Performance Analysis of the Generalised Disc Modulo Allocation Method for Multiple Key Hashing Files on Multi-disc Systems

C. C. CHANG*† AND C. Y. CHEN‡

* *Institute of Applied Mathematics, National Chung Hsing University, Taichung, Taiwan 40227, Republic of China*

‡ *Department of Electronics, Private Feng Chia University, Taichung, Taiwan 40724, Republic of China*

In this paper, we discuss the performance of the Generalised Disc Modulo (GDM) allocation method for multiple key hashing (MKH) files on multi-disc systems. A very important performance formula which can be used directly to evaluate the average response time over all possible partial match queries is derived.

Received May 1986, revised October 1986

1. INTRODUCTION

In an information retrieval system, a file is a collection of records. Since a file is usually large and cannot be wholly stored in primary memory, it is generally divided into buckets and stored in discs. Each time we need information about the file, a query is used to retrieve some records from the disc. And each time the disc is accessed, an entire bucket is brought into primary memory. If the whole file is stored on a disc, only one bucket can be accessed at a time. Since the disc access time is considerably larger than the primary memory retrieval time, the time taken to respond to a query can be simply measured in terms of the number of distinct disc accesses. The number of distinct disc accesses that must be issued to respond to a query is equal to the number of buckets which contain at least one record satisfying the query. If a file is stored on several independently accessible discs, several buckets which reside on different discs can be accessed at one disc access time. In this case the response time to a given query is proportional to the maximum number of buckets needed to be examined on a particular disc. It is obvious that different methods of allocating buckets in the file among discs always result in different response time to a given query. Therefore, given a file designed primarily for all possible queries and an m-disc ($m > 1$) system, it is important to allocate all buckets into m discs in such a way that the maximal possible disc access concurrency is achieved when examining the required buckets.

In practice, a record is always characterised by more than one attribute. By a multi-attribute file we mean a file whose records are characterised by more than one attribute. By a partial match query for an N-attribute file we mean a query of the form $(A_1 = a_1, A_2 = a_2, \ldots, A_N = a_N)$, where $1 \leqslant i \leqslant N$ and a_i is either a key belonging to D_i, the domain of the ith attribute, or is unspecified (i.e. a don't-care condition), in which case it is denoted by *. Since speeding up response times of partial match queries is getting more and more important, some researchers recently explored the problem concerning the allocation of a pre-constructed multi-attribute file into multiple disc in such a way that the average response time, over all possible partial match queries, is minimised (i.e. the concurrency of disc access is maximised).[3-5] Chang and Shieh[4] showed that the above-mentioned problem is NP-hard. Du and Sobolewski[5] proposed a heuristic method, called the Disc Modulo (DM) allocation method, to allocate an arbitrarily given Cartesian product file on to m discs ($m > 1$). Although the DM allocation method has good performance, it is not strictly optimal for all possible partial match queries when the number of available discs is greater than three. Du and Sobolewski[5] then proposed a more generalised allocation method called the Generalised Disc Modulo (GDM) allocation method. Recently, Chang and Shen[3] derived a performance formula for the use of a DM method to allocate all buckets of a Cartesian product file on to discs.

In this paper we are concerned with the performance of the GDM allocation method for multiple key hashing (MKH) files. In Section 2 we shall introduce the concept of MKH files. In Section 3 we shall introduce the DM and GDM allocation methods. In Section 4 we shall derive an important performance formula, which can be used directly to evaluate the average response time to all possible partial match queries for the use of a GDM method to allocate a pre-constructed MKH file on to multiple discs. Since the DM allocation method can be viewed as a special case of the GDM allocation method and a Cartesian product file can be viewed as a special MKH file, it will be seen that the formula derived by Chang and Shen is just a particular case of our performance formula.[3]

2. THE MULTIPLE KEY HASHING FILE CONCEPT

The MKH method for multi-attribute file system design was first proposed by Rothnie and Lozano.[7] We shall assume that each record is characterised by N attributes $A_1, A_2, \ldots, A_N$ with corresponding domains $D_1, D_2, \ldots, D_N$.

