Division of Research
Graduate School of Business Adminstration
The University of Michigan

November, 1980

A PRACTICAL GUIDE TO THE DESIGN OF DIFFERENTIAL FILES FOR RECOVERY OF ONLINE DATABASES

Working Paper No. 239

Houtan Aghili
and
Dennis G. Severance*

The University of Michigan

* This work was supported by the National Science Foundation under Grant MCS-78-26597 and by David Taylor Naval Ship Research and Development Center under Contract N00167-97-G0002.

ABSTRACT

The concept of a differential file has previously been proposed as an efficient means of collecting database updates for online systems. This paper studies the problem of database backup and recovery for such systems. An analytic model of the operations is presented. Five key design decisions are identified and an optimization procedure for each is developed. A design procedure is presented which quickly provides parameters for a near-optimal differential architecture on the basis of a series of table look-ups.

Key Words and Phrases: backup and recovery, database maintenance, differential files, hashing functions, numerical methods, optimization, reorganization

CR Categories: 3.73,3.80,4.33,5.0

OUTLINE

1. INTRODUCTION

Corporations have increasingly come to depend upon the continuous availability of online databases in support of such critical business functions as order entry, inventory control, credit verification, and customer service. Loss of such systems for even short periods of time can create havoc within the business and may result in significant financial loss. As a result, rapid database recovery is an important issue in the design of these systems.

It has been argued that the database recovery might be speeded considerably by holding updates in a separate file of changes called a differential file [8,14]. A practitioner considering this idea, however, has available little specific guidance for designing such an architecture as an alternative to a conventional update inplace strategy. This paper analyzes the backup and recovery operations required for an online database which employs a differential file and provides useful guidelines for selecting an efficient set of operating parameters. The potential for a dramatic improvement in recovery speed at a slight increase in operating cost is illustrated.

There are two basic causes of data loss in online systems: (1) partial completion of update operations caused by the program or system failures which render parts of the

database inaccurate or inaccessible, and (2) physical destruction of storage media which renders all or part of a database unreadable. Canning [6] provides insights into the nature and impact of such data losses and an overview of techniques used in commercial database systems to recover from them.

A simple and widely used database recovery procedure involves a periodic dumping of the database to a backup file. Once the dump is taken, all processed transactions are then dated and recorded on a transaction log so that in the event of a data loss the latest dump may be recopied and all transactions reapplied. Recopying of the latest dump is generally quite fast, requiring as little as a few minutes for an entire disk pack. The reprocessing of transactions, however, may require several hours if the time since the dump has been long. Sayani [11] and Lohman and Muckstadt [8] study the trade-offs between dumping cost and recovery speed by means of an analytic model which provides guidance in selecting an efficient dumping frequency.

The database recovery can be speeded considerably with the use of an after-image log, in which a sequential file is used to store an identified and dated copy of each new or changed database page, record, or record segment at the time that it is modified. With this file, transaction

reprocessing is not required once the dump is restored. Rather, the log is sorted by identifier and date, and the latest version of each modified record is selected and written directly into the database.

For large databases with moderate or naturally concentrated update activity, differential files offer an alternative strategy for rapid backup and recovery. A differential file isolates a database from the physical change by directing all new and modified records onto a separate and relatively small file of changes. Since the main file is never changed, it can always be recovered quickly from its dump in the event of a loss. Transaction reprocessing is required only in the event of damage to the differential file. Since this file is small, it can be backed up quickly and frequently to minimize the reprocessing time. It can also be duplexed at reasonable cost as insurance against physical damage to one of the copies.

While differential files offer a number of other operational advantages [14], this paper focuses exclusively upon their value in speeding backup and recovery operations for online databases. Specifically, we will analyze these operations to establish the frequency with which a main data file and its differential file should be subjected to

backup, as well as the frequency with which they should be reorganized into a new main database. An analytic cost model for this purpose is developed in Section 2. Its solution in Section 3 leads naturally to a series of tables which enable a designer to quickly determine a near-optimal differential file architecture for a typical operating environment. For environments which differ substantially, a FORTRAN program is provided in the Appendix as an alternative means of developing an efficient design.

2. A QUANTITATIVE MODEL OF DIFFERENTIAL FILE OPERATION

2.1 Previous Analysis of Operational Characteristics

Consider a database of N records. Assume that updates are independent and uniformly distributed over all records. Severance and Lohman [14] showed that the expected proportion of distinct main file records, Rn, which are updated after n updates have been received is given by

$$R_n = 1 - (1 - 1/N)^n = 1 - e^{-n/N}, \text{ for large } N . \quad (1)$$

For n ranging from 0 to N , Figure 1 depicts the growth over time of n/N and Rn. Respectively, these curves characterize the size of a differential file which maintains every record change and one which maintains only the most recent image of each changed record.

