Retrieval Performance Versus Disc Space Utilization on WORM Optical Discs

Stavros Christodoulakis

Daniel Alexander Ford

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

ABSTRACT

Steady progress in the development of optical disc technology over the past decade has brought it to the point where it is beginning to compete directly with magnetic disc technology. WORM optical discs in particular, which permanently register information on the disc surface, have significant advantages over magnetic technology for applications that are mainly archival in nature but require the ability to do frequent on-line insertions.

In this paper, we propose a class of access methods that use rewritable storage for the temporary buffering of insertions to data sets stored on WORM optical discs and we examine the relationship between the retrieval performance from WORM optical discs and the utilization of disc storage space when one of these organizations is employed. We describe the performance trade off as one of fast sequential retrieval of the contents of a block versus wasted space owing to data replication. A model of a specific instance of such an organization (a buffered hash file scheme) is described that allows for the specification of retrieval performance objectives. Alternative strategies for managing data replication that allow trade offs between higher consumption rates and better average retrieval performance are also described. We then provide an expected value analysis of the amount of disc space that must be consumed on a WORM disc to meet specified performance limits. The analysis is general enough to allow easy extension to other types of buffered files systems for WORM optical discs.

1. Introduction

Steady progress in the development of optical disc technology over the past decade has brought it to the point where it is beginning to compete directly with magnetic disc technology [3, 4, 12, 20, 22]. The two technologies have much in common, but there are differences that are significant. For example, all magnetic storage technology is rewritable, but this is not true for all optical disc technology. WORM optical discs (Write-Once-Read-Many times) allow on-line registration of information, but data cannot be modified once registered. For archival applications that require long (or sometimes infinite) archival life and on-line insertions and retrieval, this characteristic of WORM discs presents particular advantages. A number of systems utilizing WORM optical discs as their mass storage device have appeared in market and in the research laboratories [7, 13, 21, 28, 29].

The different characteristics of optical disc technology suggest that alternate or modified database design techniques for applications using them, which account for these differences, may be appropriate. For the case of WORM optical discs, for example, retrieval performance and disc space utilization are two aspects that are significantly affected by design decisions and so deserve special attention. In a previous report [8], we presented file organizations that attempt to address these issues by employing magnetic discs to buffer record insertions to files stored on WORM optical discs. Other new access methods, specifically designed for WORM storage, are also under investigation in various research centres [11, 16, 24, 26].

For both technologies, a large factor in determining the level of retrieval performance of accesses to a data set is the degree to which the data set's physical layout and organization facilitates its sequential retrieval. Movements of the access mechanisms, or seeks, on both types of discs, take a long time to complete compared to the time required to (sequentially) read data from a track. Thus, the more movements of the access mechanism required to retrieve a data set, the lower the retrieval performance. Physical database methods developed for magnetic discs frequently exploit the advantages of sequentiality by utilizing large block sizes [15, 27, 30], and physical database structures and access methods can generally be characterized by the proportion of the sequential versus random retrieval of data sets that they provide [2, 14, 25, 31].

This relationship is even more evident for data sets stored on WORM optical discs than on magnetic. The high storage densities found on optical discs and the fact that data located close to the position of the access mechanism can be read without it moving, by tilting a mirror (span access capability [6]), give them high data transfer rates (Megabits/Sec). But, their long seek times (100 to 1000 milliseconds) are ten to one hundred times slower than those of magnetic discs.

Again, for both technologies, physical reorganization of a data set can restore or maintain its physical contiguity. Because Magnetic discs are rewritable, this is an easy task and many file systems and access methods for magnetic discs (e.g. ISAM) have utilities that routinely perform physical reorganization for this purpose. This reorganization may be done for the whole file periodically for static organizations [1, 17], or locally, at insertion time, for more dynamic organizations [18, 19].

The scenario is somewhat different, however, for WORM optical discs, where the reorganization cannot be performed without incurring some overhead in terms of storage space consumption. Data registration on a WORM optical disc is performed by using the energy of an applied laser beam to form a pattern of "pits" in the disc surface. This pit formation process is physically irreversible and data registration can only be performed in units of complete disc sectors (not parts of sectors). Furthermore, because of a sophisticated error correcting code [23] stored in each sector, along with the data, once a sector has been written, its contents cannot be changed. This is because the code prevents changes by either "correcting" the change without comment or reporting it as an error.

These characteristics imply that, unlike magnetic discs, once storage space is allocated on a WORM optical disc it cannot be reused, modified or reclaimed. This further implies that all changes to data stored on the disc must be recorded in previously unwritten sectors. Since the activity of maintaining or improving the physical contiguity of a data set to attain a given level of sequentiality of access requires some change in the physical positioning of the data, doing so on a WORM optical disc, implies the consumption of disc space. Furthermore, because storage space cannot be reclaimed on a WORM disc, physical reorganization will necessitate the duplication of all or a large portion of the data set. This duplication and the consumption of the disc sectors it requires, is unnecessary from a pure data storage view point, but is an important means of achieving high retrieval performance from a WORM optical disc.

