a load on some of the disks and increase the query response time. Therefore, the focus of this paper is the design of signature file allocation on parallel disks.

Few researches have paid attention to the design of signature files on parallel machine architectures. Stanfill proposed a parallel signature file developed by the Thinking Machines Corporation for high-speed interactive querying of text databases on an SIMD computer, called the Connection Machine [8]. However, it was based on the assumption that main memory was sufficient to hold all the text signatures. To deal with this problem,

Received March 12, 1997; accepted November 3, 1997.
Communicated by Jang-Ping Sheu.

Panagopoulos et al. proposed a parallel bit-sliced signature file method on an SIMD machine [9]. 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 [10]. Only some partitions are searched concurrently for a query. The non-activated processors are available for inter-query parallelism. The Fragmented Signature File (FSF) is a frame-sliced partitioned parallel signature file approach on the Shared-Nothing architecture of multiprocessor database computers [11]. FSF is based on two signature file structures, the frame-slice approach and the dynamic partitioning scheme, called Quick Filter.

In this paper, we investigate the distribution of signature files on parallel devices. Using the parity check matrix of the error correcting code, the least frequently and simultaneously accessed pages are allocated on the same disk while the most frequently simultaneously accessed pages are distributed among diverse disks. We first review the signature file approaches in Section 2. Section 3 presents the proposed disk allocation technique. Performance evaluation is described in Section 4. Conclusions are given in Section 5.

2. SIGNATURE FILE

The signature file access method is widely used as a filter in text retrieval. It is used for content-based retrieval whenever each data object is characterized by a set of terms [2]. Content-based retrieval retrieves those objects which contain all the queried terms. In the signature file access method, each object is associated with an object signature. An object signature is produced through the transformation of terms contained in the object. A collection of object signatures is called a signature file. A query described by a set of queried terms is also transformed into a query signature using the same method of object signature generation. Queries are answered using a two-stage searching process. First, after evaluating the query signature with the object signatures in the signature file, most of the impossibly qualified objects are pruned out. Then, in the second stage, objects corresponding to the qualified signatures are further evaluated. An object whose signature seems to be qualified but is actually unqualified called a *false drop* [2]. The process of further evaluating of objects with qualified signatures is called *false drop resolution*.

The performance of the signature file method depends on efficient design of both stages. Therefore, corresponding to the two stages, two main design issues of the signature file access method are the signature storage structure and the signature extraction method. The signature storage structure deals with the reduction of the number of accessed physical pages to evaluate the query signature (the first stage) while the signature extraction method deals with the reduction of false drop probability (the second stage).

The most popular type of signature file is Superimposed Coding [12]. In Superimposed Coding, each term is hashed into a binary codeword of size F, in which m bits have a value of "1" while others have a value of "0". The binary codeword is called the *term signature*. These term signatures are OR-ed together to form the object signature. The number of bits set to "1" in the binary codeword is called the *signature weight*. If an object signature contains 1s in the same bit positions as does the query signature, then the object signature qualifies for a query signature. Fig. 1 shows an example of Superimposed Coding applied to the searching of books in a library. In the library, each book is associated

	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 System	100 010	Query Language	010 001
	Data Model	010 010	Query Language	010 001	Security	001 100
Object Signature		111 011		110 011		011 101

Fig. 1. Superimposed coding.

with a set of keywords. Users wish to search the books which contain the keywords specified by the users. In this example, the signature size F is six bits, and the term signature weight m is two bits. If the user wishes to retrieve a book which contains the keywords "Indexing" and "Query", then the query signature is generated by <100001> OR <010001> which is <110001>. After evaluating the query signature against the three object signatures, we get the qualified signatures, the object signatures of Book0 and Book1. After false drop resolution, only Book1 is actually qualified while Book0 is a false drop.

To reduce the search space for evaluation of query signatures, several approaches to the storage structure of signature files have been proposed. Above all, Quick Filter is economical in storage space and is very efficient in dealing with large files of dynamic data and high weight query signatures [13]. Quick Filter uses linear hashing to partition signatures into pages. Signatures with the same suffix are grouped together. The evaluation time can be reduced by first comparing the common suffix of pages with the suffix of query signatures. Only the pages of signatures with a qualified common suffix are retrieved. The size of the suffix is determined by the current level of hashing. Based on the property of linear hashing, Quick Filter can dynamically organize the signatures in a dynamic environment. Fig. 2 shows the result after partitioning six signatures using Quick Filter. In this example, the capacity of a page is assumed to be two signatures. All the signatures in a page have the same suffix, and in this case it has 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 comparing the 2-bit suffix of query signature with that of the signature pages. Also the pages with suffix <00>, <10> cannot contain qualified signatures.

	111100	010001 000101	011110 110110	000011
signature pages	00	01	10	11

Fig. 2. Clustering of 6 signatures into 4 signature pages using Quick Filter.

The common suffix of each signature page may be regarded as the key of the signatures in that page. It can uniquely identify the signature page. In the following, the term "*signature key*" denotes the common suffix of the signature page. Furthermore, the terms "*signature key*" and "*signature page*" are used interchangeably.

3. PROPOSED DISK ALLOCATION METHOD

Intuitively, similar signatures are likely to be accessed simultaneously. The similarity between two signatures can be measured by the number of bit positions in which they differ. Therefore, similar signatures must be clustered together to reduce the number of disk pages accessed. On the other hand, similar signatures must be distributed (declustered) among the disks to achieve concurrent access.