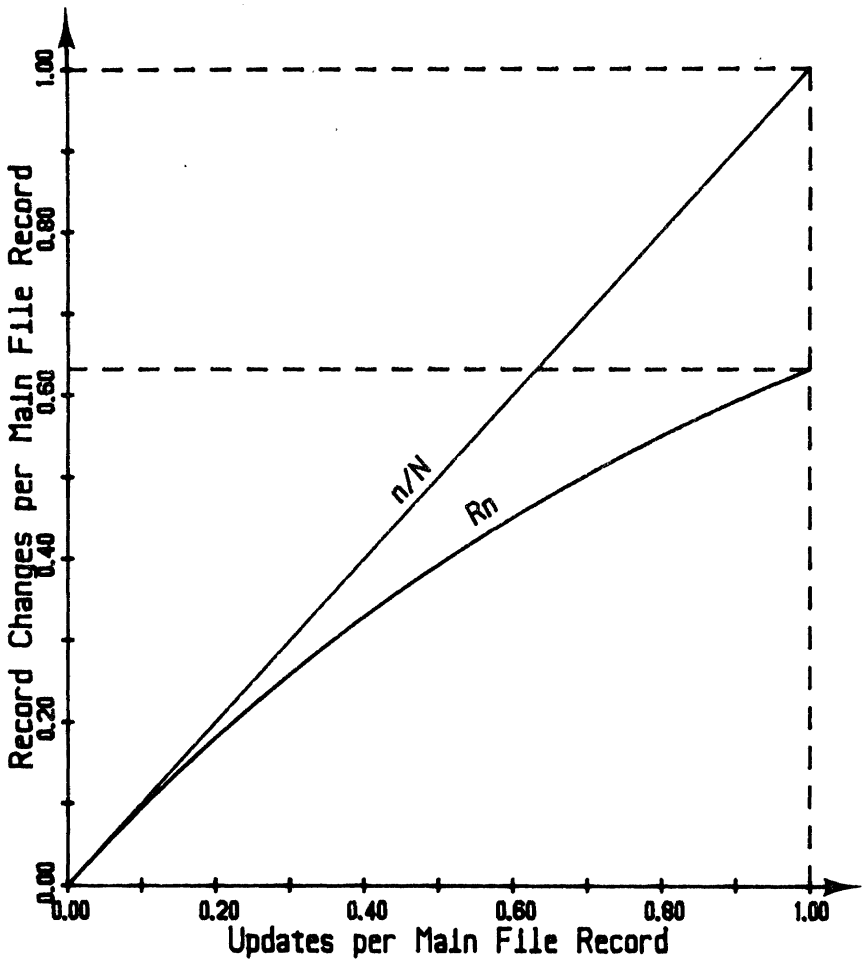

Figure 1 - General Shape of n/N and Rn

To assure access to the most current image of a record, one should search the differential file for every record retrieval. If the required record is not found there, then the original copy is accessed from the main database. In general, both the main database and the differential file utilize some form of index to speed these searches [12,13]. Unsuccessful searches of the differential file can be largely eliminated by use of a presearch filtering

mechanism. The filtering scheme, devised originally by Bloom [5], associates the differential file with a main memory bit vector of length M, and some number X of hashing functions which map record identifiers into bit addresses. When the differential file is initially empty, all bits in the filter are set to 0. Thereafter, whenever a record is stored in the differential file, each hashing transformation is applied to its identifier and each of the selected bits is set to 1.

Retrieval of the database records now proceeds as follows. The identifier of the record to be retrieved is mapped through each transformation. If any bit is 0, it is certain that the most recent version of the record still resides in the main data file; the differential file search is skipped and the main data file is accessed immediately. If all bits have value 1, then an updated copy of the record is likely to be found in the differential file, which is therefore searched. There is a possibility that this search will prove fruitless, since the bits associated with a given identifier might be set to 1 coincidentally by mappings from other updated records. Only in the event of such a filtering error are both files searched during a record retrieval.

Severance and Lohman [14] show that the probability, Pn, of a filtering error after the accumulation of n updates in the differential file is the product of the probability

that the required record is yet unchanged and the probability that all bits addressed have been previously set to 1, that is,

$$P_n(X,M) = e^{-n/N}(1-e^{-nX/M})^X . \qquad (2)$$

Figure 2 illustrates the general shape of Pn(X,M) for different values of X. This function appears as one component of the analytic cost model for the differential file operations developed in the next section.

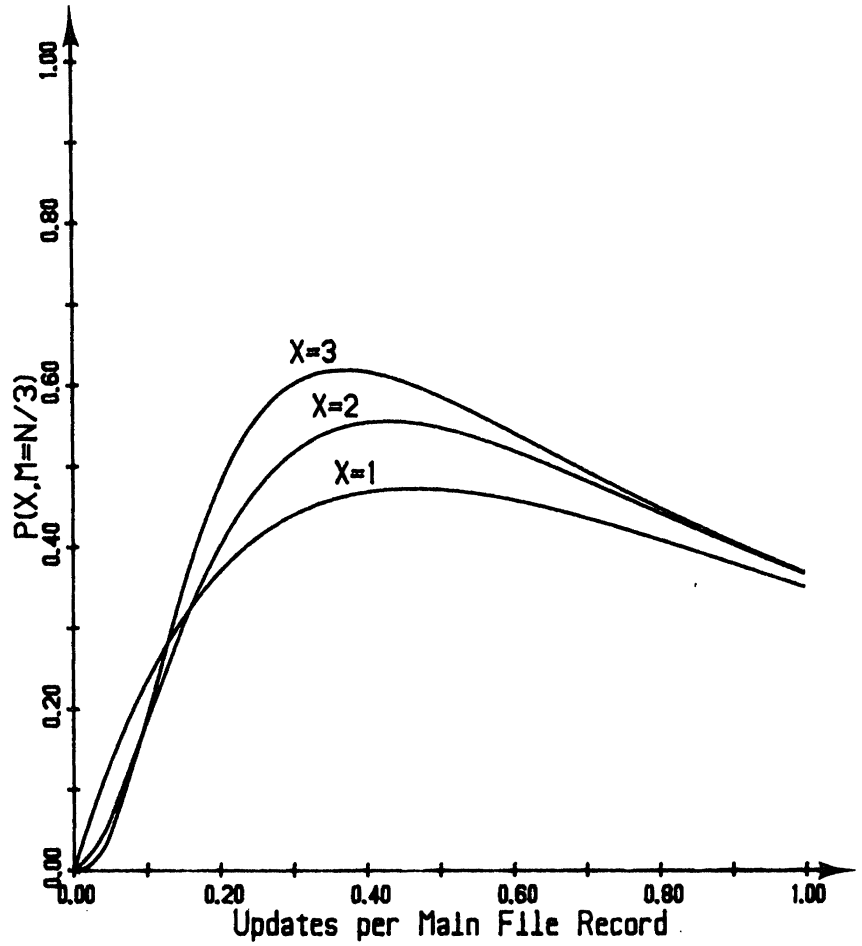

Figure 2 - General Shape of Pn(X,M)

2.2 Backup, Recovery, and Reorganization Procedures

We analyze a differential file operating environment which cycles through a series of backup and reorganization processes. Initially, the main database is dumped to a backup copy, the differential file is empty, and the Bloom filter is set to 0. As update transactions arrive over time they are recorded in a (sequential) transaction file. Resulting updates are posted both in a (direct access) differential file and on a (sequential) after-image log. Appropriate filter bits are set and, finally, the successful completion of the transaction processing is noted in the transaction log. A detailed description of processes which utilize this data to recover from a variety of data losses and system failures is provided by Lohman [7].

As the after-image log grows, the time required for differential file recovery increases. Periodically, therefore, the cumulative effect of reposting these changes is preserved by dumping the current differential file to a backup copy. If the differential file grows sequentially, maintaining a history of all changes, then only the incremental portion of the file accumulated since the last dump is copied. The entire file is copied if an update inplace strategy is used.

As updates continue, the search filter becomes less

effective and the differential file will grow to the limit of its space allocation. A periodic reorganization therefore resets the bit vector to zero and empties the differential file by merging all changes into the main database. After one or more such reorganizations, the time required to recover the main database via the after-image processing justifies another dump and the entire cycle begins again.

Figure 3 illustrates the relationships among the various files and backup procedures and highlights five important parameters whose values must be selected during the differential file system design:

M : size of the Bloom filter bit vector,
X : number of hashing transformations,
R : number of updates before reorganization,
D : number of differential file dumps during a reorganization interval, and
B : number of reorganizations before the main database dump.

