Printing Requisition/Graphic Services

26000

1. Please complete unshaded areas on form as applicable.
2. Distribute copies as follows: White and Yellow to Graphic Services. Retain Pink Copies for your records.
3. On completion of order the Yellow copy will be returned with the printed material.
4. Please direct enquiries, quoting requisition number and account number, to extension 3451.

TITLE OR DESCRIPTION: Sequentiality of access and Disc Space Consumption on WORM Optical Discs CS-88-47

DATE REQUISITIONED: Dec. 20/88	DATE REQUIRED: ASAP	ACCOUNT NO.: 1 2 6 8 9 4 1 4 1
REQUISITIONER– PRINT: S. Christodoulakis [illegible]	PHONE: 4452	SIGNING AUTHORITY: [illegible]

MAILING INFO – NAME: Sue DeAngelis DEPT.: C.S. BLDG. & ROOM NO.: DC 2314 [✓ DELIVER] PICK-UP

Copyright: I hereby agree to assume all responsibility and liability for any infringement of copyrights and/or patent rights which may arise from the processing of, and reproduction of, any of the materials herein requested. I further agree to indemnify and hold blameless the University of Waterloo from any liability which may arise from said processing or reproducing. I also acknowledge that materials processed as a result of this requisition are for educational use only.

NUMBER OF PAGES: 28 NUMBER OF COPIES: 20

TYPE OF PAPER STOCK: [✓ BOND] NCR ___ PT. [✓ COVER] BRISTOL [✓ SUPPLIED]

PAPER SIZE: [✓ 8½ x 11] 8½ x 14 11 x 17

PAPER COLOUR: [✓ WHITE] INK: [✓ BLACK]

PRINTING: 1 SIDE ___ PGS. [✓ 2 SIDES] ___ PGS. NUMBERING FROM TO

BINDING/FINISHING: 3 down left side [✓ COLLATING] [✓ STAPLING] ___ PUNCHED PLASTIC RING

FOLDING/PADDING CUTTING SIZE

Special Instructions

Math fronts and backs enclosed.

COPY CENTRE OPER. NO. BLDG. MACH. NO.

DESIGN & PASTE-UP

	OPER. NO.	TIME	LABOUR CODE
			D01
			D01
			D01

TYPESETTING

	QUANTITY			
PAP00000				T01
PAP00000				T01
PAP00000				T01

PROOF

PRF				
PRF				
PRF				

NEGATIVES	QUANTITY	OPER. NO.	TIME	LABOUR CODE
FLM				C01
FLM				C01
FLM				C01
FLM				C01
FLM				C01
PMT				
PMT				C01
PMT				C01
PMT				C01
PLATES				
PLT				P01
PLT				P01
PLT				P01
STOCK				
				001
				001
				001
				001
BINDERY				
RNG				B01
RNG				B01
RNG				B01
MIS00000				B01

OUTSIDE SERVICES

$ ___ COST

TAXES – PROVINCIAL FEDERAL

GRAPHIC SERV. OCT. 85 482-2

To Stavros

From Sue DeAngelis

Date Dec. 12

memo

University of Waterloo

Would you please sign the attached print req. & indicate how many copies you would like printed.

Thanks.

Sue

I Want 20 Copies

67991

Printing Requisition / Graphic Services

1. Please complete unshaded areas on form as applicable.
2. Distribute copies as follows: White and Yellow to Graphic Services. Retain Pink Copies for your records.
3. On completion of order the Yellow copy will be returned with the printed material.
4. Please direct enquiries, quoting requisition number and account number, to extension 3451.

TITLE OR DESCRIPTION: CS-88-47

DATE REQUISITIONED: May ~~5~~ 8 | DATE REQUIRED: ASAP | ACCOUNT NO.: 1 2 6 8 9 4 1 4 1

REQUISITIONER - PRINT: S. CHRISTODOULAKIS | PHONE: 4452 | SIGNING AUTHORITY: S. DeAngelis / S. Christodoulakis

MAILING INFO - NAME: S. DE ANGELIS | DEPT.: C. S. | BLDG. & ROOM NO.: DC 2314 | [✓ DELIVER] PICK-UP

NUMBER OF PAGES: 29 | NUMBER OF COPIES: 20

TYPE OF PAPER STOCK: [✓ BOND] NCR ____ PT. [✓ COVER] BRISTOL [✓ SUPPLIED] ____

PAPER SIZE: [✓ 8½ x 11] 8½ x 14 11 x 17 ____

PAPER COLOUR: [✓ WHITE] ____ INK: [✓ BLACK] ____

PRINTING: 1 SIDE ____ PGS. [✓ 2 SIDES] ____ PGS. NUMBERING FROM ____ TO ____

BINDING/FINISHING: 3 down left side [✓ COLLATING] [✓ STAPLING] HOLE PUNCHED ____ PLASTIC RING

FOLDING/PADDING | CUTTING SIZE

Special Instructions

Math fronts + backs enclosed.

COPY CENTRE — OPER. NO. | BLDG. | MACH. NO.

DESIGN & PASTE-UP

OPER. NO.	TIME	LABOUR CODE
		D01
		D01
		D01

TYPESETTING

	QUANTITY			
PAP000000				T01
PAP000000				T01
PAP000000				T01

PROOF

PRF				
PRF				
PRF				

	QUANTITY	OPER. NO.	TIME	LABOUR CODE
NEGATIVES				
FLM				C01
FLM				C01
FLM				C01
FLM				C01
FLM				C01
PMT				
PMT				C01
PMT				C01
PMT				C01
PLATES				
PLT				P01
PLT				P01
PLT				P01
STOCK				
				001
				001
				001
				001
BINDERY				
RNG				B01
RNG				B01
RNG				B01
MIS000000				B01

OUTSIDE SERVICES

$ ____ COST

TAXES — PROVINCIAL ☐ FEDERAL ☐

Sequentiality of access and Disc Space Consumption on WORM Optical Discs

Stavros Christodoulakis
Daniel Alexander Ford

Research Report CS-88-47
December, 1988

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

Sequentiality of access and Disc Space Consumption 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 now beginning to compete directly with magnetic disc technology. WORM optical discs in particular, which permanently register information on the disc surface, have significant advantages for applications that are mainly archival in nature but require the ability to perform frequent on-line insertions.