Definition 2.1: the MKH method[2]

The MKH method can be described as follows:

(1) Choose a hashing function $f_i: D_i \rightarrow \{1, 2, \ldots, m_i\}$ for each domain D_i, where $\prod_{i=1}^{N} m_i = NB$, the total number of available buckets.

(2) Associate each N-tuple $(s_1, s_2, \ldots, s_N)$ with a

† To whom correspondence should be addressed.

Downloaded from https://academic.oup.com/comjnl/article/30/6/535/327613 by guest on 09 April 2024

Table 1

Records (A_1, A_2, A_3)
(19, 80, *B*)
(18, 60, *A*)
(21, 90, *A*)
(20, 70, *B*)
(18, 75, *C*)
(21, 65, *A*)
(20, 85, *A*)
(18, 75, *D*)
(20, 75, *D*)
(18, 60, *C*)
(18, 80, *B*)
(19, 65, *B*)
(21, 85, *A*)
(18, 80, *A*)
(18, 60, *B*)
(18, 55, *C*)
(21, 60, *C*)
(20, 75, *C*)
(19, 75, *B*)
(20, 65, *B*)
(18, 65, *B*)
(20, 60, *C*)
(18, 65, *C*)
(20, 85, *B*)

bucket, where s_i is an integer, $1 \leqslant s_i \leqslant m_i$. For simplicity, we call this associated bucket $[s_1, s_2, \ldots, s_N]$ in the rest of this paper.

(3) If attributes $A_1, A_2, \ldots, A_N$ of some record R have attribute values $a_1, a_2, \ldots, a_N$ respectively, assign R to the bucket $[f_1(a_1), f_2(a_2), \ldots, f_N(a_N)]$, where $a_i \in D_i$.

Definition 2.2: the MKH file

An MKH file is a multi-attribute file based upon the MKH method.

Example 2.1

Let us consider a set of records depicted in Table 1. Each record consists of three attributes: A_1 = age, A_2 = calculus score and A_3 = English grade. In this case we see that $D_1 = \{18, 19, 20, 21\}$, $D_2 = \{55, 60, 65, 70, 75, 80, 85, 90\}$ and $D_3 = \{A, B, C, D\}$.

Let $m_1 = m_2 = m_3 = 2$. We can define the following hashing functions:

$$\begin{aligned} h_1(x) &= 1 \quad \text{if} \quad x = 18, 19 \\ &= 2 \quad \text{if} \quad x = 20, 21 \\ h_2(x) &= 1 \quad \text{if} \quad x = 55, 60, 65, 70 \\ &= 2 \quad \text{if} \quad x = 75, 80, 85, 90 \\ h_3(x) &= 1 \quad \text{if} \quad x = A, B \\ &= 2 \quad \text{if} \quad x = C, D \end{aligned}$$

Through the use of hashing functions h_1, h_2 and h_3 we obtain an MKH file as shown in Table 2.

Bolour[1] explored the optimality properties of MKH functions. Chang[2] explored the problem of designing optimal MKH files with the assumption that, in a partial match query, the probability of an attribute being specified depends on the attribute itself and is independent of other attributes specified in the query. It should be noted that the Cartesian product (CP) file concept introduced in Ref. 5 is, in fact, identical to the MKH file concept. The definition of CP file introduced by Du and Sobolewski[5] is given as follows.

Table 2

(A_1, A_2, A_3)	Bucket	Bucket number
(18, 60, *A*) (18, 60, *B*) (18, 65, *B*) (19, 65, *B*)	[1, 1, 1]	1
(18, 55, *C*) (18, 60, *C*) (18, 65, *C*)	[1, 1, 2]	2
(18, 80, *A*) (18, 80, *B*) (19, 80, *B*) (19, 75, *B*)	[1, 2, 1]	3
(18, 75, *C*) (18, 75, *D*)	[1, 2, 2]	4
(20, 70, *B*) (20, 65, *B*) (21, 65, *A*)	[2, 1, 1]	5
(20, 60, *C*) (21, 60, *C*)	[2, 1, 2]	6
(20, 85, *B*) (20, 85, *A*) (21, 85, *A*) (21, 90, *A*)	[2, 2, 1]	7
(20, 75, *C*) (20, 75, *D*)	[2, 2, 2]	8