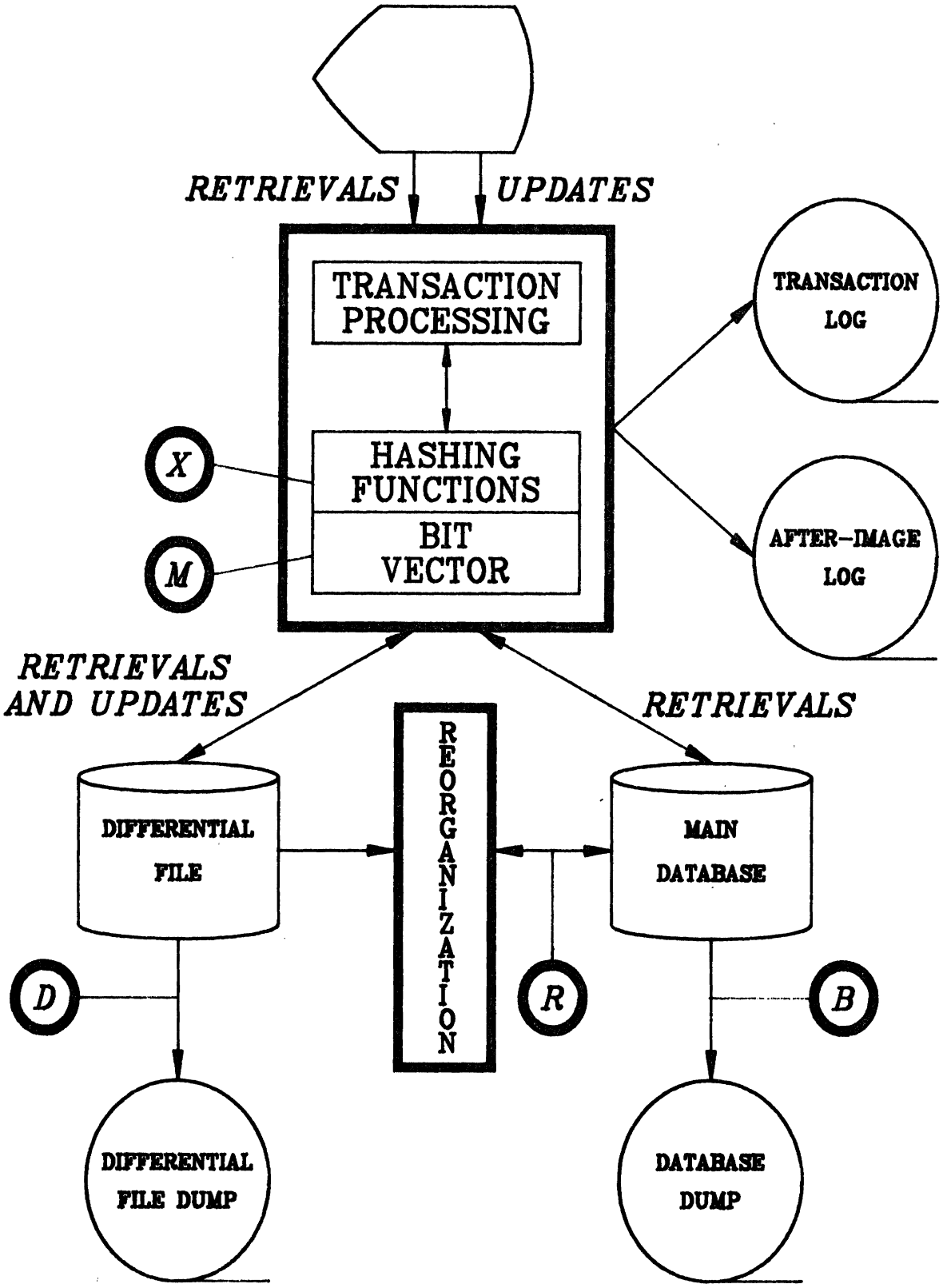

Figure 3- Interrelationship of Model Components and Decision Variables

2.3 Accounting for Costs

Selection of efficient values for the five design parameters is affected by the 21 problem variables defined in Table 1. Typical values for each are given and will be used for calculations in subsequent sections.

There are nine major components of the system cost to consider. An equation defining the expected cost per processed update for each of these is given below:

The expected main database recovery cost is the sum of the costs to recopy the latest dump and to repost from the after-image log all updates made prior to the last reorganization, and is given by

$$C1 = (\lambda/\mu)\,[rN + u'R(B-1)/2] \,. \qquad (3)$$

The expected differential file dump and recovery cost is the sum of all differential file dumps taken during a reorganization interval plus, in the event of loss, the expected cost to recopy the latest differential file dump and to repost all updates accumulated in the after-image log since that dump. A general analysis for this cost is provided in Section 3.2. In the case where no differential file dump is taken during reorganizations, the cost is given by

$$C2 = (\lambda'/\mu)u''(R-1)/2 \,. \qquad (4)$$

The expected differential file storage cost is, in general, given by

$$C3 = (s''/\mu) \sum_{j=0}^{R-1} E[\text{differential file size} \mid j \text{ updates}].$$

Parameter Description	Typical Value
N : number of records in the database	10^4 - 10^7
μ : record updates per unit of time	1-100 per minute
ρ : read-only requests per unit of time	1-100 per minute
λ : rate of main database failures	1-360 per year
λ' : rate of the differential file failures	1-360 per year
C0 : set-up cost associated with dump and reorganization operations	$2.00
d : cost of dumping one record from the main data file	$0.0005
d' : cost of dumping one record from the differential file	$0.0005
u : cost of posting an update to the main data file during reorganization	$0.01
a : cost of a record access from the main data file	$0.01
a' : cost of a record access from the differential file	$0.01
a" : cost of an unsuccessful search of the differential file	$0.01
s : main storage cost per bit per unit of time	$1.7x10^{-6}$ bit/minute
s' : cost of storing a record of the main data file per unit of time	$3x10^{-7}$record/minute
s" : cost of storing a record of the differential file per unit of time	$3x10^{-7}$record minute
h' : cost of executing a hashing function in Bloom filter	10^{-4}
w : weighting factor reflecting relative importance of recovery costs vis-a-vis other cost components	10.
r : cost of restoring a record of main data file from its dump	($0.0005)w
r' : cost of restoring a differential file record from its dump	($0.0005)w
u' : cost of posting an update to the main data file during recovery	($0.01)w
u" : cost of posting an update to the differential file during recovery	($0.002)w

Table 1- Problem Parameters

When the history of all database updates is stored sequentially in the differential file, this expression reduces to

$$C3 = (s''/\mu)(R-1)/2 \; ; \qquad (5)$$

when only the latest versions of each changed record is recorded, then

$$C3 = (s''/\mu)N[1-(1-e^{-R/N})/(R/N)] \; . \qquad (5\text{-}1)$$

The latter strategy of update inplace precludes incremental dumping of the differential file. Moreover in our environment of independent, uniformly distributed updates, the space savings that it offers is not significant. (Specifically, differential files are found to lose much of their attractiveness at sizes above 10 to 20 percent of the main data file; and below this limit the storage costs of equations (5) and (5-1) are nearly equivalent.) We shall assume in this paper that the differential file is relatively small ($R/N \leq 0.20$), that it grows sequentially, and that C3 is given by equation (5). The reader interested in an analysis of the update inplace strategy for an environment in which updates cluster on a high activity subset of records, e.g., airline reservation systems, is directed to Aghili and Severance [2].