In practice, it might not always be necessary to do the reorganization each time changes or additions decrease physical contiguity, a lower level of sequentiality and correspondingly lower retrieval performance might be tolerated by applications that have lower performance demands. The variability in performance requirements among applications leads to a direct trade off between the required level of sequential access of data sets stored on WORM optical discs and the rate at which disc storage space is consumed. Maintaining a high level of sequentiality for fast retrieval performance will consume space at a greater rate than maintaining a lower level.

In this paper, we propose a class of access methods that use a rewritable memory buffer for temporary storage of data that are to be stored on WORM optical discs. We develop in detail the trade off of retrieval performance versus WORM disc storage space utilization. Our results provide an analytic tool that allows the database designer to study the retrieval performance objectives versus the WORM storage utilization objectives as a function of the rewritable memory buffer size.

In section 2 of this paper, we discuss in general, the use of buffering of file organizations for WORM discs and then describe in detail, an example of a buffered hash file organization (BHash). Buffered organizations maintain new insertions in a rewritable storage buffer. When the buffer is full they flush the portion of the block in the buffer that is largest to the WORM optical disc and possibly merge this flush with previous flushes to improve the sequentiality of the data set. In section 3 of the paper, we analyze the space consumption on the WORM disc as a function of the number of flushes from the rewritable to the write-once memory, the merge (performance) constraints, and the expected flush size. In section 4, we analyze the problem of calculating the expected flush size and present closed form formulae based on an expected value analysis. In section 5, we use the analytic results developed throughout the paper to derive a trade off between the sequential access cost and the storage space consumption on the WORM disc; this trade off will be of particular use in the physical database design phase. We also compare our results against simulations and show that they are in excellent agreement. Finally, in section 6 of the paper, we present a summary and our conclusions.

2. Using Rewritable Storage to Buffer Insertions

A buffered file organization for WORM optical discs uses reusable storage, either a magnetic disc or semiconductor main memory, to buffer all file modifications [8]. Alterations to the contents of the file, insertions, updates, and deletions, are not applied directly to the WORM disc, but are instead, delayed until some later time (for example when the buffer is full) by storing them in the buffer. The motivation for this delay is two-fold. First, it can reduce sector fragmentation that could be a problem if record lengths do not closely match the length of a disc sector (typically 1Kbyte). By batching a series of record insertions, instead of storing them individually as they arrive, and by allowing records to cross sector boundaries, fragmentation can be greatly reduced. Secondly, buffering reduces the dispersion of the file contents over the disc. By grouping logically related records together in the buffer, and later storing them together on the disc as a unit, the rate at which a data set is divided into physically disjoint groups, as a function of the number of record insertions, will be reduced. This in turn will reduce the required number of physical reorganizations, and the resulting consumption of disc space required for each.

Although we concentrate our description in this paper on a buffered hash file organization "BHash" to make it concrete, we wish to emphasize that our results are in fact applicable to a class of organizations that maintain a proportion of their data blocks on rewritable storage to optimize long term storage and retrieval objectives for WORM storage [9]. Multiway tree organizations [10], for example, require large nodes (blocks) to guarantee a small tree height. The node contents must be nearby in WORM storage to guarantee fast node access.

2.1. Buffered Hashing (BHash)

A buffered hash file organization is an excellent vehicle for examining the trade off between maintaining a specified level of sequentiality of access to a data set and the rate of storage space consumption on a WORM optical disc. It is also an organization that is likely to be employed by the types of applications that use WORM optical discs as a storage medium; interactive multi-media applications such as the new generation of office automation systems [5], training simulations, or games, are good examples. Because of their interactive nature, these types of applications often require the fast file access provided by hashing organizations. They also rarely need to delete data, but require high storage capacities with fast transfer rates for multi-media data such as bit maps and audio segments. They use WORM optical discs to meet these needs.

Any insertion or modification to the file is first placed in the buffer. It is assumed that the hashing function will randomly distribute the records across the hash buckets in a uniform manner. When the buffer is full and the next buffer insertion occurs, a group of records belonging to a single hash bucket is removed from the buffer and placed on the WORM optical disc. This group is linked into a list of any other groups belonging to the same bucket that were previously flushed from the buffer.

We will assume that all changes to the file are equivalent to a record insertion. This is reasonable as the types of applications that use WORM optical discs are archival in nature and so both record updates and deletions will be infrequent compared to insertions.

We will ignore bucket overflow considerations as the single spiral track of a CLV format WORM optical disc will allow any amount of data, up to the capacity of the disc, to be stored and accessed contiguously. Thus, any size bucket capacity can be accommodated on such a disc. This assumption is not as reasonable for CAV format WORM discs, but since most worm discs have *span access capability* that allows almost instant access to a large portion of the disc surface (one megabyte or more), appropriate file design can usually avoid overflow problems.

