MULTI-DISK ALLOCATION METHODS

FOR BANG FILES

By

JAE-MYEONG JEON

Bachelor of Science

Air Force Academy

Chung-won, R. O. K.

1988

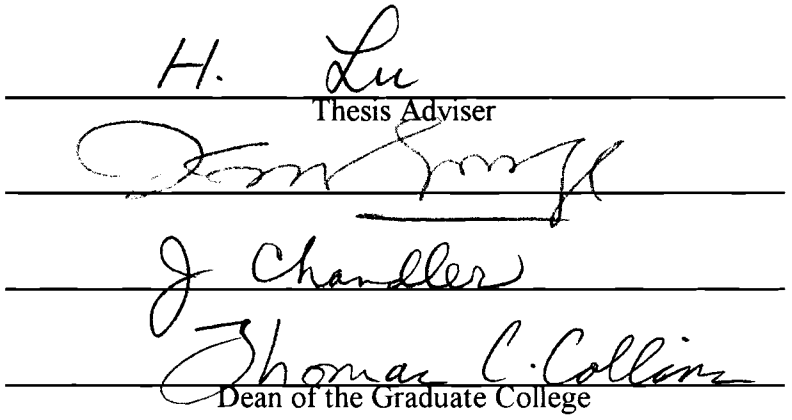
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1995

OKLAHOMA STATE UNIVERSITY

MULTI-DISK ALLOCATION METHODS

FOR BANG FILES

Thesis Approved:



Thesis Adviser

Dean of the Graduate College

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my thesis adviser, Dr. Huizhu Lu, for her constructive guidance, patience, and encouragement through this study. My sincere appreciation extends to my other committee members Dr. J. P. Chandler and Dr. K. M. George, for their guidance, assistance, and encouragement.

Moreover, I wish to express my sincere gratitude to my government, R.O.K., for their financial support.

I would like to give my special appreciation to my wife, Young-jae, for her strong encouragement at times of difficulty, love and understanding throughout this whole process. Thanks also go to my parents for their support and encouragement. Thank God for everything.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

MISCELLANEOUS

A record r is defined to be an ordered n-tuple $(r_1, r_2, \ldots, r_n)$ of values chosen from $\Sigma = D_1 \times D_2 \times \ldots \times D_n$. The set D_i is the ith domain from which the value r_i is chosen. A subset consisting of the k attributes that uniquely identifies a record forms the *key space* [9]. Query(q) is defined to be the set of records denoting the response to a query q.

Definition: Exact match query: Query(q) contains just a single record of Σ [9].

Definition: Partial match query: $\text{Query}(q) = \{r \subseteq \Sigma \mid (\forall_j,\ 1 \le j \le n)\ (q_j = \text{'*'} \text{ or } q_j = r_j)\}$ for a query $q = (q_1, q_2, \ldots, q_n)$ [9].

Definition: Orthogonal range query: $\text{Query}(q) = \{r \subseteq \Sigma \mid (\forall_j,\ 1 \le j \le n)\ (\min_j \le r_j < \max_j)\}$ for a query $q = ([\min_1, \max_1), \ldots, [\min_n, \max_n))$ [9].

Definition: Let A be a set and R a relation on A. The relation R is

(1) *Reflexive* if a R a for all a in A,

(2) *Symmetric* if "a R b" implies "b R a" for a, b in A, and

(3) *Transitive* if "a R b and b R c" implies "a R c" for a, b, c in A. The elements a, b, and c need not be distinct [15].

Definition: *Equivalence relation* is a relation which is reflexive, symmetric, and transitive [15].

Definition: *Equivalence classes* are disjoint subsets of a set A which is partitioned by an equivalence relation R [15].

To formalize queries for the BANG file, a grid transforms the Euclidean Space into a Cartesian Space with a range of 20. The axes are numbered from 0 to 19 and only the first quadrant is used. The maximum value of the data is 1.00 and the minimum value is 0.00 therefore the distance between points on the axes is 0.05. We can identify cells with a pair of cell identifier (x, y). N-dimensional space can be described with N-tuples (x_1, x_2,..., x_n), where x_i is an integer.

Definition: Orthogonal path query - An orthogonal path query is a set represented by ($x_1..x_2$, $y_1..y_2$), where either ($x_1 = x_2$) or ($y_1 = y_2$) [4].

Definition: ColSet(M) - The ColSet(M) is an orthogonal range query represented by {($x_i..x_i$, 0..M-1),| for i = j..(j+M-1)}, where j and M are natural numbers [4].

Definition: RowSet(M) - The RowSet is an orthogonal range query set represented by {(j..(j+M-1), $y_i..y_i$), for i = 0..M-1}, where j and M are natural numbers [4].

Definition: Query set Principle Diagonal PD(M) - The Query set Principle Diagonal PD(M) is a set of queries {PD_0, PD_1, PD_2, .., PD_{2M-2}} where query $PD_j = (x_i, y_i)$ | region include (x_i, y_i) AND $x_i + y_i = j$, for all j = {0,1,..2M-2}, where j and M are natural numbers [4].

Definition: Query set Anti-Diagonal PA(M) - The query set Anti-diagonal PA(M) is a set of queries {PA_0, PA_1, PA_{-1}, .., PA_M, PA_{-M}} where query $PA_j = (x_i, y_i)$ | region include (x_i, y_i) AND $x_i - y_i = j$, for all j = {0,1, -1, ..., M, -M}, where j and M are natural numbers [4].

Definition: BSR(H,W) - The query set Basic Small Rectangles BSR(H,W) is a collection of orthogonal range queries representing rectangles with height ≤ H and width ≤ W, where H and M are natural numbers [4].

Definition: BLR(H, W) - The query set Basic Large Rectangles BLR(H, W) is a collection of orthogonal range queries representing rectangles with either height = k * H or width = k * W cells, for k, H and W are natural numbers. BLR(H, W) is formally represented as ($x_1..x_2$, $y_1..y_2$), where $(x_2-x_1) \bmod W = 0$ or $(y_2-y_1) \bmod H = 0$ [4].

Definition: Balanced allocation methods - An allocation method is balaned with respect to a query if the same number of cells in the query are assigned to each disk by the allocation method. If the query has $|Q|$ cells and if there are M disks, then the allocation method is balanced with respect to the query Q if and only if the allocation method puts at least $\lfloor |Q|/M \rfloor$ cells and at most $1 + \lfloor |Q|/M \rfloor$ cells onto the disk. If $|Q|$ is divisible by M, then each disk gets ($|Q|/M$) cells [4].

Definition: Perfect allocation method - An allocation method is perfect with respect to a query set $Q_s = \{Q_1, Q_2, ..., Q_k\}$ if and only if the allocation method is balanced for all Q_i, with i = (1, 2, ..., k) [4].

CHAPTER I

INTRODUCTION

Background

Spatial databases are designed for a wide variety of observational multi dimensional data such as geographic data. The design of a successful database system calls for efficient data maintenance and information retrieval. Data maintenance means operations such as insertion, deletion, and updates of data. Information retrieval means an operation to retrieve a set of data matches a user's query. Some of the common queries are exact match, partial match, and orthogonal range queries. An exact match query specifies the attribute that uniquely identifies the record as in "Retrieve the record of the car whose license is abc123 and is registered in Oklahoma.". A partial match query specifies a subset of attributes as in "Retrieve the records of all students who are 25 year old and who are graduate students.". An orthogonal range query specifies the range of the attributes as in "Retrieve the records of those students who major in Statistics and whose grades range from 3.5 to 4.0.".

A multi-disk system has the potential of servicing a range query for data on a number of disks concurrently. The number of disks that take part in the execution of a request depends on the location of the data to be accessed and the amount of data to be transferred. When more than one disk is involved in the execution of a request, the request is divided into a number of service demands, each of which is executed

independently by a different disk [9]. A query may be composed of one or more subqueries. Each query results in a seek to the location of a data bucket and a read or a write of the bucket.

Because of the importance of range queries in data retrieval, much concentration has been devoted in recent years to the problem of designing multi-attribute file systems for such queries. Design for range queries of multi-disk allocation methods can be delineated as follows: Given a set of regions, arrange the buckets into disks in such a way that the number of continuously examined disk, over all possible range queries, is minimized. It is desirable to achieve some degree of parallelism in examining required buckets in order to reduce the access time.

The Problem

Design of spatial databases and access methods for a single disk have improved rapidly in the past fifteen years [1-3]. The past few decades have seen admirable advances in main memory access times and CPU speed. However, I/O channels have not developed as much CPUs and main memory and the design of a multi-disk database to process spatial queries is not well developed. For that reason, a relatively slow single-disk database has resulted in a processing bottleneck in I/O. We have studied I/O parallelism as a method for decreasing the bottleneck in the BANG file.

Purpose of the Study

In this thesis we focus on finding a way to allocate buckets given a multi-disk system to achieve minimum response time for all range query sets. This means the maximal possible disk access concurrence would be achieved when examining the required buckets. One of the primary goals of a multi-disk allocation method is to support efficient data retrieval, which is crucial to any applications of the computer. We propose two new allocation methods to make BANG file queries more efficient. Storage units of the BANG file can be accessed simultaneously, and parallel processing of a range query is achieved by distributing data evenly among the storage units. When data is efficiently distributed for every possible range query, maximal parallelism is obtained. Our scheme uses multiple disks that can be accessed simultaneously.

Significance of the Study

The most common method of achieving I/O parallelism is to distribute data concurrently to multiple I/O processors; that can speed I/O processing by a factor of the number of disks used. The proposed disk allocation methods partition data among several disks and maximize output of a database manager in processing various queries. We use a simple hardware architecture which has M number of independent I/O processors and the same number of disks, and suggest methods of distributing the BANG file's data buckets on this architecture. We study several disk allocation methods to distribute newly created buckets and reduce response time of the range queries to the BANG file.

Conceptual Assumptions

For the purposes of this thesis we assumed that every file is divided into pages (buckets) consisting of one or more records (i.e., we do not consider the case where a record is spread over several pages) and the whole file is stored on a secondary storage device such as magnetic disk. When the secondary device is accessed, a whole page is brought into primary memory. Since the time for disk accesses or seeks is considerably longer than the main memory access time or the internal machine instruction time, the time taken to respond to a query can be simply measured in terms of the number of distinct disk accesses issued. The number of discrete disk accesses that must be issued to respond to a query is equal to the number of buckets that contain at least one record satisfying the query.

It is assumed that one bucket can be accessed on one disk unit in one unit of time, several buckets can be accessed in one unit of time if they are on distinct and independently accessible disk units. Also, the access time of a bucket is the same no matter how many records it has and no matter on which disk it is contained. The response time to a range query in this case is no longer proportionate to the total number of buckets to be examined, but becomes proportional to the maximum number of buckets that need to be examined on a particular disk unit.

Suppose buckets 1 and 2 are stored on the first disk unit and buckets 3 and 4 are stored on the second disk unit of a two-disk system. The average time needed to respond to a partial match query (a, b, c, d) in TABLE 1, on the two-disk system, would be two

units instead of four units for a single-disk system. Both disks could be accessed simultaneously to retrieve two buckets in one unit of time. Similarly, the time response for the query (a, b, i, j), TABLE 2, is one unit instead of the two units required for the single-disk system. Note, however, that the time response for the query (a, b, e, f), TABLE 2, is two units for the single-disk and the two-disk systems. Thus the response times will be minimized if we maximize concurrence of disk accesses.

TABLE 1

BUCKETS AND RECORDS ASSIGNED TO THE BUCKETS;
FIRST QUERIES AND BUCKETS TO BE EXAMINED

Bucket 1	Bucket 2	Bucket 3	Bucket 4
a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

Queries	Buckets to be examined
a, b, c, d	1, 2, 3, 4
e, f, g, h	1, 2, 3, 4
i, j, k, l	1, 2, 3, 4
m, n, o, p	1, 2, 3, 4

TABLE 2

BUCKETS AND RECORDS ASSIGNED TO THE BUCKETS; SECOND QUERIES AND BUCKETS TO BE EXAMINED

Bucket 1	Bucket 2	Bucket 3	Bucket 4
a	e	i	m
b	f	j	n
c	g	k	o
d	h	l	p

Queries	Buckets to be examined
a, b, e, f	1, 2
c, d, g, h	1, 2
i, j, m, n	3, 4
k, l, o, p	3, 4
a, b, i, j	1, 3
c, d, k, l	1, 3
e, f, m, n	2, 4
g, h, o, p	2, 4

CHAPTER II

REVIEW OF THE LITERATURE

Independently accessed multi-disk systems with M disks provide an opportunity for M-way parallelism in disk operations. This parallelism can serve up to M-times speed-ups in processing large spatial queries [4]. Several multi-disk allocation methods have been proposed that use the values of key attributes or of coordinate space embedding [9]. Hashed-declustering, round-robin, and parallel R-trees [7] are based on key values. The coordinate, space-partitioning techniques include the Disk Modulo Method [8].