The expected main database backup cost is given by the cost of a dump divided by the number of updates between the dumps,

$$C4 = (Nd)/(BR) \; . \qquad (6)$$

The expected reorganization cost is the ratio of the cost of posting the latest version of each updated record to the total number of updates made:

$$C5 = [uN(1-e^{-R/N})+C_0]/R \; . \qquad (7)$$

The cost of unsuccessful differential file accesses for updates and read-only requests is given by

$$C6 = (1+\rho/\mu)\,(a''/R) \sum_{j=0}^{R-1} Pn(X,M) \quad ,$$

$$= (1+\rho/\mu)[a'' \sum_{j=0}^{X} C(X,j)\,(-1)^j \left(\frac{1-e^{-JXR/M-R/N}}{R(jX/M+1/N)}\right)], \quad (8)$$

where $C(i,k)$ denotes the number of k combinations from i objects ($C(i,k) = i!\,/\,(k!\,(i-k)!)$).

The expected record access cost includes filtering costs plus the cost of either a differential file or a main data file access,

$$C7 = (1+\rho/\mu)[h'X+a'+(a-a')N(1-e^{-R/N})/R]. \quad (9)$$

The cost of storing the bit vector is equal to

$$C8 = (s/\mu)M. \quad (10)$$

The main data file storage cost is a fixed cost and given by

$$C9 = (s'/\mu)N. \quad (11)$$

Combining all of these cost components, the total expected cost per processed update transaction is given by

$$C(X,M,R,B,D) = C1+C2+C3+C4+C5+C6+C7+C8+C9. \quad (12)$$

3. MINIMIZING TOTAL OPERATING COSTS

To design a differential file architecture for a given database, we select the combination of design parameter values which minimizes the total expected operating cost per processed update transactions over the operating cycle; that is,

Minimize $C(X,M,R,D,B)$

Subject to:

$R \leq$ Maximum allowable differential file growth,

$R,B \geq 1$,

$X,M,D \geq 0$,

R,B,X,M,D integers .

No exact optimization method exists to solve this nonlinear objective function in integer decision variables. In general, such problems require techniques such as Branch and Bound [2] to structure and search their large solution space, and numerical methods such as Pattern-Search [4], Newthon-Raphson [4], or Powell [9,10] to determine optimal parameter values. A precise methodology for the solution of the differential file design problem is presented in [1].

Experience with this precise analysis has yielded a number of useful insights into the nature of the optimization problem and characteristics of its solution. These insights have led to the development of a much simpler

heuristic procedure which can quickly generate a near-optimal differential file architecture via a series of table look-ups. The latter design procedure is presented here.

The solution procedure accurately decomposes our design problem into three independent parts, as follows. First, a near-optimal number of hashing functions (X*) and bit vector size (M*) is computed as a function of reorganization interval, R. Values for this function are presented in tabular form. Next, the optimal number of differential file dumps (D*) and the optimal number of reorganizations during main data file operations (B*) are computed as a closed form function of the reorganization interval, R. Finally, the functional expressions for X*, M*, D*, and B* are substituted into the total expected cost expression (equation 12), defining it entirely as a function of R. The optimal reorganization interval R* is then obtained by minimizing the total cost expression, using numerical methods. Having computed a specific value for R*, corresponding values for X* and M* can be extracted from their table, while values of D* and B* are obtained by substituting R* into their corresponding expressions.

3.1 Designing the Search Filter

Selection of an efficient combination of X (number of hashing functions) and M (bit vector size) is affected by the sum of costs: C6 for unsuccessful differential file accesses; C7 for main data file record access; and C8 for bit vector maintenance. Given a reorganization interval R, the optimal (X*,M*) combination is obtained by solving:

$$\underset{X,M}{\text{Minimize}}\ C6 + C7 + C8 \qquad (13)$$

Subject to:

$$X,M \geq 0\ ,$$

X and M integers.

Analyzing this problem precisely, Aghili [1] has obtained several useful results. He shows (1) that the optimal value of M is never greater than $[a''(\rho+\mu)]/s$; (2) that for reasonable ratios of h'/a'' (≤ 0.0001), the optimal value of X is never greater than 8; (3) that for $R \geq [a''(\rho+\mu)]/s$, filtering mechanisms are ineffective and should not be used at all; and (4) that the total cost function (12) is insensitive to changes in X and M in a relatively wide range of values surrounding X* and M*. The latter result implies that a precise determination of (X*,M*) is not of practical concern, and therefore approximations which permit us to locate a near-optimal combination quickly are reasonable.

Assuming that the size of the main data file is much larger than the reorganization interval, e.g., $(R/N) \leq 0.2$, problem (13) may be reformulated as

$$\underset{X,M}{\text{Minimize}}\ a''(\rho+\mu)\left[h'X+\sum_{n=0}^{R-1}(1-e^{-nx/M})^{X}/R\right]+sM\ , \qquad (13\text{-}1)$$

Subject to:

$$X, M \geq 0\ ,$$

X and M integers.

This problem is solved by iterating on the values of X from 1 to 8, approximating the optimal M* for each X, and selecting as "optimal" the combination with minimum cost in equation (13-1). Specifically, the approximation to M* for a given X is obtained by solving[1]

$$\underset{M}{\text{Minimize}}\ a''(\rho+\mu)\left[1-(1-e^{-XR/M})/(RX/M)\right] + sM\ , \qquad (13\text{-}2)$$

Subject to:

$$0 \leq M \leq [a''(\rho+\mu)]/s\ ,$$

M integer ;

which is easily accomplished using a Fibonacci search [4].