Let X be the number of hash buckets, and let $Buff\_cap$ be the capacity of the buffer in bytes. Let V be the number of records inserted into the file. Records are assumed to have a fixed length of r bytes and can cross sector boundaries.

Let Y be the limit on the number of groups into which a hash bucket can be divided. When adding a group of records flushed from the buffer to the list would exceed this limit, the new group and all or just some of the groups in the list are merged together. This group is then stored in a new location on the optical disc.

Let g be the expected size, in records, of the groups flushed from the buffer to the disc.

Let the disc sector size on the WORM optical disc be l_s bytes. We will assume no limit on the number of disc sectors available. Let $W = \lfloor Buff\_cap / l_s \rfloor$ be the number of records that can be buffered.

In the next section we develop formulae that give the expected disc space consumption as a function of the number of buffer flushes, the merge constraints Y and the expected buffer flush size g.

3. Analysis of Disc Space Consumption

The expected number of flushes of a group of records from the buffer per hash bucket in the BHash file organization is a function of the algorithm for merging flushed groups on the optical disc when the number would exceed Y, the number of records inserted V, the number of records that can be buffered W, and g, the expected size of each group flushed from the buffer. The expected flush size is constant when the process is in steady state and a function of the size of the buffer and the number of hash buckets. Its derivation is left to a later section.

We describe and analyze the expected space consumption of two merge algorithms below. The first performs a full merge of all flushed groups, while the second performs a partial merge. The first has the advantage of better expected random access performance while the second has a lower consumption rate.

The first flush from the buffer occurs after $W + 1$ record insertions, and the next flushes occur every $g(W,X)$ record insertions. The expected number of flushes per bucket for $V\,(V > W + 1)$ record insertions to the file is:

$$Flushes(V,W,X) = round\left\{(1 + \frac{V - (W+1)}{g(W,X)})\frac{1}{X}\right\}$$

otherwise $Flushes(V,W,X) = 0$.

3.1. Full Merge

The full merge algorithm for $Y = 3$ is illustrated in Figure 1. It merges all the groups on the disc and the flushed group into a single group whenever the number of groups would exceed Y. The expected number of groups on the disc at any one time is $Y/2$, this is also the expected number of seeks needed to retrieve them.

The merging of a bucket's buffer groups on the optical disc occurs once every Y group flushes to the bucket, except for the first merge, that occurs after the $Y + 1$'th flush.

If $V > W + 1$, then the expected number of merge operations that occur for each hash bucket is:

$$Merges(V,W,X,Y) = \left\lfloor \frac{Flushes(V,W,X) - 1}{Y} \right\rfloor$$

otherwise $Merges(V,W,X,Y) = 0$.

Knowing that $g(W,X) = g_1 = g_2 \cdots = g_n$, we can compute the expected number of consumed disc sectors per hash bucket. In doing so, we must account for sectors consumed by single groups flushed from the buffer and the larger merged groups. Each group flushed to a bucket on the disc is stored as a separate unit unless it triggers a merge operation, in which case it is stored directly in the larger group (e.g. groups g_4, g_7 and g_{10} in Figure 1.).

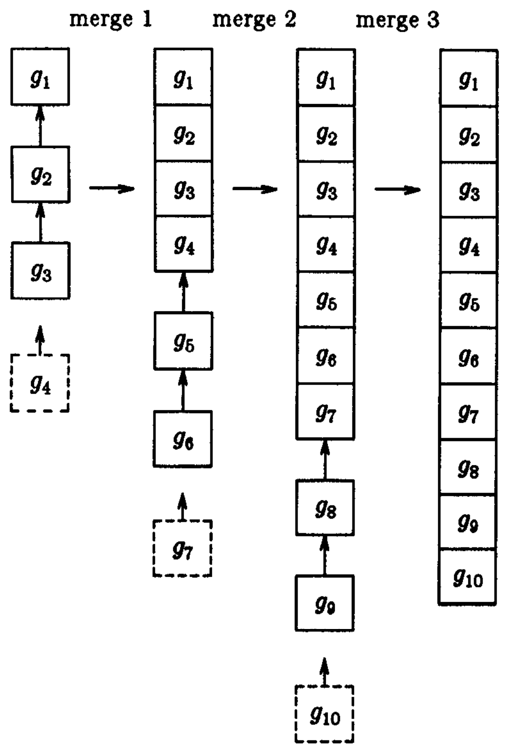

Figure 1. Full Bucket Merges for $Y = 3$

The number of separately stored buffer groups belonging to a bucket is the number of flushes to the bucket minus the number of merge operations that have occurred. Thus, the total expected number of sectors occupied by the separate groups is:

$$(Flushes(V,W,X) - Merges(V,W,X,Y)) \cdot \left\lceil \frac{g(W,X) \cdot r}{l_s} \right\rceil$$

Each merge operation, except for the first, adds Y groups of size g records to the the existing set of merged groups; in the case of the first merge operation, $Y + 1$ groups are combined. Thus, the number of groups that are combined in the i'th merge operation is $1 + i \cdot Y$.