Definition 2.3: the CP file[5]

Let D_i denote the ith attribute domain of an N-attribute file and let each D_i be partitioned into m_i disjoint subdomains $D_{i1}, D_{i2}, \ldots, D_{im_i}$. We call a file a CP file if all records in every bucket are in $D_{1s_1} \times D_{2s_2} \times \ldots \times D_{Ns_N}$, where each D_{js_j} is one of the subdomains $D_{j1}, D_{j2}, \ldots, D_{jm_j}$. The bucket $BK \subseteq D_{1s_1} \times D_{2s_2} \times \ldots \times D_{Ns_N}$ is denoted by $[s_1, s_2, \ldots, s_N]$.

3. DISC MODULO ALLOCATION METHOD AND GENERALISED DISC MODULO ALLOCATION METHOD

Du and Sobolewski[5] proposed a DM allocation method for allocating CP files on to m discs ($m > 1$).

Definition 3.1: the DM allocation method

Let file $F \subseteq D_1 \times D_2 \times \ldots \times D_N$ be a CP file, where each D_i is partitioned into m_i disjoint subsets $D_{i1}, D_{i2}, \ldots, D_{im_i}$, and let m be the number of available discs (labelled as $0, 1, \ldots, m-1$). Let $[s_1, s_2, \ldots, s_N]$ denote the bucket $F \cap (D_{1s_1} \times D_{2s_2} \times \ldots \times D_{Ns_N})$, where $1 \leqslant s_j \leqslant m_j$ for $1 \leqslant j \leqslant N$. In the DM allocation method each bucket $[s_1, s_2, \ldots, s_N]$ in file F is assigned to disc $(s_1 + s_2 + \ldots + s_N) \bmod m$.

Downloaded from https://academic.oup.com/comjnl/article/30/6/535/327613 by guest on 09 April 2024

Example 3.1

Consider a CP file with $m_1 = 2$, $m_2 = 2$ and $m_3 = 3$. Let $m = 3$. Table 3 depicts the distribution of all buckets on to the 3 discs by applying the DM allocation method.

Table 3

Bucket $[s_1, s_2, s_3]$	$s_1+s_2+s_3$	Assigned disc $(s_1+s_2+s_3) \bmod 3$
[1, 1, 1]	3	0
[1, 1, 2]	4	1
[1, 1, 3]	5	2
[1, 2, 1]	4	1
[1, 2, 2]	5	2
[1, 2, 3]	6	0
[2, 1, 1]	4	1
[2, 1, 2]	5	2
[2, 1, 3]	6	0
[2, 2, 1]	5	2
[2, 2, 2]	6	0
[2, 2, 3]	7	1

It was shown by Du and Sobolewski[5] that the DM allocation method is strictly optimal for the following cases: (1) all partial match queries with only one unspecified attribute, (2) all partial match queries with at least one unspecified attribute j for which $m_j \bmod m = 0$, (3) all partial match queries when $m_i \bmod m = 0$ or $m_i = 1$ for all $1 \leqslant i \leqslant N$, (4) all partial match queries when $m = 2$ or 3.

Definition 3.2[5]

An allocation method is said to be strictly optimal to a query if a maximum of $\lceil B/m \rceil$ buckets need to be accessed on any one of m independently accessible discs in order to examine the B buckets in response to the query. If an allocation method is strictly optimal for all possible queries it is called a 'strictly optimal' allocation method.

Evidently a strictly optimal method is superior to all other possible allocation methods. Although the DM allocation method is strictly optimal for the above cases, it is unfortunately not strictly optimal (even not optimal) in general. The following example shows that in some cases the DM allocation method may have a relatively poor performance.

Example 3.2

Table 4 depicts the distribution of assigning all buckets among m discs using the DM allocation method for a CP file in which $m_1 = m_2 = m_3 = 2$ and $m = 7$. As can be seen, the performance is relatively poor since, in fact, disc 0, disc 1 and disc 2 are never used.