The above procedure has been used to approximate (X*,M*) for a variety of reorganization intervals and access intensities $(\rho+\mu)$, using the typical values of a", h', and s

---

1 For reasonable parameter values, $s/[a''(\rho+\mu)] \ll 1$, and any given value of X, the solution of (13-2) provides a tight upper bound for M* (Aghili [1]).

given by Table 1. The results are presented in Table 2. Realize that this table is in fact a tabular representation of the function

$$CXM(R) : R \dashrightarrow (X^*, M^*) , \qquad (13\text{-}3)$$

which maps values of R into corresponding near-optimal values of X and M. It will be used as such by the procedure in Section 3.4 which determines an optimal reorganization interval R*.

Reorganization Interval (R)	Access Intensity ($\rho + \mu$)		
	1	10	100
100	2 7	2 23	2 74
200	2 9	2 33	2 103
300	2 11	2 40	2 122
400	2 12	2 46	2 151
500	3 15	2 51	2 161
600	2 14	2 55	2 180
700	2 15	2 59	2 190
800	2 16	2 63	2 209
900	2 16	2 66	2 218
1000	2 17	2 69	2 228
2000	1 17	2 94	2 324
3000	1 18	2 111	2 392
4000	1 17	2 124	2 450
5000	1 15	2 135	2 508
6000	0	2 143	2 546
7000	0	2 151	2 585
8000	0	2 157	2 623
9000	0	2 162	2 662
10000	0	2 166	2 691
20000	0	1 166	2 942
30000	0	1 175	2 1114
40000	0	1 170	2 1240
50000	0	1 148	2 1346
60000	0	0	2 1433
70000	0	0	2 1501
80000	0	0	2 1559
90000	0	0	2 1615
100000	0	0	2 1655
200000	0	0	1 1655
300000	0	0	1 1751
400000	0	0	1 1703
500000	0	0	1 1481
600000	0	0	0
700000	0	0	0
800000	0	0	0
900000	0	0	0
1000000	0	0	0
2000000	0	0	0

Table 2 - Near-Optimal (X*,M*) Combination for Different Reorganization Intervals and Access Intensities (M is given in hundreds of bits)

3.2 Selecting an Optimal Differential File Dumping Policy

The differential file is dumped periodically to speed its recovery in the event of loss. The combined cost of dumping and recovery of the differential file was referred to simply as C2 in Section 2.3. A detailed analysis of this cost is now presented.

Consider the general situation in which D dumps are taken during a reorganization interval with Yi updates accumulated into the differential file between the (i-1)-th and i-th dumps. Three cost components affect the selection of D and $Y=\{Yi \mid i=1,\ldots,D\}$:

i- the cost of differential file dump,
ii- the expected cost of recopying the latest differential file dump, and
iii- the expected cost of reprocessing all transactions since the last dump.

Specifically, for given integer values of R, D, and Y, the combined cost of these components is given by

$$C2 = R\ CD(R,D)\ , \tag{14}$$

where

$$\begin{aligned} CD(R,D) = {} & C_0 D + d'Y1 + \ldots + d'Y_D \\ & + (\lambda'/\mu) r' [Y2Y1 + Y3(Y1+Y2) + \ldots + Y_D(Y1+\ldots+Y_{D-1})] \\ & + (\lambda'/\mu) r' (R - Y1 - \ldots - Y_D)(Y1+\ldots+Y_D) \\ & + (\lambda'/\mu) u'' [Y1(Y1-1) + \ldots + Y_D(Y_D-1)]/2 \\ & + (\lambda'/\mu) u'' (R - Y1 - \ldots - Y_D)(Y1+\ldots+Y_D-1)/2\ . \end{aligned} \tag{14-1}$$

Relaxing the integer constraint on values of Yi for our environment of sequential differential file growth, Aghili [1] has shown the vector $\overline{Y^*}$ which minimizes CD(R,D) has components

$$\overline{Yi^*} = R/(D+1) \quad , i=1,\ldots,D \; ; \qquad (15)$$

that is, differential file dumps should be equally spaced over the reorganization interval. The optimal solution to the original integer problem is guaranteed to be one of the integer points surrounding $\overline{Y^*}$ (i.e., a vertex of a hyper-cube in D dimensional space); for reasonable problems these values are identical for all practical purposes. Substituting $\overline{Y^*}$ into (14-1) yields

$$\begin{aligned} CD(R,D) = {} & C_0 D + d'(RD)/(D+1) \qquad (16) \\ & +(\lambda'/\mu)r'(RD)/[2(D+1)] \\ & +(\lambda'/\mu)u''(R/2)[R/(D+1)-1] \; , \end{aligned}$$

which may be rewritten as

$$CD(R,D) = C_0 D + d1/(D+1) + d2 \; , \qquad (17)$$

where

$$d1 = \{(\lambda'/\mu)[(u''/2)R-(r'/2)]-d'\}R \; , \qquad (17\text{-}1)$$

and

$$d2 = [(\lambda'/\mu)(r'-u'')/2+d']R \; . \qquad (17\text{-}2)$$

The optimal number of differential file dumps with this

restatement of the problem is determined by solving dCD(R,D)/dD=0 to obtain

$$\overline{D^*} = \begin{cases} 0 , & \text{if } d1/C_0 \leq 1 , \\ (d1/C_0)^{1/2}-1 , & \text{otherwise.} \end{cases} \qquad (18)$$

The optimal integer value D* of the original problem (16) is guaranteed to be one of the two integers nearest $\overline{D^*}$. It can be quickly determined and substituted for D in (17) to yield

$$CD(R,D^*) = \begin{cases} d1+d2 , & \text{if } (d1/C_0) \leq 1 , \\ C_0D^*+d1/(D^*+1)+d2 , & \text{otherwise.} \end{cases} \qquad (19)$$

Observe that if $d1/C_0 \leq 1$, then D*=0 and C2=$(\lambda'/\mu)u''(R-1)/2$ in agreement with equation (14) of Section 2.3.

Tables 3(a) and 3(b) respectively present optimal values of dumps D* and time between dumps $(R/\mu)/(D^*+1)$ for the values of C_0, u", r', and d' given in Table 1. A 10-hour day, a 5-day week, a 4-week month, and a 12-month year are assumed in our tables.