The total expected number of occupied disc sectors per hash bucket is:

$$B\_Occupied(V,W,X,Y) =$$

$$(Flushes(V,W,X) - Merges(V,W,X,Y)) \cdot \left\lceil \frac{g(W,X) \cdot r}{l_s} \right\rceil$$

$$+ \sum_{i=1}^{Merges(V,W,X,Y)} \left\lceil \frac{(1 + (i \cdot Y)) \cdot g(W,X) \cdot r}{l_s} \right\rceil$$

This assumes that at least one merge operation has occurred, otherwise the second term will be zero.

The total expected number of disc sectors consumed is simply:

$$Tot\_Occupied(V, mark\ W, X, Y) = X \cdot B\_Occupied(V, W, X, Y)$$

3.2. Partial Merge

The merging process for the partial merge algorithm is illustrated in Figure 2. When the number of groups on the disc would exceed Y, the flushed group and the one or two groups with the smallest size are merged together. This reduces the number on the disc by only one group and means that the number is always Y or $Y - 1$. Implementing this merging algorithm will require a small amount of extra rewritable store to buffer the pointer links between the buffered groups of a bucket as they need to be changed more than once during the lifetime of a particular group, and so cannot be stored on the disc.

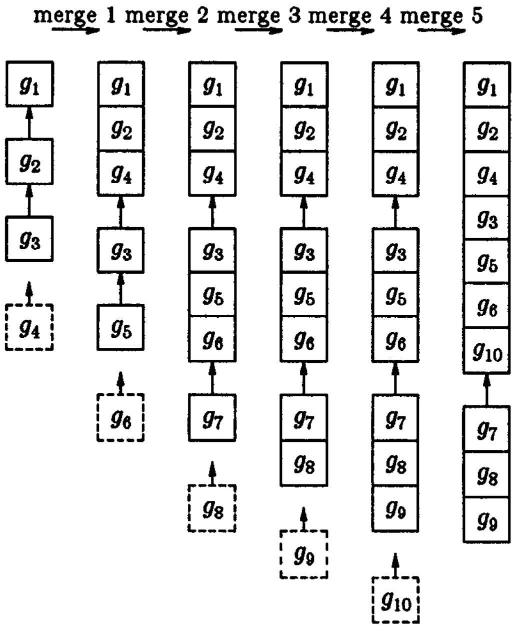

Figure 2. Partial Bucket Merges for $Y = 3$

The analysis of the expected disc space consumption for the partial merge algorithm is based on the cycle the list goes through as it changes from a list of Y equal size list elements through a series of merges and then again becomes a list of Y equal (but now larger) sized elements. Such a cycle is completed in Figure 2 after merge 4. We will call this cycle a *duplication cycle*.

Let Q be the number of complete duplication cycles of the list of length Y that occur for V insertions.

$$Q(V,W,X,Y) = \left\lfloor \log_2 \left(\frac{Flushes(V,W,X) - Y}{2Y} + 1 \right) \right\rfloor$$

Each cycle causes $Y-1$ list element merges to occur. Where i is the number of flush groups in a list element, the space consumed (number of sectors) by each merge is given by:

$$Emerge(i,W,X) = \left\lceil \frac{(2i+1)\cdot g(W,X)\cdot r}{l_s} \right\rceil$$

Each complete cycle also involves a series of $Y-2$ merges which start from a list element with a single group and finish with an element equal in size to the elements in the list at the beginning of the cycle. Where i is the size of the elements at the beginning of the cycle, the amount of space consumed by each "element build" is:

$$Ebuild(i,W,X) = \sum_{j=1}^{i} \left\lceil \frac{i\cdot g(W,X)\cdot r}{l_s} \right\rceil$$

Finally, each cycle also includes one complete "build" of an element equal in size to two merged list elements, plus one.

Thus, the amount of space consumed in the $Q(V,W,X,Y)$ complete (full) cycles is:

$$Full(V,W,X,Y) = \sum_{i=1}^{Q(V,W,X,Y)} (Y-1)\cdot Emerge(2^i-1,W,X) + (Y-2)\cdot Ebuild(2^i-1,W,X) + Ebuild(2^{i+1}-1,W,X)$$

To complete the computation of the amount of disc space consumed we must account for the amount of space consumed in the last incomplete (partial) duplication cycle, if there is one.

Let Z be the number of flushes in the last incomplete cycle.

$$Z = Flushes(V,W,X) - ((2^{Q(V,W,X,Y)+1}-2)\cdot Y + Y)$$

If $Z=0$, then the amount of consumed space is zero, otherwise $Z>0$ and we compute the consumed space for the incomplete cycle in the following manner. If $Z > (Y-2)(2^{Q(V,W,X,Y)+1}) + 1$, then all the $Y-1$ merges and the $Y-2$ builds have occurred in the cycle, what remains is a build of size $Z-1-(Y-2)(2^{Q(V,W,X,Y)+1})$.