In this paper we propose a class of access methods that employ a rewritable storage for 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 tradeoff as one of fast sequential retrieval of the contents of a block versus waisted space due 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. We then provide exact and 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 now 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; 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 employing 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, 24, 26].

On both magnetic and optical discs, 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 permits its sequential retrieval. Movements of the access mechanisms or seeks on both types of discs take a relatively 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 very 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 very 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.

On both disc types, physical reorganization of a data set can restore or maintain its physical contiguity. Because Magnetic discs are rewritable, this is an easy task, 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 [16, 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. Registration is also performed in units of complete disc sectors (not parts of sectors) which, because of a sophisticated error correcting code [23], cannot be changed once they have been written. The error correcting code associated with each sector prevents modifications by either "correcting" a 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 sequentiallity 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 a very important means of achieving high retrieval performance from a WORM optical disc.

In practice, it might not always be necessary to perform the reorganization each time changes or additions decrease physical contiguity, a lower level of sequentiallity 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 tradeoff 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 sequentiallity 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 tradeoff 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 an example of a buffered hash file organization (BHash) in detail. 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. We develop, and analyze, and prove the convergence of a Markov model that allows us to derive exact values of the flush size, and we validate our results against simulation. In section 5, we present less expensive to compute closed form formulae for the expected flush size based on an expected value analysis. We compare our results against simulations and show that they are in excellent agreement. In section 6, we use the analytic results derived throughout the paper to derive a tradeoff between the sequential access cost and the storage space consumption on the WORM disc. This tradeoff will be of particular use in the physical database design phase. Finally, in section 7 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 employs 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. Firstly, it can reduce sector fragmentation which 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" in order 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 in order to optimize long term storage and retrieval objectives for WORM storage. Multiway tree organizations, for example, require large nodes (blocks) to guarantee a small tree height. The node contents must be nearby in WORM storage in order to guarantee fast node access.

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).

2.1. Buffered Hashing (BHash)

A buffered hash file organization is an excellent vehicle for examining the tradeoff between maintaining a specified level of sequentiallity 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 which 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 employ 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 employ 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 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 W be the number of records that can be buffered. 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 a bucket can be divided into. When adding a group of records flushed from the buffer to the list would exceed this limit, the new group and all of the groups in the list are merged together to form a new single group. This group is then stored in a new location on the optical disc and becomes the first member of a new list for the bucket.

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.

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 V, the number of records inserted, W, the size of the buffer and g, the expected size of each group flushed from the buffer. The later is constant and a function of the size of the buffer and the number of hash buckets. Its derivation is left to a later section. The first flush from the buffer occurs after $W + 1$ record insertions, and the next flushes occur every $g(W,X)$ record insertions.

Thus, if $V > W + 1$, then:

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

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

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, which occurs after the $Y+1$'th flush. An example sequence of merge operations for $Y = 3$, for a single hash bucket, is depicted in Figure 1.

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.).

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 $1 + i \cdot Y$.

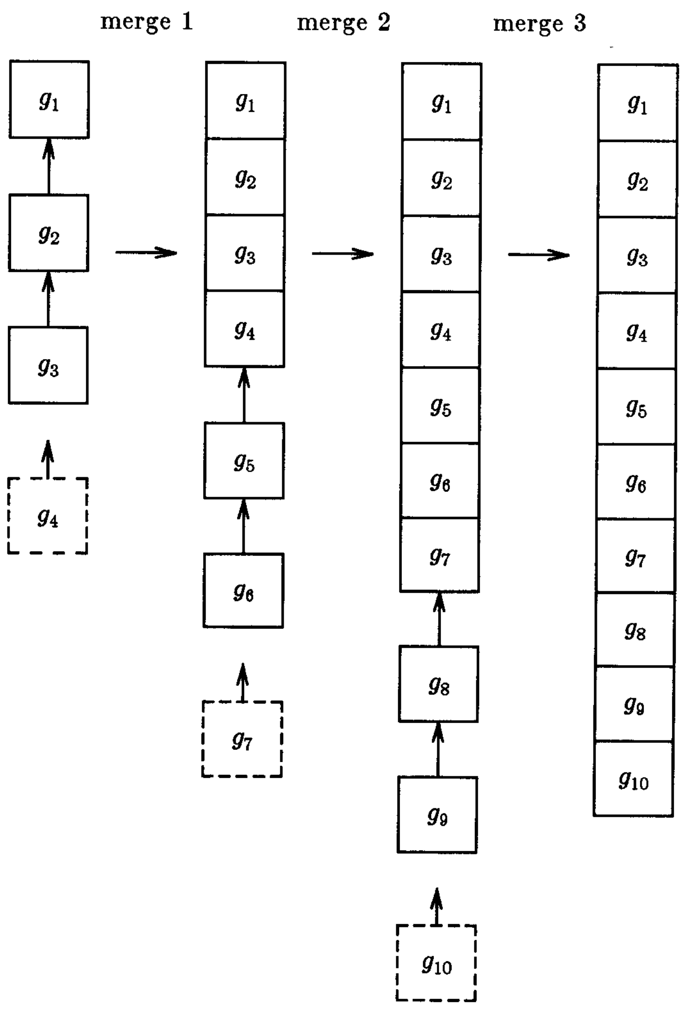


Figure 1. Bucket Merges for $Y = 3$

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:

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

4. Analysis of Flush Group Size

The BHash buffering process has a complex behavior, but by using a Markov Chain model of the BHash buffering process, we can determine the exact value of the expected flush group size g for a given buffer size W and number of hash buckets X.