Our design for allocation of signatures consists of two steps:

Step 1: Intrapage clustering: signatures are clustered into partitioned signature pages to reduce the number of disk pages accessed.

Step 2: Disk allocation: the partitioned signature pages are distributed among disks for maximum concurrent access.

As mentioned in the previous section, signature files can be partitioned into pages by Quick Filter. Signatures with the same common suffix are clustered together in a signature page. This accomplishes the first step. We assume that there are 2^n signature pages. If the number of signature pages is not equal to a power of 2, then additional split operations of Quick Filter are needed. The signature pages which have not been split in the current run of expansion are split whether there is overflow or not. This guarantees that the number of signature pages will be a power of 2 and is helpful in the design of disk allocation.

Given the partitioned signature pages, the goal of the next step is to allocate these signature pages among the disks in order to minimize the query response time.

Definition 1: The query response time R(KQ), given query signature key KQ, is defined as Max{$N_0(KQ)$, $N_2(KQ)$, ..., $N_{M-1}(KQ)$}, where $N_i(KQ)$ is the number of qualified signature pages of disk i, and M is the total number of disks.

For example, given 4 partitioned signature pages with signature keys <00>, <01>, <10>, <11>, two different allocation strategies for these pages on two disks are shown in Fig. 3. Given a query signature key, comparison of query response time between these two allocation strategies is listed in Table 1. It is observed that Strategy 2 performs better than Strategy 1 when the query signature key is <10>. Signature pages with signature keys <10>, <11> are qualified. These two pages in Strategy 1 are allocated to the same disk while those in Strategy 2 are evenly distributed between the two disks.

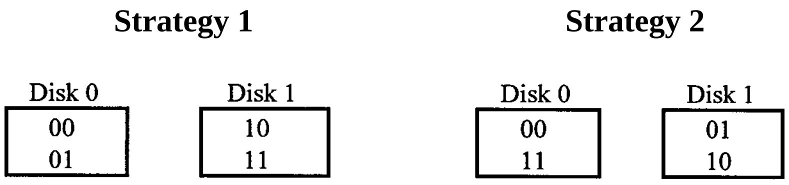


Fig. 3. Two allocations of 4 signature pages.

Table 1. Comparison of the query response time between two allocation strategies.

Query Signature Key	Qualified Pages	Query Response Time	
		Strategy 1	Strategy 2
00	4	2	2
01	2	1	1
10	2	2	1
11	1	1	1

The reason why Strategy 2 performs well lies in the Hamming distance. The Hamming distance in each disk of Strategy 1 is one while that of Strategy 2 is two. The definitions of the Hamming distance are given as follows.

Definition 2: The Hamming distance $d(x, y)$ between two signature keys x and y is defined as the number of positions in which they differ.

Definition 3: The Hamming distance d of a given set C of signature keys is defined as $d=\text{Min}\{d(x, y): x, y \in C, x \neq y\}$.

It is observed that allocation with a larger Hamming distance performs better. The reason is as follows. Allocation with a larger Hamming distance results in more dissimilar signature pages in the same disk. Owing to the characteristics of query signature evaluation, dissimilar signature pages are more unlikely to be accessed simultaneously. Thus, the least frequently and simultaneously accessed pages are allocated to the same disk. This implies that the most frequently and simultaneously accessed pages are distributed among diverse disks.

Therefore, given $M=2^l$ disks numbered as l-bit address $<a_1a_2...a_l>$ and 2^n signature pages each represented as n-bit signature key $<s_1s_2...s_n>$, we seek a linear transformation matrix H from $<s_1s_2...s_n>$ to $<a_1a_2...a_l>$. In other words,

$$H*\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} = [h_1 \quad h_2 \quad \cdots \quad h_n]*\begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} = \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{21} & z_{22} & \cdots & z_{2n} \\ \vdots & & \cdots & \\ z_{l1} & z_{l2} & \cdots & z_{ln} \end{bmatrix} * \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_l \end{bmatrix}. \quad (1)$$

This $l*n$ transformation matrix H must map any pair of signature keys with Hamming distance $\geq d$ to the same disk, where d is a pre-specified distance.

Theorem 1: Let H be the transformation matrix from $<s_1s_2...s_n>$ to $<a_1a_2...a_l>$. Then every $(d$-1) column of the matrix is linearly independent if and only if every pair of signature keys mapped to the same disk has a distance of at least d.

***Proof* :** Consider any two signature keys $<x_1x_2...x_n>$ and $<y_1y_2...y_n>$, which are mapped to the same disk. Then,

$$\sum_{j=1}^{n} z_{ij}x_j - \sum_{j=1}^{n} z_{ij}y_j = \sum_{j=1}^{n} z_{ij}(x_j - y_j) = 0, \forall i = 1,2,...,l. \tag{2}$$

In terms of column vectors,

$$\sum_{j=1}^{n} h_jx_j - \sum_{j=1}^{n} h_jy_j = \sum_{j=1}^{n} h_j(x_j - y_j) = 0 \tag{3}$$

First assume that every set of (d-1) columns of H is linearly independent. If the distance between these two signature keys is less than d, say t, then for n binary digits (x_j-y_j), $\forall j$=1, 2,...,n, t of them are nonzero. Without loss of generality, let these t digits be the 1-th, 2-th, ..., t-th digit. That is,

$$\sum_{j=1}^{t} h_j(x_j - y_j) = 0. \tag{4}$$

Therefore, we have a nontrivial linear combination of less than d columns of H, which sums to zero. This is not possible since by hypothesis, every set of (d-1) columns is linearly independent. Therefore, these two signature keys have a distance of at least d.