The amount of space consumed in that case is then:

$$Part(V,W,X,Y) = (Y-1)\cdot Emerge(2^{Q(V,W,X,Y)+1}-1,W,X) + (Y-2)\cdot Ebuild(2^{Q(V,W,X,Y)+1}-1,W,X) + Ebuild(Z-1-(Y-2)\cdot 2^{Q(V,W,X,Y)+1},W,X)$$

Otherwise, we know that at most $Y-1$ merge and $Y-2$ builds have occurred. Let A be the number of merges and builds.

$$A = \left\lfloor \frac{Z-1}{2^{Q(V,W,X,Y)+1}} \right\rfloor$$

Then the consumed space is:

$$Part(V,W,X,Y) = Emerge(2^{Q(V,W,X,Y)+1}-1,W,X)\; A\cdot(Emerge(2^{Q(V,W,X,Y)+1}-1,W,X) + Ebuild(2^{Q(V,W,X,Y)+1}-1,W,X)) + Ebuild(Z-1-A\cdot(2^{Q(V,W,X,Y)+1}),W,X)$$

If the number of flushes is greater than Y, then the total space consumed by the flushes to a single hash bucket is:

$$B\_occupied(V,W,X,Y) = Y\cdot \left\lceil \frac{g(W,X)\cdot r}{l_s} \right\rceil + Full(V,W,X,Y) + Part(V,W,X,Y)\ sectors$$

otherwise,

$$B\_occupied(V,W,X,Y) = Flushes(V,W,X)\cdot \left\lceil \frac{g(W,X)\cdot r}{l_s} \right\rceil$$

The total space consumed by V insertions to the file is as for the first algorithm simply the product of the number of hash buckets X and $B\_occupied(V,W,X,Y)$.

A comparison of the storage consumption rates of the two algorithms is given later in section 5.

4. Expected Case Steady State Analysis For Flush Group Size

The BHash buffering process has a complex behavior. A Markov Chain model of the process has been developed [9] that produces exact results for the flush group size g, given the buffer size W and number of hash buckets X. It is omitted for brevity. The derivation of an expected case *steady state* analysis of the flush size is included below.

4.1. More Than Half the Buckets Occupied $X < 2W$

The steady state form of a buffer when more than half of the buckets are occupied is illustrated in Figure 3. We shall refer to the group of records with constitute a "step" in the staircase as a *bucket set*. The diagram shows the staircase arrangement of k bucket sets in a full buffer just before a bucket with g records is flushed.

Under the steady state assumption, we expect that the bucket sets will maintain their equal sizes (same number of members) and that the g new records that will be inserted (to replace the ones that were flushed) will be randomly assigned to the X buckets by the hashing function. Thus, $l_1 = l_2 = \cdots = l_k = \frac{X}{g}$.

Using this property and the constraint that the sizes of the bucket sets must sum to the total number of buckets, we can find an expression for the number of bucket sets k.

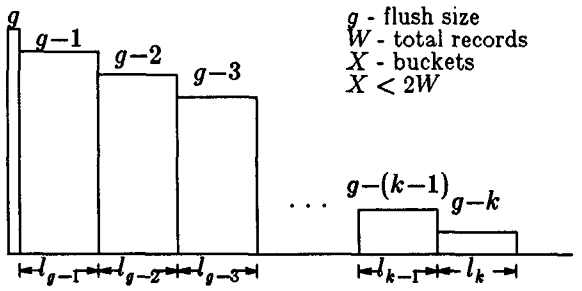

Figure 3. Bucket Sets Before S. State Flush for $X < 2W$

$$X = l_1 + l_2 + \cdots + l_k + 1 = k\left\lceil\frac{X}{g}\right\rceil + 1$$

solving for k, we find $k = g\,\dfrac{X-1}{X}$

To derive g we can use the constraint that just before the flush, the sum of the buffered records and the one that triggered the flush, must be equal the capacity W plus one.

$$W + 1 = g + l_1(g-1) + \cdots + l_k(g-k)$$

Each of the l_i's equals $\dfrac{X}{g}$ and the sum $(g-1) + \cdots + (g-k)$ is equal to $\dfrac{2g-k-1}{2}k$, thus we have:

$$W + 1 = g + \frac{X}{g}\cdot\frac{(2g-k-1)}{2}k$$

Substituting for k,

$$W + 1 = g + \frac{X}{g}\cdot\frac{(2g - g\frac{(X-1)}{X} - 1)}{2}\cdot g\frac{(X-1)}{X}$$

$$W + 1 = g + g(X-1) - g\frac{(X-1)^2}{2X} - \frac{(X-1)}{2}$$

rearranging terms,

$$W + 1 + \frac{(X-1)}{2} = g(1 + (X-1) - \frac{(X-1)^2}{2X})$$

$$W + 1 + \frac{(X-1)}{2} = g(\frac{X}{2} + 1 - \frac{1}{2X})$$

and solving for g,

$$g = \frac{W + 1 + \frac{X-1}{2}}{\frac{X}{2} + 1 - \frac{1}{2X}} = \frac{2W + X + 1}{X + 2 - \frac{1}{X}}$$

For W and X both much greater than 1, we have $g \approx \dfrac{2W + X}{X} = \dfrac{2W}{X} + 1$

4.2. Less than Half Occupied $X > 2W$

When the number of buckets is such that more than half of them are empty when the buffer is full, the set of bucket sets will assume the steady state arrangement shown in Figure 4.