4.1. Markov Chain Analysis

To compute the expected value of the flushed group size g the buffering process can be modeled as a finite, irreducible, aperiodic Markov chain. The states in the chain will represent the different possible distributions of records in the buffer to the hash buckets. For our analysis, the important characteristic of each distribution will be its break down in terms of the different buffered bucket sizes (i.e. the number of records buffered for a bucket), not the particular assignment of records by the hash function to specific buckets. In short, a state will record the exact number of buckets having each of the possible sizes. For example, distributions that place records such that the same <u>number</u> of buckets (not necessarily the same buckets) are empty, the same <u>number</u> have exactly one assigned record, etc., will all be represented in the chain by the same state. This approach reduces the state space of a Markov chain representation.

Transitions in the Markov chain will represent the insertion of single records into the buffer; each insertion will change the size of one bucket and alter the overall size distribution accordingly.

The flush group size g can be obtained from the states in the Markov chain that represent the distribution of records in a full buffer; the size of the largest bucket or buckets in such states will determine the size of the flushed group. The insertion of the record which triggers the flush from those states must also be considered however, as it may or may not be hashed to any of the buckets about to be flushed, and hence may or may not add to the flush size.

4.2. State Representation

We can represent the states of the Markov chain with a vector of numbers that records their particular bucket size distribution. The positions in the vector will be weighted by the allowable sizes of the buckets; the rightmost position will have weight zero and positions to the left will increase in weight incrementally; leading zeros in the vector can be omitted. The sum of the digits in the vector will always sum to the number of buckets being buffered, and the sum of the digits in the vector weighted by the bucket sizes will equal the number of records in the buffer for that state.

For example, the state S of a buffer containing two records distributed to two of three buckets would be represented by S_{21}. The "1" in the vector indicates that one bucket has no buffered records and the "2" indicates that two buckets have exactly one buffered record each. Similarly, if both records were assigned by the hash function to a single bucket, then the state would be S_{102}, or "1" bucket with two records, "0" buckets with one record and "2" buckets with zero assigned records. The state representing the empty buffer would simply be S_3, indicating three buckets with zero records.

As mentioned previously, transitions occur when records are inserted and each insertion causes exactly one bucket to increase in size. For example, there are two possible transitions from the state S_{102}. The first occurs if the inserted record is assigned to the single bucket with two records, causing the next state to be S_{1002}, and the second occurs, if the inserted record is assigned to one of the two empty buckets causing the next state to be S_{111}.

The transition probabilities are taken from the ratios between the elements of the state vector and the total number of buckets. In the example above, the probability of the transition $S_{102} \rightarrow S_{1002}$, is the probability of the inserted record being assigned (hashed) to the single bucket with one record, or 1/3. Similarly, the probability of the transition $S_{102} \rightarrow S_{111}$, is 2/3.

4.3. Parametric State and Transition Description

Definition 1: Given a state $S_{k_W, \ldots, k_2, k_1, k_0}$, then it is a ***non-flushing*** state if $\sum_{i=0}^{W} i \cdot k_i < W$, otherwise, $\sum_{i=0}^{W} i \cdot k_i = W$ and it will be a *flushing* state.

From a non-flushing state $S_{k_W, \ldots, k_j, \ldots, k_2, k_1, k_0}$, there will be transitions to the states $S_{k_W, \ldots, k_{j+1}+1, k_j-1, \ldots, k_2, k_1, k_0}$, for each $k_j \neq 0$, each with probability $\frac{k_j}{X}$.

From a flushing state $S_{k_W, \ldots, k_q, \ldots, k_2, k_1, k_0}$, where $k_q \neq 0$, $k_i = 0$ for $i > q$ (i.e. q is the size of the largest bucket in the buffer), there will be transitions to the states $S_{k_W, \ldots, k_q-1, \ldots, k_{j+1}+1, k_j-1, \ldots, k_2, k_1, k_0+1}$, for each $k_j \neq 0$, $q > j$, each with probability $\frac{k_j}{X}$, and to the state $S_{k_W, \ldots, k_q-1, \ldots, k_2, k_1, k_0+1}$, with probability $\frac{k_q}{X}$.

Transitions to the state $S_{k_W, \ldots, k_j, \ldots, k_2, k_1, k_0}$ will occur from the non-flushing states $S_{k_W, \ldots, k_j-1, k_{j-1}+1, \ldots, k_2, k_1, k_0}$, for each $k_j > 0$, $j > 0$ with probability $\frac{k_{j-1}+1}{X}$. Also, there will be transitions from flushing states, if $k_0 \neq 0$ and q, the size of the largest bucket in the buffer ($k_q \neq 0$, $k_i = 0$, for $i > q$), is less than or equal to $g = W + 1 - \sum_{i=0}^{W} i \cdot k_i$, the size of a possible flush (i.e. if a flushing transition to particular state is to exist, then the largest bucket size q in the state cannot be greater than the maximum size of the flush; the maximum size is simply $W + 1$ less the number of records in the buffer). If so, the flushing states from which transitions will occur from are $S_{k_W, \ldots, k_g+1, \ldots, k_j-1, k_{j-1}+1, \ldots, k_2, k_1, k_0-1}$, for each $k_j \neq 0$, $j > 0$, each with probability $\frac{k_{j-1}+1}{X}$, and, if $g > q$, also from the flushing state $S_{k_W, \ldots, k_{g-1}+1, \ldots, k_2, k_1, k_0-1}$, with probability $\frac{k_{g-1}+1}{X}$.