Next assume that these two signature keys have a distance of at least d. If $t \le (d$-1) columns of H are linearly dependent (without loss of generality, let these t columns be column 1, 2, ..., t), then there exist t binary scales λ_1, λ_2, ..., λ_t, not all zero, such that

$$\sum_{j=1}^{t} h_j\lambda_j = 0. \tag{5}$$

Let λ_j be equal to (x_j-y_j). Therefore, there exist two signature keys with distance less than d. This is a contradiction since by hypothesis, these two signature keys have a distance of at least d. Therefore, (d-1) columns of H are linearly independent.□

Example 1: Consider the following transformation matrix H from the signature page $<s_1s_2s_3s_4s_5>$ to the disk numbered as $<a_1a_2a_3>$; every pair of signature keys mapped into the same disk has a distance of at least three, and it can be verified that every two columns of matrix H are linearly independent:

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Of course, the value of d is set as large as possible. To seek the transformation matrix H in which (d-1) columns are linearly independent, error correcting code in coding theory is helpful. The parity check matrix of error correcting code shares a common property with the linear transformation matrix of Theorem 1.

The theory of error correcting code mainly deals with the reliable transmission and storage of data. Errors can be detected and corrected by incorporating redundancy into the original data. Fig. 4 shows an example. In this example, 2 bits of information are transmitted. Before transmission, each 2 bits of information is encoded by appending 3 redundant bits.

After transmission, the transmitted message is decoded by a maximum likelihood decoder. Assume that a single error occurs. Now, if the receiver receives <01000>, then it seems reasonable to decode the transmitted message as <00000>. This set of four codewords is an error correcting code, which can detect and correct a single error.

The larger the code distance of an error correcting code the more errors it can detect. Therefore, one major problem in research in error correcting code is to determine how to add this redundancy to messages in order to produce code with a maximum code distance.

Definition 4: An [*n*, *k*, *d*] code is a set of codewords with length *n* and code distance *d*, where the size of information bits is *k*.

The example shown in Fig. 4 is a [5,2,3] code.

Linear code is the most general class of error correcting code. Many error correcting codes in use and under investigation are subclasses of linear code defined by imposing additional structural constraints.

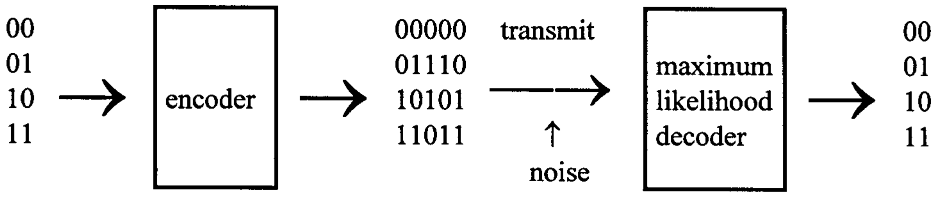


Fig. 4. An example of error correcting code for reliable transmission [14].

Because a signature is a binary string, we emphasize binary linear code only. Let Z_2 denote the finite field formed by integers modulo 2, under the operations of standard modulo 2 addition and multiplication . Also, let $V_n(Z_2)$ denote the vector space of an *n*-bit binary codeword over finite field Z_2.

Definition 5: A binary linear code, an [*n*, *k*, *d*] code, is a *k*-dimensional subspace of $V_n(Z_2)$.

Theorem 2: If *C* is an [*n*, *k*, *d*] code over Z_2, then there exists a $(n-k)*n$ matrix *H* such that for each codeword, $x \in C$, $Hx^T=0$ [15].

Example 2: The parity check matrix for the [5,2,3] code in Fig. 4 is

$$H=\begin{bmatrix}1 & 1 & 1 & 0 & 0\\ 0 & 1 & 0 & 1 & 0\\ 1 & 0 & 0 & 0 & 1\end{bmatrix}.$$

Theorem 3: Let *H* be a parity check matrix for an [*n*, *k*, *d*] code over Z_2. Then, every set of (*d*-1) columns of *H* is linearly independent [14].

From Theorem 3, the parity check matrix *H*, in which every set of (*d*-1) columns is linearly independent, seems to satisfy the requirement of the linear transformation matrix we seek in Theorem 1. However, the linear transformation matrix must transform 2^n *n*-bit signature keys (i.e., the vector space $V_n(Z_2)$) into 2^l disks while in Theorem 3, the parity check matrix only transforms 2^k n-bit codewords (i.e., the k-dimensional subspace $V_n(Z_2)$) into one disk numbered as zero. We must show that the parity matrix indeed works in the same way as does the linear transformation matrix.

Note that C may be regarded as a subgroup of order 2^k in the group $V_n(Z_2)$. C has $2^n/2^k=2^{(n-k)}$ cosets.

Definition 6: The coset of an $[n, k, d]$ code C over Z_2 which contains the element $y\in V_n(Z_2)$ is defined as the equivalent class $[y]=\{x+y|\ x\in C\}$.

Example 3: The coset of the [5,2,3] code in Fig. 4 which contains the element <00001> is the set of codewords {<00001>, <10100>, <01111>, <11010>}.

