UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

LOAD BALANCED AND OPTIMAL DISK ALLOCATION STRATEGY FOR PARTIAL MATCH QUERIES ON MULTI-DIMENSIONAL FILES

Sajal K. Das and Cristina M. Pinotti

October 2002

Technical Report # DIT-02-0077

Also: to be published on IEEE transaction on Parallel and Distributed Systems in October 2002

Load Balanced and Optimal Disk Allocation Strategy for Partial Match Queries on Multi-dimensional Files *

Sajal K. Das
Dept. of Computer Science & Engineering
University of Texas at Arlington
Arlington, TX 76019-0015, USA
das@cse.uta.edu

Cristina M. Pinotti
Dept. of Computer Science & Telecom.
University of Trento
Povo, 38050, Trento, ITALY
pinotti@science.unitn.it

Abstract

A *multi-dimensional file* is one whose data are characterized by several attributes, each specified in a given domain. A *partial match query* on a multi-dimensional file extracts all data whose attributes match the values of one or more attributes specified in the query. The *disk allocation* problem of a multi-dimensional file, F, on a database system with multiple disks accessible in parallel is the problem of distributing F among the disks such that the data qualifying for each partial match query are distributed as evenly as possible among the disks of the system. We propose an optimal solution to this problem, for multi-dimensional files with pairwise prime domains, based on a large and flexible class of maximum distance separable codes, namely *the redundant residue codes*. We also introduce a new family of residue codes, called the *redundant non-pairwise prime residue codes*, to deal with files whose attribute domains are non-pairwise prime.

Keywords: Non-uniform multi-dimensional file, partial match query, strictly optimal disk allocation, redundant pairwise-prime residue code, redundant non-pairwise prime residue code.

1 Introduction

Parallel database systems are essential to important real-life applications that require to manage extremely large volumes of data, often of the order of terabytes, which cannot be stored in a single disk. The applications include spatial databases, airline reservation systems, cartography, world wide web, and so on. Queries on such large databases involve the retrieval of a big portion of data, often under critical time constraints. Therefore, to retrieve qualifying data in parallel is mandatory. Before proceeding further, let us define a few terminology and notations.

*This work is partially supported by Texas Advanced Research Program TARP-003594013 and MIUR REAL-WINE research grant in Italy. A preliminary version of this paper appeared in [4].

In this paper, we consider database systems of *multi-dimensional files*, each made up of *buckets*, each, in turn, made up of *records*.

A *record* is an n-tuple $< r_1, \ldots, r_n >$ of *attributes* such that the value of the jth attribute r_j belongs to the interval $I_j = [L_j, R_j]$, for $1 \leq j \leq n$.

In order to define a bucket, let us partition uniformly the values of the jth attribute of a record into m_j groups, each of size $\left\lceil \frac{R_j - L_j + 1}{m_j} \right\rceil$, and let m_j be termed the *domain* of r_j. The *bucket* $< x_1, \ldots, x_n >$ of respective domains $m_1, \ldots, m_n$ is the set of the records $< r_1, \ldots, r_n >$ such that the value of r_j, for $1 \leq j \leq n$, belongs to the x_j-th group of I_j. More formally,

$$< x_1, \ldots, x_n >= \left\{ < r_1, \ldots, r_n > \left| \begin{array}{ll} L_j + \left\lceil \frac{R_j - L_j + 1}{m_j} \right\rceil x_j \leq r_j \leq L_j + \left\lceil \frac{R_j - L_j + 1}{m_j} \right\rceil (x_j + 1) - 1 & \text{if } 0 \leq x_j \leq m_j - 2, \\ L_j + \left\lceil \frac{R_j - L_j + 1}{m_j} \right\rceil x_j \leq r_j \leq R_j & \text{if } x_j = m_j - 1, \end{array} \right. \forall j \right\}$$

Thus, the bucket $< x_1, \ldots, x_n >$ stores at most $\Pi_{j=1}^{n} \left\lceil \frac{R_j - L_j + 1}{m_j} \right\rceil$ records. Finally, a *multi-dimensional file* $F(m_1, \ldots, m_n)$ of domains $m_1, \ldots, m_n$ is the set of all $\Pi_{i=1}^{n} m_i$ buckets $< x_1, \ldots, x_n >$ where $x_i \in [0, \ldots, m_i - 1]$ and $1 \leq i \leq n$.

Example 1: Consider two attributes that range, respectively, in $I_1 = [1 \ldots 100]$ and $I_2 = [1 \ldots 50]$. Fixing the domains $m_1 = 20$ and $m_2 = 10$, partition I_1 and I_2, respectively, into 20 and 10 groups of equal size. Specifically, the ith group of I_1 contains the values $[5i+1, 5i+2, 5i+3, 5i+4, 5(i+1)]$, for $0 \leq i \leq 19$, of the first attribute. Thus, the bucket $< 2, 3 >$ contains the records $< r_1, r_2 >$, where $11 \leq r_1 \leq 15$ and $16 \leq r_2 \leq 20$. The multi-dimensional file $F(20, 10)$ consists of all the 200 buckets $< x_1, x_2 >$, where $0 \leq x_1 \leq 19$ and $0 \leq x_2 \leq 9$, and each bucket stores 25 records.

A *uniform p-ary multi-dimensional* file $F(p, \ldots, p)$ is defined as one in which all the domains assume the identical value p. Otherwise it is called a *non-uniform multi-dimensional file*. If the n domains $m_1, \ldots, m_n$ are pairwise prime, $F(m_1, \ldots, m_n)$ is said to be a *non-uniform, pairwise prime multi-dimensional* file.

The most common operation on a database system is the information retrieval. Formally,

Definition 1 *Given a multi-dimensional file* $F(m_1, \ldots, m_n)$, *a* match query $q =< q_1, \ldots, q_n >$ *on* F *retrieves the bucket* $< q_1, \ldots, q_n >$. *A* partial match query *(PMQ) is one whose ith attribute*

q_i *is either* specified *or* unspecified*, and retrieves from* F *the set of buckets that match the query* q *on the specified attributes. Such buckets are said to* qualify *for* q*. The* query response set*,* $QR(q)$*, of a PMQ with the unspecified attributes* $q_{i_1}, q_{i_2}, \ldots, q_{i_k}$ *consists of* $N(q) = \Pi_{j=1}^{k} m_{i_j}$ *buckets that qualify for* q*.*

As an example, given the file $F(20, 10)$, the PMQ $q =< 5, * >$ extracts $N(< 5, * >) = 10$ buckets defined as $\{< 5, i > | 0 \leq i \leq 9\}$.

In real database systems, however, the number of attributes is much larger than two. Typically, there are thousands of buckets, and a considerably large number of them may qualify for each PMQ. In such a context, database systems need to store their files on multiple disks.