Du and Sobolewski[5] then proposed a more generalised method called the GDM allocation method, which can be stated as follows.

Definition 3.3: the GDM allocation method

In the GDM allocation method the bucket $[s_1, s_2, \ldots, s_N]$ in a CP file is stored on disc $(\sum_{j=1}^{N} s_j v_j) \bmod m$, where each positive integer v_j is relatively prime to m.

Table 4

Bucket	Assigned disc
[1, 1, 1]	3
[1, 1, 2]	4
[1, 2, 1]	4
[1, 2, 2]	5
[2, 1, 1]	4
[2, 1, 2]	5
[2, 2, 1]	5
[2, 2, 2]	6

Table 5

Bucket	Assigned disc
[1, 1, 1]	0
[1, 1, 2]	5
[1, 2, 1]	6
[1, 2, 2]	4
[2, 1, 1]	3
[2, 1, 2]	1
[2, 2, 1]	2
[2, 2, 2]	0

From the above definition it can easily be seen that $v_j > m$ is useless, since reduction modulo m is used. This reduces the number of possible v_is to $\phi(m)$, where ϕ denotes the Euler function. Thus there exists only a finite number $\phi^N(m)$ of choices. At least theoretically it is always possible to solve the problem of the optimal $v_1, v_2, \ldots, v_N$ in finite time. Readers who are interested in computing $\phi(m)$ are recommended to consult Ref. 6, section 1.2.4.

If we choose $v_1 = v_2 = \ldots = v_N = 1$, the GDM allocation method reduces to the DM allocation method. By the choice of 'good' $v_1, v_2, \ldots, v_N$, the superiority of the GDM allocation method can easily be exhibited.

Example 3.3

Consider the CP file in Example 3.2 again. Choose $v_1 = 3$, $v_2 = 6$ and $v_3 = 5$. Table 5 shows the distribution of all buckets on to seven discs by applying the GDM allocation method. The reader can verify that the GDM allocation method used in this example is strictly optimal for all possible partial match queries.

Du and Sobolewski[5] also give a sufficient (but not necessary) condition for the GDM allocation to be strictly optimal.

Unfortunately, up to now there has been no general method to determine the proper $v_1, v_2, \ldots, v_N$ for a given set of subdivision sizes $\{m_1, m_2, \ldots, m_N\}$ and the number of available discs m such that the sufficient condition is satisfied, or even to determine if such a set of v_is exists.

In the next section we shall explore the performance of the GDM allocation method for MKH files.

4. PERFORMANCE ANALYSIS OF THE GDM ALLOCATION METHOD FOR MKH FILES

Let F be an MKH file in which each record is characterised by N attributes $A_1, A_2, \ldots, A_N$. Let D_i be

Downloaded from https://academic.oup.com/comjnl/article/30/6/535/327613 by guest on 09 April 2024

the domain of A_i and let each D_i be partitioned, by a predefined function, into m_i disjoint subdomains $D_{i1}, D_{i2}, \ldots, D_{im_i}$ for $1 \leqslant i \leqslant N$. Also, let d_i be the number of elements in D_i, $1 \leqslant i \leqslant N$.

We shall assume that both overflow and underflow problems are ignored. Also, we shall assume that it is impossible to foresee whether a bucket is empty; hence, for example, when $N = 4$, the set of buckets which must be examined in response to the query $(A_1 = *, A_2 = a, A_3 = b, A_4 = *)$, where $a \in D_{2j}, b \in D_{3k}$, will be $\{[s_1, s_2, s_3, s_4] \mid 1 \leqslant s_1 \leqslant m_1, s_2 = j, s_3 = k, 1 \leqslant s_4 \leqslant m_4\}$.

