File Organizations and Access Methods for CLV Optical Disks.

CONFERENCE PAPER · JANUARY 1989

Source: DBLP

FILE ORGANIZATIONS AND ACCESS METHODS FOR CLV OPTICAL DISKS

Stavros Christodoulakis
Daniel Alexander Ford

Research Report CS-88-21

April, 1988

File organizations and Access Methods for CLV Optical Disks

Stavros Christodoulakis
Daniel Alexander Ford

Department of Computer Science
University of Waterloo,
Waterloo, Ontario, N2L 3G1

ABSTRACT

A very large and important class of optical disk technology are CLV type disks such as CD ROM and WORM.

In this paper, we examine the different issues surrounding the implementation and use of several different file organizations on CLV type optical disks such as CD ROM and WORM. The organizations examined are hashing, B-trees and ISAM.

The CLV recording scheme is shown to be a good environment for efficiently implementing hashing. Single seek access and storage utilization levels approaching 100% can be achieved.

It is shown that a B-tree organization is not a good choice for WORM disks, but that a modified ISAM approach can be.

Expressions for the expected retrieval performance of both hashing and B-trees are also given.

1. Introduction

Continuing progress in the development of optical disk technology is resulting in ever increasing capacities and lower costs for direct access secondary storage. This new availability of large inexpensive storage is fueling the development of more ambitious and demanding applications such as multimedia data bases [3] which until recently, were not technically nor financially feasible.

Along with new applications, the advance of optical disk technology is spawning a need to understand the issues involved in providing efficient file organizations and access methods for the new storage medium. Optical disks are similar in nature to conventional magnetic disks, but some types have two significant differences that affect the efficiency and feasibility of implementing the conventional file organizations employed on magnetic disks such as hashing and tree structures like B-trees and ISAM. Firstly, the recording scheme used on these disks to register data on the disk surface is different than that used on magnetic disks. Many optical disks use a CLV (Constant Linear Velocity) recording scheme

rather than the CAV (Constant Angular Velocity) scheme used on virtually all magnetic disks. Secondly, at the current level of technology, most generally available optical disks are not erasable. Two common examples of optical disks that combine both of these characteristics are CD ROM and CLV WORM (Write Once Read Many times) disks.

CLV optical disks have the advantage over their CAV cousins of maximizing, within the limits of the recording technology, the utilization of a disk's recording surface. For the same size disk platter, more data can be stored on a CLV disk than a CAV disk. This is achieved by using a uniform recording density throughout the disk and then relying upon the drive's ability to vary the speed at which the disk rotates to ensure that all recordings, regardless of their position on the disk, pass beneath the sense mechanism at the same rate. The data recordings on a CLV disk are usually laid down in a single spiral pattern.

The disadvantage of this approach is that it makes movements of the access mechanism or seeks on CLV disks slightly slower than on CAV disks. Reading the data on the disk requires the speed at which the disk rotates to accurately match the position of the sense mechanism, but determining the position with the required precision is difficult without first being able to read the identification information stored in each sector. This "chicken and egg" problem can be solved by reading the identification information as the access mechanism is moved incrementally across the surface of the disk, adjusting the rotation speed to match. Other delays come from problems in determining the exact position of a track as adjustments might be required for proper alignment. The exact method employed by CLV optical disk drive manufacturers is proprietary information and is generally not available.

For most optical disks, the seek time is generally much larger than the disk's rotational latency. Though some WORM disks are faster, delays of up to one second or more are possible with the current CD ROM technology. As a result, the seek time is usually the most important factor in the time required to answer a query. The issue is complicated slightly however, by some optical disk drives which are capable of reading from more than one track without performing a seek. The viewing mechanism (in the access mechanism) of such a drive is equipped with an adjustable mirror that allows slight deflections of the beam of laser light used to read the data. This enables it to be aimed at any one of a small set of the

spirally concentric tracks immediately beneath the viewing mechanism. This set of tracks is called a *span* and the number of tracks in the set is called the *span size*. Typical values of the span size are 10, 20 and 40 tracks. This span access capability can be likened to cylinders on magnetic disk packs except that for optical disks the sets of tracks or sectors in a span can be overlapping. It is provided by manufacturers of both CAV and CLV disk drives (although currently, it is more frequently encountered with CAV drives).

Neither CD ROM nor WORM disks allow the information stored on them to be altered. Information on CD ROM disks is registered by a physical pressing process similar to that used to create audio LP's. A master disk is created using a photochemical etching process from which other pressing masters can be made. The pressing masters are then used to imprint the CD ROM disk with the desired data recording patterns. The recording surface is given a thin coating of aluminum to make it reflective and is then covered with a protective plastic layer.

On a WORM disk, data is recorded on the disk surface by way of permanent reflective changes made by a laser beam. WORM disks are manufactured and received by the user in a blank state; information is then permanently registered on them in an incremental sector by sector manner at the user site. To improve error rates, error detection and correction information is incorporated within a sector when it is written. It is not possible for the user to erase, reuse or even add to data that has been previously stored on a WORM disk. Any attempt to write over a previously written sector (if the disk drive will allow such an operation), will cause the error detection and correction information to become inconsistent with the sector contents. This will cause an error to be reported during subsequent retrievals of the rewritten sector.