Table 3(b) shows that the time between differential file dumps is rather insensitive to reorganization interval R. This is reasonable, since for large values of R, $\overline{D^*}$ is

approximately

$$\overline{D^*} = \begin{cases} 0 , & \text{if } d1/C_0 \leq 1, \\ R[(\lambda' u'')/(2\mu C_0)]^{1/2} , & \text{otherwise ;} \end{cases} \tag{20}$$

and hence the time between dumps is approximated by

$$(R/\mu)/D^* = \begin{cases} \text{undefined} , & \text{if } d0/C_0 \leq 1, \\ [(2C_0)/(\mu\lambda' u'')]^{1/2} , & \text{otherwise .} \end{cases} \tag{20-1}$$

Reorganization Interval (R)	Failure Rate(λ')											
	1/Day			1/Week			1/Month			1/Year		
100	0	0	0	0	0	0	0	0	0	0	0	0
200	0	0	0	0	0	0	0	0	0	0	0	0
300	0	0	0	0	0	0	0	0	0	0	0	0
400	0	0	0	0	0	0	0	0	0	0	0	0
500	0	0	0	0	0	0	0	0	0	0	0	0
600	1	0	0	0	0	0	0	0	0	0	0	0
700	1	0	0	0	0	0	0	0	0	0	0	0
800	1	0	0	0	0	0	0	0	0	0	0	0
900	2	0	0	0	0	0	0	0	0	0	0	0
1000	2	0	0	0	0	0	0	0	0	0	0	0
2000	5	1	0	2	0	0	0	0	0	0	0	0
3000	8	2	0	3	0	0	1	0	0	0	0	0
4000	11	3	0	4	0	0	1	0	0	0	0	0
5000	13	3	0	5	1	0	2	0	0	0	0	0
6000	16	4	0	7	1	0	3	0	0	0	0	0
7000	19	5	1	8	2	0	3	0	0	0	0	0
8000	22	6	1	9	2	0	4	0	0	0	0	0
9000	25	6	1	11	2	0	5	0	0	0	0	0
10000	28	8	1	12	3	0	5	0	0	0	0	0
20000	57	17	4	25	7	0	12	2	0	2	0	0
30000	86	26	7	38	11	2	18	4	0	4	0	0
40000	114	35	10	51	15	3	25	7	0	6	0	0
50000	143	45	13	63	19	4	31	9	0	8	0	0
60000	172	54	16	76	23	6	38	11	0	9	0	0
70000	201	63	19	89	27	7	44	13	1	11	0	0
80000	230	72	22	102	31	8	50	15	2	13	1	0
90000	259	81	25	115	35	10	57	17	2	15	1	0
100000	288	90	27	128	40	11	63	19	3	17	2	0
200000	576	181	56	257	80	24	128	39	10	36	8	0
300000	865	273	85	386	121	37	192	60	16	54	14	0
400000	1153	364	114	515	162	50	257	80	23	73	20	0
500000	1442	455	143	644	203	63	322	100	29	91	26	0
600000	1731	546	172	773	244	75	386	121	36	110	32	0
700000	2019	638	201	903	284	88	451	141	42	129	38	0
800000	2308	729	229	1031	325	101	515	162	49	147	44	4
900000	2597	820	258	1160	366	114	580	182	55	166	50	7
1000000	2885	912	287	1289	407	127	644	203	62	185	56	9
2000000	5772	1824	576	2580	815	256	1289	407	126	371	115	29

Table 3(a) - Optimal Number of Differential File Dumps, D*, during a Reorganization Interval at Update Intensities 1, 10, and 100 per Minute

Reorganization Interval (R)	Failure Rate(λ')			
	1/Day	1/Week	1/Month	1/Year
100				
200				
300				
400				
500				
600	0.5			
700	0.6			
800	0.7			
900	0.6			
1000	0.6			
2000	0.6 0.2	1.1		
3000	0.6 0.2	1.2	2.5	
4000	0.6 0.2	1.3	3.3	
5000	0.6 0.2	1.4 0.4	2.8	
6000	0.6 0.2	1.2 0.5	2.5	
7000	0.6 0.2 0.06	1.3 0.4	2.9	
8000	0.6 0.2 0.07	1.3 0.4	2.7	
9000	0.6 0.2 0.08	1.2 0.5	2.5	
10000	0.6 0.2 0.08	1.3 0.4	2.8	
20000	0.6 0.2 0.07	1.3 0.4 0.17	2.6	11
30000	0.6 0.2 0.06	1.3 0.4 0.17	2.6 1.1	10
40000	0.6 0.2 0.06	1.3 0.4 0.17	2.6 1.0	9.5
50000	0.6 0.2 0.06	1.3 0.4 0.14	2.6 0.8	9.3
60000	0.6 0.2 0.06	1.3 0.4 0.15	2.6 0.8	10
70000	0.6 0.2 0.06	1.3 0.4 0.14	2.6 0.8 0.6	9.7
80000	0.6 0.2 0.06	1.3 0.4 0.14	2.6 0.8 0.4	9.5 6.7
90000	0.6 0.2 0.06	1.3 0.4 0.14	2.6 0.8 0.5	9.4 7.5
100000	0.6 0.2 0.06	1.3 0.4 0.13	2.6 0.8 0.4	9.3 5.6
200000	0.6 0.2 0.06	1.3 0.4 0.13	2.6 0.8 0.4	9.0 3.7
300000	0.6 0.2 0.06	1.3 0.4 0.13	2.6 0.8 0.3	9.1 3.3
400000	0.6 0.2 0.06	1.3 0.4 0.13	2.6 0.8 0.3	9.0 3.2
500000	0.6 0.2 0.06	1.3 0.4 0.13	2.6 0.8 0.3	9.1 3.0
600000	0.6 0.2 0.06	1.3 0.4 0.13	2.6 0.8 0.3	9.0 3.0
700000	0.6 0.2 0.06	1.3 0.4 0.13	2.6 0.8 0.3	9.0 3.0
800000	0.6 0.2 0.06	1.3 0.4 0.13	2.6 0.8 0.3	9.0 3.0 2.7
900000	0.6 0.2 0.06	1.3 0.4 0.13	2.6 0.8 0.3	9.0 2.9 1.9
1000000	0.6 0.2 0.06	1.3 0.4 0.13	2.6 0.8 0.3	9.0 2.9 1.6
2000000	0.6 0.2 0.06	1.3 0.4 0.13	2.6 0.8 0.3	9.0 2.9 1.1

Table 3(b) - Optimal Time (in Days) Between Differential File Dumps at Update intensities 1, 10, and 100 per Minute

3.3 Selecting the Frequency of Main File Backup

The main data file is dumped periodically to speed recovery in the event of loss. The expected main database recovery cost, C1, and the main data file backup cost, C4, combine to affect the selection of the optimal main data file dump frequency, B*.