4.4. Conditional Probabilities

The conditional probabilities $p_{k_W, \ldots, k_2, k_1, k_0}$ of the states are calculated as follows. If the state does not have transitions from flushing states (i.e. $q > g$ above), then its conditional probability is:

$$p_{k_W, \ldots, k_2, k_1, k_0} = \sum_{\substack{k_j \neq 0 \\ j > 0}} \frac{k_{j-1} + 1}{X} \cdot p_{k_W, \ldots, k_j - 1, k_{j-1} + 1, \ldots, k_2, k_1, k_0}$$

If the state does have transitions from flushing states (i.e. $k_0 \neq 0$, $g \geq q$), and if $g > q$ the conditional probability is:

$$\begin{aligned} p_{k_W, \ldots, k_2, k_1, k_0} = {} & \sum_{\substack{k_j \neq 0 \\ j > 0}} \frac{k_{j-1} + 1}{X} \cdot p_{k_W, \ldots, k_j - 1, k_{j-1} + 1, \ldots, k_2, k_1, k_0} \\ & + \frac{k_{g-1} + 1}{X} \cdot p_{k_W, \ldots, k_{g-1} + 1, \ldots, k_2, k_1, k_0 - 1} \\ & + \sum_{\substack{k_j \neq 0 \\ j > 1}} \frac{k_{j-1} + 1}{X} \cdot p_{k_W, \ldots, k_g + 1, \ldots, k_j - 1, k_{j-1} + 1, \ldots, k_2, k_1, k_0 - 1} \\ & + \left[(if\ k_1 \neq 0) \frac{k_0}{X} \cdot p_{k_W, \ldots, k_g + 1, \ldots, k_1 + 1, k_0} \right] \end{aligned}$$

Otherwise, $g = q$ and the conditional probability is:

$$\begin{aligned} p_{k_W, \ldots, k_2, k_1, k_0} = {} & \sum_{\substack{k_j \neq 0 \\ j > 0}} \frac{k_{j-1} + 1}{X} \cdot p_{k_W, \ldots, k_j - 1, k_{j-1} + 1, \ldots, k_2, k_1, k_0} \\ & + \sum_{\substack{k_j \neq 0 \\ j > 1}} \frac{k_{j-1} + 1}{X} \cdot p_{k_W, \ldots, k_g + 1, \ldots, k_j - 1, k_{j-1} + 1, \ldots, k_2, k_1, k_0 - 1} \\ & + \left[(if\ k_1 \neq 0) \frac{k_0}{X} \cdot p_{k_W, \ldots, k_g + 1, \ldots, k_1 + 1, k_0} \right] \end{aligned}$$

4.5. Expected Flush Size from Markov Model

We can compute the exact expected flush size from the stationary (steady state) probabilities of the flushing states in the Markov chain. The basic idea is to form a sum of the flush sizes associated with each flushing state weighted by the probability of each state in relation to the probabilities of the other flushing states.

Thus, the expected flush size, where q is as above, the largest number of records assigned to any bucket in the buffer, and P_F is the set of stationary probabilities for the flushing states, is given by:

$$g(W,X) = \frac{\sum\limits_{p_{k_W,\ldots,k_2,k_1,k_0} \epsilon P_F} p_{k_W,\ldots,k_2,k_1,k_0} \left[(q+1)\frac{k_q}{X} + q\frac{X-k_q}{X} \right]}{\sum\limits_{p_{k_W,\ldots,k_2,k_1,k_0} \epsilon P_F} p_{k_W,\ldots,k_2,k_1,k_0}}$$

4.6. Proof of Steady State Convergence

We can show that the Markov chain will converge to a steady state by showing that it is *finite, irreducible* and *aperiodic.*

Theorem 1: The Markov chain representation for the buffering process is finite.

Proof: The number of records that can be buffered is fixed and finite, and as such, there can only be a finite number of bucket size distributions. Hence, the number of states and transitions in the chain is finite.

Theorem 2: The Markov chain representation for the buffering process is irreducible.

Proof: A Markov chain is *irreducible* if it contains no closed set of states other than the set of all states. In other words, each state can be reached by any other through a series of transitions. To show this, it will suffice to demonstrate that the state representing the empty buffer can reach any other state and that any state can reach the empty buffer state. Since we are free to choose the transitions we as we like, the first condition can be shown to be true by simply assigning incoming records (selecting transitions) to buckets to create the distribution corresponding to the state to be reached. Similarly for the second condition, for any given state, if we continually assign the incoming records only to the bucket with the most records, then eventually there will be only one bucket with assigned records, the other buckets will have each been flushed in turn when the buffer reached its capacity. When that last bucket is flushed the empty buffer state will have been reached.

To complete our proof of steady state convergence, we need to show that the Markov chain is *aperiodic*. To prove that a chain is aperiodic we must show that none of its states are periodic. A *periodic* state is one which the chain can only enter periodically, for example, every n'th transition for some value of $n > 1$.

To show that the states in a chain are aperiodic, we need to show that the lengths of the return paths of each state have a greatest common divisor of at most one. The basic idea behind this requirement is that when it is true, then when the chain is in a particular state we could be in that state again after any of a suitably large number of transitions by selecting an appropriate series of return paths. More formally, a state S_j is an *aperiodic* state if the greatest common divisor (g.c.d.) of the set $\{n \mid P^{(n)}{}_{jj} > 0,\, n \geq 1\}$ is equal to 1, where $P^{(n)}{}_{jj}$ is the probability of starting from state S_j and being in state S_j after n transitions.