An important difference between WORM and magnetic disks that affects the use of pointer linked structures is the inability to detect bad sectors on WORM disk until an attempt to write the sector is made. On a magnetic disk, unusable (bad) sectors can be detected before the disk is used and corrective (preventive) measures taken. This implies that all pointers on WORM disk must point to previously written sectors; to ensure that recorded sector addresses are valid.

In this paper we examine the efficiency, feasibility and possible implementation strategies of providing some of the conventional file organizations and access methods typically available for CAV magnetic disks. We also give approximate expressions for the expected retrieval performance of these organizations when implemented on CLV optical disk. The next section briefly presents a model of CLV type optical disks. Section 3 examines the issues raised when providing a hashing file access method. The performance and space utilization implications of using pointer linked tree file organizations such as B-trees and ISAM are discussed in section 4; particular attention is paid to the difficulties of employing such structures on WORM type optical disks.

2. A Model and a Schedule

In this section we present an abstract model of optical disks upon which our retrieval performance analysis was based and give a physical description of CLV disks. We also present an optimal schedule for answering a query.

An optical disk is a device composed of T ordered tracks, an access mechanism and a viewing mechanism. A track is represented by its sequence number i within the tracks of the device, i=1,2, . . . ,T (to avoid discussion about boundary conditions we assume that track numbers are extended above T and below 1). Each track is composed of a number of sectors (or blocks). The access mechanism can be positioned at any track. When the access mechanism is positioned at a certain track i, the device can read data which completely exist within Q consecutive tracks (track i is one of them). In order to do that, the viewing mechanism *focuses* to a particular track with qualifying data (within the Q tracks). We call the Q consecutive tracks a *span* and this capability of optical disks *span access capability*. An anchor point a of a span, is the smallest track number within a span. The largest track number within this span is $a + Q - 1$. The anchor point of a span completely defines the tracks of the span. A span can therefore be described by its anchor point.

A number N of objects (or records) may qualify in a query. N is called the object selectivity (or record selectivity) of the query. In the case of optical disks the number of times that the access mechanism has to be moved for accessing the data which qualifies in a query is called the *span selectivity* of the query. Therefore, a first approximation of the cost of evaluating a query is given by the span selectivity

of the query.

When the access mechanism is moved a seek cost and a rotation delay cost are incurred as in the case of magnetic disks. The seek cost depends on the distance travelled by the access mechanism. Transferring a track of data from the device involves a track transfer cost as is also the case in magnetic disks. When more than one track within a given span has to be transferred to main memory, the access mechanism does not have to be moved as we mentioned before. However, there is a small additional delay (of the order of one millisecond) involved for focusing the viewing mechanism on each additional track (within a span) that has to be read. We call this delay a viewing cost and will ignore it for the remaining part of this paper. This model reduces to a model of magnetic disks when the number of tracks in a span is one and the track size is equal to the cylinder size.

A spiral scheme is generally used on CLV optical disks for the storage of data rather than the concentric rings found on CAV disks. The spiral consists of one long physical track of data recordings that begins at a distance from the centre of the disk called the *principal radius*, and continues until close to the outer edge of the disk where it terminates.

A track, for purposes of our analysis, starts from the intersection of the spiral with the radial line that starts at the centre of the disk and passes through the very beginning of the physical spiral. The track ends at the next intersection with this radial line.

A scheduling algorithm for retrieving the data which qualify in a query is an ordered sequence of anchor points which define the spans used for answering the query (and their order). Observe that there are many different sets of anchor points (and sequences of anchor points/schedules) that can be used for answering a given query. The span selectivity of the query (as well as the distance travelled by the access mechanism) depends on the scheduling algorithm used for answering the query.

In the following, we show optimal criteria for a scheduling algorithm when retrieving data in this model. In the algorithm, the access mechanism moves continuously in one direction. The criteria minimizes the number of times that the access mechanism is moved (span selectivity), as well as the total distance travelled by the access mechanism.

Theorem {optimal scheduling}

The number of spans required to access all qualifying objects in a query is minimized if:

A. Spans do not overlap and

B. The anchor point is always positioned on a track with qualifying sectors.

In addition, the total distance travelled by the access mechanism is also minimized (within one span length) if the access mechanism is moved so that in addition to the conditions A and B, the anchor points $a_1, \ldots, a_s$ of the schedule satisfy $a_1 \leq a_2 \cdots \leq a_s$.

Proof found in [4].

The theorem shows that if a schedule satisfies the conditions A and B it results in a minimum number of spans.

In [5] exact and approximate results have been derived for retrievals from CLV optical disks that follow optimal schedules. It was shown that the conventional CAV solution for the expected number of spans required for the retrieval of sector boundary crossing or non-crossing objects from a CAV disk as found in [4] was a good approximation to the CLV solution.

The expected number of spans $\overline{K}$ is given by:

$$\overline{K}(\overline{B}) = \frac{\overline{B}}{1 + \frac{Q-1}{T-1}(\overline{B}-1)}$$

Where $\overline{B}$ is the expected number of tracks to be retrieved. It is calculated differently for non-crossing and crossing objects.