To determine B* in terms of reorganization interval R, we define

$$CB(R,B) = C1+C4 \qquad (21)$$
$$= (\lambda/\mu)[rN+u'R(B-1)/2]+(Nd)/(BR) \ .$$

Relaxing the integer constraint on B and solving dCB(R,B)/dB=0, we obtain

$$\overline{B^*} = \begin{cases} d3/R \ , & \text{if } R < d3 \ , \\ 1 \ , & \text{otherwise} \ ; \end{cases} \qquad (22)$$

where

$$d3 = [(2Nd\mu)/(\lambda u')]^{1/2} \ . \qquad (22\text{-}1)$$

Tables providing optimal values of $\overline{B^*}$ for a wide variety of problems have been produced. Rather than including them here, we present a single table for a normalized problem from which entries of the other tables can be generated. Assuming the typical values for r, d, u', and w given in Table 1, Table 4 displays optimal values $\overline{B1^*}$ for a number of reorganization intervals R1, assuming $N=10^7$ records,

$\mu 1$=100/minute, and $\lambda 1$=1/year. This table and equation 22 permit us to closely approximate an optimal $\overline{B2^*}$ for arbitrary values of R2, N2, $\mu 2$, and $\lambda 2$. Specifically,

$$\overline{B2^*} = \begin{cases} 1 \text{ , if } \alpha\overline{B1^*} < 1 \text{ ,} \\ \alpha\overline{B1^*} \text{ , otherwise,} \end{cases} \qquad (23)$$

where α is given by

$$\alpha = 0.9\text{x}10^{-5}(R1/R2)[N2(\mu 2/\lambda 2)]^{1/2} \text{ ,}$$

and $\mu 2$ and $\lambda 2$ are, respectively, the average number of updates per minute and the average number of main data file failures per month.

Given an optimal value $\overline{B^*}$, B* is guaranteed to be one of the two integer values nearest $\overline{B^*}$. Substituting B* for B in (21), the expected main data file backup and recovery cost per processed update transaction is given by:

$$CB(R,B^*) = \begin{cases} (\lambda/\mu)[rN-u'R/(B^*-1)/2]+(Nd)/(B^*R), & R<d3 \text{ ,} \\ (\lambda/\mu)(rN)+(Nd)/R \text{ , otherwise .} \end{cases} \qquad (24)$$

This is the final cost component needed by the procedure to locate R*, as described in the next section.

Reorganization Interval (R)	Number of Reorganizations B*
100	12000.00
200	6000.00
300	4000.00
400	3000.00
500	2400.00
600	2000.00
700	1714.29
800	1500.00
900	1333.33
1000	1200.00
2000	600.00
3000	400.00
4000	300.00
5000	240.00
6000	200.00
7000	171.43
8000	150.00
9000	133.33
10000	120.00
20000	60.00
30000	40.00
40000	30.00
50000	24.00
60000	20.00
70000	17.14
80000	15.00
90000	13.33
100000	12.00
200000	6.00
300000	4.00
400000	3.00
700000	2.40
600000	2.00
700000	1.71
800000	1.50
900000	1.33
1000000	1.20
2000000	1.00

Table 4 - Optimal Number B* of Main Data File Reorganizations Between Dumps for N=10,000,000 , μ=100/minute, λ=1/year

3.4 Selecting the Reorganization Interval

Section 3.1 describes a table look-up procedure for selecting a near-optimal (X*,M*) combination for a given reorganization interval R. Similarly, Sections 3.2 and 3.4 provide near optimal D* and B* values as functions of R. As a result, the total cost equation (12) can be reduced to the nonlinear function of R given by

```
C(R) = R CD(R,D*) ,  differential file dump and recovery cost;
     +CB(R,B*) ,     main data file dump and recovery cost;
     +CXM(R) ,       sum of unsuccessful differential file
                     access cost, record access cost, and
                     cost of maintaining the bit vector;
     +C3 ,           differential file storage cost;
     +C5 ,           reorganization cost; and
     +C9 ,           main data file storage cost;     (25)
```

where CD(R,D*), CB(R,B*), CXM(R), C3, C5, and C9 are given by expressions (19), (24), (13-3), (5), (7), and (11), respectively.

The optimal reorganization interval, R*, can now be found by minimizing C(R) over the range of integers $1 \leq R \leq 0.2N$. Although the nonlinear nature of C(R) precludes a closed form solution of this function, a complete enumeration on R is not necessary. Figure 4 illustrates the typical shape of C(R). One finds that while C(R) generally has more than one local minimum, the curve is relatively shallow at all minima. A simple search procedure (as an alternative to Pattern-Search [2], Newthon-Raphson [3], or

Powell [9,10]) can therefore locate an efficient reorganization interval quickly. Rather than enumerating all values of R, determining (M*,X*,D*,B*) for each and selecting the R* with minimal cost, one can take sample values of R -- say 100 updates apart -- and confidently choose the sample value with the minimum cost as effectively optimal. Using this technique for the parameter values given in Table 1, R* has been computed for a wide variety of problem situations. Results are presented in Table 5 and corresponding values for X*, M*, D*, and B* can be readily obtained using Tables 2, 3, and 4.

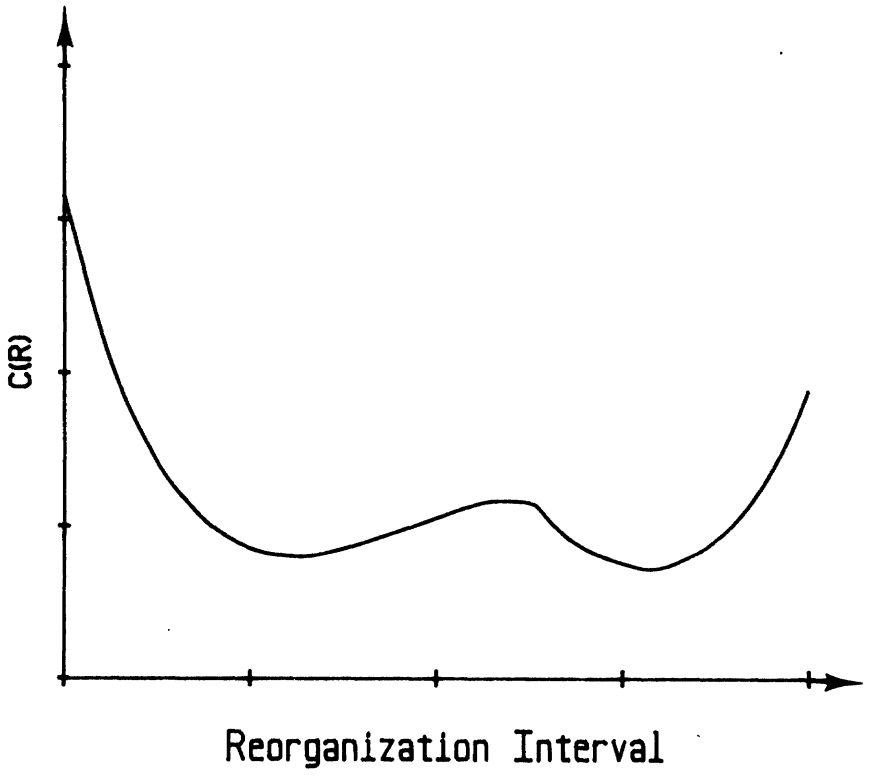

Figure 4 - General Shape of Cost Function C(R)