Let Q denote the set of all possible partial match queries and $Q_{i_1 i_2 \ldots i_h}$ denote the set of all queries for which $A_{i_1}, A_{i_2}, \ldots, A_{i_h}$ are specified and other attributes are unspecified, where $i_1, i_2, \ldots, i_h \in \{1, 2, \ldots, N\}$ and $i_1 < i_2 < \ldots < i_h$ for $h = 0, 1, \ldots, N$. Notice that the case $h = 0$ represents the particular query to list the entire file and the case $h = N$ represents the exact match query which is equivalent to the retrieval of a particular record.

Let us denote $Q_{(w_1, w_2, \ldots, w_N)}$, where w_i is either * or $1 \leqslant w_i \leqslant m_i$, to be the set of all queries of the following form: $(A_1 = a_1, A_2 = a_2, \ldots, A_N = a_N)$, with $a_i \in D_{iw_i}$ if w_i is not * and $a_i = *$ if otherwise. Then

$$Q_{i_1 i_2 \ldots i_h} = \bigcup_{\substack{1 \leqslant w_{ij} \leqslant m_{ij},\ j=1,2,\ldots,h \\ w_{ij}=*,\ j=h+1,h+2,\ldots,N}} Q_{(w_1, w_2, \ldots, w_N)};$$

for example, let

$$N = 3, D_1 = \{a_1, a_2, a_3\}, \quad D_2 = \{b_1, b_2, b_3, b_4\},$$
$$D_3 = \{c_1, c_2, c_3, c_4, c_5\}, \quad D_{11} = \{a_1\}, \quad D_{12} = \{a_2, a_3\},$$
$$D_{21} = \{b_1, b_2\}, \quad D_{22} = \{b_3, b_4\},$$
$$D_{31} = \{c_1\}, \quad D_{32} = \{c_2, c_3\}$$

and $D_{33} = \{c_4, c_5\}$, *then*

$$Q_{(1,1,*)} = \{(a_1, b_1, *), (a_1, b_2, *)\},$$
$$Q_{(1,*,2)} = \{(a_1, *, c_2), (a_1, *, c_3)\}$$

and

$$Q_{12} = Q_{(1,1,*)} \cup Q_{(1,2,*)} \cup Q_{(2,1,*)} \cup Q_{(2,2,*)}.$$

Let $Q_{(w_1, w_2, \ldots, w_N)} \subseteq Q_{i_1 i_2 \ldots i_h}$. It is important to note that every query in $Q_{(w_1, w_2, \ldots, w_N)}$ must examine the same set of buckets

$$B_{(w_1, w_2, \ldots, w_N)} = \{[s_1, s_2, \ldots, s_N] \mid s_{i_j} = w_{i_j} \text{ for } j = 1, 2, \ldots, h \text{ and } 1 \leqslant s_{i_j} \leqslant m_{i_j} \text{ for } j = h+1, h+2, \ldots, N\}.$$

Therefore we see that the response times of all queries in $Q_{(w_1, w_2, \ldots, w_N)}$ are the same. Furthermore, it can be shown in the following lemma that the response time is independent of the values of $w_{i_1}, w_{i_2}, \ldots, w_{i_h}$. Consequently, we can conclude that every query in $Q_{i_1 i_2 \ldots i_h}$ has a common value of response time.

Lemma 4.1

Let F be an MKH file. If F is allocated by the use of the GDM allocation method on to m discs ($m > 1$), then every partial match query in $Q_{i_1 i_2 \ldots i_h}$ has a common value of response time which can be expressed as

$$t_{i_1 i_2 \ldots i_h} = \max\{\sum_{r \bmod m = 0} C_r, \sum_{r \bmod m = 1} C_r, \ldots, \sum_{r \bmod m = m-1} C_r\},$$

where C_r denotes the coefficient of x^r in the polynomial

$$P(x) = \prod_{k \in \{1,2,\ldots,N\} \setminus \{i_1, i_2, \ldots, i_h\}} (x^{v_k} + x^{2v_k} + \ldots + x^{m_k v_k})$$

in which each $v_k < m$ is a positive integer and is relatively prime to m.

Proof

Let $Q_{(w_1, w_2, \ldots, w_N)} \subseteq Q_{i_1 i_2 \ldots i_h}$ such that $w_{i_j} = *$ for $j = h+1, h+2, \ldots, N$, and let $\sum_{j=1}^{h} w_{i_j} v_{i_j} = w$.