The properties of multidimensional storage systems are abstracted in terms of their ability of carrying out M-independent disk operations in parallel [4]. We consider the storage systems as being a collection of logical disks, each with an independent read/write head, and an independent channel to transfer data to and from the processor's memory. Disk blocks accessed from different logical disks are independent, and are carried out in parallel. Multi-dimensional data refers to a collection of data values embedded in a coordinate space which has dimension ≥ 2. The coordinate space may represent a Euclidean space or a logical space. Data values embedded in an Euclidean space represent the measured properties of a physical system. For example, the measurements of temperature, pressure, and chemical concentration for different points of a physical system such as an ocean are spread over three-dimensional Euclidean space. The data values embedded in a logical space may represent data assembled by experiments and models that illustrate the relationship among state variables of the phenomenon under study. For

example, the value of a volume of gas at different temperatures and pressures gives rise to a set of data points spread across three-dimensional Cartesian space, representing orthogonal dimensions of temperature, pressure and volume.

BANG file

Data partitioning schemes can be described by a partitioning algorithm of the BANG file [3,5]. We will work with a two-dimensional coordinate space to depict our approach. The result can be generalized to higher dimensions. The BANG file maintains the correspondence between the data bucket and grid region. If a region overflows, splitting is performed until the best balanced condition is achieved by the partitioning algorithm. Figure 1 shows how the BANG file system handles the data.

A region numbering method of the BANG file provides the following properties [5].

- Each region is represented by a unique number pair $<r, l>$, where r is a region number and l is the level number of the region.
- The number can be computed from a given set of n key values ($k_1, k_2, .., k_n$) and the partial level of each dimension ($l_1, l_2, .., l_n$), where level number $l = \sum l_i$.
- Given a region number r at level l, then the region R encloses region r at level l-1 can be computed by removing the most significant bit.

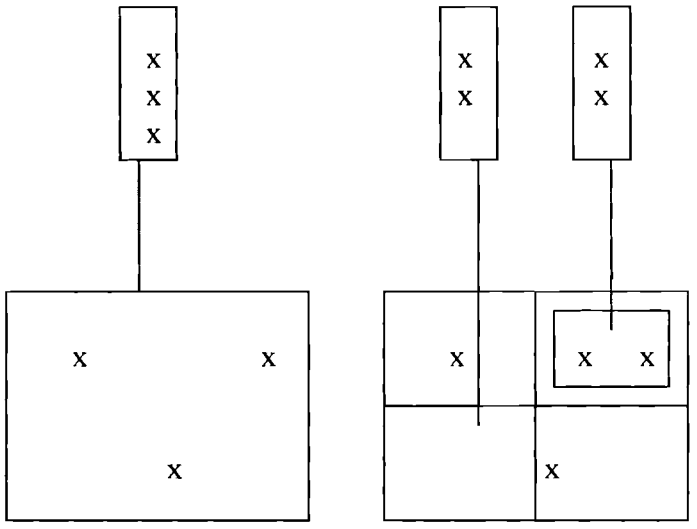
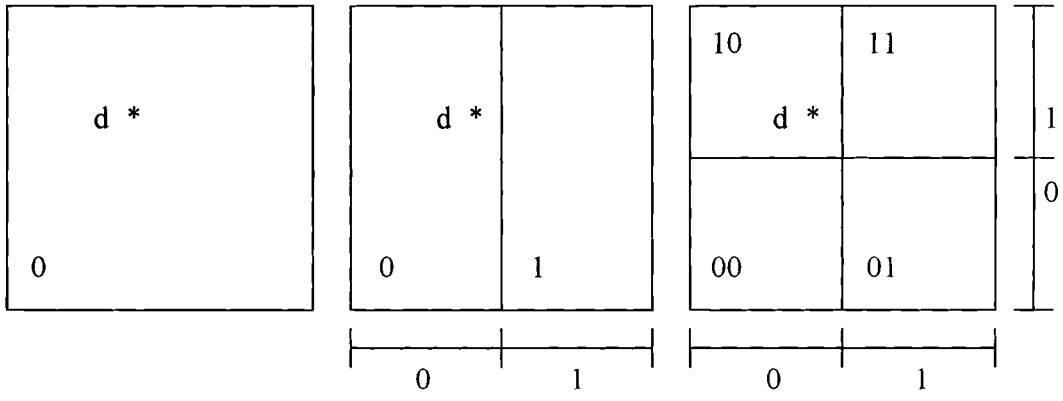


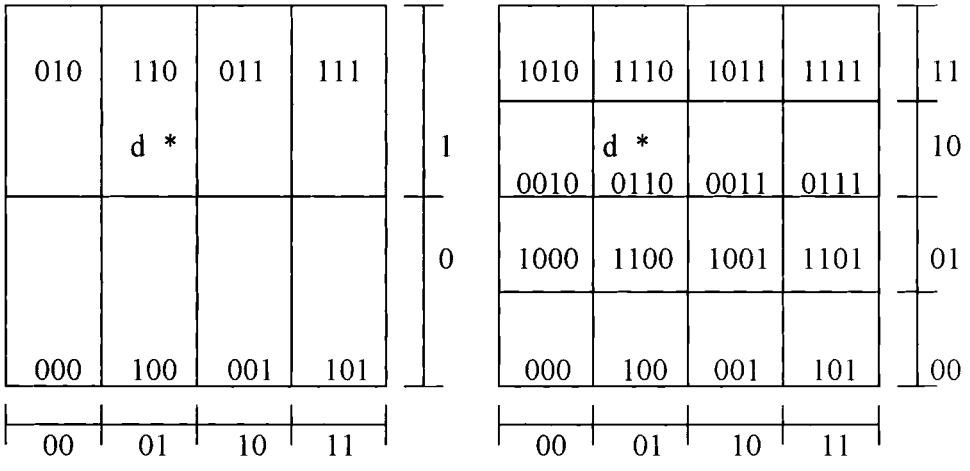
Figure 1. The BANG file structure(Max block capacity : 3)

Figure 2 shows an example of this numbering method. Assuming that there is no preferred attribute, then a cyclic partitioning through the dimensions is suitable [3]. All regions and their corresponding coordinate numbers are given in binary representation. Based on figure 2, we obtain some conclusions.

- The total number of the grid regions at level l is 2^l.
- Level l is generated from level (l-1) by splitting each grid region into two in some chosen dimension.
- The grid region r at level l can be split into two unique region numbers r and $r + 2^l$ at level (l+1). This can be achieved by concatenating the least significant bit of the newly-formed coordinate in dimension i at level $l + 1$ to the most significant bit of the corresponding region number at level l. For example, a splitting of the grid region 110 in Figure 2 (d) produces 2 grid regions numbered 0110 and 1110.



(a) level = 0 (b) level = 1 (c) level = 2



(d) level = 3 (e) level = 4

Figure 2. Numbering method

- If a record d falls into a region *r* at level *l*, then the region that enclosed d at level *l*-1 can be computed by removing the most significant bit from *r*. For example, a record d in Figure 2 (e) that is enclosed in region 0110 at level *l* = 4 is also enclosed in region 110 in level *l*=3.

Figure 3 depicts a tree representation of the splitting history of the data space with dimension = 2. Each directory entry of the BANG file structure is a number pair <*r*, *l*>, where *r* is a unique region identifier and *l* is a level number. To avoid high pointer overhead, the BANG file system maintains a one-to-one correspondence between the directory entry and the data block. In order to maintain the correspondence, the BANG

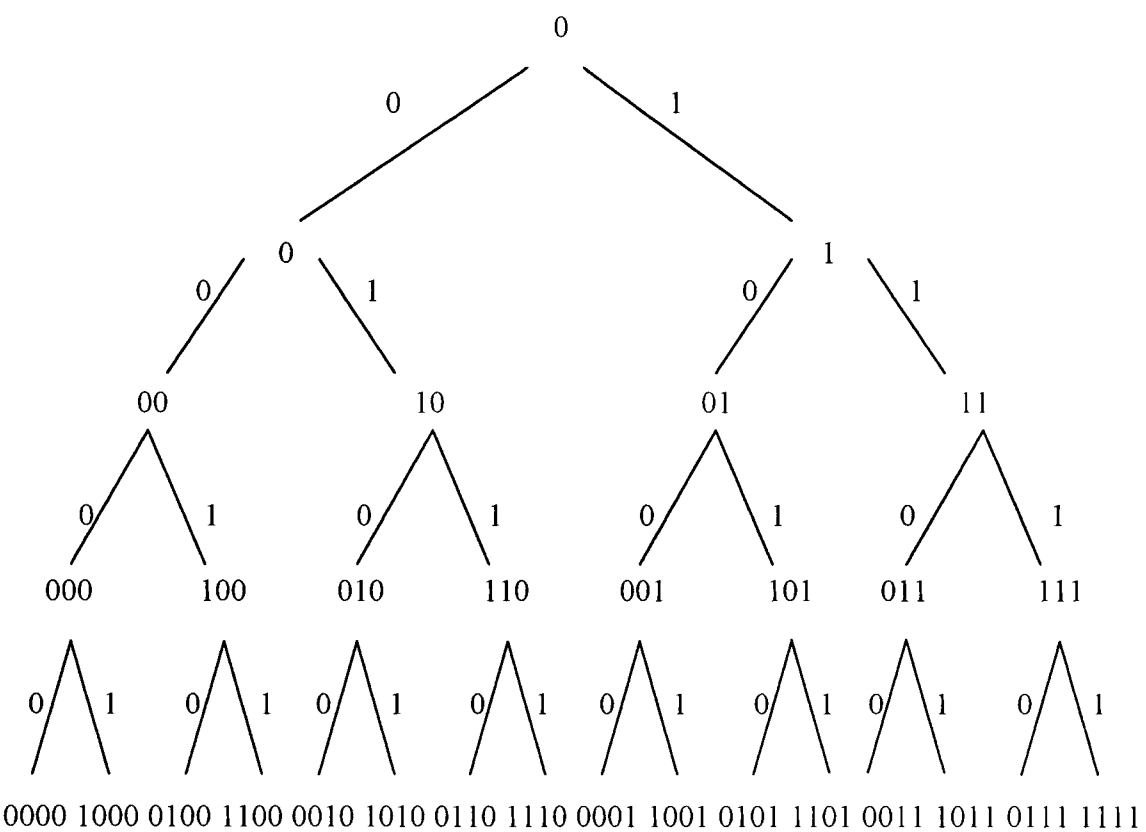


Figure 3. Splitting history for 2 dimensional data space

file system does not require the subspace corresponding to a data region to be a hyperrectangle.

Consider the current state of a data space organized by the BANG file system as shown in Figure 4 (a). The data space is partitioned into two block regions R1 and R2. R1 encloses the entire data space and R2 is enclosed within R1. S2 is the subspace enclosed by region R2, and S1 the subspace enclosed by region R1 minus subspace S2. The directory of the BANG file in this state contains 2 entries. The first entry <0, 0> points to a bucket that contains all records included in the subspace S1, and the second entry <1, 2> points to a data bucket that contains all records included in subspace S2. Since the data d, in Figure 4 (a), is enclosed by <1, 2>, <1, 1>, and also <0, 0>, an ambiguity may arise during the search of the directory to find an entry that points to a correct data bucket which contains data d. For example, a directory of the BANG file with the current state as shown in Figure 4 (b) has 4 entries (<0, 1>, <1, 2>, <3, 2>, <4, 3>). Suppose the directory entries are arranged as above, and a query for a record d1 is issued. The

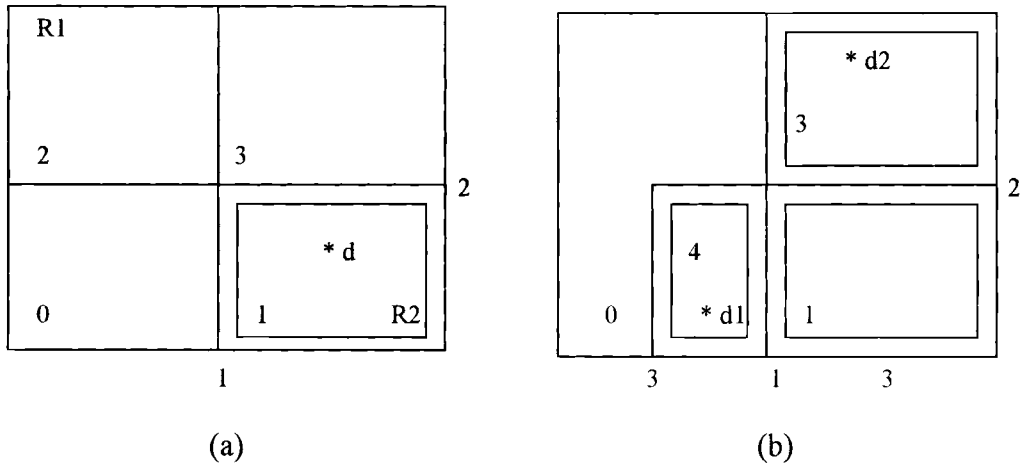


Figure 4. Nested Directory

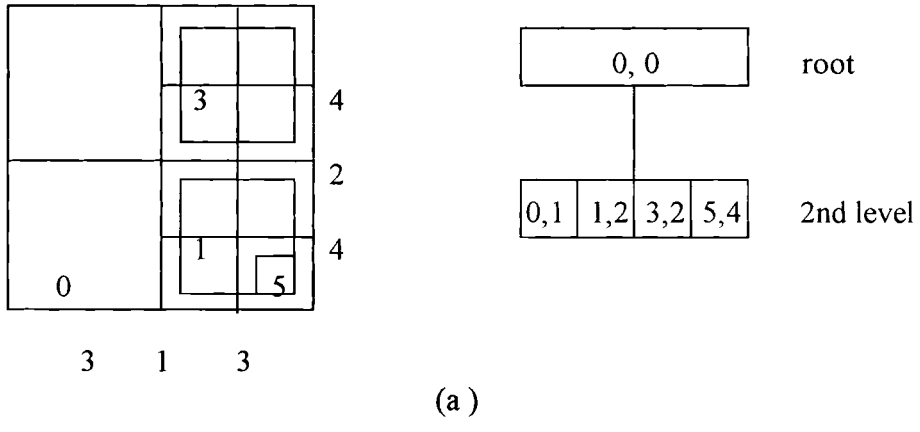
smallest region at current level *l* that encloses the location of d1 in the data space can be computed by transforming the set of its key values into a unique region identifier by using the mapping function. Based on this identifier, all regions that may enclose d1 at every level can be derived. In this case they are <0,0>, <0,1>, <0,2>, and <4, 3>. During the search of the directory, the first entry that matches with these regions is <0, 1>, but the record d1 is not located in the data bucket pointed to by <0,1>. To avoid this ambiguity, the directory entries of the BANG file structure are arranged in order of increasing partition level. During a search fora data d1, the directory is scanned to find the smallest region that encloses d1. In this example the directory entry <4, 3> is examined first.