A *parallel database system* is a database system that stores multi-dimensional files on multiple disks that can be read or written simultaneously. This additional capability may improve significantly the time required for data transmission if a suitable scheme is adopted for data distribution among the disks. In other words, if the data that qualify for a query are almost evenly distributed among the disks available in the database system, the response time is reduced by a factor equal to the number of disks. In fact, distributing $F(m_1, \ldots, m_n)$ into a parallel database system of D disks, denoted as $d_0, \ldots, d_{D-1}$, the time to retrieve the query response set $QR(q)$ of a partial match query, q, is proportional to the number of buckets that qualify for q in each disk. Precisely, the *response time* of a query is proportional to $RT(q) = \max_{0 \leq i \leq D-1}\{N_i(q)\}$, where $N_i(q)$ is the number of buckets that qualify for q stored on disk d_i. Thus, the response time is minimum when $RT(q) = \left\lceil \frac{N(q)}{D} \right\rceil$, i.e., when the query response set $QR(q)$ is balanced among all the available disks in the system. Therefore, an optimal solution to the disk allocation problem is equivalent to finding a load balanced distribution of the buckets among the multiple disks of the system.

We adopt the following criteria for optimality.

Definition 2 [1, 7] *Given a system with* D *disks that can be accessed simultaneously, a disk allocation strategy is termed* optimal for a specific partial match query q *if it evenly distributes* $N(q)$ *among the* D *disks, thus achieving a response time of* $\left\lceil \frac{N(q)}{D} \right\rceil$ *for* q*. Moreover, a disk allocation strategy is called* strictly optimal *if it is* optimal for all partial match queries.

From now on, whenever the domains of the attributes are clear from the context, $F(m_1, \ldots, m_n)$ will be referred to as simply F.

2 Previous Work on Disk Allocation Problem

The problem of minimizing the response time of PMQs on uniform multi-dimensional files has received a lot of attention in the past. For example, one of the first approaches to the problem proposed of distributing F among D disks with the help of a (pseudo)random number generator, with $1/D$ as the probability of assigning a bucket to a particular disk. This solution had no constraints on the number of disks as well as on the cardinality of the domains of the attributes. However, it did not guarantee optimality for any class of partial match queries.

Subsequently, Fang et al. [6] interpreted each bucket of $F(m_1, \ldots, m_n)$ as a point of the n-dimensional space $m_1 \times \ldots \times m_n$, and suggested to partition the buckets into two *geometrical similar* groups, that is in two groups with almost the same spanning tree and almost the same set of short spanning paths. They proved that, as a consequence of the geometrical similarity in each group, almost the same number of buckets qualifies for each PMQ. This solution, however, can at most halves the query response time. Moreover, the partitioning of F in more than two groups seems quite intricate and therefore such a method is not suitable for generalizations.

A different approach, due to Du and Sobolewski [5], assumed a D-disk system and assigned the bucket $< x_1, \ldots, x_n >$ to the disk d_j where $j \equiv (\sum_{i=1}^{n} x_i) \bmod D$. This method has been shown to be always strictly optimal for PMQs with exactly one unspecified attribute. Moreover for PMQs with two unspecified attributes, the method is strictly optimal when the disk system as $D = 2$ or $D = 3$, or when the attribute domains satisfy $m_i \equiv 0$ or $1 \bmod D$, for $1 \leq i \leq n$.

Later, Kim and Pramanik [8] solved the disk allocation problem optimally for both PMQs with exactly two unspecified attributes and for PMQs with the domain of one unspecified attribute larger than D. Their solution assigns the bucket $< x_1, \ldots, x_n >$ to the disk d_j, such that $j = ([+]_{i=1}^{n} x_i) \bmod D$, where $[+]$ denote the bitwise Exclusive-OR operation.

In [7], Faloutsos and Metaxas followed a completely different approach. Given a binary multi-dimensional file F with n-attributes and $D = 2^k$ disks, their solution assigns to each disk a set of buckets which form a linear $(k-1)$-error correcting binary code, C. The code definition implies that any two buckets stored in the same disk differ by at least k attributes. This approach is always optimal for PMQs with at most $(k-1)$ unspecified attributes, and strictly optimal when C is a *maximum distance separable* (MDS) code [10]. Unfortunately, for many pair of values n, k, there are no MDS codes, so rarely a strictly optimal solution can be found.

Abdel-Ghaffar and El Abbadi [1] extended the results of [7] to uniform p-ary multi-dimensional files, with $p > 2$, and formally established the equivalence between *strictly optimal allocation strategies* and *maximum distance separable* codes. Based on such an equivalence and using the fact that the *Reed-Solomon* codes form a large family of p-ary MDS codes applicable to p-ary multi-dimensional files with at most $p-1$ attributes, the designed strategy is strictly optimal for files with at most p^{p-1} buckets and uses $D = p^t$ disks, where $t \leq p-1$. This solution does not work for non-uniform files as well as for files whose records have more than $p-1$ attributes.

3 Our Contributions

This paper presents the first systematic solution to the disk allocation problem for non-uniform pairwise-prime multi-dimensional files. It can be considered as an extension of the results in [7, 1]. Our method takes full advantage of all (possibly a very large number) disks, and hence the response time is drastically reduced. The solution is based on a very large and flexible class of semi-linear MDS codes, called the *redundant residue (pairwise prime) codes* [2]. The existence of such a family of codes guarantees strictly optimal solution and alleviates the limitations of the previous code-based disk allocation strategies which mainly suffer from the lack of enough MDS codes for files of different sizes. Our solution handles non-uniform multi-dimensional files that model real data much better than the uniform multi-dimensional files, as argued below with an example.

Example 2: Let us consider a multi-dimensional file, F, having three integer attributes whose intervals are very skewed. Let the interval of the first attribute be $I_1 = [1..1000]$, that of the second be $I_2 = [1..21]$, and that of the third be $I_3 = [1..9]$. Thus, there are $1000 \times 21 \times 9$ records altogether. Assuming that each disk page contains no more than 100 records, let us determine a suitable domain for each attribute such that a bucket has almost the same size as a disk page and each bucket is full.

First, we consider the scenario when all the three attributes have the same domain value $m_1 = m_2 = m_3 = p$. Then the bucket $< x_1, x_2, x_3 >$, with $0 \leq x_i \leq p-1$ for $1 \leq i \leq 3$, contains the records $< r_1, r_2, r_3 >$ such that $L_i + \left\lceil \frac{R_i - L_i + 1}{p} \right\rceil x_i \leq r_i \leq L_i + \left\lceil \frac{R_i - L_i + 1}{p} \right\rceil (x_i + 1) - 1$. Specifically,

- For $p = 4$, we obtain very large buckets that do not fit in one disk page. Indeed, there are $250 \times 6 \times 3$ records stored in each bucket.
- For $p = 10$, each bucket consists of $100 \times 3 \times 1$ records, which is still too large in size.
- For $p = 20$, each bucket has at most $50 \times 2 \times 1$ records. Although a bucket perfectly fits in a page, the file F is mapped into 20^3 buckets, most of which are empty (consider, for example, all the buckets with $x_3 \geq 1$).