Lemma 1: According to the definition of a coset, two vectors in the same coset of an $[n, k, d]$ code have a distance of at least d.

Definition 7: Let H be a parity check matrix for an $[n, k, d]$ code over Z_2. For the vector $x\in V_n(Z_2)$, the *syndrome* S of x is defined as $S = Hx^T$.

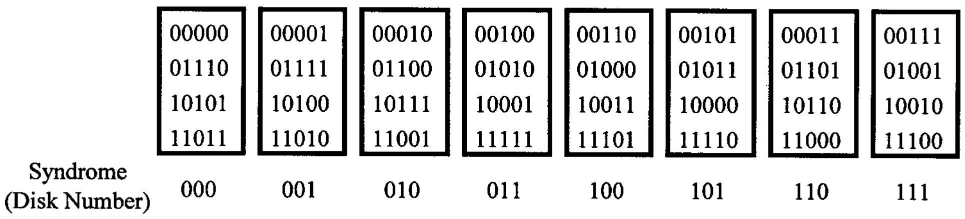
Theorem 4: Let H be a parity check matrix for an $[n, k, d]$ code C over Z_2. Then, two vectors x and y are in the same *coset* of C if and only if they have the same syndrome, that is, $Hx^T=Hy^T$ [14].

Example 4: <00001> and <10100> are in the same coset because

$$\begin{bmatrix}1&1&1&0&0\\0&1&0&1&0\\1&0&0&0&1\end{bmatrix}*\begin{bmatrix}0\\0\\0\\0\\1\end{bmatrix}=\begin{bmatrix}1&1&1&0&0\\0&1&0&1&0\\1&0&0&0&1\end{bmatrix}*\begin{bmatrix}1\\0\\1\\0\\0\end{bmatrix}=\begin{bmatrix}0\\0\\1\end{bmatrix}.$$

According to Theorem 4 and Lemma 1, two vectors with the same syndrome have a distance of at least d. Since the group (also a vector space) $V_n(Z_2)$ is partitioned into $2^n/2^k=2^{(n-k)}=2^l$ subgroups, each subgroup of vectors corresponds to the signature keys of the signature pages stored in a disk. The value of a syndrome may be treated as the disk number. This indicates that the parity check matrix is the same as the linear transformation matrix which transforms 2^n signature keys into 2^l disks. Therefore, to allocate 2^n signature keys into 2^l disks, the parity check matrix of an $[n, n\text{-}l, d]$ error correcting code is used. Fig. 5 shows an example of allocating 2^5 signature pages to 2^3 disks along with the syndrome value using a [5, 2, 3] code. For example, signature pages with signature keys <00000>, <01110>, <10101> and <11011> are allocated to the disk numbered as <000>.

Given the $l*n$ parity check matrix, the computation of a syndrome takes $O(l*n)$ multiplication and addition operations. It is possible to use $O(l*(n-l))$ computation time if the code of this parity check matrix is a cyclic code. Cyclic code is a subclass of linear code. An $[n, k, d]$ cyclic code is generated by a generating polynomial $g(x)$ of degree $(n-k)$, which is a monic divisor of (x^n-1) over Z_2 [15]. For example, the generating polynomial g(x) = 1



	00000	00001	00010	00100	00110	00101	00011	00111
	01110	01111	01100	01010	01000	01011	01101	01001
	10101	10100	10111	10001	10011	10000	10110	10010
	11011	11010	11001	11111	11101	11110	11000	11100
Syndrome (Disk Number)	000	001	010	011	100	101	110	111

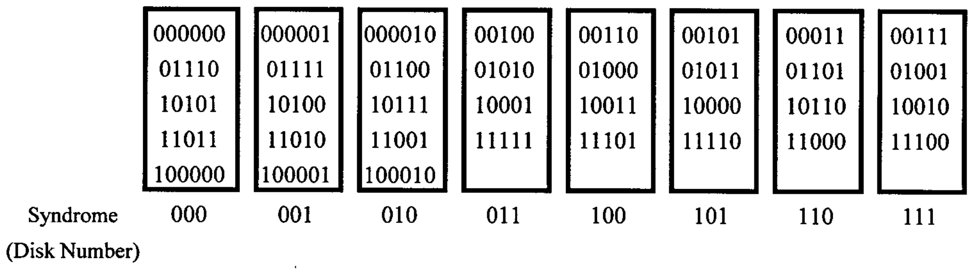
Fig. 5. Allocation of 2^5 signature pages to 2^3 disks.

+ x generates a [5,4,2] code since g(x) is a monic divisor of (x^5-1) of degree (5-4). Given the information bits <1010> expressed as polynomial a(x) = 1 + x^2, it is encoded to the codeword <11110> expressed as polynomial $1 + x + x^2 + x^3 = (1 + x)(1 + x^2)$.

Let c(x) and s(x) be the polynomial representations of a codeword and its syndrome, respectively. Then, s(x) is the remainder polynomial when c(x) is divided by generating polynomial g(x). Therefore, we divide the signature key c(x) by generating polynomial g (x) over Z_2 and examine the remainder s(x). The binary representation of remainder s(x), which is also a polynomial of degree less than l, represents the allocated disk number.