If the directory does not contain the entry for the smallest region *r* at the current level *l* that encloses data d, the search is continued until the smallest region that encloses d is encountered. As an example, the smallest region that encloses d2 as shown in Figure 4 (b) is <3, 3> since the smallest level in the directory is 3. This region identifier is not directly recorded in the directory. The search is continued for the next smallest region <3, 2> that encloses d2. This example shows that, although the directory does not contain an entry for the smallest region that encloses a data d at current level, there is no ambiguity in locating the correct directory entry as long as the entries are arranged in increasing level number and some merging constraints are maintained.

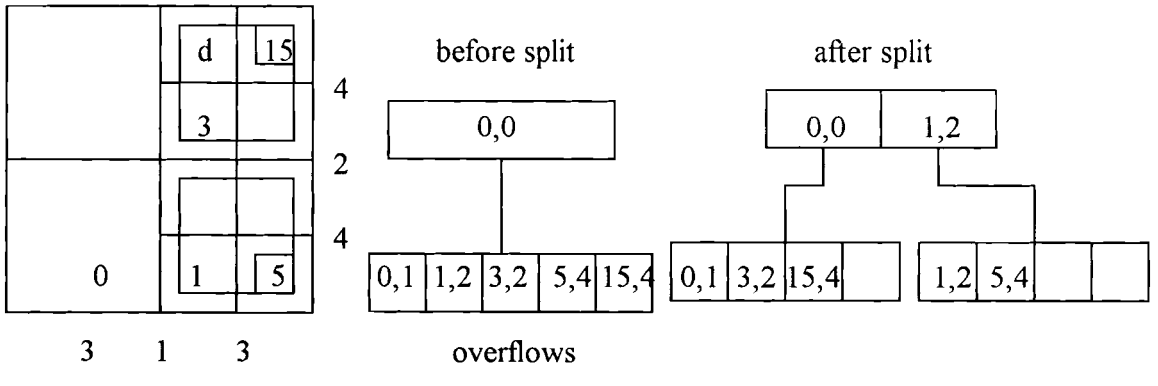
In a large database system, the size of the BANG file directory is too large to be stored in primary memory. Therefore, the directory entries should be distributed among a sequence of disk blocks. To have the same balancing algorithm as for the data buckets, the BANG file directory buckets are managed by another BANG file. In the following discussion, we name the directory of the BANG file as root directory and the directory

that manages the data points as second-level directory. When a directory bucket overflows, it is split by using the same method applied to the data bucket. Each directory entry is treated as a data point. A partition algorithm is invoked recursively until the best balance condition is achieved.

In the root directory, each entry points to a second-level directory bucket. All entries of the second-level directory, in which corresponding regions are enclosed in the region represented by the entry in the root directory, have to be stored in the same directory bucket. The arrangement of the entries of the root directory is the same as the second-level directory, that is, all entries are arranged in increasing partition level.



(a)



(b)

Figure 5. Directory split

In Figure 5 we show an example of how the secondary-level directory bucket is split and its effect on the root directory. Assume that maximum capacity of a directory bucket is four entries, and the current state of the BANG file system is shown in Figure 5 (a).

At first, the root directory contains a single entry that represents the whole data space. Suppose insertion of a data d causes a data bucket to overflow. The partitioning of this data bucket introduces a new entry into the second level directory bucket, which in turn causes this directory bucket to overflow. The result of the partitioning of this directory bucket is shown in Figure 5 (b). If the content of the regions falls below a certain threshold, then the merging algorithm is invoked to maintain a responsible storage utilization.

In a dynamic file system, insertions and deletions of records are intermixed with the queries. If an insertion of a record causes the bucket to overflow, the partitioning algorithm is invoked to partition the corresponding block region into two new regions. This procedure is repeated until the best balance condition is achieved. In order to maintain a responsible storage utilization, a merging algorithm is invoked if a deletion causes the bucket capacity to fall below a certain threshold.

An insertion of a record into a data bucket may cause it to overflow. In order to maintain a one-to-one correspondence between a data bucket and the directory entry, the overflowing data bucket has to be partitioned as follows:

- The partitioning of the data space is performed until the best balance condition is achieved. If n, $n > 1$, partitions are needed to achieve this balance, then the directory

entry <r, l> of the old region is modified into two new entries <r, l> and <$r1$, $l+n$>, where $r1$ is the identifier of the newly formed region.

- If the balance condition is achieved at the first level of division, then the directory entry <r, l> of the old region is replaced by the identifiers of the two resulting regions <r, $l+1$> and < $r+2^l$, $l+1$>.
- The partitioning of a region is treated as a continuation of a higher level split.

Basic concepts of multi-dimensional Query

A query on multi-dimensional data represents a subset of the data. To process the query, the database manager has to retrieve the data points contained in each region which intersects with the query. Two queries are identical if these intersect the same set of regions [4]. The database manager spends the same disk-access cost in processing identical queries. For studying the performance of various data distribution techniques, any query in an equivalence class has the same effectiveness (see MISCELLANEOUS). Informally, a query with respect to a grid specifies a collection of cells.

Definition: Query - A query Q denotes a set of cells [4].

Definition: Query Set - A Query Set is a set of queries [4].

A data-distribution method assigns a disk-id to each region. The disk-ids for a set of M disks can be chosen from the range of integers 0 to M-1. The region in two dimensions can be represented by a region number.

Definition : Allocation method - An allocation method **f** in two dimensions is a mapping from a region in two dimensions to a set of integer disk-id's, say 0 .. (M-1).

f : [(n)] $\rightarrow$ [Disk-id = 0..(M-1)], where n is a region number.

The allocation method should be designed to satisfy the objectives of a database manager in processing various queries. One of the important goals of the database is to maximize output, which imposes two restraints on the allocation methods, fairness and efficiency [7]. A fair allocation method spreads the regions that are distributed under a query as uniformly as possible among the various disks. An efficient allocation method imposes a light load on the database system by accessing as few disks as possible for small queries.

Definition : Periodic allocation methods - Periodic allocation methods map regions (n) and (n mod M) to a common disk, where n is a region number and M is the number of disks..

Definition : Equivalent allocation method - If there exists a one-to-one function **f** : {disk-id's} $\rightarrow$ {disk-id's}, such that D1(n) = f(D2(n)) for all regions n, two allocation methods D1(n) and D2(n) are equivalent, where n is a natural number.

Disk striping

Disk striping has been studied as a disk allocation method to improve I/O bandwidth. Disk striping is a general purpose facility for achieving parallel I/O [6]. A striping unit (the number of consecutive data units allocated to each disk) is used to spread data among the disks. For example, with a striping unit of 1 byte, each block is partitioned among the N disks with each disk storing byte k, byte k+1*N, byte k+2*N. Disk striping stores the data as a one-dimensional stream of bytes, and thus does not

support efficient I/O queries. In the domain of multi-dimensional data, disk striping may partition the data along one dimension. Range queries of the dimension may be processed efficiently, but many range queries of the other dimensions may display poor efficiency. The allocation methods of disk stripping at bit or byte level are paired, but not efficient. Furthermore, the choice of a data unit restricts on the maximum possible parallelism via striping.

Latin squares

Latin Squares are square arrays which contain M distinct elements with each element occurring M times, but, with no element occurring twice in the same column or

TABLE 3

SEVERAL LATIN SQUARES FOR M=4.

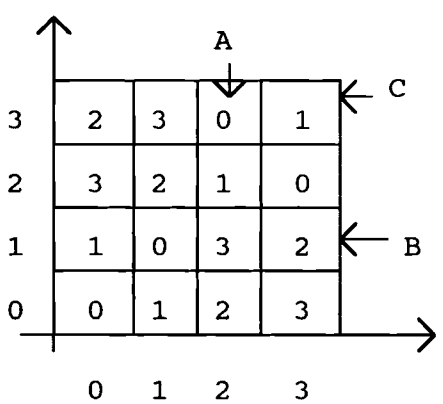
3 2 1 0 1 0 3 2 2 3 0 1 0 1 2 3	1 0 3 2 3 2 0 1 2 3 1 0 0 1 2 3	1 2 3 0 2 3 0 1 3 0 1 2 0 1 2 3
3 0 1 2 2 3 0 1 1 2 3 0 0 1 2 3	3 2 0 1 2 3 1 0 1 0 3 2 0 1 2 3	2 0 1 3 1 3 0 2 3 2 1 0 0 1 2 3

row [4]. TABLE 3 shows six distinct Latin Squares for M = 4, with items 0, 1, 2, and 3. The Latin Squares are perfect (see MISCELLANEOUS) for RowSet(M) and ColSet(M).

The cells of Latin Squares are denoted by ordered pairs, (i, j), and the lower left coner is considered to be the origin, as shown in TABLE 4. Each Latin Square can be considerd to be a function from the coordinate (i, j) to the set (0, 1, ..., M-1). For example, Latin Square (1,1) = 0, Latin Square (2,1) = 3 and Latin Square (1,3) = 3, in the Latin Square as shown in TABLE 4.

TABLE 4

THE LATIN SQUARE AS A MAPPING FUNCTION



Point	Cell ID	Disk ID
A	(2,3)	0
B	(3,1)	2
C	(3,3)	1

We can choose one of the Latin Squares for mapping the cells to the disks. The cells of the Latin Square are distributed uniformly over the disks. The data located in cell (i, j) in the grid of coordinate space can be mapped in disk-id = Latin Square (i mod M, j mod M). A Latin Square allocation method is not guaranteed to be perfect with respect to

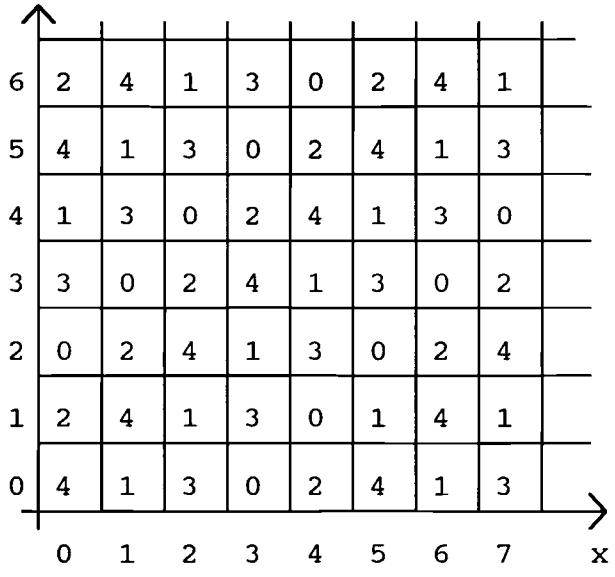
many query sets of interest [4]. Furthermore, the Latin Square allocation method can be used only when the number of disk is five or fewer [4].

Linear allocation method

The Linear allocation method is a round-robin allocation of cells in a given row or column to multiple disks and is characterized by two features in 2-dimensions: a disk-id permutation cycle and a shift [4]. The disk-id permutation cycle describes the round-robin allocation along the striping units in one dimension. The shift informs the difference between the starting positions of the permutation cycle between adjacent columns. The Linear allocation method for all M disks can be described as a function Disk-Id(x, y) = (px + qy + r) mod M, where p, q, and r are parameters which determine a set of lines [4]. Linear allocation methods can be composed which are perfect with respect to the Query set union {PD(M), PA(M), BLR(M,M), RowSet(M), and ColSet(M)}(see MISCELLANEOUS).

TABLE 5

LINEAR ALLOCATION METHOD WITH P=2, Q=3, R=4, AND M=5.



CHAPTER III

METHODOLOGIES

Chapter Overview

Figure 6 depicts the relation between a CPU and an I/O processor in a single-disk system. The CPU controls the I/O processor to read/write and after the I/O processor finishes the job, the CPU executes another job. The CPU is waiting during the I/O processing time which decreases CPU utilization and increases turn around time.