Let $[s_1, s_2, \ldots, s_N] \in B_{(w_1, w_2, \ldots, w_N)}$. Since

$$\sum_{j=1}^{N} s_j v_j = \sum_{j=1}^{h} w_{i_j} v_{i_j} + \sum_{j=h+1}^{N} s_{i_j} v_{i_j} = w + \sum_k s_k v_k,$$

where $k \in \{1, 2, \ldots, N\} \setminus \{i_1, i_2, \ldots, i_h\}$ and $1 \leqslant s_k \leqslant m_k$, $[s_1, s_2, \ldots, s_N]$ is allocated on to disc $(w + \sum_k s_k v_k) \bmod m$.

Now, consider x^r in the polynomial $P(x)$. Since x^r is a product of $x^{s_k v_k}$s, where each $x^{s_k v_k}$ is selected from the factor $(x^{v_k} + x^{2v_k} + \ldots + x^{m_k v_k})$, the term x^r is of the form $x^{\sum_k s_k v_k}$, where $k \in \{1, 2, \ldots, N\} \setminus \{i_1, i, \ldots, i_h\}$ and $1 \leqslant s_k \leqslant m_k$. Thus C_r equals the number of $(N-h)$-tuples $(s_{i_{h+1}}, s_{i_{h+2}}, \ldots, s_{i_N})$ satisfying $\sum_{j=h+1}^{N} s_{i_j} v_{i_j} = \sum_k s_k v_k = r$.

Let n_i, $i = 0, 1, 2, \ldots, m-1$ be the number of buckets in $B_{(w_1, w_2, \ldots, w_N)}$ which are allocated into disc i. Then $n_i = \sum_{(w+r) \bmod m = i} C_r$. Therefore the response time for any query in $Q_{(w_1, w_2, \ldots, w_N)}$ can be expressed as

$$t_{(w_1, w_2, \ldots, w_N)} = \max\{n_0, n_1, \ldots, n_{m-1}\}$$
$$= \max\{\sum_{(w+r) \bmod m = 0} C_r, \sum_{(w+r) \bmod m = 1} C_r, \ldots, \sum_{(w+r) \bmod m = m+1} C_r\}.$$

Since $(a+b) \bmod m = (a+c) \bmod m$ if and only if $b \bmod m = c \bmod m$, we obtain

$$t_{(w_1, w_2, \ldots, w_N)} = \max\{\sum_{r \bmod m = 0} C_r, \sum_{r \bmod m = 1} C_r, \ldots, \sum_{r \bmod m = m-1} C_r\}.$$

As we can see, the above expression is independent of the values of $w_{i_1}, w_{i_2}, \ldots, w_{i_h}$. Therefore the response times of all queries in $Q_{i_1 i_2 \ldots i_h}$ are the same. The common value is

$$t_{i_1 i_2 \ldots i_h} = \max\{\sum_{r \bmod m = 0} C_r, \sum_{r \bmod m = 1} C_r, \ldots, \sum_{r \bmod m = m-1} C_r\}.$$

Q.E.D.

By Lemma 4.1 we obtain the average response time for queries by using the GDM allocation method for MKH files.

Theorem 4.2

Let T_{GDM} denote the average response time for all possible partial match queries by using the GDM allocation method for an MKH file F among m discs ($m > 1$). Let $P_{i_1 i_2 \ldots i_h}$ be the probability of occurrence for queries in $Q_{i_1 i_2 \ldots i_h}$. Then T_{GDM} can be computed as

$$T_{GDM} = \sum_{h=0}^{N} \sum_{\substack{i_1 < i_2 < \ldots < i_h \\ \{i_1, i_2, \ldots, i_h\} \subseteq \{1, 2, \ldots, N\}}} P_{i_1 i_2 \ldots i_h} \max\{\sum_{r \bmod m = 0} C_r, \sum_{r \bmod m = 1} C_r, \ldots, \sum_{r \bmod m = m-1} C_r\}. \quad (4.1)$$

Downloaded from https://academic.oup.com/comjnl/article/30/6/535/327613 by guest on 09 April 2024

Proof

Since

$$T_{GDM} = \sum_{h=0}^{N} \sum_{\substack{i_1 < i_2 < \ldots < i_h \\ \{i_1, i_2, \ldots, i_h\} \subseteq \{1, 2, \ldots, N\}}} P_{i_1 i_2 \ldots i_h} t_{i_1 i_2 \ldots i_h}.$$

Theorem 4.2 follows immediately from Lemma 4.1.