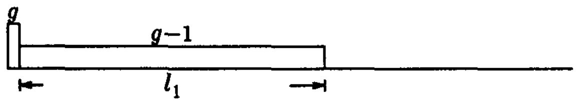

Figure 4. Bucket Sets Before S. State Flush for $X > 2W$

As before we have $l_1 = \dfrac{X}{g}$ and $l_g = 1$, and the number of records must sum to $W + 1$,

$$W + 1 = (g-1)\frac{X}{g} + g$$

solving for g, we find $g = \dfrac{X}{X + g - W - 1}$

For values of X and W much larger than 1 or 2 (the value of g), we have, $g \approx \dfrac{X}{X - W}$

4.3. Boundary Point Comparison

We observe that at the boundary point dividing the two cases, $X = 2W$, for values of W and X much greater than one, the values of the expected case steady state flushing size computed for each case should agree and be equal to two.

Using the first approach with $X = 2W$ we compute $g = \dfrac{2W}{2W} + 1 = 2$ and using the second approach, $g = \dfrac{2W}{2W - W} = \dfrac{2W}{W} = 2.$

4.4. Simulation Comparison

We compared our analysis with the results of simulations. The graph in Figure 5. plots the values for the flush size resulting from simulations of the buffering process for different buffer sizes. The graph in Figure 6. plots the values computed using the expected case analysis.

5. Comparison of Different Levels of Sequentiality

The graphs in Figures 7. and 8. compare the expected number of consumed sectors computed using the formulae derived in section 3 with the results of simulations. The graph in Figure 7. shows the consumption for the full merge algorithm while the graph in Figure 8. shows the consumption for the partial merge algorithm. Each line in the respective graphs corresponds to a different value of Y (the partition limit).

As can be seen from the graphs, the analysis and simulation are in excellent agreement. The graphs also show that the highest level of sequentiality ($Y = 1$) has a dramatically higher level of disc consumption than the lower levels for both the full and partial merging algorithms. The rate of disc space consumption is significantly reduced for lower levels of sequentiality ($Y > 1$).

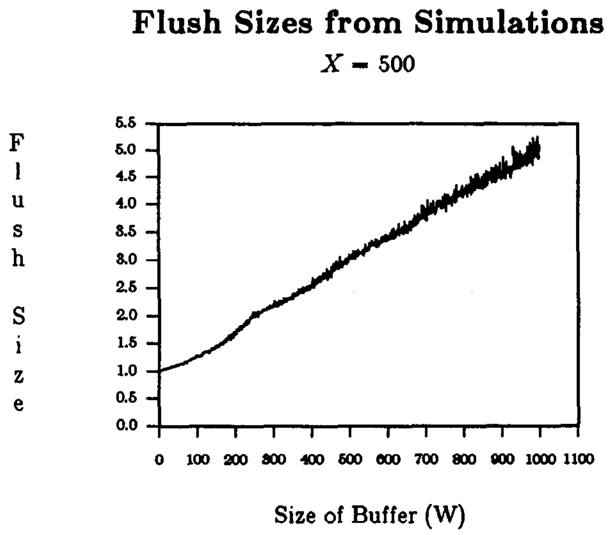

Figure 5. Example Flush Sizes From Simulations

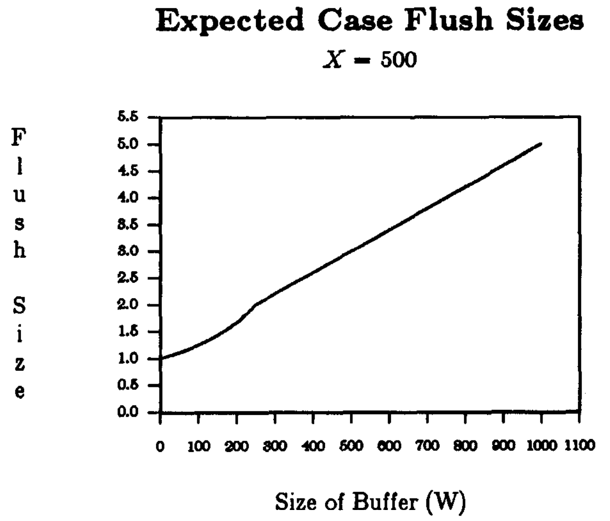

Figure 6. Example Expected Case Flush Sizes

The consumption rates of the two algorithms are compared in the graph in Figure 9. For $Y = 1$, the two algorithms are equivalent and have identical consumption rates. For larger values of Y, the consistently lower consumption rate of the partial merging algorithm is evident. This lower rate, however, comes at the price of higher average retrieval cost. The partial merging algorithm with a value of $Y = 2$ seems to be a good compromise between consumption rates and retrieval performance.