For an irreducible chain it is sufficient to show that at least one of the states is aperiodic since that that would imply that the rest can also be reached after any of a large number of transitions. To prove that our Markov chain is aperiodic we will show that the state representing the empty buffer has two return paths with lengths that have a greatest common divisor of one and then use the fact that the chain is irreducible (Theorem 2) to conclude that the chain is aperiodic.

Lemma 1: If $B \geq 2$ and $W \geq 2$, then there exists two return paths from the state S_W (empty buffer) of lengths β and $\beta + 1$ that lead back to the state S_W.

Proof: Consider the path from state S_W to flushing state $S_{k_{W-1}=1,0,\ldots,0,1,W-2}$ (or if $W = 2$, state $S_{2,0}$) in W transitions, then to state $S_{1,W-1}$ in one transition, from that state to the flushing state $S_{k_W=1,0,\ldots,0,W-1}$ in $W-1$ transitions and then from there back to state S_W with one more transition for a total number of $W + 1 + W - 1 + 1 = 2W + 1$ transitions. Also, consider the path from state S_W to flushing state $S_{k_W=1,0,\ldots,0,W-1}$ in W transitions, to the state S_W again in one transition, from there again to the state $S_{k_W=1,0,\ldots,0,W-1}$ in W transitions and finally back to state S_W in one transition for a total number of $W + 1 + W + 1 = 2W + 2$ transitions. Thus, for $B \geq 2$ and $W \geq 2$, two return paths exist from state S_W of lengths β and $\beta + 1$, where $\beta = 2W + 1$.

Theorem 3: The Markov chain representation for the buffering process where $B \geq 2$ and $W \geq 2$ is aperiodic.

Proof: From Lemma 1 above, we know that for state S_W (empty buffer) that β and $\beta + 1$ are the lengths of two return paths ($\beta = 2W + 1$). From elementary number theory we know that the g.c.d. of two successive integers is 1, thus the g.c.d. of the lengths of the return paths must be 1 and so the state S_W is aperiodic. Since from Theorem 2 the chain is irreducible, all of the states must be aperiodic, therefore, the Markov chain is aperiodic.

Theorem 4: The Markov chain representation for the BHash buffering process converges.

Proof: A well known theorem of Markov chains is that a chain will converge if if is finite, irreducible and aperiodic. By theorems 1, 2, and 3 we know these conditions to be true.

4.7. Example

The Markov chain for a BHash organization with the number of buckets X equal to three and a buffer capacity W of six records is shown in Figure 2. The state labeled 1 at the top of the diagram represents the empty buffer state where all three buffers are empty (vector "(3)"). The numbers beside the state transition arrows are the single step transition probabilities. For example, the transition from state 1 to state 2 is 1.0 (i.e. the probability that a record insertion into an empty buffer will create one bucket with a single record and leave the other two empty is 1.0). The flushing states in this example are the states labeled 17 to 23. The transitions from them go to the states labeled along the bottom. For example, from state 19 (10020) (one bucket with 4 records, two buckets with 1 record, buffer at capacity), the next state will be either state 4 (21) (probability .33) if the next record is assigned to the bucket with 4 records (flush size of 5), or state 6 (111) (probability .67) if the record is assigned to one of the two buckets with only one assigned record (flush size 4). In either case the largest bucket in the buffer will be flushed.