Consider an example in which we allocate 2^7 signature pages to 2^3 disks. We may use a [7,4,3] cyclic code generated by polynomial $g(x) = 1 + x + x^3$. The signature page <1011011> will be allocated to the disk numbered as <001>. This is because $c(x) = 1 + 0x^1 + 1x^2 + 1x^3 + 0x^4 + 1x^5 + 1x^6 = g(x) * (1 + x + x^2 + x^3) + x^2$, $s(x) = 0 + 0x^1 + 1x^2$.

One issue not addressed yet is addition or deletion of signatures. Addition/deletion of signatures may cause signature pages to grow/shrink owing to page overflow/underflow. For example, assume the signature page <00000> is split into two pages <000000> and <100000>. We can allocate these two pages to the original disk. Fig. 6 shows an example. When growth continues and reaches the full run of expansion, the length of all the signature keys increases from n to n+1. Then, either a reorganization is performed using a new transformation matrix or no reorganization is performed by sacrificing efficiency. In the case of the former, the new transformation matrix consists of n+1 columns. In the case of the latter, efficiency declines. The reasons is that the determination of the old transformation matrix is based on only an n-bit signature key.



	000000	000001	000010	00100	00110	00101	00011	00111
	01110	01111	01100	01010	01000	01011	01101	01001
	10101	10100	10111	10001	10011	10000	10110	10010
	11011	11010	11001	11111	11101	11110	11000	11100
	100000	100001	100010					
Syndrome (Disk Number)	000	001	010	011	100	101	110	111

Fig. 6. Allocation of split signature pages.

4. PERFORMANCE ANALYSIS

We have measured the performance of disk allocation based on the average query response time. We have derived expressions for calculating the average response time. Several variables may affect the query response time. The total number of signatures, the signature size, the disk page size and the loading factor determine the total number of pages. The signature size and the page size are fixed before the signature file is designed. The number of disks, of course, also affects the response time. The number of pages, in turn, affects the number of bits (the length of the signature key) considered in disk allocation. Besides, it is apparent that the weight of a query signature affects the query response time. Due to the nature of Quick Filter, which is the intrapage clustering mechanism of proposed allocation scheme, the weight of a query signature determines the search space and, thus, affects the response time. The search space is invariant under different parallel storage structures. However, the query signature weight also affects the disk allocation effect. The query signature weight depends on the term signature weight and the number of query terms and the signature size. Therefore, the term signature weight, the number of query terms affect the query response time.

For performance evaluation, we considered a sample signature file with parameters listed in Table 2. We assumed that there were 64 disks and 40960 2048-bits signatures in the sample signature file. We also assumed that the capacity of the disk page was 2048 bytes, and that the loading factor was 80%. We will first give some definitions related to the average query response time.

Definition 8: The query response time R(Q) , given a query signature Q, is defined as Max $\{N_i(Q) \mid i=0,1,2,..., M-1\}$, where $N_i(Q)$ is the number of accessed pages of disk i, and M is the total number of disks.

Therefore, the average response time is measured by

$$\sum_{\forall Q} \text{Prob}(Q) * \text{R}(Q), \tag{6}$$

Table 2. List of symbols and parameters of the sample signature file.

Symbol	Definition	Value
F	Signature length	2048 bits
sn	Number of signatures	40960
bs	Page size	16384 bits (2048 bytes)
α	Loading factor	80%
M	Number of disks ($M=2^l$)	64
n	Length of signature key	12
m	Weight of term signature	10~30
TQ	Number of query terms	47~140
w(Q)	Expected weight of query signature Q	
kw	Weight of query signature key	

where Prob(Q) is the probability of occurrence of a query signature Q and R(Q) is the response time of query signature Q. In general, the probability of occurrence of each query signature is assumed to be the same as that of any other query signature.

Note that Definition 8 is different from Definition 1. Definition 1 measures the response time of query signature key KQ while Definition 8 measures that of query signature Q. The probabilities of occurrence of query signatures are assumed to be the same while those of query signature keys are not the same.

A lower bound for the disk allocation problem is described as follows.

Definition 9: An allocation method is strictly optimal for a query signature Q if the response time R(Q) is equal to [N(Q) / M], where N(Q) is the total number of pages accessed and M is the number of disks.

Definition 10: An 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. However, the requirement of being strictly optimal is too strict to be attainable in general.

The derivation of the average response time of equation (6) can be simplified as

$$\sum_{\forall kw} \mathrm{Prob}(Q^{kw}) * \mathrm{R}_{\mathrm{av}}(Q^{kw}), \tag{7}$$

where Q^{kw} is the query signature with key weight kw and Prob(Q^{kw}) is the probability of occurrence of Q^{kw}. $\mathrm{R}_{\mathrm{av}}(Q^{kw})$ is the average response time of Q^{kw}.

We will first derive the probability Prob(Q^{kw}). Given a signature size of F bits, a term signature weight of m bits and TQ query terms specified in a query, then the expected weight of query signature w(Q) can be estimated as [13]:

$$\mathrm{w}(Q) = F * [1-(1-m/F)^{TQ}]. \tag{8}$$

Given a signature size F, n bits of common suffix (signature key) and the expected query signature weight of w(Q), the probability Prob(Q^{kw}) can be estimated as

$$\mathrm{Prob}(Q^{kw}) = \frac{C^{n}_{kw} * C^{F}_{\mathrm{w}(Q)-kw}}{C^{F}_{\mathrm{w}(Q)}}, \tag{9}$$

where C^{i}_{j} is the combinatorial function. Equation (9) can be explained as follows. Totally, there are $C^{F}_{\mathrm{w}(Q)}$ combinations of signatures with weight w(Q). Also, there are $C^{n}_{kw} * C^{F}_{\mathrm{w}(Q)-kw}$ combinations of signatures in which the n-bit suffix has w bits set to 1.