The approximate expected number of tracks for non-crossing objects where: t is the number of tracks in the file, n is the number of objects to be retrieved and c is the track capacity, is given by :

$$\overline{B} = b(t,n,c) = t \left(1 - \frac{\binom{(tc - n)}{n}}{\binom{tc}{n}} \right)$$

For crossing objects, the same formula for spans is used but the approximate expected number of tracks is calculated differently.

$$\overline{B} = \overline{B}_c + b(T - \overline{B}_c, N \left(1 - \frac{T-1}{TC}\right) \frac{T - \overline{B}_c}{T}, C)$$

Where

$$\overline{B}_c = \frac{N}{TC}[(T-1)\,(2 - \frac{N}{TC}) + 1]$$

3. Hashing on CLV type Optical Disks

As a primary access mechanism, hashing has the ability of directly determining the location of a selected record without the need to perform disk accesses to consult file indices. This is of particular advantage on CLV optical disks as they typically require relatively long periods of time to reposition their access mechanisms.

Consideration of the details of providing a hashing access method for a file on a CLV optical disk produces some interesting observations. Firstly, that to ensure efficient accesses and space utilization, some way will be needed of accounting for the varying number of sectors that can pass under the viewing mechanism, depending upon its position, during one rotation of an optical disk. This situation would be similar to having variable capacity tracks on a CAV disk. The number of sectors increases as the viewing mechanism becomes positioned closer to the outer edge of the disk. The difference can be as much as three to one between the two extremes of the disk surface.

Secondly, we observe, that while the slow seek times of CLV optical disks make hashing attractive as an file access method, they also make the cost of employing overflow chaining to handle the inevitable bucket overflows to be expensive, or any other collision resolution method for that matter, that requires the access mechanism to move. The process of following a list of pointers from overflow block to overflow block across the surface of the disk could cause considerable delay in resolving a query.

One possible way of accounting for the variable number of sectors passing under the viewing mechanism is with some form of nonuniform hashing function. As is also the case for conventional CAV type disks (e.g. magnetic), movements of the CLV disk drive's access mechanism (seeks) will be minimized if there is both a one-to-one mapping between hash buckets and some physical division (track, cylinder) and a further correspondence in terms of their respective storage capacities. This means that on CLV type disks, the variable number of sectors in one revolution of the disk surface implies variable record capacities for the hash buckets. A special nonuniform hash function that distributes records to hash buckets in proportion to their capacity would help to avoid over utilizing the lower capacity buckets, causing overflows and further delays in resolving them, and under utilizing the buckets with greater capacity, causing lower file utilization. Something similar to Spiral Hashing [8] would seem appropriate.

Upon greater reflection, we realize that these observed complications are really a result of conforming to restrictions imposed by physical divisions that are not found on the spiral CLV recording scheme. Typical physical divisions found on conventional CAV type disks are concentric *tracks* which are rings of stored data that can be read completely with one movement of the access mechanism and one rotation of the disk surface, and *cylinders* which are groups of tracks on different disk platters that are accessible by one position of a multihead access mechanism. On a CLV disks such divisions are not present. The data recordings do not form concentric rings[1] and the disks are not usually found in multiplatter configurations. A more appropriate definition of a "track" for file organizations on CLV disks is "a set of sequentially accessed sectors". The number of sectors in the set being variable and independent of the number of rotations required of the disk actually to read them. For CAV type disks, this definition is equivalent to the conventional notion of a track.

[1] concentric CLV schemes are rare

The advantage of this definition is that more closely mirrors the way in which sectors are addressed and allocated on a CLV type optical disk. In a spiral recording scheme it is unnecessary to divide sectors into units which are related to the rotation of the disk platter; there is no physical reason to do so. The constant rate at which sectors pass under the access mechanism, independent of their position on the disk surface and hence the number of disk rotations (that is what constant linear velocity means), implies no need to relate a group of sectors to one revolution of the disk. Any such relationship would be somewhat arbitrary in any event, since any sector could be designated the "start" of a track, and since there is no guarantee that a whole number of sectors will pass under the access mechanism during one revolution, almost any sector could be designated the last. Further, on CLV disks, sector addressing and seek requests are expressed in terms of offsets from the beginning of the spiral, not in units called "tracks". On CD ROM's, this offset is expressed in the form of "Minute, Second and Data Block", and reflects the technology's origins as a medium for music delivery.

Note that it is still convenient for the purposes of analysis to employ the CAV definition of a track as an approximation to the CLV case. As mentioned above, it has been shown [5] that the CAV solution for the expected retrieval performance from a disk is very close to the solution for the CLV case. We will employ it in our analysis below.

The CLV concept of what constitutes a track allows them to be of any arbitrary size, as small as one sector, or more than can be read with one or even several rotations of the disk; this characteristic is very useful for hashing. Note that because the data is recorded in a spiral, reading an amount of data that requires the disk to rotate more than once incurs no penalty in terms of a seek cost. Provided we read sequentially, which is always the case for our definition of a "track", the access mechanism will for as long as is required slowly follow the recorded data as it spirals out to the edge of the disk. The access mechanism itself will periodically make small adjustments in its position while its associated optics will make even smaller adjustments to stay focused on the spiral. The important point is that no time delay is incurred by this following process; it is different from performing a seek.