On the other hand, if different domains are allowed a much better load factor can be obtained:

- For $m_1 = 20$, $m_2 = 11$ and $m_3 = 9$, the bucket $< x_1, x_2, x_3 >$, for $0 \leq x_1 \leq 19$, $0 \leq x_2 \leq 10$ and $0 \leq x_3 \leq 8$, consists of the records $< r_1, r_2, r_3 >$ such that $1 + x_1 \left\lceil \frac{1000}{20} \right\rceil \leq r_1 \leq (x_1 + 1) \left\lceil \frac{1000}{20} \right\rceil$, $1 + x_2 \left\lceil \frac{21}{11} \right\rceil \leq r_2 \leq (x_2 + 1) \left\lceil \frac{21}{11} \right\rceil$, and $1 + x_3 \leq r_3 \leq (x_3 + 1)$. Hence, each bucket $< x_1, x_2, x_3 >$ consists at most of 100 records and the file F has $20 \times 11 \times 9 = 1980$ buckets, all of which are full.

This example shows that partitioning all attributes of the file with the same domain, one can get many empty buckets. On the contrary, if different domains are used for different attributes, a good load factor of the buckets can be obtained.

Back to the proposed disk allocation strategy, note that the family of redundant residue codes applies only to files whose attributes have pairwise-prime domains. It is worth to point out that this is not a serious limitation since the attribute domains are selected when the disk allocation strategy is designed. For the sake of completeness, we introduce a new class of *Redundant Non-Pairwise Prime Residue Codes* in order to handle attributes with non-pairwise prime domains. This may lead to a slowdown with respect to the optimal solution by a predetermined factor.

The remainder of the paper is organized as follows. Section 4 reviews basic concepts of the coding theory that are relevant to our work. Based on the *redundant pairwise prime residue* codes, Section 5 proposes a strictly optimal disk allocation strategy for non-uniform multi-dimensional files whose attributes have pairwise-prime domains. In Section 5.1, we study how to guarantee strict optimality when either the number of attributes of the file or the number of disks in the system changes. Finally, a new family of redundant residue codes, called the *redundant non-pairwise prime residue codes*, is introduced in Section 6 to solve the disk allocation strategy for non-uniform multi-dimensional files whose attributes have non-pairwise prime domains. Finally, conclusions are offered in Section 7.

4 Coding Theory Framework

Given a set of n positive integer *radices*, denoted as $m_1, m_2, \ldots, m_n$, let $S = m_1 \times m_2 \times \ldots \times m_n$ be the space of all n-tuples of size $M = \Pi_{i=1}^{n} m_i$. A *code*, C, is a subset of S. Each n-tuple of S which is also in C, is called a *codeword*. Since S is defined on n radices, C has length n. Let the *size*, γ, of C be its cardinality. If all radices are equal to $p \geq 2$, C is called a p-ary code.

A code C is *linear* if it is closed under the addition and subtraction operations, as defined below. Given two codewords $x =< x_1, x_2, \ldots, x_n >$ and $y =< y_1, y_2, \ldots, y_n >$ of C, both the n-tuples $x + y$ and $x - y$ as defined below are also codewords of C.

$x + y =< (x_1 + y_1) \bmod m_1, (x_2 + y_2) \bmod m_2, \ldots, (x_n + y_n) \bmod m_n >$, and

$x - y =< (x_1 - y_1) \bmod m_1, (x_2 - y_2) \bmod m_2, \ldots, (x_n - y_n) \bmod m_n >$.

Let the information set $I_{S_k} = \{m_{i_1}, m_{i_2}, \ldots, m_{i_k}\}$ of a code C be a set of k indices such that for any k-tuple $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$, where $a_{i_j} \in [0, m_{i_j} - 1]$, there is a *unique* codeword $< x_1, x_2, \ldots, x_n > \in C$ such that $x_{i_j} = a_{i_j}$ for $1 \leq j \leq k$. Hence, if a code C has an information set I_{S_k}, then the size of C is larger than or equal to $\gamma = \Pi_{j=1}^{k} m_{i_j}$. Such a code C is said to be a *systematic* code.

For a linear code C, the *Hamming weight* of a codeword $x =< x_1, x_2, \ldots, x_n >$ is the number of non-zero components x_i; and the *Hamming distance* between two codewords is the number of components in which they differ. The *minimum distance*, d, of C is the minimum Hamming distance between all pairs of distinct codewords in C. Since the all-zero codeword always belong to a linear code, the minimum distance $d > 0$ and the minimum Hamming weight of the codewords of C coincide.

The concept of minimum distance is essential for defining the error control capabilities of a code. In fact, representing the error as an n-tuple $e =< e_1, e_2, \ldots, e_n >$, and denoting a codeword x subject to error e as $x + e$, a code of minimum distance d is able (i) to detect at least all errors e with Hamming weight at most $d - 1$, and (ii) to correct all errors e with Hamming weight $\leq \lfloor (d-1)/2 \rfloor$. It follows directly that among all the codes of a given length n and size γ, the codes with the highest error control are those with the largest minimum distance d.

From now on, $C = [n, \gamma, d]$ will denote a linear code of length n, size γ and minimum distance d. Furthermore, when $\max \left\{\Pi_{i=1}^{k} m_{j_i}\right\} \leq \gamma < \max \left\{\Pi_{i=1}^{k+1} m_{j_i}\right\}$, the inequality $d \leq n - k + 1$ holds, which is known as the *Singleton bound* [10]. The class of codes that satisfy the equality $d = n - k + 1$ plays an important role in our solution approach as evident from the following definition.

Definition 3 [10] *A code $C = [n, \gamma, d]$ with minimum distance $d = n - k + 1$ is called a* maximum distance separable *(MDS) code which is also a systematic code. Any k radices of an MDS code, C, form an information set.*

Among the MDS codes, of particular interest are the *Redundant Pairwise-Prime Residue Codes*, which are defined on a *Redundant Residue Number System* (RRNS). An RRNS has n

pairwise-prime positive radices, $m_1, m_2, \ldots, m_k, m_{k+1}, \ldots, m_n$, called *moduli*. Let the first k moduli be termed as the *non-redundant* moduli, while the remaining $n-k$ moduli be the *redundant* moduli. Let $M_{I_S} = \Pi_{i=1}^{k} m_i$ and $M_R = \Pi_{i=k+1}^{n} m_i$ be the product of, respectively, the non-redundant and the redundant moduli. Also let $M = \Pi_{i=1}^{n} m_i$ be the product of all moduli.