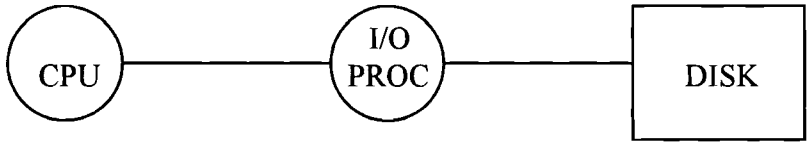


Figure 6 single-disk access system

Figure 7 describes a multi-disk access BANG file system which has M independent I/O processors and the same number of disks. Each I/O processor controls its own disk. In the multi-disk access system, the CPU can manage M parallel I/O processors.

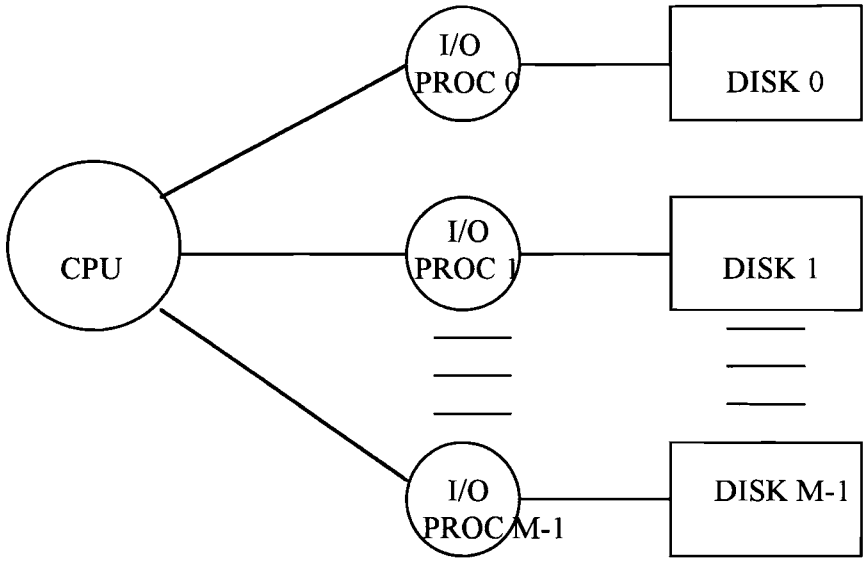


Figure 7 Multi-disk access system

The I/O processors are assigned jobs one after another by the CPU. When an I/O processor is executing a job, the CPU assigns jobs to the other I/O processors. If a job is assigned before the I/O processor has finished, the job is waiting until the I/O processor has finished the current job. When a query request several buckets which are in one disk, the access time is almost the same as the single disk access method. When requested buckets are distributed equally among the disks, the query access time is the best.

Research Methodologies

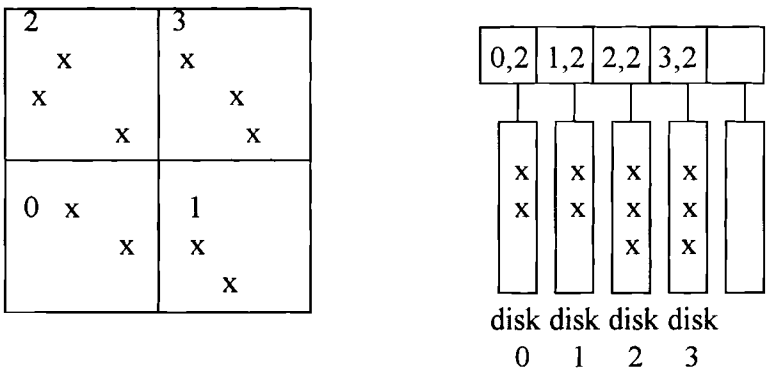
Region Number Allocation Method

Each region in the BANG file has a unique ordered pair $<r, l>$, where r is a region number and l is a level number. The region number is also used as a key in the BANG file directory. The Region Number Allocation Method is characterized by a periodic allocation of regions to multiple disks. The Region Number Allocation Method for all M disks can be described as a function Disk-id $(r) = (r + k)$ mod M, where r is a region number, k insures that each new bucket in a split is assigned to a different disk. k is set to 1 if the difference of two new region number is zero or equal to a multiple of the number of disks else k is set to 0, and M is the number of disks. An insertion of a record into a data bucket may cause it to overflow and the overflowing data bucket has to be partitioned as follows:

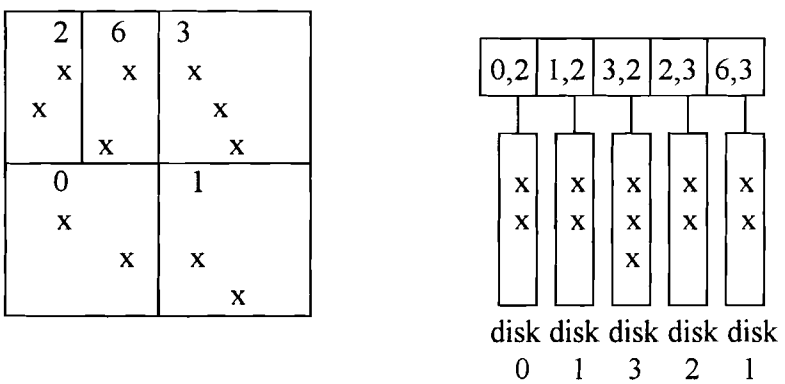
a) If a balanced condition is achieved at the first level of division, then the directory entry $<r, l>$ of the old region is replaced by the identifiers of the two resulting regions $<r, l+1>$ and $< r+2^l, l+1>$.

b) The partitioning of the data space is performed until the best balance condition is achieved. If n, $n > 1$, partitions are needed to achieve this balance, then the directory entry $<r, l>$ of the old region is modified into two new entries $<r, l>$ and $<r1, l+n>$ for nested partition, where $r1$ is the identifier of the newly formed region.

The difference of the two new region numbers would be 2^l (for $n = 1$) or $r1 - r$. If either difference is zero or equal to a multiple of the number of disks, the region $<r, l+1>$ or $<r,$



(a) Before splitting region 2



(b) After splitting region 2

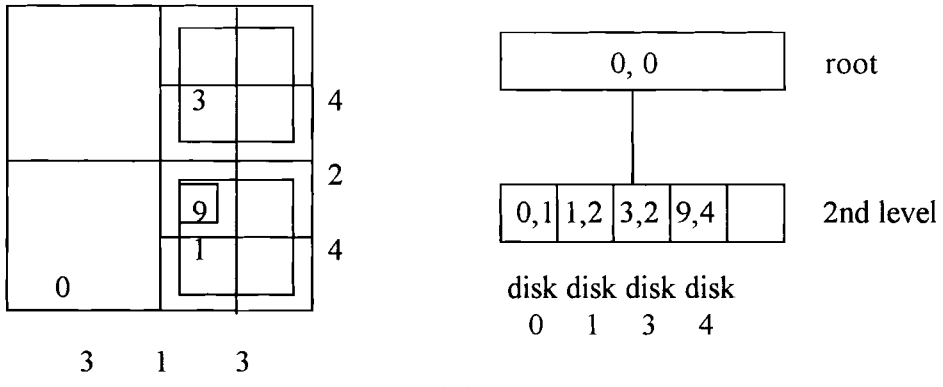
Figure 8. Region partitioning and disk allocation of RNAM according to region number with 5 disks (case a, p.23).

l> is assigned to the current disk and the region <$r+2^l$, $l+1$> or <$r1$, $l+n$> is assigned to the disk N, where N is $(r + k)$ modulo M, and set to k 1. Therefore, the Region Number Allocation Method assigns the two new buckets to different disks.

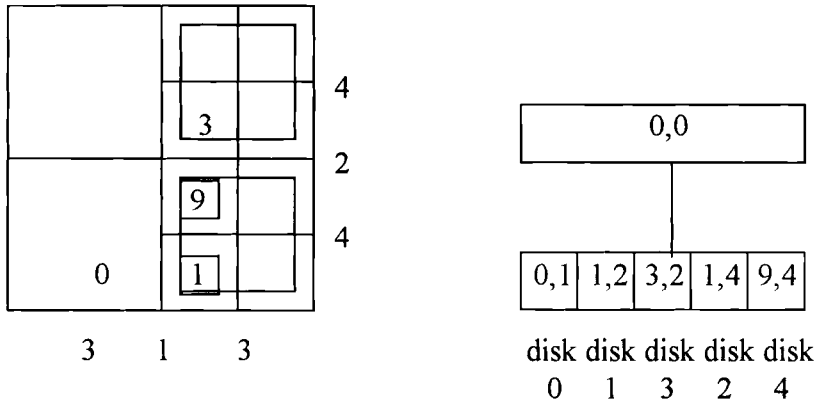
In figure 8, we show an example of how the Region Number Allocation Method assigns buckets to the disks. The splitting entry is <2,2> and the new entries are <2, 3> and <6,3>. The splitting bucket was assigned to disk 2 since the remainder of the

region number 2 is 2 for the five available disks (Disk-id (2) = (2+0) mod 5 = 2). The two new buckets are assigned to disk 2 and disk 1 since the region numbers are 2 and 6, respectively (Disk-id (2) = (2+0) mod 5 = 2, and Disk-id (6) = (6+0) mod 5 = 1).

We show another example of how the Region Number Allocation Method assigns buckets to the disks in figure 9. Assume that maximum capacity of the directory entry is 5, and the current state of the BANG file system is shown in Figure 9 (a). The splitting



(a)



(b)

Figure 9 Region partitioning and disk allocation of RNAM according to region number with 5 disks (case b, p.23).

entry is <1,2> and the new entries are <1,2> and <1, 4>. The bucket corresponding to the splitting entry was assigned to disk 1 for the five available disks. The new bucket corresponding to the higher level number is assigned to disk 1 since the region number is 1. To avoid the buckets being assigned to the same disk, we set the variable k to 1 and assign disk number 2 (Disk-id (1) = (1+1) mod 5 = 2). The new bucket corresponding to the higher level is assigned to disk 2.

Page Distribution Allocation Method

The Region Number Allocation Method is a good method for uniform data types but is not good for skewed data types. The Page Distribution Allocation Method is designed to parallelize the set of rectangle queries. Given M disks, the method is efficient for BSR(H,W), and BLR(H,W) (see MISCELLANEOUS). BANG file has a tree-structured directory which has the self-balancing property of a B-tree and has the information of each region on its leaf nodes. The leaf nodes of the BANG file are arranged in increasing order by region number and entries in the leaf nodes are also ordered by region number. The ordering of the BANG file is one-dimension and the entries corresponding to adjacent region numbers are also adjacent on the tree. The Page Distribution Allocation Method distributes the entries in a node to as many disks as possible. For that reason, we choose the number of entries of the directory node to be the same as the number of disks, therefore each entry in a node is assigned to a different disk.

When a bucket splits into two new buckets, the bucket corresponding to the smaller region number (if the two new region numbers are same, the lower level number is

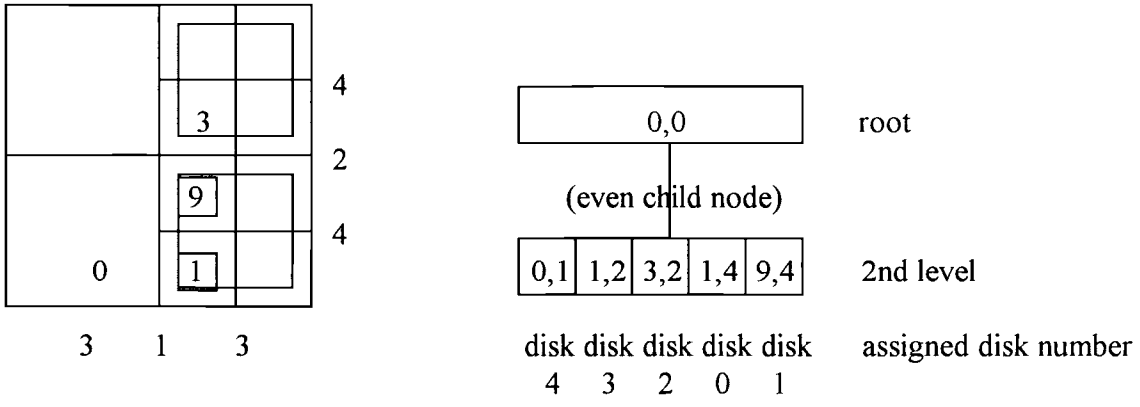
used) stays in the disk and the other new bucket is assigned to a disk which is not assigned to its node. If the node is an odd child of its parent, a disk from disk 0 to disk (M-1) is assigned to the bucket corresponding to the new larger region number (if the two region numbers are same, the higher level number is used). If the node is an even child of its parent, a disk from disk (M-1) to disk 0 is assigned for the balanced allocation of the disks.