Results such as these will be very useful in the physical design phase of a database. Using the formulae derived in this paper, a designer can explore the impact of design decisions and constraints. For example, if an upper limit on the amount of wasted space is specified as a constraint, one can develop estimates for the expected retrieval performance (response time). And similarly, given a constraint on the maximum number of seeks on the WORM discs allowed by an application, one can obtain an estimate for the expected space overhead required to meet that constraint.

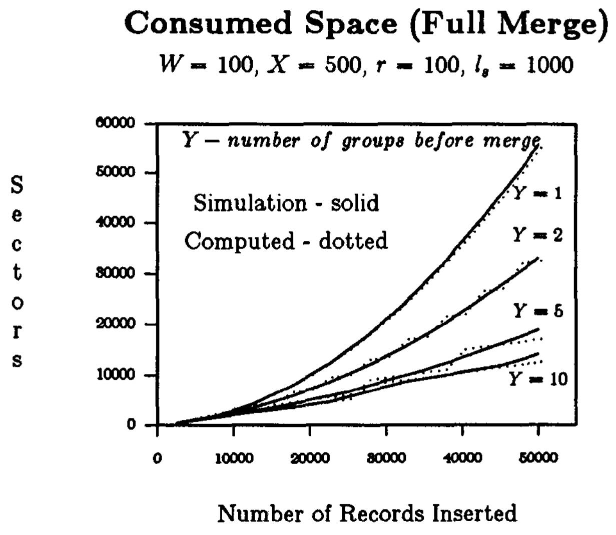

Figure 7. Consumed Space for Full Merge

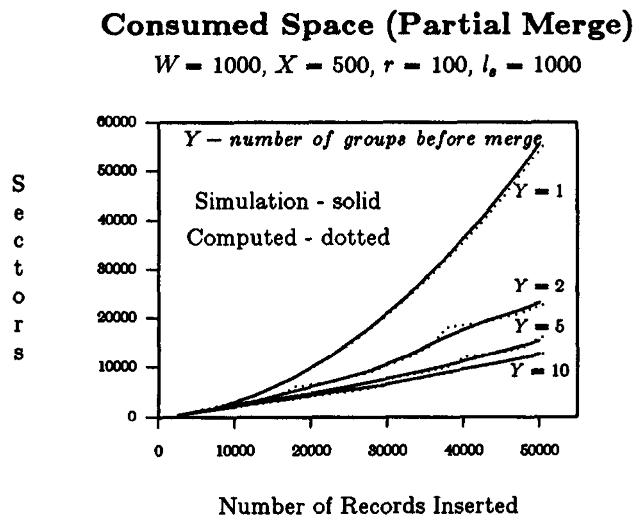

Figure 8. Consumed Space for Partial Merge

6. Summary

In this paper, we proposed a class of access methods that use rewritable storage for temporary buffering of insertions to data sets stored on WORM optical discs and examined the relationship between the expected retrieval performance and the utilization of disc storage space for such organizations. For the sake of concreteness of description and space limitations we concentrated on just one organization, buffered hashing (BHash).

We showed that the cost of guaranteeing high retrieval performance from a WORM optical disc is a dramatic increase in the expected amount of wasted storage space on the disc owing to data duplication. The description of the method, the development of the model to study and the derivation of analytic formulae describing the trade off between retrieval performance and WORM disc space utilization constitute the major contributions of

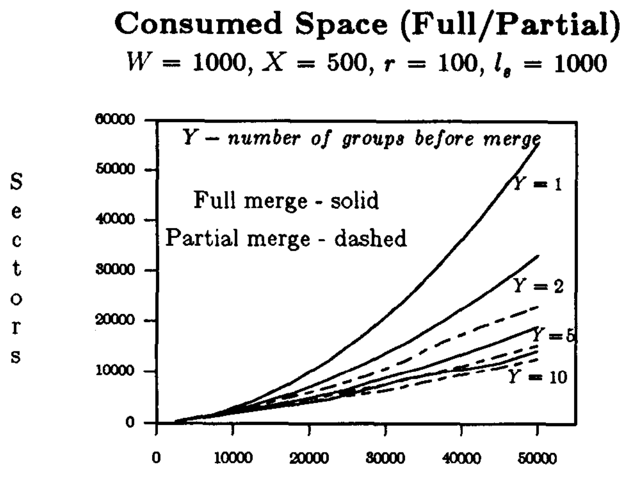

Figure 9. Consumed Space Full/Partial Merging

this paper.

Our analysis will be useful in the physical database design phase of applications that use WORM optical discs, where it will allow one to examine and explore the implications of design constraints and decisions on the expected retrieval performance and disc space consumption. For example, increasing the size of the rewritable buffer reduces the storage consumption rate; our analysis allows one to determine the size necessary to meet performance objectives.