The disadvantage of employing tracks with variable numbers of sectors, lies in possibly increased retrieval times due to longer transmission delays and more complicated main memory management

schemes. Tracks with large numbers of sectors will take longer to retrieve then ones with smaller numbers. Buffer management also becomes an issue when potentially large amounts of buffer space could be required. Organizing the space to ensure the ability to buffer the largest track may not be a simple matter, especially in a storage system shared among many concurrent users. Fixed size tracks do not have these problems.

Exploiting the ability to arbitrarily specify the capacity of a track allows us to avoid the two complications observed above. A special nonuniform hashing function that skews the distribution of records to hash buckets in proportion to their different, but fixed, capacities will not be required. And further, overflow resolution will not be a problem since overflows simply need not occur. On CLV type optical disks, buckets all have the same potentially (almost) unlimited capacity so a conventional uniform (i.e. well understood) hashing function is all that need be employed. In fact, to minimize the size of any given track assigned to hold the data of a hash bucket, and in turn, the expected amount of time required to read it in from the disk and be searched, it will be important that hash function selected distribute records to buckets in as uniform a manner as possible.

Bucket overflow can be completely eliminated when employing hashing as a file access mechanism on CLV type optical disks because tracks can be expanded to accommodate any number of records assigned to a bucket. The original motivation to restrict the size of hash buckets on CAV disks was to keep the number of records assigned to a physical division within the capacity of the division and reduce the probability of moving the access mechanism to resolve overflows. This is not necessary on CLV type optical disks; the flexibility allowed in selecting track sizes allows them to be adjusted to fit the number of records hashed to a bucket. All records can be allocated to sequential disk sectors and read with one disk access.

3.1. Ideal Environment

The ability to flexibly specify track capacities on CLV type optical disks makes it an almost ideal environment for implementing hashing as a file access method. It allows the elimination of two of the major problems encountered when employing it as an access method on conventional CAV disks: under utilization of file space and delays incurred during overflow resolution.

For efficiency reasons, hashing schemes employed on CAV disks typically do not use a small but substantial portion of the disk space allocated to the file. This under utilization tends to reduce the expected length of overflow chains that hold records assigned to an already full hash bucket by reducing the number of hash buckets expected to be full. This in turn, improves the expected performance of accessing the file as the probability of incurring the cost required to traverse a long overflow chain[2] is reduced. Typical values for the file utilization factor on CAV disks lie between 70% and 90%.

For certain types and applications of CLV optical disks, the file utilization factor can be 100%.[3] With variable capacity tracks bucket overflows should not occur, so no unused space need be allocated to reduce the probability of such an event. Tracks need only be as big as the capacity of their corresponding hash bucket and no bigger. Even space in the last sector of a track can be used if the first sector of one track overlaps with the last sector of the previous track. Again, main memory management may be a problem when using variable capacity tracks.

Allowing such an overlap however, introduces the potential for records to cross sector boundaries. For objects or records which are larger than a sector this will not be a significant source of overhead as at most one extra sector will be retrieved, but for objects smaller than one sector it represents a doubling of the required storage to be retrieved and buffered in main memory. A careful mapping between hash buckets and tracks can reduce the occurrence of records that cross sector boundaries. This is possible because the mapping does not need to reflect the logical order of the hash buckets. The track for the logically last hash bucket, for example, could occupy the first sectors allocated to the file or any other position that might prove advantageous. A method for implementing such a flexible scheme and its storage overhead are discussed below.

The problem of mapping the logical order of the hash buckets to a physical sequence is essentially a bin packing problem. The objective is to select an assignment that reduces the overhead introduced by sector boundary crossing. This is a difficult problem in general though the linear ordering of hash buckets may help to simplify it.

---

[2] just one collision resolution method but the same is true for other mechanisms

[3] unfortunately though, not for all as file expansion will still be an issue

3.2. Hashing Implementation

Implementing the variable track capacity hashing scheme for CLV disks requires a small amount of storage overhead to hold a table that implements the mapping function between the hash buckets and the CLV tracks. If hash buckets are laid out on the spiral in their logical order, the table will contain one sector address per hash bucket, successive table entries will delimit the boundaries of each track. If a more complex physical arrangement is called for, perhaps due to a need to reduce sector crossings, it can be accommodated by including more information in each table entry. It is expected that the mapping tables, even for a great many hash buckets, will be small enough to fit in main memory, but could be maintained on magnetic disk if so required by an implementation.

3.2.1. Hashing on CD ROM

CD ROM disks have similar characteristics as those used for audio compact disks (CD's). The nature of the manufacture and use of CD ROM type optical disks make them a prime platform for implementing hashing as a file access mechanism. The contents of a CD ROM disk are never updated or altered so a great deal of time, effort and resources can be spent preprocessing the disk contents to produce an organization that will ensure good access performance. Once expended, this effort is not repeated until a new disk is issued to physically replace the old one. This characteristic and the preprocessing stage make it possible to achieve the 100% file utilization possible with variable capacity disk tracks. Having no need to handle insertions or deletions to or from the file, implementing hashing on CD ROM optical disks, becomes a one time job of preprocessing the file by computing the contents of each hash bucket and constructing the hash bucket/sector address mapping table.