Definition: Page Distribution Allocation Method

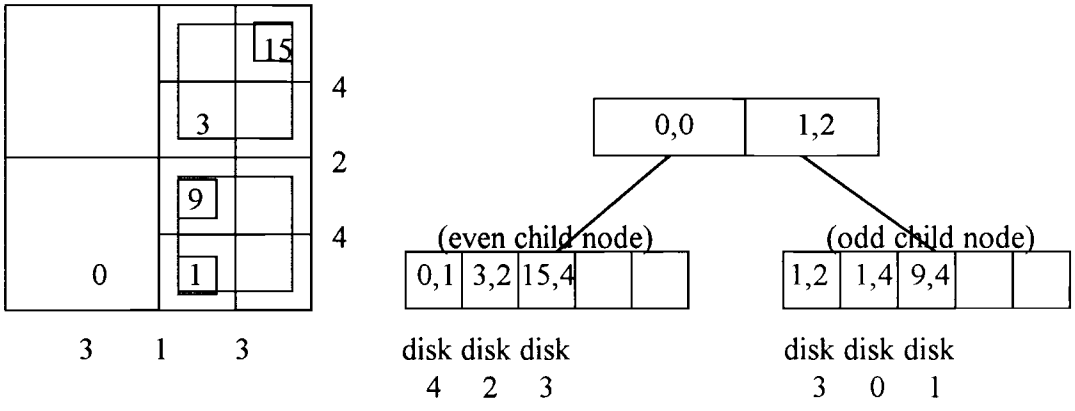
$$\text{Disk-Id}(n) = \begin{cases} \text{lowest unassigned disk number of the node} & : \text{if an odd child of its parent} \\ \text{highest unassigned disk number of the node} & : \text{if an even child of its parent} \end{cases}$$

Figure 10 (a) depicts five regions and their corresponding entries. The node is filled with entries until the node contains maximum capacity. The entries in the node are assigned to each different disk. When the entry <3,2> is split into <3,2> and <15,4>, the node is split into two nodes as described figure 10 (b) since the number of entries in the node exceeds the maximum capacity. In that case, the new bucket corresponding to the larger region number pair <15,4> is included in the even child node and assigned to disk 3 because it is the highest unassigned disk number in the node.

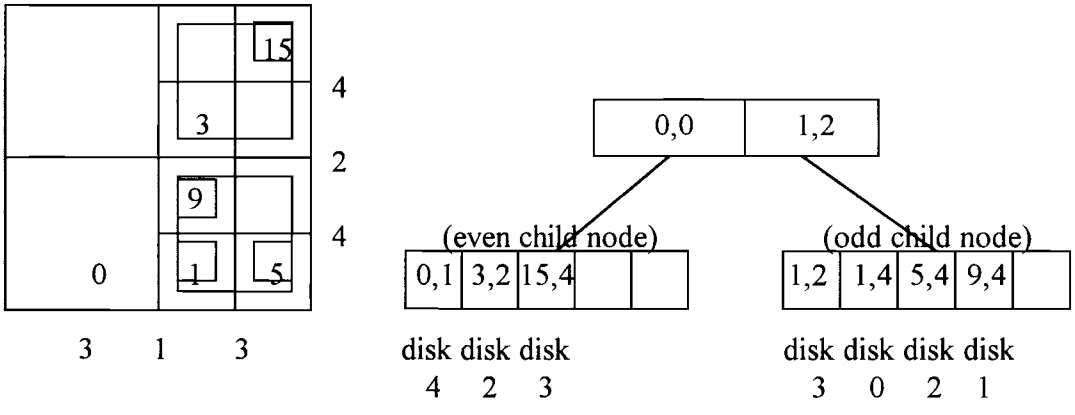
In figure 10 (c) we show an example of how the disks are assigned to the odd child of its parent. The entry <1,2> is split into <1,2> and <5,4>. A bucket corresponding to the larger region number pair <5,4> is assigned to disk 2 which is the lowest unassigned disk number to the node, since the node is an odd child of its parent and disk 0 and disk 1 were assigned already. The bucket corresponding to the smaller region number pair <1,2> is in the disk 3 as it was.



(a) before splitting



(b) after splitting an even entry of its parent which causes the node splitting



(c) after splitting an odd entry of its parent

Figure 10. Data partitioning and disk allocation of Page Distribution Allocation Method (disk number = 5)

Odd number nodes and even number nodes in the directory are assigned to the opposite direction to get balanced load of the disks and to increase disk parallelism for the set of queries. The Page Distribution Allocation Method (PDAM) is efficient for the queries accessing consecutive records by region number because of the ordering property of the BANG file directory. The order of the entries in a node is dependent on region number. If region number is lowest, the entry point of the region is leftmost in the node.

Analysis of Simulation

Data sets

Four sets of data distributions are considered in the experiment. These are called UU, UN, NN, and HS (see APPENDIX A). The UU data-set is a collection of points (x, y), where x and y are independent, uniformly distributed random variables. The NN data-set is a collection of points (x, y), where x and y are independent, normally distributed random variables. The UN data-set is a collection of points (x, y), with independent random variables x and y, where x is distributed uniformly and y is distributed normally. An HS data-set of K points is generated from an initial uniform distribution UU(K/4) over the unit square. We generate and insert (3K/4) other points from the NN distribution, with a small standard deviation.

Uniformly distributed data is generated with a maximum value of 1.00 and minimum value of 0.00, and normally distributed data is generated with means of 0.50 and standard deviation of 0.15. The (3K/4) points of the HS data set are generate with

standard deviation of 0.05 (see APPENDIX B). Data sets are 500, 1000, 2000, 4000, 8000, and 16000 data points. These data sets are stored in the BANG file via a sequence of insert operations with the BANG file access method. If a data bucket overflows, the splitting is performed until the best balanced condition is achieved.

Query sets

Six query sets are applied to represent path and range queries. The query sets are PD(N), PA(N), RowSet(N), ColSet(N), BSR(N), and BLR(N), where N is 9. The Figure 11 shows shapes of the query sets. To formalize queries for the BANG file, a grid transforms the Euclidean Space into a Cartesian Space with a range of 20. The axes are numbered from 0 to 19 and only the first quadrant is used. The maximum value of the data is 1.00 and the minimum value is 0.00 therefore the distance between points on the axes is 0.05. We can identify cells with a pair of cell identifier (x, y). N-dimensional space can be described with N-tuples (x_1, x_2,..., x_n), where x_i is an integer.

- Basic Small Rectangle BSR(N) - The query set small rectangles BSR(N) is a collection of orthogonal range queries representing N rectangles with height H=N/3 cells and width W=N/3 cells.
- Basic Large Rectangle BLR(N) - The query set Large Rectangles BLR(N) is a collection of orthogonal range queries representing rectangles with height H=2N/3 cells and width W=2N/3 cells.
- ColSet(N) - The ColSet(N) is a set of orthogonal range queries described by $\{x_i..x_i, j..(j+\lceil N/2 \rceil\}$, where i, j and N are integers.

- RowSet(N) - The RowSet(N) is a set of orthogonal range queries described by $\{ j..(j+\lceil N/2 \rceil), y_i..y_i \}$, where i, j and N are integers.
- Principle Diagonal PD(N) - The query set Principle Diagonal PD(N) is represented by a set of queries (PD_0, PD_1, ..., PD_{n-1}), where $PD_k = \{(x_i,y_j)|$ cell (x_i, y_j) AND $i+j = k$ AND $0 \leq j - i \leq 1$ AND $0 \leq i, j < \lceil N/2 \rceil \}$, for all $k = \{0, 1, ..., N-1\}$.
- Anti-diagonal PA(N) - The query set Anti-Diagonal PA(N) is represented by a set of queries $\{ PA_0, PA_1, .., P_{n-1}\}$where $PA_k = \{(x_i, y_j)|$ cell (x_i, y_j) AND $0 \leq i, j < \lceil N/2 \rceil$ AND $i \times j = 0\}$, for all $k = \{0, 1, ...,N-1\}$.

Number of disks: 5, 8, and 10

Assumptions

- all M disk units are identical
- the bucket access time is constant
- the CPU time is negligible because the CPU much faster than disk access time. Thus, only I/O time is counted when we simulate a number of disks. In this study, we assign the same number of disks and the same number of I/O processors to each allocation method and compare the performance of each allocation method.

Evaluation

The relative access time of each query depends on the allocation method. Access time is parallel I/O cost, where I/O cost is the number of data buckets accessed. The access time provided by the best method for each data set and query is treated as unit in computing the relative access time.

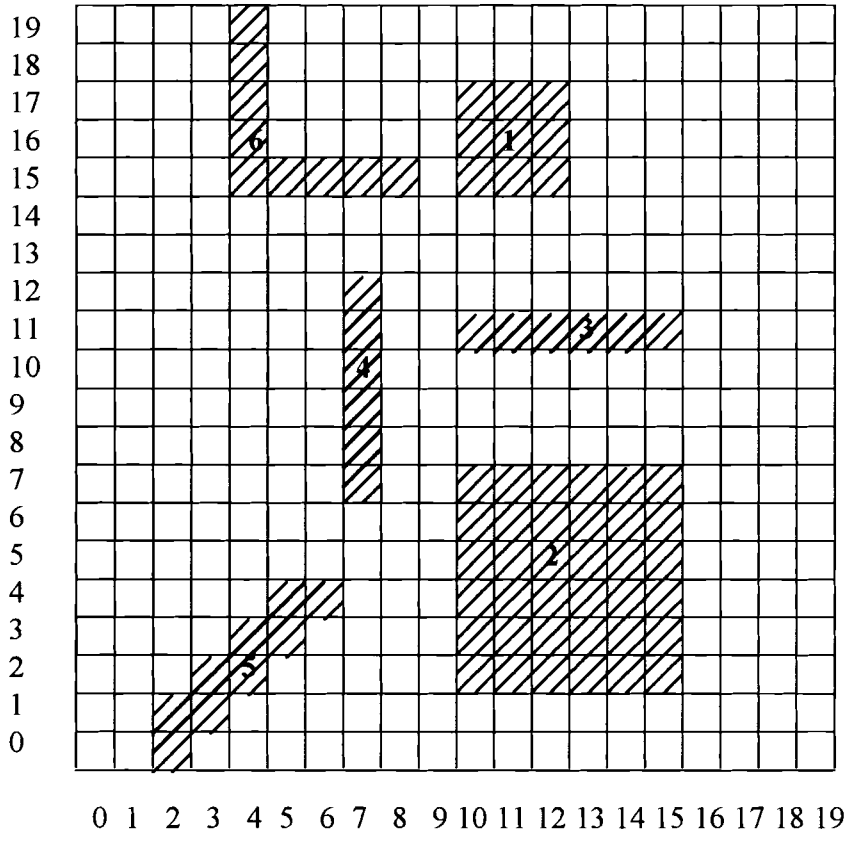


Figure 11. Shapes of the Queries

CHAPTER IV

FINDINGS

We evaluated the proposed PDAM and RNAM along with well-known allocation methods such as Sequential Allocation Method (SAM) and Random Allocation Method (RAM) for three sets of disks. The relative access time are evaluated for the effect of queries, data-set size, data distribution, and number of disks. We compared access time for each allocation method by considering the range query sets.

Effect of Queries

Figure 12 shows the performance of the alternative disk allocation function for several queries. The queries include small rectangles, large rectangles, rows, columns, principle diagonals and anti-diagonals of the coordinate space [0..1, 0..1] where the data is located. The data distribution is a uniform-normal with 10 disks. The observations show that the PDAM performs the best overall. The RNAM performs very well. The SAM and the RAM show similar performance.

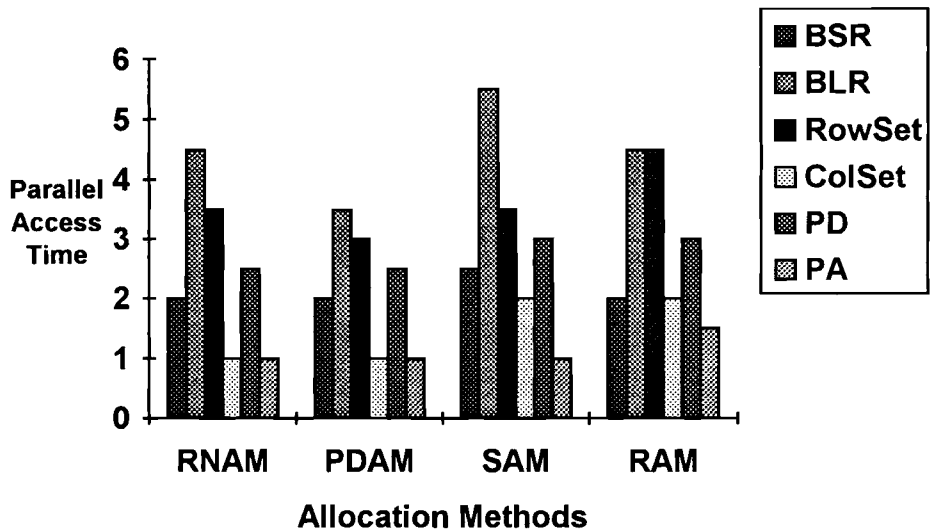


Figure 12. Relative access time for queries with uniform-normal distribution UN(8000) and 10 disks

Figure 13 shows the performance of the alternative disk allocation function for several queries. The queries include small rectangles, large rectangles, rows, columns, principle diagonals and anti-diagonals of the coordinate space [0..1, 0..1] where the data is located. The data distribution is a normal-normal with 8 disks. The observations show that the PDAM performs the best overall. The RAM performs poorly on PD which is unexpected.