Finally, $\mathrm{R}_{\mathrm{av}}(Q^{kw})$, the average response time of query signature key Q^{kw}, is estimated by physically collecting the result from the query evaluation of the sample signature file. In the sample signature file, there are 2^{12} signature pages since 40960*2048/(2*2^{10}*80%) is equal to 2^{12}. Each signature page is represented by a 12-bit common suffix. These 2^{12} signature pages are distributed among 2^6 disks using a 6*12 linear transformation matrix. This linear transformation matrix is the parity check matrix of an [12, 6, 6] cyclic code with generating polynomial $g(\mathrm{x}) = 1 + \mathrm{x} + \mathrm{x}^2 + \mathrm{x}^4 + \mathrm{x}^5 + \mathrm{x}^6$. The page access time of the overflowing pages is ignored. We have measured the performance for all the combinations of query signature keys (i.e., all the 2^{12} query signature keys) according to Definition 1. The average response time was calculated over all query signature keys with the specified key weight. That is,

$$\mathrm{R}_{\mathrm{av}}(Q^{kw}) = \sum_{\forall KQ,\mathrm{w}(KQ)=kw} \frac{\mathrm{R}(KQ)}{\mathrm{n}(KQ)}, \tag{10}$$

where n(KQ) is total number of query signature keys with key weight kw. Table 3 lists the average response time $\mathrm{R}_{\mathrm{av}}(Q^{kw})$ of the query signature keys for each query key weight kw.

Fig. 7 and Fig. 8 show a performance comparison between the average response time of the proposed allocation method and optimal response time. From the analysis, it can be seen that the performance of the proposed method is nearly optimal. Fig. 7 shows a comparison of the response time as a function of the number of query terms, TQ. From Fig. 7, it is obvious that the proposed method performs well with increasing number of query terms. Fig. 8 compares the response time as a function of term signature weight m. It is well know that given signature length value of F and D terms per object, the optimal value of term signature weight m that minimizes the false drop probability can be derived using the following formula [11]:

$$F^{*}\ln 2 = m^{*}D. \tag{11}$$

The values of weight in Fig. 8 range from 10 to 30, which implies that the number of terms ranges from 47 to 141. A typical value of the number of terms is about 40 in a traditional text document [14]. From Fig. 8, we can observe that the response time decreases with increasing weight and an increasing number of terms per object for a given signature length.

In addition, we have run an experimental simulation to verify the mathematical analysis of the average response time. In the experiment, sn signatures of bits with size F were generated by sampling T terms from a vocabulary of 10000 terms (key words). There were

Table 3. Comparison of the average and optimal response time for a query key (64 disks, 2^{12} pages).

Query key weight kw	Average response time $\mathrm{R}_{\mathrm{av}}(Q^{kw})$	Optimal response time	Query key weight kw	Average response time $\mathrm{R}_{\mathrm{av}}(Q^{kw})$	Optimal response time
12	1	1	5	2.23	2
11	1	1	4	4.17	4
10	1	1	3	8	8
9	1	1	2	16	16
8	1.02	1	1	32	32
7	1.24	1	0	64	64
6	1.51	1			

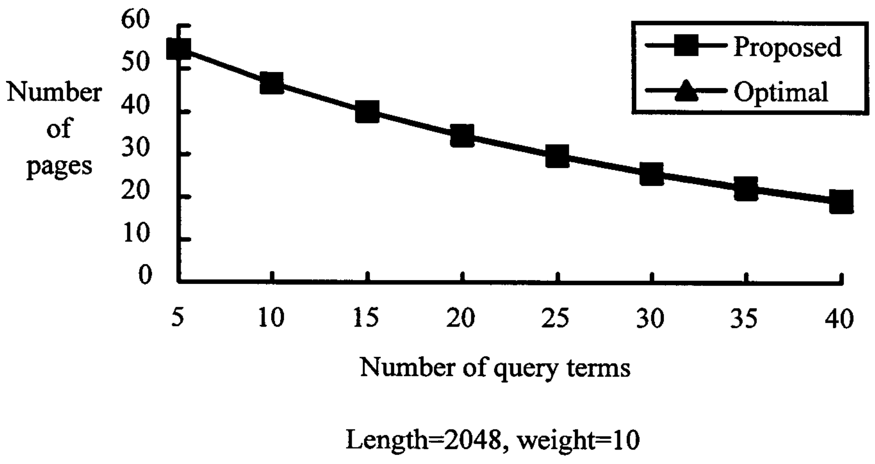


Fig. 7. Average response time versus number of query terms.