Before the disk is physically pressed, a certain degree of optimization and improvement can be injected into the expected access performance during the preprocessing stage. During this period, it is possible to examine and experiment with the performance of a variety of different combinations of hashing functions and numbers of hash buckets, selecting the one that provides the greatest expected access performance for the file contents. We can also give a great deal of attention to the bin packing problem of assigning hash buckets to physical locations to reduce sector boundary crossings.

As there is no requirement to allocate disk space to hash buckets that have not been assigned records, a simple "NULL" entry in the table is all that is required to accomplish this, one can employ a large number of hash buckets in an attempt to reduce the expected number of records per bucket, and hence the time to retrieve and search each one. There is virtually no penalty for having a large number of buckets other than a larger mapping table. Even for large files, the mapping table size for a CD ROM implementation can be quite manageable. On a CD ROM disk, complete sector addresses require at least twenty bits, 7 bits for the minute (0-99), 6 bits for the second (0-59) and 7 bits for the data block number (0-74), or two and a half bytes. A hashing scheme employing 20,000 hash buckets would only require 45000 bytes of main memory, a size easily accommodated by today's generation of microcomputers, the primary type of computer associated with CD ROM use. If each bucket was allocated a track of just one sector, the size of the file would be 40 megabytes and the table overhead, a mere 0.1%. If needed, the size of the table could be reduced by storing sector run lengths in fewer bytes per table entry (i.e. store the number of sectors in each track). The savings in space that would result need to be measured against the effort to calculate an absolute sector addresses.

3.2.2. Hashing on CLV WORM

Providing an efficient hashing access mechanism for a WORM type CLV optical disk will be more difficult than for CD ROM. The obvious complication is the existence of dynamically changing files and the necessity to allow for the insertion and deletion of file contents on a medium that does not allow for the reuse of previously allocated space. Also, for the applications in which WORM disks are typically employed, there will usually be no preprocessing stage in which to explore alternate hashing strategies, so in general, it will not be possible to improve the expected access performance by adjusting the number of hash buckets or the hashing function to fit the data set as it was for CD ROM. As a result, it will be a nearly impossible task to achieve the potential of 100% space utilization.

To improve efficiency, and in particular space utilization, implementing hashing for a file stored on a WORM disk will probably involve some degree of buffering on a magnetic disk for both the mapping table and the contents of hash buckets. A pointer in each entry of the mapping table would lead to what is essentially an overflow chain stored on the magnetic disk of records that are assigned to the hash

bucket but which have not yet been archived on the optical disk. Information on which records have been logically deleted from the hash bucket (but which cannot be erased from the physically unerasable disk) might also be stored in the bucket's entry in the mapping table. At some point, as the buffer space gradually fills up, a portion or all of the buffered information would be flushed. The "new" records will be merged with the old (logically undeleted) ones already on the optical disk and written together on new tracks; the mapping table will also be updated to reflect the changes.

With regard to space utilization on the WORM disk, we speculate that the extent of buffer flushing may have some, as yet undetermined, decision points. It may not be profitable, for example, to invalidate the contents of a long track and allocate a completely new one, simply to add a few more records to it. The deciding factors will be the size of the track on the WORM disk, the proportion of deleted records in the track, the amount of buffered information associated with the track's hash bucket and the amount of magnetic storage available.

For the archiving of large objects in a hashing file access scheme on WORM disks, it will be efficient in terms of space utilization to store a pointer to the object in the hash bucket rather than the object itself. As it would be wasteful to recopy large objects (records) which might be megabytes in size simply for the benefit of merging buffered information with that on the optical disk. A pointer would add another disk access to the retrieval process but the savings in storage space may be worth the extra effort.

3.3. Expected Performance of Hashing

We can employ the results of [5] to analyze the expected retrieval performance of hashing as a file access method. As a primary access mechanism the analysis will be fairly simple, as a secondary access method it is slightly more complex.

3.3.1. Hashing as a Primary Access Method

When used as a primary access method, we expect that the retrieval performance of hashing to be very good. For the scheme outlined above, the delay will consist of the time to perform one seek and the time to read one hash bucket.

The expected delay is given by:

$$expected\ delay = S_t + Bk_t$$

Where S_t is the expected seek time and Bk_t is the expected bucket transfer time.

A typical value for the expected seek time for CD ROM disks is 400msec. The expected bucket transfer time depends upon the expected number of sectors occupied by the bucket, this in turn is determined by the size and average number of records hashed to a bucket. The time required to transfer one sector from a CD ROM disk is exactly 13.3msec.

If records are small enough to be contained in a sector and a sufficient number of hash buckets are employed to reduce the expected number in a bucket to one, then at most two sectors will need to be retrieved if sector boundaries can be crossed by records, and at most one, if not. A delay between 413msec and 426msec is expected for the retrieval of one hash bucket.

3.3.2. Hashing as a Secondary Access Method

When employed as a secondary file access method, the dispersal of qualifying records to different parts of the file (because they have been ordered by some other primary access method) will lower the expected retrieval performance of hashing. In such a situation the contents of a hash bucket will not be records but pointers to records in the file. These pointers cause many more accesses beyond the one to retrieve the hash bucket contents.