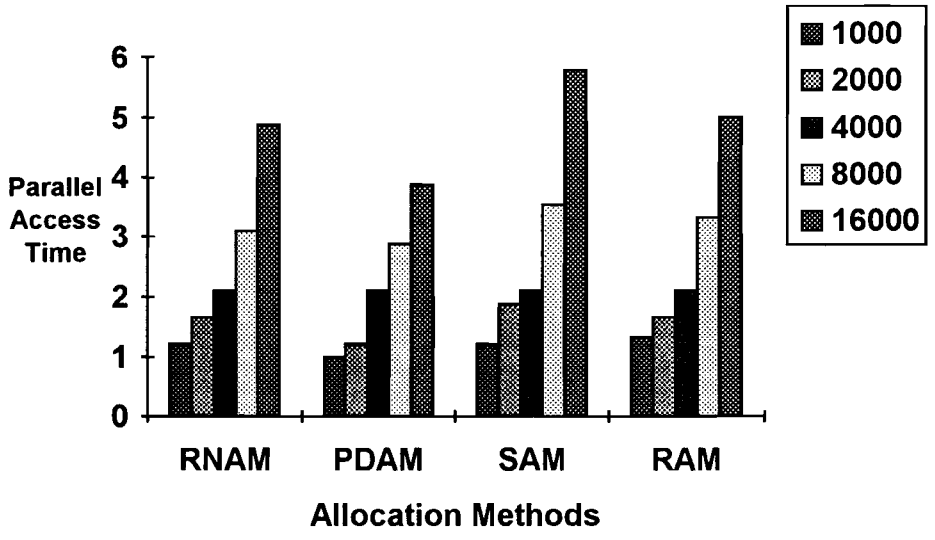


Figure 14. Relative access time for different sizes of data-sets uniform-uniform distribution with 10 disks

Figure 15 shows the performance of the alternative disk allocation function for data sets of five different sizes: 500 points, 1000 points, 2000 points, 4000 points, and 8000 points; the number of disks is 5. Each data-set is a collection of points (x, y), where x and y are independent random variables. The observations show that the PDAM performs well on all sizes of data sets. The performances of the PDAM, RNAM, and RAM improve reasonably on the set of all queries, as the size of the data sets increases. Even though the SAM performs well for 4000 points, the performance is hard to accept because the response time of SAM for 4000 points is less than that for 2000 points. The PDAM access time is the fastest among the four allocation methods on the set of all queries, as the size of the data-sets increase.

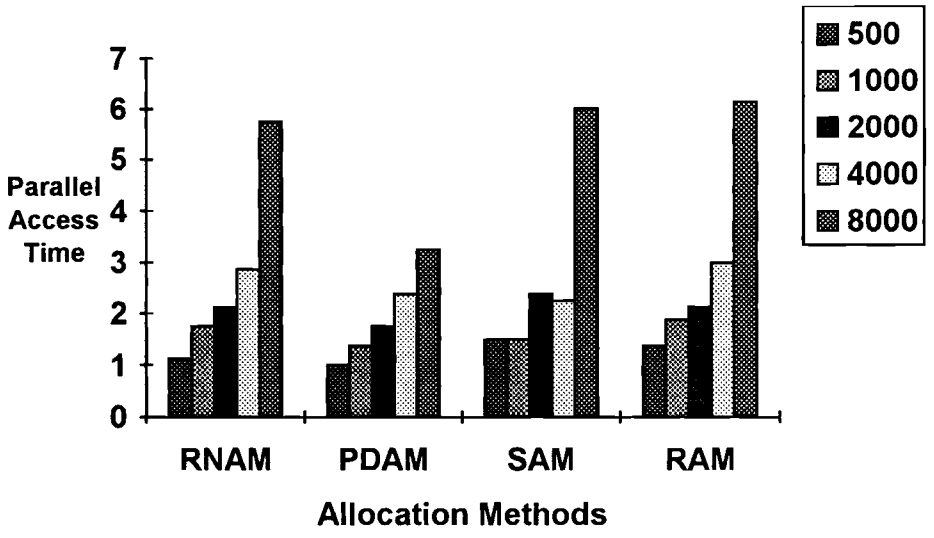


Figure 15. Relative access time for different sizes of data-sets uniform-normal distribution with 5 disks

Effect of Data Distribution

Figure 16 shows the performance of the alternative disk allocation methods for four kinds of data: UU, UN, NN, HS; the number of disks is 5. The UU data-set is a collection of points (x, y), where x and y are independent, uniformly distributed random variables. The NN data-set is a collection of points (x, y), where x and y are independent, normally distributed random variables. The UN data-set is a collection of points (x, y), with independent random variables x and y, where x is distributed uniformly and y is distributed normally. The HS data-set of K points is generated from an initial uniform distribution UU(K/4) over the unit square. We generate and insert (3K/4) other points from the NN distribution, with a small standard deviation. The observations in Figure 16 show that the PDAM performs very well on the set of all data distributions.

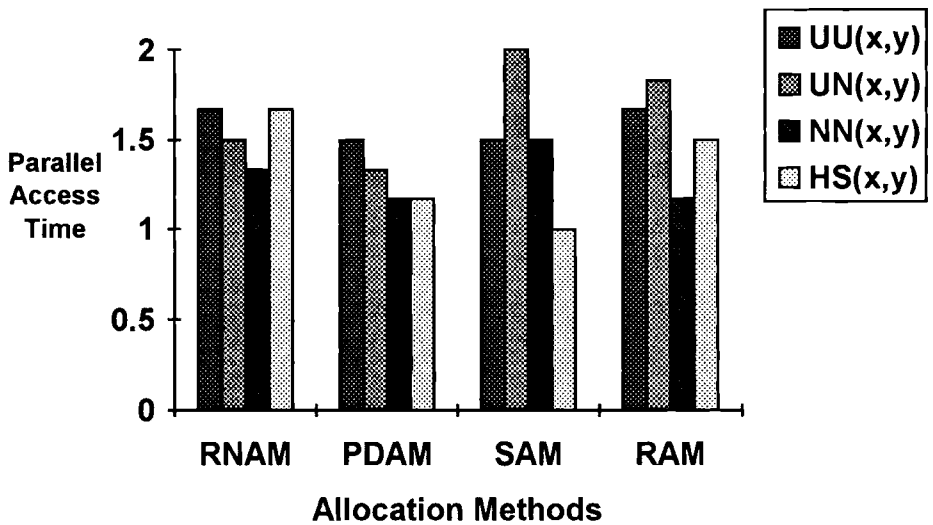


Figure 16. Relative access time for different data distributions each data-set has 500 points with 5 disks

The SAM performs reasonably well on UU and HS. The performance of the PDAM has the smallest deviation for each data distribution and the access time of the PDAM is the fastest among the allocation methods. The access time for the SAM show fluctuation with a 100% difference between UN and HS.

The observations in Figure 17 show that the PDAM performs very well on the set of all data distributions with 8 disks. The RAM performs reasonably well on UN and HS. The SAM performs poorly for the data distributions. The access time of the PDAM are maximally 90 % of difference for each data distribution, and the access time of the PDAM is 5.8. The PDAM performs the best among the allocation methods, which also shown by Figure 16. The access time of the SAM are 160% of difference between UU and HS and the performance of SAM is 8.2. The performance of RAM is better than that of RNAM, which is unexpected.

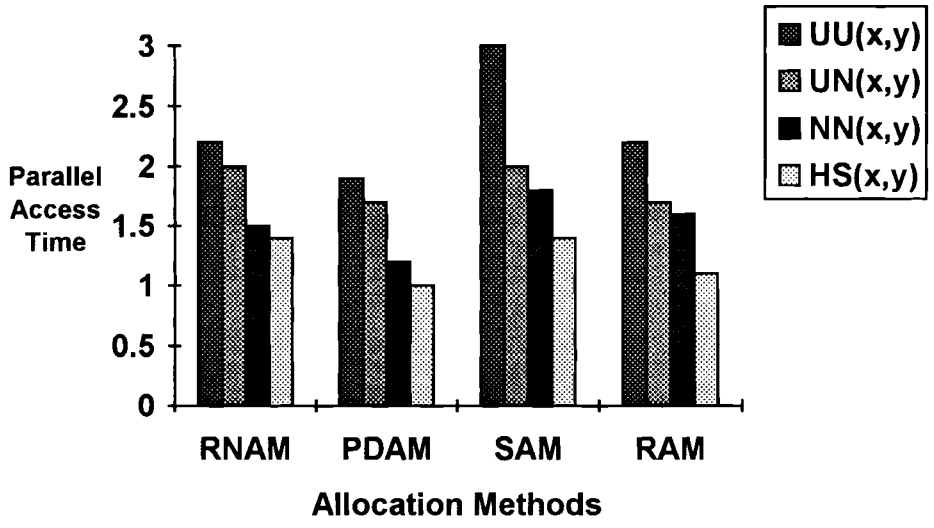


Figure 17. Relative access time for different data distributions
each data-set has 4000 points with 8 disks

Effect of Number of Disks

This section is included to show how varying the number of disk drives affects each allocation method's performance. Figure 18 shows the performance of the alternative disk allocation methods for different number of disks: 5, 8, 10. The distribution is normal-normal. The RNAM performs very well on 5 disks, and the PDAM performs very well on 8 disks. The RAM performs well on 10 disks, better than RNAM and PDAM. However, the performance of the allocation methods improves as the number of disks increases. For example, the PDAM improves 34 % when the number of disks increases from 5 to 10. The increase in performance is direct proportion to the increase in the number of disks.

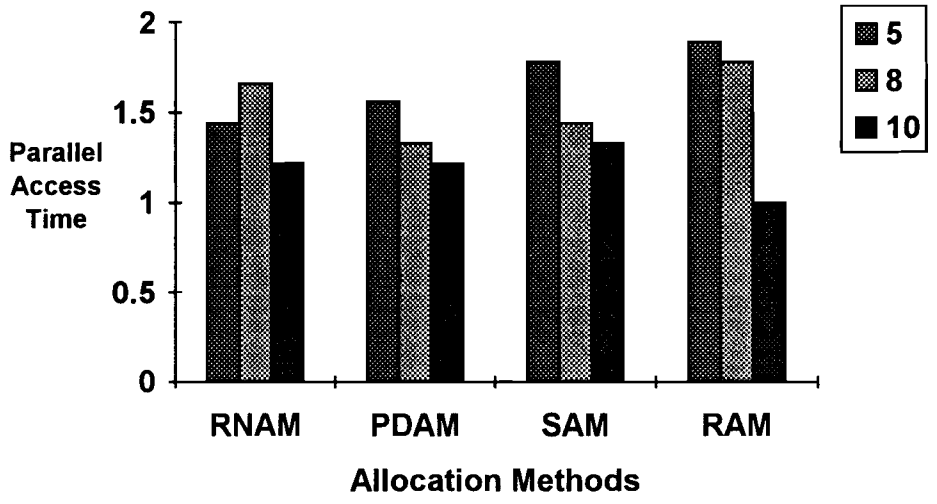


Figure 18. Relative access time for different number of disks normal-normal distribution(2000)

Figure 19 shows the performance of the alternative disk allocation methods for different number of disks: 5, 8, 10. The distribution is uniform-normal. The PDAM performs very well on 5 disks and 10 disks. The RAM performs almost evenly on the set of all disks, which is unexpected. The performance of the PDAM improves very well on the set of all queries, as the size of data-sets increases.

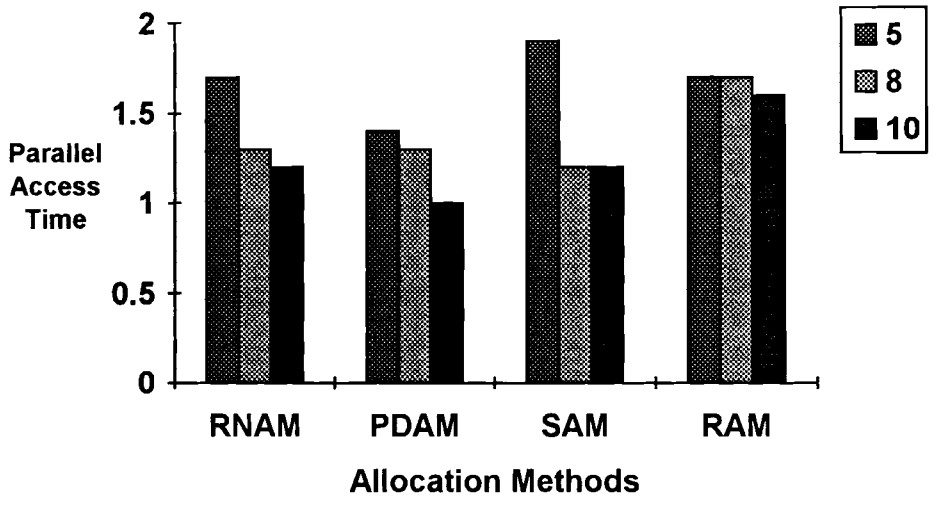


Figure 19. Relative access time for different number of disks uniform-normal distribution(2000)

We have evaluated the effect of queries, data-set size, data distribution, and number of disks. Access time of the SAM and the RAM are faster under some circumstances than that of PDAM but the performance is not consistent. The access time of the PDAM are fast and consistent. The performance of the RNAM is better than the SAM and the RAM but not better than PDAM. The access time of the PDAM is the fastest among the allocation methods for all the queries, data-set sizes, the data distributions, and the number of disks. This is not surprising since the PDAM directory leaf node entries are assigned to different disks. The queries retrieve a data region which is ordered by region number. Therefore, the PDAM queries a large number of disks in a time unit.