It is well known that every integer $X \in [0, M)$ can be uniquely represented by a residue vector $x =< x_1, x_2, \ldots, x_n >$ such that $x_i \equiv X \bmod m_i$, for $1 \leq i \leq n$. Clearly, $0 \leq x_i < m_i$. Similarly, given a residue vector x, the corresponding integer X can be uniquely determined by applying the *Chinese Remainder Theorem* [2]. The residue representations of the integers in the range $[0, M)$ can be partitioned into codes as follows:

Definition 4 *For fixed $k \in [1, \ldots, n-1]$ and given a set of n pairwise-prime redundant moduli, $m_1, m_2, \ldots, m_k, m_{k+1}, \ldots, m_n$, let $M_{I_S} = \Pi_{i=1}^{k} m_i$, $M_R = \Pi_{i=k+1}^{n} m_i$, and $M = \Pi_{i=1}^{n} m_i$. For fixed $\alpha \in [0, M - M_{I_S})$, the* redundant residue *RR-(n, k, α)-code (or briefly, RR-(n, k)-code) consists of all the residue vectors representing integers in the range $[\alpha, \alpha + M_{I_S})$. In particular, for $\alpha = 0$, RR-$(n, k, 0)$-code consists of all the residue vectors representing integers in $[0, M_{I_S})$.*

Every codeword of an RR-(n, k, α)-code has an information part consisting of the first k residue digits and a parity part, consisting of the remaining $n-k$ residue digits. Moreover, as proved in [9], C is *semi-linear*, i.e., it is linear under certain conditions. All the main properties of the linear codes also hold for the semi-linear codes.

In particular,

Lemma 1 [9] *An RR-(n, k)-code C has a minimum distance d if and only if*

$$max\left\{\Pi_{i=1}^{d} m_{j_i}\right\} > M_R \geq max\left\{\Pi_{i=1}^{d-1} m_{j_i}\right\}, \quad \text{for } 1 \leq j_i \leq n.$$

Therefore,

Lemma 2 [2, 9] *For fixed $k \in [1, \ldots, n-1]$, the RR-(n, k)-code C defined by the moduli $m_1, \ldots, m_k, m_{k+1}, \ldots, m_n$ such that $m_1 < m_2 < \ldots < m_k < m_{k+1} < \ldots < m_n$ has minimum distance $d = n - k + 1$ and hence it is an MDS code. Moreover, varying the number k of non-redundant moduli from 1 to $n-1$, we obtain $n-1$ different MDS codes.*

Procedure Decluster (k, $F(m_1, \ldots, m_k, m_{k+1}, \ldots, m_n)$, $C_0 = [n, \Pi_{i=1}^{k} m_i, d]$, $D = \Pi_{i=k+1}^{n} m_i$)

1. Based on the seed code C_0, partition $S = m_1 \times \ldots m_k \times m_{k+1} \times m_n$ into D codes $C_0, \ldots, C_{D-1}$ such that C_j consists of the residue representations of the integers in the range $[j\Pi_{i=1}^{k} m_i, (j+1)\Pi_{i=1}^{k} m_i)$.
2. For $0 \leq j \leq D-1$, assign code C_j to disk d_j.

Figure 1: Procedure Decluster

Fact 1 *Given n moduli $m_1, \ldots, m_k, m_{k+1}, \ldots, m_n$ and a fixed integer $k \in [1, \ldots, n-1]$, let $M_{I_S} = \Pi_{i=1}^{k} m_i$ and $M_R = \Pi_{i=k+1}^{n} m_i$. Then, the entire space $S = m_1 \times m_2 \times \ldots m_k \times m_{k+1} \times \ldots \times m_n$ can be partitioned into M_R subspaces, denoted as $C_0, C_1, \ldots C_{M_R-1}$ such that C_j = RR-(n, k, jM_{I_S}).*

5 Disk Allocation Strategy Based on Redundant Residue Codes

We are now ready to present the Decluster procedure, which solves the disk allocation problem for multi-dimensional files whose attributes have pairwise prime domains. For fixed $k \in [1, \ldots, n-1]$, the procedure Decluster in Figure 1 distributes a file $F(m_1, \ldots, m_k, m_{k+1}, \ldots, m_n)$ among a set of $D = \Pi_{i=k+1}^{n} m_i$ disks denoted as $d_0, d_1, \ldots, d_{D-1}$, with the help of the RR-$(n, k, 0)$-code = $C_0 = [n, \Pi_{i=1}^{k} m_i, d]$, whose codewords are the residue representations of the integers in the range $[0, \Pi_{i=1}^{k} m_i)$. In this paper, C_0 will be called the *seed code*.

Let us prove some properties of the above procedure.

Lemma 3 *The Decluster procedure guarantees that every partial match query (PMQ), q, on the file F with at most $d-1$ unspecified attributes has response time $RT(q) = 1$.*

Proof. No more than one bucket in each disk qualifies for q because the seed code has minimum distance d, and thus, two buckets stored in the same disk must differ in at least d attributes. □

Theorem 1 *Let the pairwise-prime domains of the attributes of the file $F(m_1, \ldots, m_k, m_{k+1}, \ldots, m_n)$ be arranged in an increasing order such that $m_1 < \ldots < m_k < m_{k+1} < \ldots < m_n$. For fixed $k \in [1..n-1]$, given the seed code $C_0 = [n, \Pi_{i=1}^{k} m_i, n-k+1]$, the associated Decluster procedure leads to a strictly optimal disk allocation strategy for every PMQ q on F. Hence, using $D = \Pi_{i=k+1}^{n} m_i$ disks, the response time for every PMQ, q, whose response set has size $N(q)$ is given by $RT(q) = \left\lceil \frac{N(q)}{D} \right\rceil$.*

Proof. Fixed k, for any partial match query with at most $n-k$ unspecified attributes, the result follows from Lemma 3 and the fact that $N(q) \leq D$.

For PMQs with more than $n-k$ unspecified attributes, let q be an arbitrary query whose attributes $q_{i_1}, \ldots, q_{i_u}$ are specified, and the remaining $n-u$ attributes satisfy $q_{i_j} = a_{i_j}$ for $u+1 \leq j \leq n$.