Where $\overline{K}$ and $\overline{B}$ are as calculated in section 2, ignoring the transmission delay, the approximate expected retrieval performance is given by:

$$expected\ delay = (\overline{K}(\overline{B}) + 1)\, S_t$$

$\overline{B}$ depends upon the expected number of records (pointers) per hash bucket and whether sector boundaries are crossed. The transmission delay is ignored because the delay due to seeks is expected to be much larger.

4. Tree Index File Organizations on CLV Optical Disks

Depending upon the type of optical disk, either CD ROM or WORM, the task of providing an indexed sequential file structure such as B-trees and ISAM will be either straight forward or very difficult. Just as for hashing, the static nature of the contents of a CD ROM disk make it relatively easy to organize them into pointer linked tree organizations. For WORM disks, the situation is completely different. Modification of files, mostly updates, is possible and is expected to occur frequently. The inability to modify information stored on WORM disks has profound implications on the efficiency of pointer linked structures when they undergo continuous modification.

Implementations of structures on WORM disks that employ pointers must address two implications that arise from WORM disk characteristics. Firstly, pointers can only point to sectors that have been written. Secondly, any change to a member of a linked list structure will require all preceding members of the list or lists that include the changed member to be changed themselves to update their pointers; members of the structure that come after the changed member do not require alteration.

4.1. B-trees on CLV type Optical Disks

The data structure known as Balanced Multiway trees or simply B-trees, is an important file organization method that uses pointers to maintain its structure. Like hashing, it provides efficient access to large amounts of data. Unlike hashing however, it has the major advantage of allowing access to file contents in the sequential order of their keys; since many applications require this feature, B-trees are widely employed.

A feature of B-trees that it shares with hashing when employed on conventional CAV disks is a low storage space utilization rate, in the worst case, about 50%, in the average case, approximately ln 2 or 69%. Newer variations have reported utilizations of 85%.

4.1.1. Implementation on CLV optical disks

As is also the case for the hashing file access method, the variable capacity tracks available on CLV disks can be used to improve the space utilization factor. However, the complexity and efficiency of a B-tree implementation, depends heavily upon the targeted type of CLV optical disk, CD ROM or WORM.

The static nature of the files stored on CD ROM optical disks simplifies things considerably as the requirement of supporting insertions and deletions is removed and the file contents can be preprocessed to improve expected access performance and space utilization. For CLV WORM type optical disks the situation is more complex. Insertions and deletions do need to be supported, but as mentioned previously, WORM disks only allow modification of data by writing the new information on another portion of the disk surface, not by writing over the old. Any change to a B-tree, even to delete a record, will cause changes to the tree and consequently more new disk space will be occupied to incorporate those changes. Unrestricted modification of a file organized as a B-tree on a WORM disk (either CLV or CAV) will quickly fill the disk.

4.1.1.1. Implementation on CD ROM

The file utilization rate of a B-tree file structure when employed on a CD ROM disk can be close to 100%. The lower file utilization rates quoted above are the result of unused space left in the nodes of the balanced tree. This unused space varies from node to node depending upon the pattern of insertions and deletions experienced by the file. Since files on CD ROM disks are static, only the space actually occupied in a node need be allocated on the disk. Again, we can eliminate logically allocated but physically unoccupied space by employing variable capacity tracks, and in the process raise the file utilization rate to very near 100%.

4.1.1.2. Implementation on WORM

The advantages offered to a B-tree implementation by the characteristics of CD ROM disks are not present when using WORM disks. In fact, certain characteristics of the B-tree insertion and deletion process make it a particularly undesirable choice for WORM disks, especially if the file contents are expected to be volatile.

Inserting or deleting records in a Balanced Tree has the potential of causing a great deal of reorganization, requiring the contents of many nodes to change to keep the tree in balance. Unfortunately, a B-tree node, once written on a WORM disk and even if to add just one record, cannot be changed without completely rewriting it on another part of the disk and occupying additional storage. Of course,

not all insertions will cause the tree to be reorganized, but even if its balance is not affected by the insertion of a record, all of the nodes in the tree along the path between the root and the affected node will need to be rewritten to update their pointers and maintain the integrity of the tree. Unrestrained, this duplication process would quickly fill up even the immense capacity of a WORM disk.

As was the case for employing the hashing file access method on WORM optical disks, it might be possible to alleviate some of the problems associated with the unerasability of WORM disks, and the resulting poor space utilization, by using magnetic disks as an accompanying storage medium. The links between nodes, for example, could be stored on the magnetic disk and updated as necessary thereby eliminating the necessity of duplicating nodes simply to modify a single pointer. Through the use of such a table it would also be possible to buffer some nodes on the magnetic disk.

4.1.2. Expected Performance of B-trees on CLV optical disks

The slow seek and data transfer rates typical of CLV optical disks will tend to make B-trees a poor performing file access method. Because of their large capacities, optical disks will generally store a very large number of records (objects). A B-tree for a file with a large number of records (objects) will have several levels of nodes, each with a high branching factor and hence a large number of records. Traversing the tree will usually require as many seeks as there are levels in the tree, save for one if the root is kept in main memory, each of which will require a relatively long time to complete. The time to actually read the node and search it will add further to the expected delay since they will generally be large themselves.