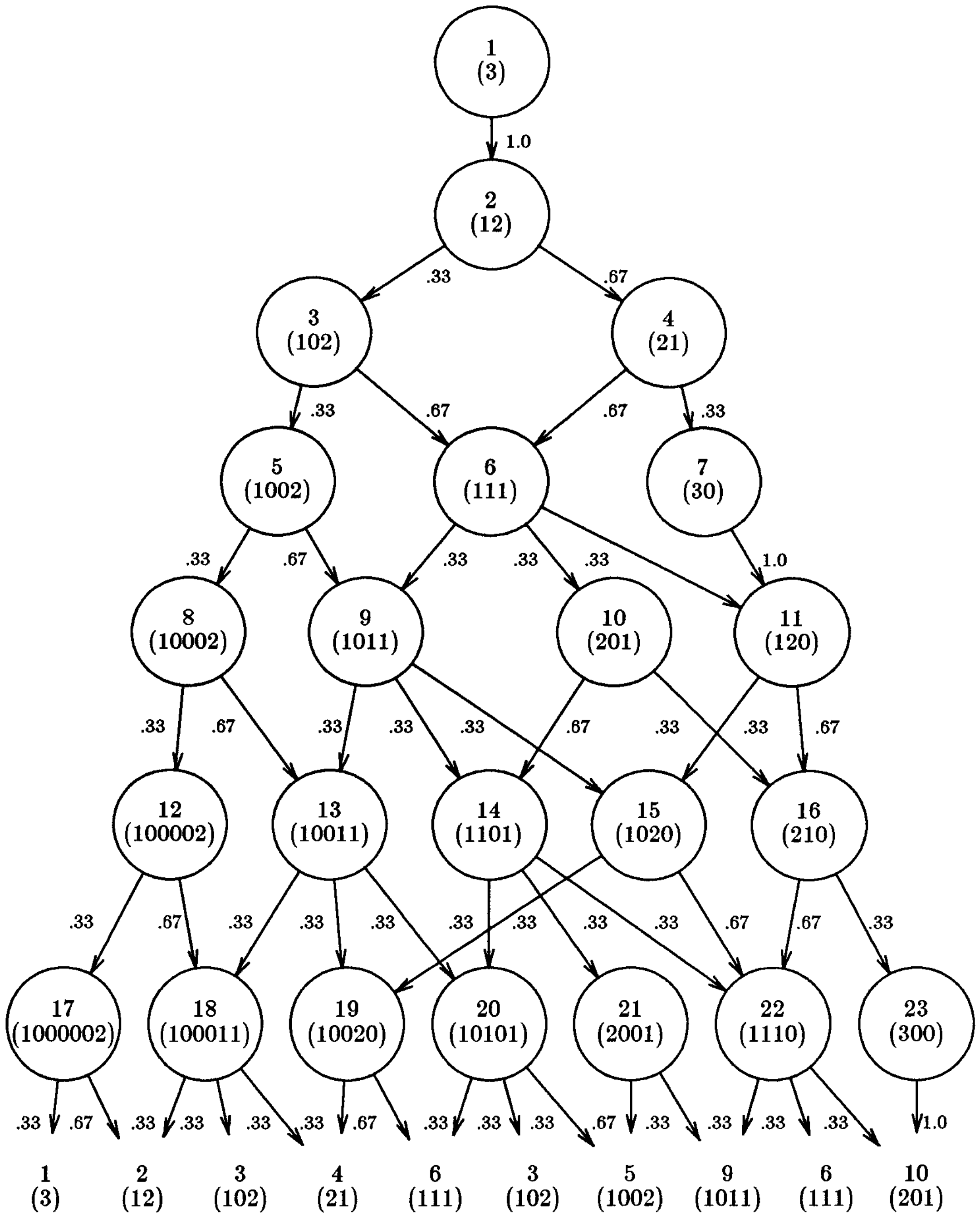


Figure 2. Markov Chain for W=6, X=3

Table 1. has the stationary probabilities for each of the states in the chain computed using iterated matrix multiplication. The transition matrix and the computation are omitted for brevity.

Stationary Probabilities					
State	Probability	State	Probability	State	Probability
1	0.0006	9	0.1124	17	0.0017
2	0.0082	10	0.0903	18	0.0193
3	0.0253	11	0.0429	19	0.0332
4	0.0230	12	0.0051	20	0.0485
5	0.0463	13	0.0478	21	0.0325
6	0.1059	14	0.0976	22	0.1062
7	0.0077	15	0.0518	23	0.0196
8	0.0154	16	0.0587		

Table 1. Stationary Probabilities

We can compute the exact expected flush size for this example using the expression for $g(W,X)$. The numerator of the expression is:

$$\begin{array}{ll}
0.0017 \cdot (7 \cdot 0.33 + 6 \cdot 0.67) & = 0.0107 \\
0.0193 \cdot (6 \cdot 0.33 + 5 \cdot 0.67) & = 0.1029 \\
0.0332 \cdot (5 \cdot 0.33 + 4 \cdot 0.67) & = 0.1439 \\
0.0485 \cdot (5 \cdot 0.33 + 4 \cdot 0.67) & = 0.2102 \\
0.0325 \cdot (4 \cdot 0.67 + 3 \cdot 0.33) & = 0.1192 \\
0.1062 \cdot (4 \cdot 0.33 + 3 \cdot 0.67) & = 0.3540 \\
0.0196 \cdot (3 \cdot 1.0) & = 0.0588 \\
\hline
 & = 0.9997
\end{array}$$

And the denominator, the sum of the stationary probabilities of the flushing states, is:

$$\sum_{p_{k_W,\ldots,k_2,k_1,k_0} \in P_F} p_{k_W,\ldots,k_2,k_1,k_0} = 0.0017 + 0.0193 + 0.0332 + 0.0485$$

$$+ 0.0325 + 0.1062 + 0.0196$$

$$= 0.261$$

The exact expected flush size $g = \dfrac{0.9997}{0.261} = 3.83\ \textit{records}$.

4.8. Simulation

A simulation of the BHash buffering process was implemented. The results it provided gave both support for the correctness of the exact analysis and insight into the nature of the process' steady state behavior.

For example, simulations of the process for $W = 6, X = 3$ (the example given in the exact analysis of the previous section) produced values for g ranging from 3.78 to 3.87 records. This range compares very favorably with the computed exact value of 3.83 records.

Insight into the steady state behavior of the buffering process was given by taking "snap shots" of the distribution of the bucket sizes of full buffers (i.e. just before flushing). The graph in Figure 3. shows an buffer size distribution taken from a simulation in steady state just before a buffer flush. It has been sorted so that the largest buffer sizes are to the left and the smaller sizes are to the right. An important point to notice on the graph is that only one or two buckets have the very largest bucket sizes (five in this case), while the rest of the sizes are evenly distributed among the rest of the buckets (approximately one hundred buckets of each size).

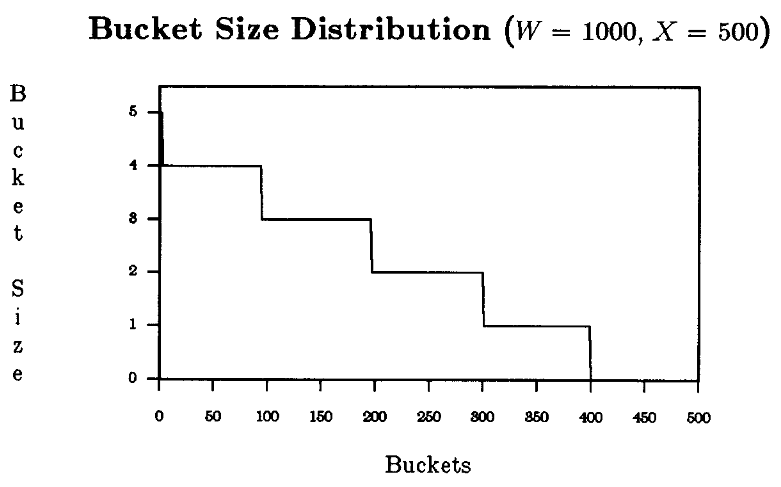


Figure 3. Bucket Size Distribution at Steady State

The explanation for this behavior is that in steady state, the uniform distribution of the incoming records among the buckets by the hash function will assign an equal number of records to each "step" of the staircase. This causes each step, in an escalator-like fashion, to lose some buckets to next higher step (because those buckets now have more records) and gain an equal number from the next lower step, The largest step at the top of the staircase will eject a bucket which is the one to be flushed. Only one or a very small number of buckets can contain g records when the buffer is full otherwise there is the possibility of one of the newly records to be inserted after the flush being assigned to one of those buckets. If this occurred then the flush size would be $g + 1$ which would violate the steady state assumption that the flush size was g.