Let $M_s = \Pi_{j=u+1}^{n} m_{i_j}$ be the product of the domains of the unspecified attributes of q. By the Chinese Remainder Theorem [2], there is a unique integer X in the range $[0, M_s)$ such that $X \equiv a_{i_{u+j}} \bmod m_{i_{u+j}}$, for $1 \leq j \leq n-u$. In the range $[0, M)$, where $M = \Pi_{i=1}^{n} m_i$, there are $\frac{M}{M_s}$ strings $< x_1, \ldots, x_n >$ such that $x_{i_{u+j}} = a_{i_{u+j}}$ for $1 \leq j \leq n-u$. Those n-tuples correspond to the integers $X + kM_s$ where $0 \leq k < \frac{M}{M_s}$, and verify $(X + kM_s) \equiv X \bmod m_{i_{u+j}}$ for $1 \leq j \leq n-u$. Recall that by Fact 1, the code C_j associated with the disk d_j, for $0 \leq j \leq D-1$, consists of the residue representations of the integers in $\left[j\Pi_{i=1}^{k} m_i, (j+1)\Pi_{i=1}^{k} m_i\right)$. There are at most $\left\lceil \frac{\Pi_{i=1}^{k} m_i}{M_s} \right\rceil$ buckets qualifying for q among the codewords of C_j. Hence the response time is $RT(q) = \left\lceil \frac{\Pi_{i=1}^{k} m_i}{M_s} \right\rceil$. Since $N(q) = \frac{M}{M_s}$ and $D = \Pi_{i=k+1}^{n} m_i$, for every PMQ q, the claim follows:

$$RT(q) = \left\lceil \frac{\Pi_{i=1}^{k} m_i}{M_s} \right\rceil = \left\lceil \frac{M/D}{M_s} \right\rceil = \left\lceil \frac{M/M_s}{D} \right\rceil = \left\lceil \frac{N(q)}{D} \right\rceil .$$

□

5.1 Capturing Dynamic Scenario

Our solution to the disk allocation problem can efficiently manage situations in which either (i) the number of disks available in the system changes, or (ii) the number of attributes in a file changes, without redistributing the entire multi-dimensional file from scratch among the disks. Nonetheless, in either cases, some buckets must be moved from one disk to another. We say that such buckets must be *relocated*.

Assume a pairwise-prime multi-dimensional file $F(m_1, \ldots, m_k, m_{k+1}, \ldots, m_n)$ with $m_1 < \ldots < m_k < m_{k+1} < \ldots < m_n$, that has been distributed by the Decluster procedure

among $D = \Pi_{i=k+1}^{n} m_i = M_R$ disks using the MDS seed code $C_0 = [n, M_{I_S} = \Pi_{i=1}^{k} m_i, n-k+1]$.

First, suppose the number of available disks increases from D to $m_k D$. To reduce the response time, fewer buckets will be assigned to each disk. The file F will be declustered using the new MDS seed code $C^+ = [n, \frac{M_{I_S}}{m_k}, n-k+2]$, which assigns $\frac{M_{I_S}}{m_k}$ buckets to each disks in the new configuration. This means that each code stored at the beginning into a single disk d_j, for $0 \leq j \leq M_R - 1$, is now spread out among the m_k disks, $\{m_k d_j, m_k d_j + 1, \ldots, (m_k+1)d_j - 1\}$, Moreover, the new solution is strictly optimal because the new response time is $RT^+(q) = \left\lceil \frac{N(q)}{D m_k} \right\rceil$.

Similarly, if the number of available disks reduces from M_R to $\frac{M_R}{m_{k+1}}$, a new MDS seed code $C^- = [n, M_I m_{k+1}, n-k]$ will be used to decluster F. The buckets distributed in m_{k+1} consecutive disks in the initial disk system, are now collapsed into a single disk of the new disk system. The new disk allocation remains strictly optimal. Note that the response time increases because fewer disks are available, and only the PMQ with no more than $n-k-1$ unspecified attributes can now exhibit constant response time.

Finally, observe that when the number (D) of disks available is not a multiple of the attribute domains, the fastest disk allocation strategy achievable with our strategy will involve $\overline{D} = \Pi_{i=n-u+1}^{n} m_i$ disks where $\overline{D}$ is the product of the u largest domains of F such that $\Pi_{i=n-u}^{n} m_i > D \geq \Pi_{i=n-u+1}^{n} m_i$. For every PMQ, q, the response time $RT(q)$ using only $\overline{D}$ disks is slower than the response time $RT^*(q)$ achieved using all the D disks. However, the slowdown is upper bounded by m_{n-u} because $RT(q) < RT^*(q) m_{n-u}$.

Suppose now that D remains unchanged, while the number of attributes of the file F changes. Although this situation may have less impact in practice, this could be the case when more refined searches, using a new attribute of F, are performed or a previously used attribute of the file becomes obsolete. If F is searched according to a new attribute $n+1$, whose domain is m_{n+1}, each bucket $< a_1, \ldots, a_n >$ is partitioned into the m_{n+1} buckets. These are $< a_1, \ldots, a_n, 0 >$, $< a_1, \ldots, a_n, 1 >$, $\ldots$, $< a_1, \ldots, a_n, m_{n+1} - 1 >$, one for each value of the $(n+1)$-th attribute. The number of buckets increases from $M = \Pi_{i=1}^{n} m_i$ to $M' = M m_{n+1}$. Since the number of

disks is unchanged, more buckets must be assigned to each disk. Assuming the MDS $C = [n, M_{I_S}, n-k+1]$, as the seed code according to which F was distributed, the new seed code will be $C^* = [n+1, M_{I_S} m_{n+1}, t]$ where $t \leq n-k+1$. Depending on the value of m_{n+1}, C^* may or may not be an MDS code, and therefore optimality is no longer guaranteed. About the relocation process, the bucket $< a_1, \ldots, a_n >$ initially stored into the disk d_j is now partitioned into m_{n+1} buckets, as mentioned above. Precisely, the buckets $< a_1, \ldots, a_n, x_{n+1} >$, which is the residue representation of $X + kM$ for fixed $k \in [0, m_{n+1} - 1]$, will be assigned to the disk $d_{\left\lfloor \frac{X+kM}{M_{I_S} m_{n+1}} \right\rfloor}$, according to Fact 1.

A similar reasoning applies when the number of attributes of F decreases.

6 Redundant Non-Pairwise Prime Residue Codes

The solution proposed in the last section works for pairwise-prime files. This is not a strong constraint because the domains of the attributes are selected while the database is designed depending on the original interval of the attributes and on the size of the disk page (see Example 2). Nevertheless, for the sake of completeness, let us extend our solution to the case of non-pairwise prime domains.

Consider a set of n non-pairwise prime moduli $m_1, m_2, \ldots, m_k, m_{k+1}, \ldots, m_n$ and also consider all the n-tuples $< x_1, \ldots, x_n >$ for $x_i \in [0, m_i)$, of the n-dimensional space $S = m_1 \times \ldots \times m_n$. Let $M = \Pi_{i=1}^{n} m_i$ and $\tau = \text{l.c.m.}(m_1, m_2, \ldots, m_n)$ be, respectively, the product and the *least common multiple* (l.c.m.) of the n moduli.