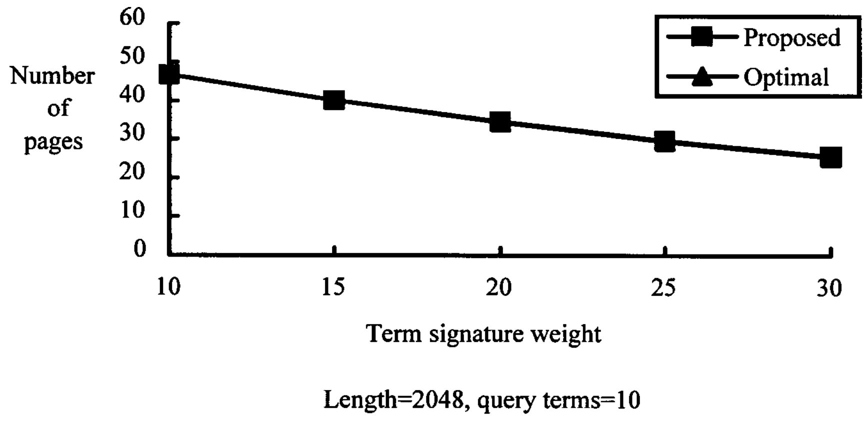


Fig. 8. Average response time versus term signature weight.

M disks and the disk page size was *bs* bytes. The average response time is measured over a sample of 10000 queries. Each query signature is generated by uniformly choosing *TQ* terms from the vocabulary. In order to compare the performance measured by means of simulation and mathematical analysis, we measured the average response time as three types of precision. Let Avg_{sim}, Avg_{math} be the average response time measured by means of simulation and mathematical analysis and let Opt_{sim}, Opt_{math} be the optimal response time measured by means of simulation and mathematical analysis, respectively. Math-Opt precision is defined $(Avg_{math}\text{-}Opt_{math})/Opt_{math}$. Sim-Opt precision is defined as $(Avg_{sim}\text{-}Opt_{sim})/Opt_{sim}$ while Math-Sim precision is defined as $(Avg_{math}\text{-}Avg_{sim})/Avg_{sim}$.

Fig. 9 and Fig. 10 compare these types of precision as a function of the number of query terms and the term signature weight. Note that in these figures, the value of y-axis denotes the base 10 logarithm of precision. In Fig. 9, both types of precision for simulation and mathematical analysis decline with an increasing number of query terms. In Fig. 10, both types of precision for simulation and mathematical analysis also degenerate with increasing term signature weight. The reason lies in the fact that increasing the number of query terms and term signature weight increases the query signature weight. Increasing the query signature weight results in more query signatures with higher-weight query keys. If the query key weight of most of the query signatures falls between 4 and 8, the precision will decline. However, the precision shows that the average response time comes close to the optimal response time. Furthermore, in both figures, the Math-Sim precision shows that the performance result obtained by mathematical analysis is very close to that obtained by experimental simulation.

5. CONCLUSIONS

In this paper, we have proposed a method for allocating signatures file to parallel disks. First, the signature file is clustered into signature pages. Then, the clustered signature pages are distributed among the disks using the parity check matrix of error correcting code. The parity check matrix of an error correcting code [n, n-l, d] transforms 2^n signature pages into 2^l disks such that the hamming distance of the signature pages of the same disk is d. Thus, the least frequently simultaneously accessed pages are allocated to the same disk. This implies that the most frequently simultaneously accessed pages will be

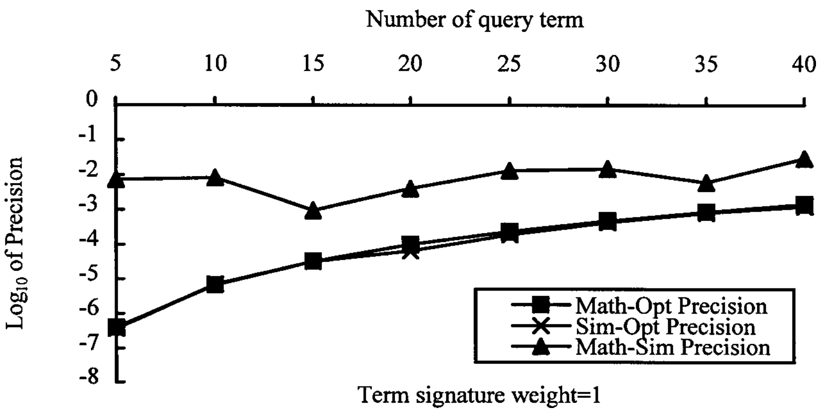


Fig. 9. Precision versus number of query terms.

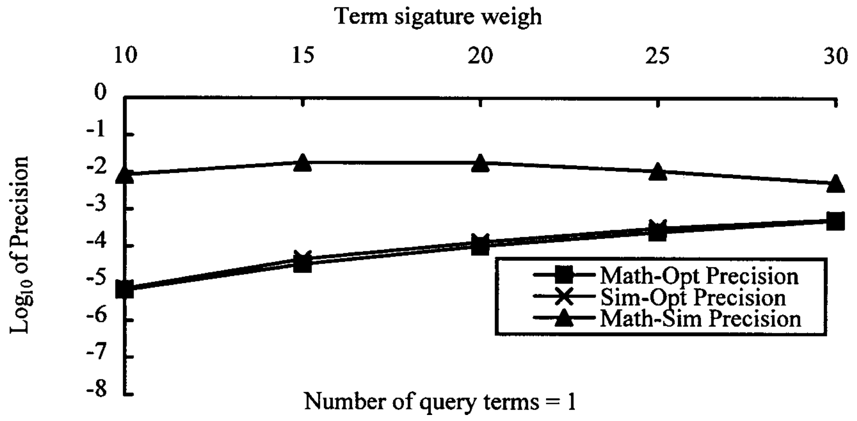


Fig. 10. Precision versus term signature weight.

distributed among diverse disks. We have measured the performance based on the average query response time. The results of performance analysis show that performance of the proposed allocation strategy is not far from optimal.