The staircase structure described above degenerates when the number of buckets is more than twice the capacity of the buffer or $X > 2W$. This occurs because more than half of the buckets will not have any records assigned to them when the buffer is at capacity. From simulations, the steady state form of the degenerate staircase contains at most a single bucket with two assigned records, the rest of the buckets have either one or zero records.

The reason for this is that at the boundary point and beyond, after the bucket with the most records is flushed, any newly inserted record has half a chance or more of being assigned to an empty bucket. Thus, in steady state, it would not be possible to have a large flush size. It is easy to see that at the boundary point the steady state gives a flush size of two. When two records are flushed, and two are (eventually) inserted to replace them, we expect one of the inserted records to be assigned to an empty bucket and one to a bucket already assigned one other record. The flush size will again be two and the cycle will repeat. In a similar manner, it is also easy to see that as X increases in relation to $2W$, increasing the proportion of empty buckets to occupied buckets, that the steady state flush size will decrease and eventually approach one.

4.9. Buffered ISAM

For an ISAM organization our analysis needs to account for the fact that the distribution of records to buckets in not uniform. The exact distribution of the probability mass assigned to a bucket and the correct distribution of the number of records per bucket after a sequence of random insertions were first derived by Batory [1]. The analysis in [32] is based on an approximate model. In [32] it is shown that the asymptotic distribution is a negative binomial distribution. Let α, $\alpha \geq 0$, denote the *file expansion factor*, defined as N/M where M is the total number of records originally loaded and N is the total number of records inserted thereafter. The probability that a bucket loaded with m records contains n, $n \geq m$, records after the file has expanded by a factor α is then asymptotically:

$$P_m^*(n,\alpha) = \binom{m+n-1}{n} \frac{\alpha^n}{(1+\alpha)^{m+n}}$$

We can use this and a more detailed (larger) Markov chain to compute the expected flush size. The states in the more detailed chain will need to account for the size of each bucket individually. The transitions probabilities in the chain will be be those obtained from the expression given above.

5. Expected Case Steady State Analysis For Flush Group Size

While the Markov analysis produces exact results for the expected flush size, its application for large buffer sizes and numbers of buckets becomes unwieldy as the number of states in the chain grows quickly with the buffer size. To deal with these situations, we can use our knowledge of the form of the steady state solution to derive an expected case *steady state* analysis.

5.1. 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 4. 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.

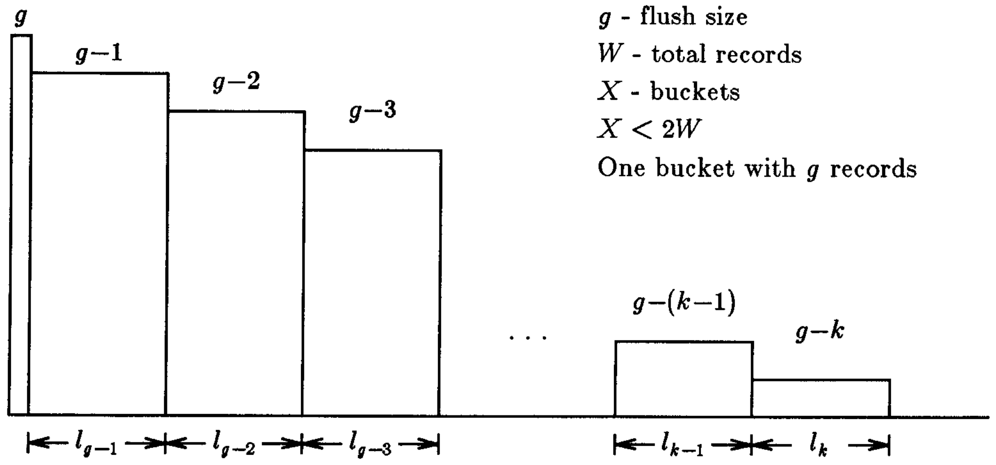


Figure 4. Buffer Bucket Sets Before Steady State Flush for $X < 2W$

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.

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

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

To derive g we can use the constraint that just prior to 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 $\frac{X}{g}$ and the sum $[(g-1) + \cdots + (g-k)]$ is equal to $\frac{2g-k-1}{2}k$, 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 $g = \frac{2W + X}{X} = \frac{2W}{X} + 1$

5.2. 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 5.

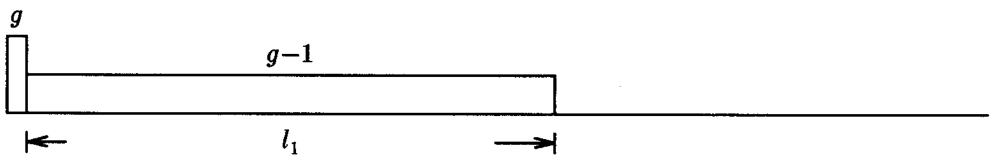


Figure 5. Buffer Bucket Sets Before Steady State Flush for $X > 2W$

As before we have $l_1 = \frac{X}{g}$ 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 $g = \frac{X}{X + g - W - 1}$

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

5.3. 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 $g = \frac{2W}{2W}+1 = 2$ and using the second approach, $g = \frac{2W}{2W-W} = \frac{2W}{W} = 2$.