CHAPTER V

ADVANTAGES AND DISADVANTAGES

Executing a query involves two parts: separating it into one or more service demands and dispatching to the appropriate disks by the I/O processor (overload activity), and executing the accesses by the disks (transfer activity) [10]. A multi-disk system trades an improvement in the properties of the execution pattern of requests with an increase in the request overload activity. Multi-disk systems preempt an improvement in the execution pattern since it leads to concurrent execution of a request, it reduces the coefficient of variation of service demands, and it disperses accesses evenly over all disks.

The first advantage of multi-disk system is that in a lightly loaded system the response time of a query depends on its execution time since requests rarely compete for the service when utilization is low. Parallelism reduces execution time and therefore multi-disk systems will reduce the expected response time of a lightly loaded system. The second advantage of a multi-disk system comes into play when system utilization is medium. In a multi-disk system each range query is decomposed into several service demands that are executed on multiple disks. This means that each disk has a smaller variation in access demands. For a system with a medium level of utilization the reduction in service demand variation due to a multi-disk system will entail a significant reduction in expected response time in spite of the increased overhead. In a highly utilized system, the single-disk system can exceed memory capacity more easily than multi-disk system.

The main disadvantage is that the amount of information to be maintained for the multi-disk system is lager because a file is represented as several subfiles that reside on independent disks. Also, there is the added complexity imposed by potentially frequent moves of data across disks as subfiles grow. Since our interest is in improving I/O performance we ignore this primarily CPU disadvantage in this thesis. The second disadvantage is an effect of a single disk failure. If redundancy is applied to deal with disk failures then it is easy to guarantee that the two copies of data will not reside on the same disk.

CHPATER VI

CONCLUSIONS

In this paper, we have introduced two multi-disk allocation methods, the PDAM and the RNAM, for the BANG file. Implementations of the allocation methods are straightforward and the functions used in the algorithms are very simple to compute. Given a multi-disk system with M independently accessible disks, it is desirable to allocate buckets among the M-disk equally for range queries. The maximum concurrency of disk access is achieved in examining the required buckets to minimize the response time. Our goal is to maximize the parallelism for query sets.

Experimental evaluation shows that the access time provided by an allocation method depends on the query sets as well as on the data distribution. Table 6 shows the average performance of the alternative disk allocation functions for several queries. The searching areas for processing queries include small rectangles, large rectangles, rows, columns, principle diagonals and anti-diagonals of the coordinate space [0..1, 0..1]. Sequential Allocation Method (SAM) and Random Allocation Method (RAM) gave inferior overall performance as shown in Table 6. RNAM performs well on ColSets, RowSets, and PDs. When a bucket overflows and must be split, RNAM assigns the buckets corresponding to the two new regions to difference disks. This gives improved performance on ColSets, RowSets, and PDs.

The BANG file entries corresponding to adjacent region numbers are also adjacent on the tree. Most of the adjacent regions are assigned to different disks since the PDAM

TABLE 6

AVERAGE RELATIVE ACCESS TIME FOR QUERIES WITH NORMAL-NORMAL DISTRIBUTION NN(8000) AND 8 DISKS, AND UNIFORM-NORMAL DISTRIBUTION UN(8000) AND 10 DISKS

Query Set ==>	Small	Large	Row	Col	PD	PA
RNAM	1.67	2.83	1.83	1.33	2.17	1.33
PDAM	1.33	2.50	1.33	1	2.17	1.33
SAM	1.83	2.83	2.17	1.67	2.33	1.33
RAM	1.67	2.83	2.17	1.33	3.33	1.50

assigns each entry in a node to a different disk. Range queries access data in adjacent regions which are distributed to different disks by the PDAM; Therefore the PDAM achieved maximal parallelism and had the best average performance on the set of queries used in this simulation.

Table 7 shows how varying the number of disk drives affects each allocation method's performance. SAM gaves inferior performance as shown in Table 7. RNAM performs well on UU and UN distributions. The UU distribution divides the regions almost equally in the manner of the grid files. Then the region numbers are apt to achieve parallelism. Since the RNAM assign regions to the disks by means of their region

TABLE 7

AVERAGE RELATIVE ACCESS TIME FOR DIFFERENT DATA DISTRIBUTIONS; DATA-SETS HAVE 500 POINTS WITH 5 DISKS AND 4000 POINTS WITH 8 DISKS

Data Distribution ==>	UU	UN	NN	HS
RNAM	1.78	1.61	1.30	1.26
PDAM	1.57	1.40	1.09	1
SAM	2.07	1.84	1.52	1.10
RAM	1.78	1.63	1.28	1.20

numbers, the UU and UN distributions are approximately balanced (see MISCELLANEOUS) among the disks.

PDAM performs best on the set of all data distributions since adjacent entries in a node are assigned to different disks by the PDAM. Buckets corresponding to the adjacent entries are perfectly (see MISCELLANEOUS) distributed to the disks for the queries no matter what data distributions are used.

When designing multi-disk systems with parallel I/O processors, it is important to ensure that similar buckets are assigned to different disks to maximize concurrency in

retrieval and similar records should be clustered into the same or similar buckets while similar buckets should be distributed among disks. Therefore we attempt to assign similar buckets to different disks. Our future research could focus on clustering similar records into the same or similar buckets.

BIBLIOGRAPHY

1. Guttman, A. "R-Trees: A dynamic Index Structure for Spatial Searching.", Proc. ACM SIGMOD 1984 Annual Conference SIGMOD Record 14, (1984), 47-57

2. Nievergelt, J., and Hinterburger, H., "The Grid File: An Adaptable, Symmetric Multikey File Structure.", ACM Trans. on Database Systems 9, 1 (1984), 38-71.

3. Freeston, M. "The BANG file: a new kind of grid file." Proc. ACM SIGMOD 1987 Annual Conference SIGMOD Record 16, 3(Dec. 1987), 260-269.

4. Zhou, Y., Shekhar, S., and Coyle, M. "Disk Allocation Methods for Parallelizing Grid Files.", Proc. 10th Int. Conf. on Data Engineering, (1994), 243-252

5. Lian, T. "Implementation and Evaluation of Balanced and Nested Grid(BANG) File Structures." Stillwater, OK: University of Oklahoma; (1988), 40-94.

6. Salem, K., Garcia-Molnia, H. "Disk Striping.", Proc. 2nd Int. Conf. on Data Engineering, (1986), 336-342.

7. Kamel, I., Faloutsos, C. "Parallel R-Trees.", Proc. ACM SIGMOD 1992 Annual Conference SIGMOD Record 21, 2(1992) 195-202.

8. Du, H. C. "Disk Allocation for Product Files on Multiple Disk Systems.", ACM Trans. on Database Systems 7, (March 1982).

9. Wu, C. T., Burkmard, W. A. "Associative Searching in Multiple Storage Units." ACM Trans. on Database Systems 12, (January 1987) 38-64.

10. Livny, M., Khoshafian, S., and Boral, H. "Multidisk management algorithms", Proc. ACM SIGMETRICS, (1987) 69-77.

11. Kumar, A. "G-Tree: A New Data Structure for Organizing Multidimensional Data", IEEE Trans. on Knowledge and Data Engineering 6, 2(April 1994) 341 - 347.

12. Comer, D. "The ubiquitous B-tree." ACM Comput. Surveys 11, 2 (June 1979), 121-137.

13. Li, J., Rotem, D., and Srivastava, J. "Algorithms for Loading Parallel Drid Files", Proc. ACM SIGMOD 1993 Annual Conference SIGMOD Record 22, (May 1993) 347-356.

14. Jagadish, H. V. "Linear Clustering of Objects with Multiple Attributes", ACM SIGMOD 1990 Annual Conference SIGMOD Record 19 (1990) 332-342.

15. Aho, A. V., Ullman, J. D. "The Theory of Parsing, Translation, and Compiling, Volumn I: Parsing", Englewood Cliffs, NJ: PRENTICE-HALL, INC; 1972.

APPENDIXES

APPENDIX A

DATA DISTRIBUTION GRAPHES

DATA DISTRIBUTION GRAPH

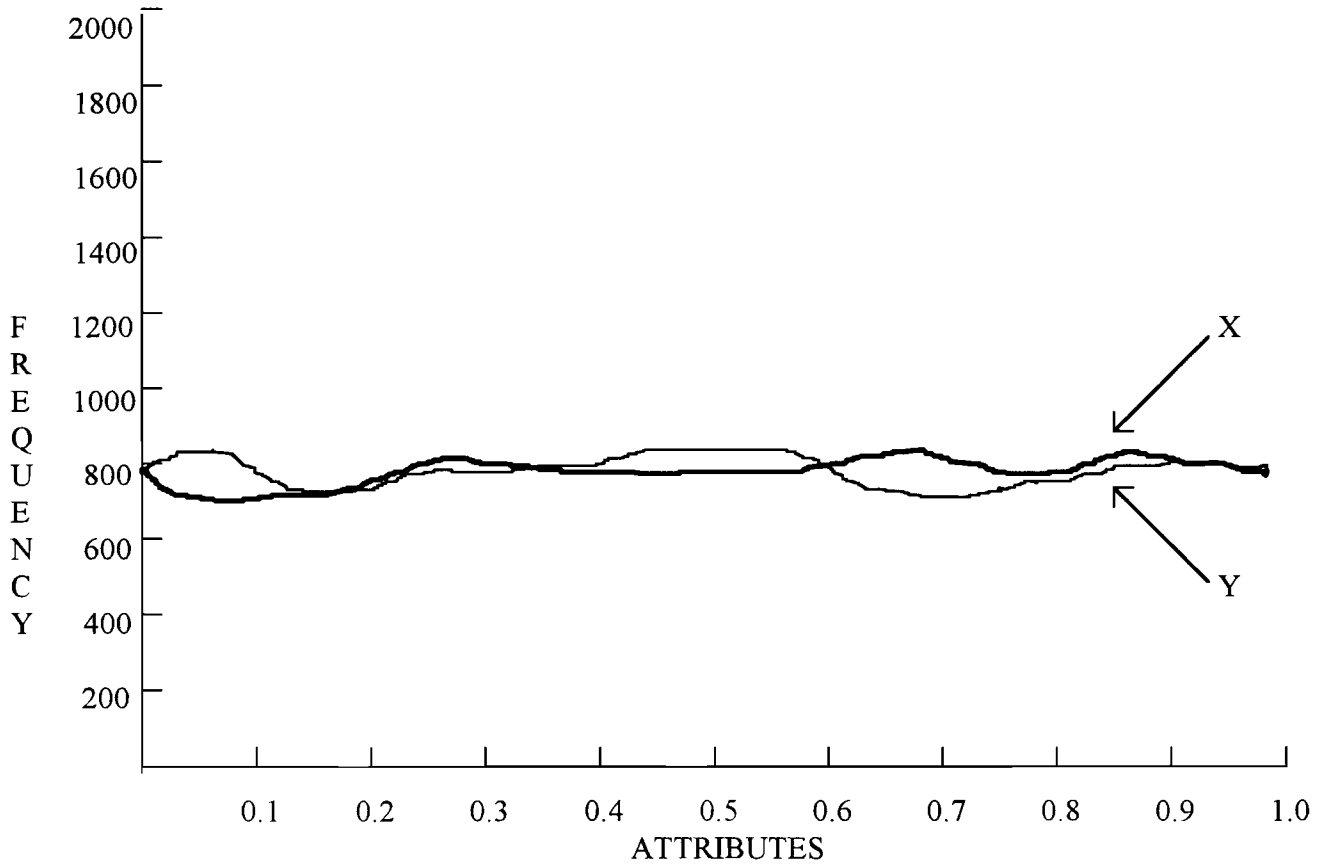


Figure 20. Data Distribution of UU 8000 (X = Uniform, Y = Uniform)
(Similar for all UU Data-sets)