Future directions for research include examining buffer flushing strategies that reduce sector fragmentation even further by not flushing incomplete sectors and models of buffered file organizations that allow arbitrary probability distributions for the assignment of records to buckets. This may be useful for file ISAM organizations where the insertion probability of each bucket is not uniform [9]. ISAM organizations may have advantages over B-tree organizations for WORM optical disc storage [8]. We are currently investigating these extensions.

7. References

[1] Batory, D.S., "Optimal File Designs and Reorganization Points", *ACM Transactions on Database Systems*, Vol. 7, No. 1, pp. 60-81.

[2] Batory, D.S., Gotlieb, C.C., "A Unifying Model of Physical Databases", *ACM Transactions on Database Systems*, Vol. 7, No. 4, pp. 509-539.

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

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

[5] Christodoulakis, S., Vanderbroek, J., Li, J., Li, T., Wan, S., Wang, Y., Papa, M., Bertino, E., "Development of a Multimedia information system for an office environment", *Proc. of VLDB '84*, Aug 1984, pp. 261-270.

[6] Christodoulakis, S., "Analysis of Retrieval Performance for Records and Objects Using Optical Disk Technology", *ACM Transaction on Database Systems*, Vol. 12, No. 2, June 1987, pp. 137-169.

[7] Christodoulakis, S., Elliott, K., Ford, D.A., Hatzilemonias, K., Ledoux, E., Leitch, M., Ng, R., "Optical Mass Storage Systems and their Performance", *IEEE Database Engineering*, March 1988.

[8] Christodoulakis, S., Ford, D.A., "File organizations and Access Methods for CLV Optical Disks", Technical Report CS-88-21, Department of Computer Science, University of Waterloo, March 1988. (Also to appear in *Proceedings of SIGIR*, Cambridge, Mass. June 1989).

[9] Christodoulakis, S., Ford, D.A., "Sequentiality of access and Disc Space Consumption on WORM optical Discs", *Technical Report CS-88-47*, Department of Computer Science, University of Waterloo, December 1988.

[10] Comer, D., "The Ubiquitous B-tree", *Computing Surveys*, Vol. 11, No. 2, pp. June 1979.

[11] Easton, M.C., "Key-sequence data sets on indelible storage", *IBM J. Res. Develop*, Vol. 30, No. 3, (May 1986).

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

[13] "HITFILE 650 Optical Disk Filing System", *Hitachi Review*, Vol. 36 (1987), No. 4, pp. 213-220.

[14] Hsiao, D., Harary, F., "A formal system for information retrieval from files", *Communications of the ACM*, Vol. 13, No. 2, pp. 67-73; Corrigendum, *Communications of the ACM*, Vol. 13, No. 4, pp. 266.

[15] Katz, R.H., Wong, E., "Resolving Conflicts in Global Storage Design through Replication", *ACM Transactions on Database Systems*, Vol. 8, No. 1, pp. 110-135.

[16] Lomet, D., Salzberg, B., "Access Methods for Multiversion Data", *Technical Report NU-CCS-88-7*, Northeastern University, December 16, 1988.

[17] Larson, P.-Å., "Analysis of index-sequential files with overflow chaining", *ACM Transactions on Database Systems*, Vol. 6, No. 4, pp. 671-680, 1981.

[18] Litwin, W., "Linear Hashing: A New Tool for File and Table Addressing", In Proc. 6th Conf. on Very Large Databases, ACM, New York, pp. 212-223, 1980.

[19] Lomet, D., "Partial Expansions for File Organizations with an Index", *ACM Transactions on Database Systems*, Vol 12, No. 1, pp. 65-84.

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

[21] "Optifile: Technical Reference Manual", KOM Inc. Ottawa, Version 1.0, (February 1986).

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

[23] Reed, I.S., Solomon, G. "Polynomial Codes over Certain Finite Fields", *J. Soc. Indus. Appl. Math., 8:3000-304, 1960.*

[24] Salzberg, B.J., *File Structures*, Prentice Hall, Englewood Cliffs, N.J., 1988

[25] Severance, D.G., "A Parametric Model of Alternative File Structures", *Information Systems*, Vol. 1, No. 2, pp. 51-55.

[26] Stonebraker, M., "The Design of the POSTGRES Storage System", *Proc. 13th VLDB Conference*, Brighton, 1987, pp. 289-300.

[27] Teorey, Fry, *Design of Database Structures*, Prentice Hall, 1982.

[28] Thomas, D., "A High-Speed Data Management System with On-Line Optical Disk Storage", *IEEE 1985 Symposium On Mass Storage Systems*, pp. 38-42.

[29] de Vos, J., "Megadoc, a modular system for electronic document handling", *Philips Technical Review*, **39** 329.

[30] Wiederhold, G., *File Organization for Database Design*, McGraw-Hill, 1987.

[31] Yao, S.B., "An Attribute Based Model for Database Access Cost Analysis", *ACM Transactions on Database Systems*, Vol. 2, No. 1, March 1977, pp. 45-67.