In contrast to the pairwise-prime case, there are n-tuples $< x_1, \ldots, x_n > \in S$ which do not represent any integer X. That is, there is no integer X such that $x_i \equiv X \bmod m_i$, for $1 \leq i \leq n$. In fact, a general form of the Chinese Remainder Theorem [11] implies that if S consists of M strings, only the integers in $[0, \tau)$ have a residue representation $< x_1, \ldots, x_n > \in S$. Precisely, if $g_{ij} = \text{g.c.d}(m_i, m_j)$ is the *greatest common divisor* (g.c.d.) of the two moduli m_i and m_j, only

those n-tuples which satisfy the congruence

$$x_i \equiv x_j \bmod g_{ij}, \text{ for } 1 \leq i, j \leq n$$

are *valid* integer representations in the n-dimensional space S.

Note that those τ valid residue representations form a subset of S, which is a code by definition. However, the minimum distance of such a code is unknown (if the code itself is not generated). To overcome this limitation, some explicit redundancy can be added as follows.

Definition 5 *Given n non-pairwise prime moduli $m_1, \ldots, m_k, m_{k+1}, \ldots, m_n$, for fixed $k \in [1, \ldots, n-1]$, let the first k moduli be called* non-redundant*, and the remaining $n-k$ moduli be called* redundant*. Moreover, let $\Lambda = l.c.m(m_1, m_2, \ldots, m_k)$ and $\tau = l.c.m(m_1, m_2, \ldots, m_n)$. The* non-pairwise prime redundant residue code *or NPRR-(n,k)-code, C, consists of the set of valid integer representations, i.e. the set of representations of all integers in $[0, \Lambda)$.*

It is easy to see that the NPRR-(n,k)-code C is a semi-linear code. To derive the minimum distance of the NPRR-(n,k)-code C, let

- $\wp$ denote the l.c.m. of the $n-k$ redundant moduli;
- Λ denote the l.c.m. of the k non-redundant moduli;
- τ denote the l.c.m. of all n moduli;
- partitioning the n moduli into two arbitrary sets s_{d-1} and $s_{\overline{d-1}}$ of sizes $d-1$ and $n-d+1$ respectively, let $\mu_{s_{d-1}}$ and $\mu_{s_{\overline{d-1}}}$ be the l.c.m of the moduli in s_{d-1} and $s_{\overline{d-1}}$,respectively. In other words, $\mu_{s_{d-1}} = l.c.m.(m_{i_j} | m_{i_j} \in s_{d-1})$ and $\mu_{s_{\overline{d-1}}} = l.c.m.(m_{i_j} | m_{i_j} \in s_{\overline{d-1}})$;
- partitioning the n moduli in two arbitrary sets s_d and $s_{\overline{d}}$ of sizes d and $n-d$ respectively, let ν_{s_d} and $\nu_{s_{\overline{d}}}$ be the l.c.m of the moduli in s_d and $s_{\overline{d}}$, respectively. That is, $\nu_{s_d} = l.c.m.(m_{i_j} | m_{i_j} \in s_d)$ and $\nu_{s_{\overline{d}}} = l.c.m.(m_{i_j} | m_{i_j} \in s_{\overline{d}})$;
- $\mathcal{S}_d$ be the set of all possible s_d subsets among $\{m_1, \ldots, m_n\}$;
- $\mathcal{S}_{d-1}$ be the set of all possible s_{d-1} subsets among $\{m_1, \ldots, m_n\}$.

Theorem 2 *Given the radices $m_1, \ldots, m_n$ for fixed $k \in [1..n-1]$, the NPRR-(n,k)-code C has the minimum distance d if and only if the following relation holds:*

$$max_{s_d \in \mathcal{S}_d} \left(\frac{\nu_{s_d}}{\gcd(\nu_{s_d}, \nu_{s_{\overline{d}}})} \right) > \frac{\wp}{\gcd(\Lambda, \wp)} \geq max_{s_{d-1} \in \mathcal{S}_{d-1}} \left(\frac{\mu_{s_{d-1}}}{\gcd(\mu_{s_{d-1}}, \mu_{s_{\overline{d-1}}})} \right).$$

Proof. The semi-linear NPRR-(n,k)-code, C, has minimum distance d if no codeword has Hamming weight $\leq d-1$ (except for the all-zero codeword) and if there is at least one codeword of Hamming weight d. A codeword with $d-1$ non-zero digits represents an integer $X = X' \ \mu_{s_{\overline{d-1}}}$, for some s_{d-1}. The smallest value of X is obtained for $X' = 1$ and $\min_{s_{d-1} \in \mathcal{S}_{d-1}} \left(\mu_{s_{\overline{d-1}}} \right)$. Consequently, the corresponding word is not a codeword if and only if $\min_{s_{d-1} \in \mathcal{S}_{d-1}} \left(\mu_{s_{\overline{d-1}}} \right) \geq \Lambda$, or equivalently $\frac{\tau}{\max_{s_{d-1} \in \mathcal{S}_{d-1}} \left(\frac{\mu_{s_{d-1}}}{\gcd(\mu_{s_{d-1}}, \mu_{s_{\overline{d-1}}})} \right)} \geq \Lambda$ where $\tau = \frac{\mu_{s_{d-1}} \mu_{s_{\overline{d-1}}}}{\gcd(\mu_{s_{d-1}}, \mu_{s_{\overline{d-1}}})}$.
Now, if a codeword of Hamming weight d exists, the above condition must be denied for any subset of d digits. Hence, $\frac{\tau}{\max_{s_d \in \mathcal{S}_d} \left(\frac{\nu_{s_d}}{\gcd(\nu_{s_d}, \nu_{s_{\overline{d}}})} \right)} < \Lambda$.
From the above two conditions and recalling that τ can be expressed as $\tau = \frac{\Lambda \wp}{\gcd(\Lambda, \wp)}$, the claim follows. □

For non-pairwise prime redundant residue codes, Fact 1 can be rewritten as follows:

Fact 2 *For fixed $k \in [1, \ldots, n-1]$ and given a set of n non-pairwise prime radices $m_1, \ldots, m_n$, let $m_1, \ldots, m_k$ and $m_{k+1}, \ldots, m_n$ be respectively the non-redundant and the redundant moduli. Considering the NPRR-(n,k)-code $C = [n, \Lambda, d]$, all the integers in $[0, \tau)$, can be partitioned into $\frac{\tau}{\Lambda}$ codes, each having the same size and the same distance as C. More precisely, code C_j for $0 \leq j \leq \frac{\tau}{\Lambda} - 1$, consists of the set of residue representations of the integers $X \in [j\Lambda, (j+1)\Lambda - 1]$. Note that $C_0 = C$.*

Proof. Follows directly by observing that the Hamming distance of the two codewords x_1 and x_2 belonging to C_0 is the same as the Hamming distance between the two codewords $x_1 + j\Lambda$ and $x_2 + j\Lambda$ belonging to C_j, where $0 \leq j \leq \frac{\tau}{\Lambda} - 1$. □

Procedure Partition ($S = m_1 \times \ldots \times m_n$, C_0)