DATA DISTRIBUTION GRAPH

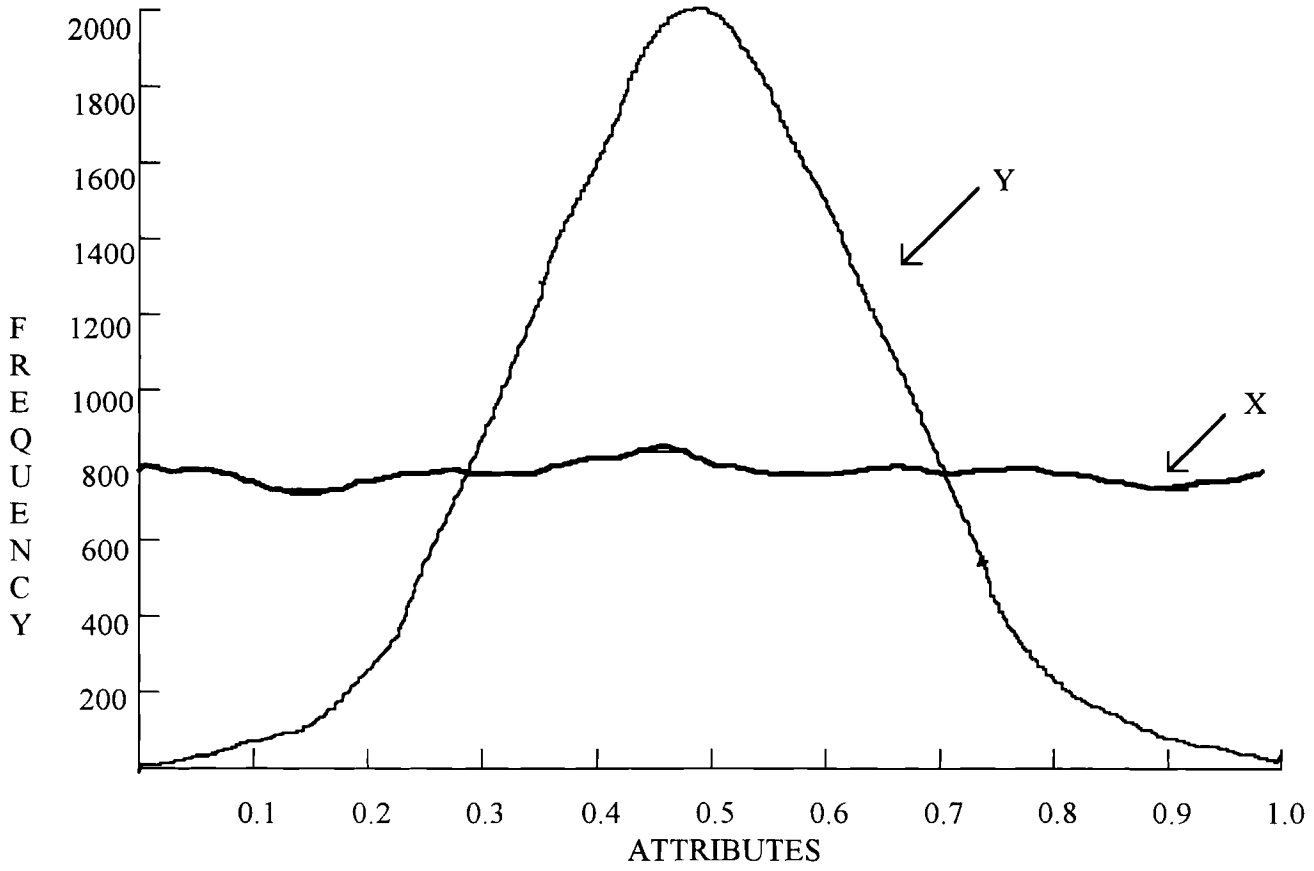


Figure 21. Data Distribution of UN 8000 (X = Uniform, Y = Normal)
(Similar for all UN Data-sets)

DATA DISTRIBUTION GRAPH

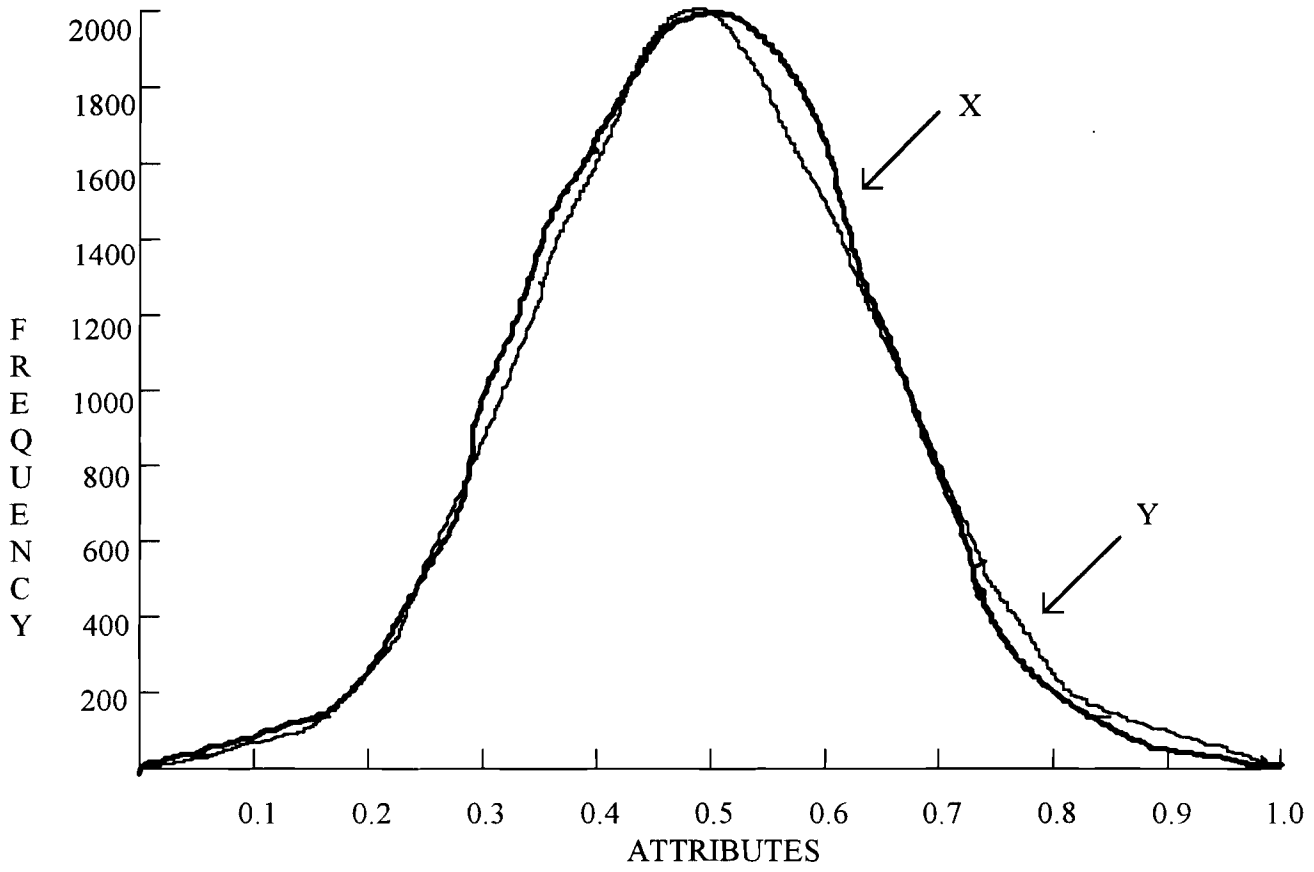


Figure 22. Data Distribution of NN 8000 (X = Normal, Y = Normal)
(Similar for all NN Data-sets)

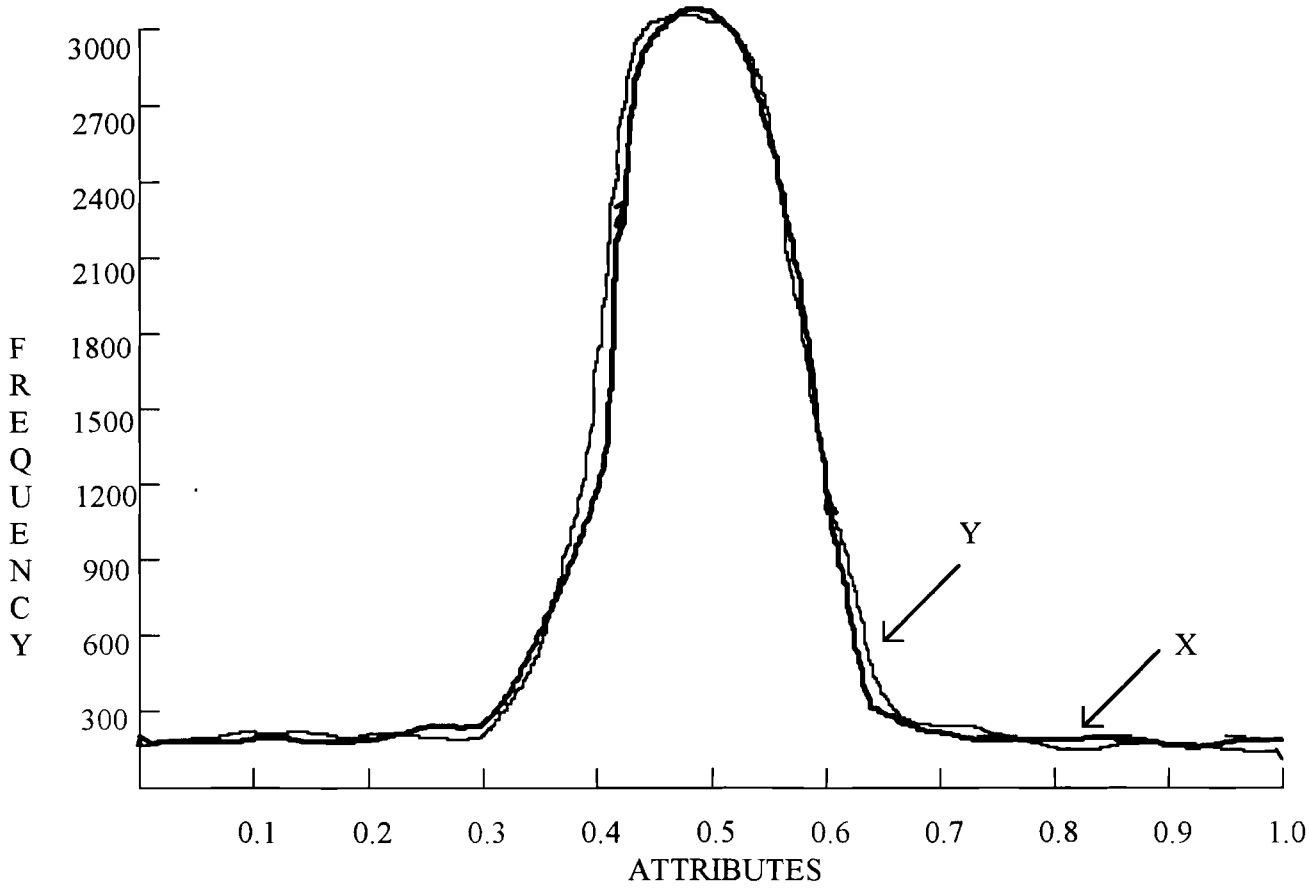


Figure 23. Data Distribution of HS 8000 (X = HS, Y = HS)
(Similar for all HS Data-sets)

APPENDIX B

DATA-SET GENERATION PROGRAMS

This appendix shows four sets of data distributions with 8000 points. The data distributions are similar for all data points. Each data-set is generated by the SAS Statistical program.

- Uniform-Uniform data distribution

```
data uu8000;  /* data name */
do i = 1 to 8000 by 1;
   x = (ranuni(680036)*1.0);  /*  Uniformly distributed random data with standard
                        deviation of 1.0 */
   y = (ranuni(670001)*1.0);
   output uu8000;  /* write data */
end;
stop;
proc print data = uu8000;   /* print data */
run;
```

- Uniform-Normal data distribution

```
data un8000;
do i = 1 to 8000 by 1;
   x = (ranuni(665036)*1.0); /*  Uniformly distributed random data with standard
              deviation of 1.0 */
    y = (rannor(770000)*0.15) + 0.5;  /* Normally distributed random data with standard
              deviation of 0.15 */
    output un8000;  /* write data */
end;
stop;
proc print data = un8000;  /* print data */
run;
```

- Normal-Normal data distribution

```
data nn8000;
do i = 1 to 8000 by 1;
   x = (rannor(775036)*0.15) + 0.5; /* Normally distributed random data with standard
                                       deviation of 0.15 */
   y = (rannor(770000)*0.15) + 0.5;
   output nn8000;
end;
stop;
proc print data = nn8000;
```

```
run;
```

- Hot-Spot data distribution

```
data hs8000;
do i = 1 to 2000 by 1;
    x = (ranuni(770000)*1.0); /*  Uniformly distributed random data with standard
                        deviation of 1.0 (K/4)*/
    y = (ranuni(660000)*1.0);
    output hs8000;
end;

do i = 1 to 6000 by 1;
    x = (rannor(770000)*0.05) + 0.5; /* Normally distributed random data with standard
                                        deviation of 0.05 (3K/4) */
    y = (rannor(898000)*0.05) + 0.5;
    output hs8000;
end;
stop;
proc print data = hs8000;
run;
```

VITA

Jae-myeong Jeon

Candidate for the Degree of

Master of Science

Thesis: MULTI-DISK ALLOCATION METHODS FOR BANG FILES

Major field: Computer Science

Biographical:

Personal Data: Born in Chul-won, Kangwondo, R.O.K., On January 16, 1966, the son of Byoung-sop and okja.

Education: Graduated from Chul-won High School, Chul-won, Kangwondo in February 1984; received Bachelor of Science degree in Computer Science from Air Force Academy, Chung-won, R.O.K. in February 1988. Completed the requirements for the Master of Science degree with a major in Computer Science at Oklahoma State University in December 1995.

Experience: Raised on a farm in Chul-won City, Kangwondo, R.O.K.; worked for R.O.K. Air Force as a programmer and as the head of Department of Computer at an Air Base; Oklahoma State University, Department of Computer Science, 1993 to present.