From [10], with the root not in main memory, we find the expected cost of retrieval from a Balanced Multiway tree to be approximately:

$$(S_t + B_t \left\lceil m \frac{E_s}{B_s} \right\rceil + D \log_2 m) \log_m N$$

Where, S_t is the expected seek time, B_t the block transfer time, m the branching factor, E_s the size of one of the entries (records) in the node and B_s is the size of a block (sector). $D \log_2 m$ is the time to perform a binary search on the node when it is in main memory and N is the number of records in the file.

Differentiating and setting the derivative to zero to find the branching factor (m) that will minimize the expected delay, we produce the following expression:

$$m \ln m - m = \frac{S_t}{B_t} \frac{B_s}{E_s}$$

4.1.2.1. B-tree Access Performance on CD ROM

Using typical performance parameters of a CD ROM disk, we can calculate the optimal branching factor for a given number of records of a particular size. A typical value for the expected seek time S_t is 400msec. The time to transfer one block or sector, B_t, is exactly 1/75 of a second, and the size of sector, B_s is 2048 bytes when using the highest level of error correction, as will be the case for record oriented data; lower levels of error correction are used for objects which can tolerate a small rate of error such as bit maps or recorded audio.

For a record size of 200 bytes we calculate the following:

$$m \ln m - m = \frac{0.4}{1/75} \frac{2048}{200}$$

$$= 307.2$$

which corresponds to a value of m equal to 88 (i.e. each node should have 87 or $m-1$ records).

If the entire 550 megabyte capacity of the disk was occupied by the file, that would include 2750000 records. The expected number of accesses is therefore:

$$= \log_m N \quad or \quad \frac{\ln N}{\ln m}$$

$$= \frac{\ln 2750000}{\ln 88}$$

$$= 3.31$$

Using the performance parameters for given above, the expected delay due to disk accesses is:

$$expected\ delay = (0.4 + (1/75) \left\lceil 87 \frac{200}{2048} \right\rceil) 3.31$$

$$= 1.72\ seconds$$

Because 87 record entries do not completely fill all of the nine sectors assigned to them, we can improve the expected performance and space utilization by increasing m to a value that does. In this case, we can increase m to $\left\lfloor 9 \frac{2048}{200} \right\rfloor = 92$, reserving space for the m'th pointer in the node. There would now be 91 records per node.

The expected delay for $m = 92$ is 1.71 seconds, a slight improvement. For comparison purposes, the expected delay for $m = 70$ is 1.72 and for $m = 100$ it is also 1.72 seconds.

When the root of the B-tree is stored in main memory, we can reduce the number of disk accesses by one. This will have an effect upon our analysis.

The expected delay when the root of the B-tree is kept in main memory is given by:

$$expected\ delay = (S_t + B_t \left\lceil m \frac{E_s}{B_s} \right\rceil + D \log_2 m) (log_m N - 1) + D \log_2 m$$

Ignoring the time to search the node (i.e. $D = 0$), we obtain:

$$= (S_t + B_t \left\lceil m \frac{E_s}{B_s} \right\rceil) (log_m N - 1)$$

Differentiating this expression and setting the derivative to zero, we find:

$$m (\ln m - \frac{\ln^2 m}{\ln N} - 1) = \frac{S_t}{B_t} \frac{B_s}{E_s}$$

Using the values from our previous example, we calculate that $m = 134$ will produce a minimal expected delay of 1.19 seconds. However, as before, our calculated value of m branches per node does not completely fill all of the sectors assigned. The value of m that does is 143, for an expected delay of 1.16 seconds. For comparison purposes, a slightly higher value of $m = 160$, gives a higher expected delay of

1.18 seconds, showing that we have indeed found the value of m that produces a minimal delay.

4.1.2.2. Expected Secondary Access Performance

When used as a secondary access method, the expected retrieval performance of B-trees well be reduced by the need to perform additional accesses to follow pointers to records in the file. Again, the expressions given in section 2 can be used to determine the expected delay.

For a file of N records and a B-tree with a branching factor of m and also assuming fully populated lowest level nodes, the expected retrieval performance is:

$$expected\ delay = S_t(\log_m N + \overline{K}(\overline{B}))$$

Where the expression used for $\overline{B}$ depends upon whether sector boundaries are crossed by records, the size of the file, the size of a span and the number of records in the lowest level node (which is at most $m-1$).

4.2. ISAM on CLV Optical Disks

An alternative sequential access method to B-trees is ISAM (Index Sequential Access Method). Like B-trees, it allows sequential access to file contents and primarily consists of a pointer linked tree structure. Unlike B-trees, the index structure of ISAM has the useful property of remaining unchanged when the file undergoes modification. This is not a great advantage for files stored on CD ROM which cannot change, but for files stored on WORM disks, it is ideal.

4.2.1. ISAM on CLV WORM

It is not possible to implement a "pure" ISAM file organization on a WORM disk, but a slight variation can be easily accommodated. The changes necessary to adapt to the characteristics of WORM disks lie in the method of handling overflows and in the frequency of periodic file maintenance.