1. Build $\frac{\tau}{\Lambda}$ codes $C_0, C_1, \ldots, C_{\frac{\tau}{\Lambda}-1}$ as defined in Fact 2, and let $Z = C_0 \cup C_1 \cup \ldots \cup C_{\frac{\tau}{\Lambda}-1}$.
2. From the remaining $(M - \tau)$ n-tuples of the space $S = m_1 \times \ldots \times m_n$, find a set of $\frac{M}{\tau}$ n-tuples $T = \{T(0), T(1), \ldots, T(\frac{M}{\tau} - 1)\}$ such that
   - $T(0)$ is the all-zero codeword;
   - for every pair i, j, with $1 \leq i < j < \frac{M}{\tau}$, $T(i) - T(j) \notin Z$ or $T(j) - T(i) \notin Z$.
3. Compute the multiple-sum between each code $C_0, C_1, \ldots, C_{\frac{\tau}{\Lambda}-1}$ and each n-tuple of T. That is, compute the M/Λ sets $C_j[T(i)]$, where $0 \leq j \leq \tau/\Lambda - 1$ and $0 \leq i \leq M/\tau - 1$.

Figure 2: Procedure Partition

Finally, let the *multiple-sum* set, $C[x] = \{x + w | w \in C\}$, be obtained by the modular sum, digit-by-digit, of the n-tuple x over all the n-tuples of the code C.

Now, for a given NPRR-(n, k)-code C_0, the entire n-dimensional space $S = m_1 \times \ldots \times m_n$ can be partitioned by the algorithm in Figure 2.

To prove the correctness of the above procedure, let us prove the following.

Lemma 4 *The set of subsets* $\{C_j[T(i)] \mid 0 \leq j \leq \tau/\Lambda - 1$ *and* $0 \leq i \leq M/\tau - 1\}$ *forms a partition of the space* $S = m_1 \times \ldots \times m_n$. *Moreover, each subset* $C_j[T(i)]$ *is a code* $[n, \Lambda, d]$.

Proof. First note that $C_j[T(0)] = C_j$, for $0 \leq j \leq \tau/\Lambda - 1$. Next let us prove, by contradiction, that $C_j[T(i)] \cap C_r[T(s)] = \emptyset$ for any choice of i, j, r, s. Suppose that the two n-tuples $X(j, i) \in C_j[T(i)]$ and $Y(r, s) \in C_r[T(s)]$, obtained by the residue vectors $X, \ Y \in C_0$, are equal. W.l.o.g. assume $j \geq r$. Note that $X(j, i)$ is the sum of the residue representation of $(X + j\Lambda)$ and of the n-tuple $T(i)$, while $Y(r, s)$ is the sum of the residue representation of $(Y + r\Lambda)$ and of the n-tuple $T(s)$. Now $X(j, i)$ is equal to $Y(r, s)$ if and only if $X - Y + (j - r)\lambda = T(s) - T(i)$ belongs to C_{j-r}. This cannot be true since it is against the criteria used for the selection of the n-tuples in T. Hence, the subsets in $\{C_j[T(i)] \mid 0 \leq j \leq \tau/\Lambda - 1$ and $0 \leq i \leq M/\tau - 1\}$ form a partition of the space $S = m_1 \times \ldots \times m_n$.

It still remains to prove that, for any pair j, i, the set of subsets $C_j[T(i)]$ has the same minimum distance as C_0. Consider two arbitrary n-tuples $X(j, i)$ and $Y(j, i)$, both belong-

Procedure **Non-Pairwise Prime Decluster**
$(k,\ F(m_1, \ldots, m_k, m_{k+1}, \ldots, m_n),\ C_0 = [n, \Lambda, d],\ D = \frac{M}{\Lambda})$

1. Partition $(S = m_1 \times \ldots \times m_n,\ C_0)$;
   /* This step builds the D codes $\{C_j[T(i)] \mid 0 \leq j \leq \tau/\Lambda - 1$ and $0 \leq i \leq M/\tau - 1\}$. */
2. For $0 \leq j \leq \tau/\Lambda - 1$ and $0 \leq i \leq M/\tau - 1$,
   assign code $C_j[T(i)]$ to the disk $d_{i\frac{\tau}{\Lambda}+j}$.

Figure 3: Non-Pairwise Prime Decluster Procedure

ing to $C_j[T(i)]$, and derived from X and Y in C_0. Repeating the above reasoning, it is easy to see that the n-tuples $X(j,i) - Y(j,i)$ and $X - Y$ are the same. Hence, all the subsets in $\{C_j[T(i)] \mid 0 \leq j \leq \tau/\Lambda - 1$ and $0 \leq i \leq M/\tau - 1\}$ inherit the same minimum distance d as the set C_0. □

Finally, the decluster procedure for non-pairwise primes is described in Figure 3.

Theorem 3 *For a fixed $k \in [1..n-1]$, let $F(m_1, \ldots, m_k, m_{k+1}, \ldots, m_n)$ be a file whose attributes are not pairwise prime and let $C_0 = [n, \Lambda, d]$ be the seed code. Distributing F among $D = \frac{M}{\Lambda}$ disks using the Non-Pairwise Prime Decluster procedure, the response time for a PMQ, q, with s specified attributes $q_{i_1}, \ldots, q_{i_s}$ is given by $RT(q) = \left\lceil \frac{\Lambda}{M_s} \right\rceil$, where $M_s = l.c.m(m_{i_1}, \ldots, m_{i_s})$. Finally, the ratio between the response time of this procedure, say $RT(q)$, and the best achievable response time, say $RT^*(q)$, is given by $\left\lceil \frac{RT(q)}{RT^*(q)} \right\rceil = \left\lceil \frac{\left\lceil \frac{\Lambda}{M_s} \right\rceil}{\left\lceil \frac{\Lambda}{\Pi_{j=1}^{s} m_{i_j}} \right\rceil} \right\rceil$.*

Proof. Consider the n-tuple q' belonging to $S = m_1 \times m_2 \times \ldots \times m_n$ and obtained by substituting all the unspecified attributes of the PMQ, q, with 0. By the previous discussion, there must be i and j such that $q' \in C_j[T(i)]$, and q' is stored into disk $d_{i\frac{\tau}{\Lambda}+j}$ and there exists $x \in C_0$ such that $q' = x + j\Lambda + T(i)$. Then, $d_{i\frac{\tau}{\Lambda}+j}$ stores q' along with the buckets $q' + uM_s = x + j\Lambda + T(i) + uM_s$, for all values of u such that $0 \leq (x + uM_s) \leq \Lambda$. Note that uM_s has always a valid residue presentation since $uM_s < \tau$. Therefore, at most $\left\lceil \frac{\Lambda}{M_s} \right\rceil$ buckets qualify for the query q on the same disk.