Q.E.D.

Corollary 4.1

If the probability of occurrence of all partial match queries is identical,

$$T_{GDM} = \frac{1}{\prod_{i=1}^{N} (1+d_i)} \sum_{h=0}^{N} \sum_{\substack{i_1 < i_2 < \ldots < i_h \\ \{i_1, i_2, \ldots, i_h\} \subseteq \{1, 2, \ldots, N\}}} d_{i_1} d_{i_2} \ldots d_{i_h} \max\{ \sum_{r \bmod m = 0} C_r, \sum_{r \bmod m = 1} C_r, \ldots, \sum_{r \bmod m = m-1} C_r\}. \quad (4.2)$$

Proof

Since the probability of occurrence of partial match queries is the same,

$$P_{i_1 i_2 \ldots i_h} = \frac{\text{total number of queries in } Q_{i_1 i_2 \ldots i_h}}{\text{total number of queries}} = \frac{d_{i_1} d_{i_2} \ldots d_{i_h}}{\prod_{i=1}^{N} (1+d_i)}.$$

By substituting $P_{i_1 i_2 \ldots i_h}$ into (4.1), we have the proof.

Q.E.D.

Example 4.1

Consider the MKH file F as shown in Table 2, for which we have $N = 3$, $d_1 = 4$, $d_2 = 8$, $d_3 = 4$ and $m_1 = m_2 = m_3 = 2$. Let $m = 7$ and apply the GDM method with $v_1 = 3$, $v_2 = 6$ and $v_3 = 5$ to F. If the probability of occurrence of all partial match queries is equal, then by (4.2) we have

$$T_{GDM} = \frac{1}{5 \times 9 \times 5} (\max\{2,1,1,1,1,1,1\} + 4 \times \max\{0,1,1,1,1,0,0\} + 8 \times \max\{0,1,1,0,1,0,1\} + 4 \times \max\{0,1,1,0,1,1,0\} + 4 \times 8 \times \max\{0,0,0,1,0,1,0\} + 4 \times 4 \times \max\{0,0,0,0,0,1,1\} + 8 \times 4 \times max\{0,0,0,1,0,0,1\} + 4 \times 8 \times 4) = \frac{226}{225} \sim 1.004.$$

Corollary 4.2

If the probability of attribute A_i specified in a query is q_i, $i = 1, , \ldots, N$, and this probability depends on A_i itself and is independent of other attributes specified in the query, then

$$T_{GDM} = q \sum_{h=0}^{N} \sum_{\substack{i_1 < i_2 < \ldots < i_h \\ \{i_1, i_2, \ldots, i_h\} \subseteq \{1, 2, \ldots, N\}}} r_{i_1} r_{i_2} \ldots r_{i_h} \max\{ \sum_{r \bmod m = 0} C_r, \sum_{r \bmod m = 1} C_r, \ldots, \sum_{r \bmod m = m-1} C_r\}, \quad (4.3)$$

where

$$q = \prod_{j=1}^{N} (1-q_j) \quad \text{and} \quad r_j = \frac{q_j}{1-q_j}, \quad j = 1, 2, \ldots, N.$$

Proof

Since, under the sufficient conditions in this corollary,

$$P_{i_1 i_2 \ldots i_h} = q_{i_1} q_{i_2} \ldots q_{i_h} (1-q_{i_{h+1}})(1-q_{i_{h+2}}) \ldots (1-q_{i_N}) = \frac{q_{i_1}}{1-q_{i_1}} \frac{q_{i_2}}{1-q_{i_2}} \ldots \frac{q_{i_h}}{1-q_{i_h}} \prod_{j=1}^{N} (1-q_j) = r_{i_1} r_{i_2} \ldots r_{i_h} q.$$

By substituting $P_{i_1 i_2 \ldots i_h}$ into (4.1) we have the proof.