In this paper, we have assumed that the number of disks M is a power of two. If the number of disks is not a power of two, some strategies may be employed. If the number of disks is a multiple of a power of two, say, $f*2^l$, then the signature file can be stripped to f frame signature files first [16]. Each of the frame signature files can then be clustered and distributed among the 2^l disks. Otherwise, if the number of disks is not a multiple of a power of two, we may consider an allocation strategy where the number of disks is $2^{\lfloor \log_2 M \rfloor}$.

Future research directions include dynamic allocation of signature pages and the allocation policy based on consideration of the characteristics of Chinese text retrieval.

ACKNOWLEDGMENTS

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

REFERENCES

1. C. Faloutsos, “Access method for text,” *ACM Computing Surveys*, Vol. 17, No. 1, 1985, pp. 49-74.
2. C. Faloutsos, “Signature-based text retrieval methods, a survey,” *IEEE Computer Society Technical Committee on Data Engineering*, Vol. 13, No. 1, 1990, pp. 25-32.
3. J. C. R. Tseng and W. P. Yang, “2D random filter and analysis,” *International Journal of Computer Mathematics*, Vol. 42, No. 1, 1991, pp.33-45.
4. S. Y. Lee and M. K. Shan, “Access method of image database,” *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 4, No. 1, 1990, pp. 27-44.
5. M. K. Shan and S. Y. Lee, “Placement of signature file for parallel retrieval of image database,” in *Proceedings of Storage and Retrieval for Image and Video Databases II, Imaging Science & Technology/Society of Photo-optical Instrumentation Engineers Symposium on Electronic Imaging Science & Technology*, 1994, pp. 120-126.

6. T. Liang, S. Y. Lee and W. P. Yang, “On the design of effective Chinese textual retrieval based on the signature method,” *Computer Processing of Chinese and Oriental Language,* Vol. 8, No. 1, 1994, pp. 87-96.
7. L. F. Chien, “Fast and quasi-natural language search for gigabits of Chinese texts,” in *Proceedings of the ACM Special Intereset Group on Information Retrieval ‘95: International Conference on Research and Development in Information Retrieval*, pp. 112-120.
8. S. Stanfill and B. Kahle, “Parallel free-text search on the connection machine system”, *Communication of ACM,* Vol. 29, No. 12, 1986, pp. 1229-1239.
9. G. Panagopoulos and C. Faloutsos, “Bit-sliced files for very large text databases on a parallel machine architecture,” Technical Report CSC-809, Department of Computer Science, University of Maryland, 1992.
10. D. L. Lee, “A word-parallel, bit-serial signature processor for superimposed coding,” in *Proceedings of the 2nd IEEE International Conference on Data Engineering,* 1986, pp. 352-359.
11. F. Grandi, P. Tibertio and P. Zezula, “Frame-sliced partitioned parallel signature files”, in *Proceedings of the ACM Special Interset Group on Information Retrieval ’92: International Conference on Research and Development in Information Retrieval* , 1992, pp. 286-297.
12. C. Faloutsos and S. Christodoulakis, “Description and performance analysis of signature file methods for office filing”, *ACM Transactions on Office Information System,* Vol. 5, No.3, 1987, pp.237-257.
13. P. Zezula, F. Rabitti and P. Tiberio, “Dynamic partitioning of signature files,” *ACM Transactions on Information Systems,* Vol. 9, No. 4, 1991, pp. 336-369.
14. S. A. Vanstone and P. C. van Oorschot, *An Introduction to Error Correcting Codes with Applications,* Kluwer Academic Publishers, 1989.
15. C. Faloutsos and S. Christodoulakis, “Signature files: an access method for documents and its analytical performance evaluation”, *ACM Transactions on Office Information Systems,* Vol. 2, No. 4, 1984, pp. 267-288.
16. K. Salem and H. Garcia-Molina, “Disk striping,” in *Proceedings of IEEE Conference on Data Engineering,* 1986, pp. 336-342.

Man-Kwan Shan (沈錳坤) received the BS degree in computer engineering and the MS degree in computer and information science both from National Chiao Tung University, Taiwan, in 1986 and 1988, respectively. From 1988 to 1990, he served as a lecturer in the Army Communications and Electronics School. Then, he worked as lecturer at the Computer Center of National Chiao Tung University, where he supervised the Research and Development division. He received the Ph. D. degree in Computer Science and Information Engineering from National Chiao Tung University in 1998. He is now an assistant professor in the Department of Computer Science at National Chengchi University. His research interests include multimedia information systems, information retrieval and data mining.

Suh-Yin Lee (李素瑛) received her BSEE degree from National Chiao Tung University, Taiwan, in 1972, and her MS degree in computer science from the Univerisyt of Washington, Seattle, in 1975. She joined the faculty of the Department of Computer Engineering at National Chiao Tung Univerisyt in 1976 and received the Ph. D. degree in electronic engineering there in 1982. Dr. Lee is now a professor in the Department of Computer Science and Information Engineering at National Chiao Tung University. She chaired the department from 1991 to 1993. Her current research interests include multimedia information systems, object-oriented databases, image/spatial databases, and computer networks. Dr. Lee is a member of Phi Tau Phi, ACM, and the IEEE Computer Society.