It remains to show how far is the response time for non-pairwise prime decluster procedure

from the optimal response time. Using $D = \frac{M}{\Lambda}$ disks, the optimal response time for q would be $RT^*(q) = \left\lceil \frac{N(q)}{D} \right\rceil = \left\lceil \frac{\frac{M}{\Pi_{j=1}^{s} m_{i_j}}}{\frac{M}{\Lambda}} \right\rceil = \left\lceil \frac{\Lambda}{\Pi_{j=1}^{s} m_{i_j}} \right\rceil$.

Therefore, our solution is $\left\lceil \frac{\lceil \frac{\Lambda}{M_s} \rceil}{\lceil \frac{\Lambda}{\Pi_{j=1}^{s} m_{i_j}} \rceil} \right\rceil$ times slower than the strictly optimal solution. □

Example 3: Consider the redundant residue number system (RNS) associated with the non-pairwise prime radices, $m_1 = 3, m_2 = 6, m_3 = 5$. For fixed $k = 2$, let m_1, m_2 be the non-redundant moduli, and let m_3 be the redundant modulus. According to our definitions, $M = 90$, $\tau = 30$, and $\Lambda = 6$. In other words, the space S consists of $3 \times 6 \times 5$ number of 3-tuples, but the RNSs associated with the moduli m_1, m_2, m_3 and m_1, m_2, can represent the integers in $[0, 29]$ and $[0, 5]$, respectively. Let the code $C = [3, 6, 2]$ consist of the residue representations of the integers in $[0, 5]$ in the RNS of radices $m_1 = 3, m_2 = 6, m_3 = 5$. That is, $C = \{< 0,0,0 >, < 1,1,1 >, < 2,2,2 >, < 0,3,3 >, < 1,4,4 >, < 2,5,0 >\}$.

From Fact 2, the valid residue representations are partitioned into $\frac{90}{6} = 15$ codes. Precisely, code $C_j[0,0,0]$ corresponds to the residue representations of the integers in the range $[6j, 6(j+1) - 1]$, where $0 \leq j \leq 4$. In particular,

$C_1 = \{< 0,0,1 >, < 1,1,2 >, < 2,2,3 >, < 0,3,4 >, < 1,4,0 >, < 2,5,1 >\}$,

$C_2 = \{< 0,0,2 >, < 1,1,3 >, < 2,2,4 >, < 0,3,0 >, < 1,4,1 >, < 2,5,2 >\}$,

$C_3 = \{< 0,0,3 >, < 1,1,4 >, < 2,2,0 >, < 0,3,1 >, < 1,4,2 >, < 2,5,3 >\}$, and

$C_4 = \{< 0,0,4 >, < 1,1,0 >, < 2,2,1 >, < 0,3,2 >, < 1,4,3 >, < 2,5,4 >\}$.

Now let $T = \{T(0) =< 0,0,0 >, T(1) =< 1,0,3 >, T(2) =< 2,3,0 >\}$ be the set of 3-tuples such that $T(1)$, $T(2)$, $T(1) - T(2) =< 2,3,3 >$ and $T(2) - T(1) =< 1,3,2 >$ does not belong to $Z = C_0 \cup C_1 \cup C_2 \cup C_3 \cup C_4$. Hence, the codes $C_j[1,0,3] = \{< 1,0,3 > + C_j[0,0,0]\}$ and $C_j[2,3,0] = \{< 2,3,0 > + C_j[0,0,0]\}$, for $0 \leq j \leq 4$, complete the partition of S.

Finally, the non-pairwise prime decluster procedure assigns the code $C_j[T(i)]$ to the disk d_{5i+j}. Altogether there are 15 codes and hence $D = 15$ disks are required.

Now, consider the PMQ $q =< *, *, 2 >$. Since $M_s = 5$, there are at most $\left\lceil \frac{\Lambda}{M_s} \right\rceil = 2$ qualifying buckets for each disk. Hence, $RT(q) = 2 = RT^*(q)$ in this case.

7 Conclusion

In this paper, we studied the disk allocation problem for distributing non-uniform multi-dimensional files on to parallel database systems that exploit the ability of accessing multiple disks simultaneously. Based on a large and flexible class of maximum distance separable codes, called the redundant residue codes, a strictly optimal allocation method is derived for every query q when the attribute domains of the multi-dimensional file are pairwise prime. We also introduced a new family of residue codes, called the *redundant non-pairwise prime residue codes*, that can be applied with multi-dimensional files that have attribute domains non-pairwise prime.

Acknowledgment: The second author is grateful to Prof. Barsi to introduce her to the area of the redundant residue number systems. The authors would also like to thank the reviewers for helpful suggestions and the associate editor for timely handling the paper.

References

[1] K.A.S. Abdel-Ghaffar and A. El Abbadi, "Optimal Disk Allocation for Partial Match Queries", *ACM Trans. on Database Systems*, Vol. 18, No.1, March 1993, pp. 132-156.

[2] F. Barsi and P. Maestrini, "Error Correcting Properties of Redundant Residue Number Systems", *IEEE Trans. on Computers*, Vol. 22 , 1973, pp.307-315.

[3] F. Barsi and P. Maestrini, "Error Codes in Residue Number Systems with Non-Pairwise-Prime Moduli", *Information and Control*, Vol. 46, No. 1, July 1980, pp. 16-25.

[4] S. K. Das and M. C. Pinotti, "An Optimal Disk Allocation Strategy for Partial Match Query of Non-Uniform Cartesian Product Files," *Proc. IEEE Int'l Parallel Processing Symposium*, San Juan, Puerto Rico, pp. 550-554, Apr 1999.

[5] H.C. Du and J.S. Sobolewski, "Disk Allocation for Cartesian Product Files" *ACM Trans. on Database Systems*, Vol. 7, No. 1, March 1982, pp. 82-101.

[6] M.F. Fang, R.C.T. Lee and C.C. Chang, "The Idea of De-Clustering and its Applications", in *Proceedings 12th Int'l Conf. VLDB*, Kyoto, Japan, Aug. 1986, pp. 181-188.

[7] C. Faloutsos and D. Metaxas, "Disk Allocation Methods using Error Correcting Codes", *IEEE Trans. on Computers*, Vol. 40, No. 8, August 1991, pp. 907-913.

[8] M.H. Kim and S. Pramanik, "Optimal File Distribution for Partial Match retrieval" in *Proceedings of the ACM-SIGMOD Int'l Conference on Management of Data*, Chicago, 1988, pp. 173-182.

[9] H. Khrisna, K. Lin and J. Sun, "A Coding Theory Approach to Error Control in Redundant Residue Number Systems – Part I: Theory and Single Correction", *IEEE Trans. on Circuits and Systems, II*, Vol. 39, No. 1, January 1992, pp. 8-17.

[10] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes, Parts I and II*, North-Holland, New York, 1977.

[11] O. Ore, "The General Chinese Remainder Theorem", *Amer. Math. Monthly*, 1952, pp. 365-370.