Insertion of records can cause the disk space allotted to a particular range of keys to overflow. The conventional ISAM approach to this problem is to reserve an overflow area and shuffle the positions of old and new records in the primary and overflow areas to maintain physical sequential ordering of the records. Periodic maintenance on conventional ISAM files is performed to clear the overflow areas by reorganizing the file and its index. The maintenance improves the expected retrieval performance by

reducing the expected amount of overflow processing required.

On WORM disks, it is not possible to "shuffle" the position of records to maintain physical as well as logical ordering. Rewriting the new and old records to do so, will quickly use up the disk space and decrease expected insertion performance.

Much like the arrangement previously described for hashing, a modified ISAM approach for CLV WORM optical disks could use magnetic disk storage as the overflow medium. It can also take advantage of the variable track capacities to reduce the need for periodic file reorganization. In such an approach, the top level of the ISAM index would be kept in main memory and the rest on the optical disk. A small table would also be kept in main memory to map between the lowest levels of the index and both physical (optical) disk locations and overflow chains stored on magnetic disk.

When a record is inserted into the file, it will be placed on the overflow chain kept for its index position on the magnetic disk. When a record is retrieved, the optical disk will be accessed, if the record is not found then the overflow chain on the magnetic disk will be searched. Sequential access will required both disks to be accessed if there are records buffered on the magnetic disk.

The deletion of a record could be handled by storing information in the mapping table or on magnetic disk. The exact method should not prove critical as the type of applications that to employ WORM disks tend to be archival in nature so deletions are expected to be rare.

When an overflow chain becomes excessively long by some criteria, its contents and the contents of its corresponding primary area on the optical disk will be merged and stored on a new WORM disk track. The mapping table would be updated to reflect the change and the buffer space on the magnetic disk flushed. Note that the index will not change.

The availability of variable capacity tracks on CLV WORM optical disks allows the capacity of a track to expand to meet the load imposed upon it. Thus, the flushing and merging operation to continue without requiring a reorganization of the file and its index until the capacity of the disk is exhausted. When the disk is full and further insertions are pending, the file and its index (along with the new records) can be transferred to a new disk and be reorganized in the process.

Reorganization may be desirable before the disk is full to adjusted the index to better match the actual contents of the file. This will shorten the length of tracks that have grown excessively long due to insertion patterns that did not match the organization reflected in the current index. A track may be considered to be too long if its capacity exceeds that that can be buffered in main memory or if the transmission time required to read it from the disk exceeds some threshold. Shortening the track length by file reorganization will lessen buffering problems and improve the expected retrieval performance by reducing the transmission delay and the expected length of overflow chains.

5. Conclusion

We have shown that conventional file access methods found on conventional magnetic disk that use a CAV recording scheme can be adapted to the characteristics of CLV optical disks, such as CD ROM and WORM.

In particular we have shown that particularly good retrieval performance and file utilization close to 100%, can be achieved by using hashing as a primary file organization method on CLV type optical disks.

For tree like file organizations, particularly B-trees, it was shown that they would not have particularly good access times when used on CLV optical disks. B-trees were shown to be a bad choice access method for WORM disks are they are expected to require large amounts of disk space to accommodate file changes. The ISAM method was discussed and an implementation strategy that made better use of the disk was described.

We also illustrated throughout the paper, the implementation advantages imparted by the static nature of files on CD ROM to all of the file access mechanisms; as well as the optimization that can be performed during the preprocessing stage of the disk's contents.

The spiral recording scheme used on almost all CLV optical disks, was shown to have a major positive impact on the expected retrieval performance and space utilization of the all of the file access methods. This was due to the flexibility allowed in selecting the capacity of disk tracks.

We are pursuing further investigations this this area.

6. References

[1] Bell, A., Marrello, V., "Magnetic and Optical Data Storage: A comparison of the Technological Limits", *Proceedings IEEE Compcon*, Spring 1984, 512-517.

[2] BYTE86, Collection of Articles, *Byte*, May 86.

[3] Christodoulakis, S., Ho, F., Theodoridou, M., "The Multimedia Object Presentation Manager of MINOS: A Symmetric Approach", *Proc. of ACM SIGMOD '86*, May, 1986, pp 295-310.

[4] Christodoulakis, S., "Analysis of Retrieval Performance for Records and Objects Using Optical Disk Technology", *ACM Transactions on Data base Systems*, June 1987.

[5] Christodoulakis, S., Ford, D.A., "Performance Analysis and Fundamental Performance Trade Offs for CLV Optical Disks", *Proceedings ACM SIGMOD*, Chicago, June 1988.

[6] Fujitani. L., "Laser Optical Disks: The coming Revolution in On-Line Storage", *CACM 27*, 6 (June '84), 546-554.

[7] Maier, D., "Using Write-Once Memory for Data base Storage", *Proceedings ACM PODS 82*, 1982.

[8] Martin, G.N.N.: "Spiral Storage: Incrementally Augmentable hash addressed Storage", *Theory of Computation Rep. 27*, University of Warwick, England, 1979

[9] OPTIMEM1000, "Optical Disk Drive (OEM MANUAL)", Optimem, 435 Oakmead Parkway, Sunnyvale CA 94086.

[10] Reingold, E.M., Hansen, W.J., "Data Structures in Pascal", Little Brown, Boston, 1986.