Q.E.D.

Example 4.2

Consider the MKH file in Table 2 again. There are three attributes: $A_1 =$ age, $A_2 =$ calculus score and $A_3 =$ English grade. If the probabilities of these three attributes are $q_1 = 0.6$, $q_2 = 0.9$ and $q_3 = 0.4$ respectively, $r_1 = 1.50$, $r_2 = 9.00$, $r_3 = 0.67$ and $q = 0.4 \times 0.1 \times 0.6 = 0.024$. Therefore, from (4.3), we have

$$T_{GDM} = 0.024 \times (\max\{2,1,1,1,1,1,1\} + 1.50 \times \max\{0,1,1,1,1,0,0\} + 9.00 \times \max\{0,1,1,0,1,0,1\} + 0.67 \times \max\{0,1,1,0,1,1,0\} + 1.50 \times 9.00 \times \max\{0,0,0,1,0,1,0\} + 1.5 \times 0.67 \times \max\{0,0,0,0,0,1,1\} + 9.00 \times 0.67 \times \max\{0,0,0,1,0,0,1\} + 1.50 \times 9.00 \times 0.67) = 1.026.$$

Notice that, when $v_1 = v_2 = \ldots = v_N = 1$, the GDM allocation method reduces to the DM allocation method and (4.1), (4.2), (4.3) are reduced to be performance expressions for the DM allocation method. It should be noted that the performance formula derived by Chang and Shen[3] is identical with the result in Corollary 4.1, with $v_1 = v_2 = \ldots = v_N = 1$ being identical.

5. CONCLUSIONS

In this paper we are concerned with the file allocation problem. In particular, we concentrate on the performance of the GDM allocation method for MKH files. Taking the probability of occurrence of queries into consideration, we have derived an important performance formula of the GDM allocation method for MKH files. We saw that the performance formula derived by Chang and Shen[3] is simply a particular case of our results. Nevertheless, there are still many relevant problems to be solved. These include deciding how to construct an optimal GDM allocation method and design an optimal MKH file under a predefined GDM allocation method; and, finding out whether there is another heuristic allocation method which is more applicable than the GDM method.

Acknowledgements

The authors wish to express their gratitude to a referee for his excellent suggestions and comments, which greatly improved this paper.

Downloaded from https://academic.oup.com/comjnl/article/30/6/535/327613 by guest on 09 April 2024

REFERENCES

1. A. Bolour, Optimality properties of multiple-key hashing functions. *Journal of the ACM* **26** (2), 196–210 (1979).
2. C. C. Chang, Optimal information retrieval when queries are not random. *Information Sciences* **34**, 199–233 (1984).
3. C. C. Chang and J. J. Shen, Performance analysis of the disk modulo allocation method for concurrent accessing on multiple disk systems. *Journal of the Chinese Institute of Engineers* **8** (3), 271–283 (1985).
4. C. C. Chang and J. C. Shieh, On the complexity of the file allocation problem. *Proceedings of the International Conference on Foundations of Data Organization, Kyoto, Japan, May 1985*, 113–115 (1985).
5. H. C. Du and J. S. Sobolewski, Disk allocation for Cartesian product files on multiple disk systems. *ACM Transactions on Database Systems* **7**, 82–101 (1982).
6. D. E. Knuth, *The Art of Computer Programming*, vol. 1, *Fundamental Algorithms*, 2nd edn. Addison-Wesley, Reading, Mass. (1973).
7. J. B. Rothnie and T. Lozano, Attribute based file organization in a paged memory environment. *Communications of the ACM* **17** (2), 63–69 (1974).

Downloaded from https://academic.oup.com/comjnl/article/30/6/535/327613 by guest on 09 April 2024