5.4. Simulation Comparison

We can compare our analysis with the results of simulations. The graph in Figure 6. plots the values for the flush size resulting from simulations of the buffering process for different buffer sizes. The graph in Figure 7. plots the values computed using the expected case analysis.

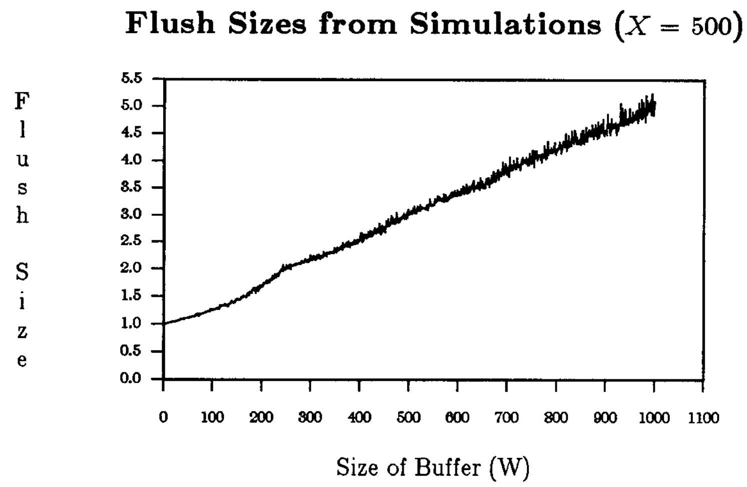
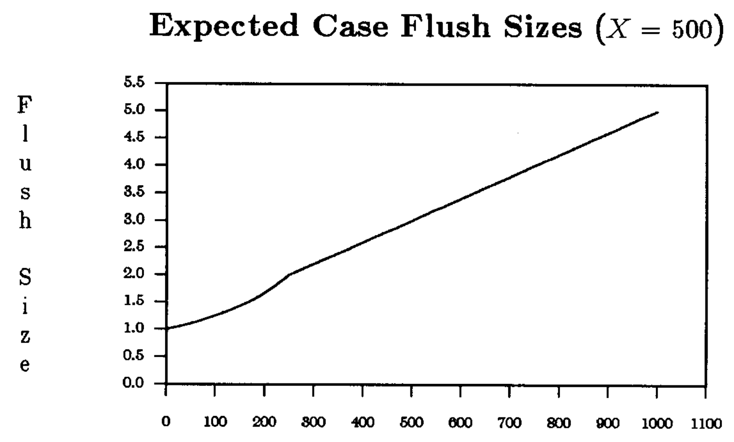


Figure 6. Example Flush Sizes From Simulations

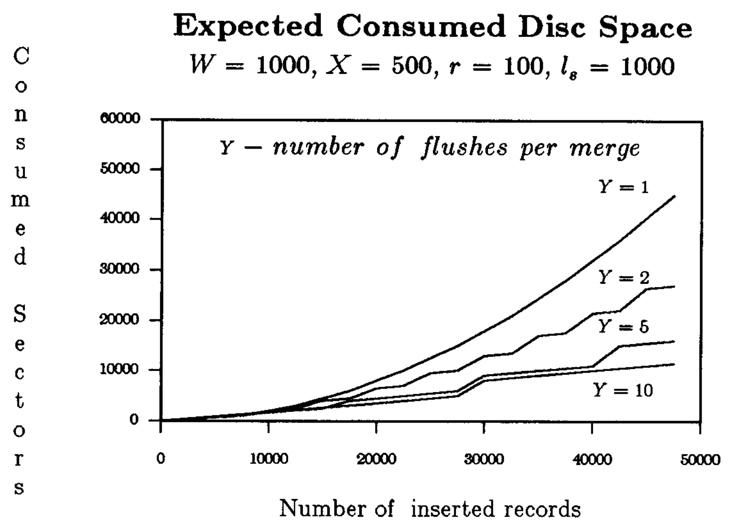


Size of Buffer (W)

Figure 7. Example Expected Case Flush Sizes

6. Comparison of Different Levels of Sequentiality

The graph in Figure 8. shows the expected amount of consumed disc space as a function of the number of inserted records for different required levels of sequentiality. The curves in the graph were computed using the formulae derived in section 3. As is evident from the graph, the highest level of sequentiality ($Y = 1$) has a dramatically higher level of disc consumption than the lower levels of sequentiality. The rate of disc space consumption significantly reduces for lower levels of sequentiallity ($Y > 1$).



Number of inserted records

Figure 8. Consumed Disc Space For Different Retrieval Performance Limits

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.

7. Summary

In this paper we proposed a class of access methods that employ 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 one of them, 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 due to data duplication. The description of the method, the development of the model to study and the derivation of analytic formulae describing the tradeoff between retrieval performance and WORM disc space utilization constitute the major contributions of this paper.

Our analysis will be very useful in the database physical design phase of applications that employ WORM optical discs, where it will allow one to examine and explore the implications of design constraints and decisions upon 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.

8. 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.

[10] Commer, 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] Larson, P.-A., "Linear Hashing with Partial Expansions", In Proc. 6th Conf. on Very Large Databases, ACM, New York, pp. 224-232, 1980.

[17] Larson, P.A., "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.

[32] Christodoulakis, S., manolopoulos, Y., Larson, P.-A., "Analysis of Overflow Handling for Variable Length Records", *Information Systems*, Vol. 14, No. 2., To appear.