Database Size (N) in 1000 Records	Failure rate(λ)											
	1/Day			1/Week			1/Month			1/Year		
10	2	2	2	1.8	2	2	2	2	2	0.9	2	2
20	3.8	4	4	3.8	4	4	4	4	4	0.9	2.8	4
30	6	6	6	6	6	6	6	6	6	0.9	2.3	6
40	8	8	8	8	8	8	8	8	8	0.9	2.2	8
50	10	10	10	10	10	10	10	10	10	0.9	2.1	10
60	12	12	12	12	12	12	12	12	12	0.9	2.1	12
70	14	13.3	14	14	13.3	14	14	14	14	0.9	2.1	13.8
80	16	16	16	16	16	16	16	16	16	0.9	2	15.8
90	18	18	18	18	18	18	18	18	18	0.9	2	8.3
100	20	20	20	20	20	20	20	20	20	1	2	6
200	28	37	40	28	36	40	28	36	40	1	2	5
300	33	60	60	33	60	60	33	60	60	33	2	4
400	38	80	80	38	80	80	38	80	80	38	2	4
500	42	100	100	42	100	100	42	100	100	42	2	4
600	46	120	120	46	120	120	46	120	120	46	2	4
700	49	140	136	49	140	130	49	140	130	49	2	4
800	53	160	160	53	160	160	53	160	160	53	2	4
900	56	180	180	56	180	180	56	180	180	56	2	4
1000	59	200	200	59	200	200	59	200	200	59	2	4
5000	131	424	1000	130	425	1000	130	422	1000	130	2	4
10000	184	590	2000	184	590	2000	183	588	2000	183	588	4

Table 5 - Optimal Reorganization Interval, R*, in Thousands of Updates for Different Failure Rates and Database Sizes at Update Intensities 1, 10, and 100 per Minute

4. AN EXAMPLE OF HEURISTIC DESIGN

The rather detailed analysis given in the last section was not intended to suggest that heuristics are inappropriate when designing a database backup and recovery strategy. On the contrary, we describe elsewhere [1] a more complex and time-consuming design procedure which determines optimal differential file parameters more precisely than the table look-up procedure presented here. We believe, however, that such precision and effort are rarely justified .

Model optimality is generally not a practical concern. Seldom with any model will all assumptions be completely satisfied (e.g., independent and uniformly distributed updates in our case) or will all problem parameters be precisely known (e.g., update and retrieval intensity, failure rates, etc.). Moreover, in a changing environment, optimality is always short-lived. An analyst's primary responsibility in database design is to avoid "bad" designs; all "good" solutions are effectively equivalent, and "exact" answers to imprecise problems provide only unwarranted comfort. An exemplary problem serves to clarify the spirit in which we offer our results and the means by which we feel "good" designs can be achieved.

Given a design problem in which updates are reasonably independent and uniformly distributed over an online

database, an analyst may proceed to design an efficient differential file architecture in one of two ways. He may accept the operating parameters of Table 1 as reasonably descriptive of the problem environment and select design values (X*,M*,D*,B*,R*) via interpolation in Tables 2, 3, 4, and 5. Alternatively, the Appendix provides the FORTRAN source code for a program which will accept an arbitrary operating environment and problem description and return parameters for a near-optimal differential file design directly.

Consider a real-time inventory system which maintains 125,000 records and receives an average of 40 updates per minute. Assume that the expected failure rate for any data file is once per month, and that other problem parameters are suitably described by Table 1. For this problem N=125,000 , μ=40/minute, λ=1/month, and R* may be obtained by interpolating on the relevant entries from Table 5 shown below:

Main Data File Size N	λ = 1/Month	
	μ=10	μ=100
100,000	20,000	20,000
200,000	36,000	40,000

Specifically, optimal reorganization intervals R1* and R2* for μ=40 and N=100,000 and 200,000, respectively, are approximated by

$$R^*1 = [(100-40)/(100-10)](20000-20000) + 20000 = 20000 ,$$

$$R^*2 = [(40-10)/(100-10)](40000-36000) + 36000 = 37333 .$$

An optimal R* for 125,000 records is then computed from

$$R^* = [(125000-100000)/(200000-100000)](R^*2-R^*1) + R^*1 ,$$
$$= 24300 .$$

With R* now established, a similar interpolation in Table 4 provides the normalized main data file dumping frequency $\overline{B^*1}$=51.4. Inserting this value and our problem parameters into equation 23, we obtain

$$\overline{B^*} = 0.9\times10^{-5}[(125000)(40)]^{1/2}(51.4) = 1.034 ,$$

and hence, B*=1. In a similar manner, optimal values X*=1, M*=45200 bits, and D*=0 are obtained from Tables 2 and 3.

Operationally, this solution translates to a differential file architecture in which the main data file is reorganized and dumped once a day, while two hashing functions and a 5,700-character bit vector are used as a Bloom filter. Calculations show the expected recovery time for the differential file to be less than 1 minute, while 10.5 minutes is required to recopy the main data file in the

event of loss.[1] An efficient update inplace strategy [7] would also dump the main data file at the end of each day and would cost 3.6 percent less to operate each month. Again, 10.5 minutes is needed to recopy the dump in the event of loss; but now, on average, a much larger 8.2 minutes are required to recover the updates.

With either strategy above, recopying of the main file could be avoided if a second disk copy were made (and moved offline if necessary) at the time the main data file was dumped to tape.[2] A differential file architecture would now look especially attractive, offering approximately an 88 percent improvement in recovery speed. Table 6 summarizes operating statistics for both the conventional and differential file strategies, with and without a duplexed main data file.

Table 6 also demonstrates solution sensitivity to w, the recovery cost weighting factor (see Table 1), by providing statistics for model solutions obtained with each strategy when w takes on alternative values 10, 5, and 1. As w decreases from our assumed value of 10, file dumps are

---

1 Since the times required to diagnose data loss and to load and ready appropriate storage devices are affected by many operational factors, they are not included here.

2 The probability of more than one loss in a day is less than 0.1 percent for this problem.

taken less frequently, while reorganization occurs more frequently and filtering errors decrease as a result. As one would expect, improvement in recovery speed diminishes as this speed is valued less; when w=1 (recovery operations and normal operations have equal weight), the improvement is a modest 7 percent. We clearly observe here that the differential file architecture offers its greatest recovery speed when the main data file is dumped with each reorganization so that transaction reprocessing is not required for its recovery (i.e. B*=1, and T2=0).

w	Design Variables (R*,X*,M*,B*,D*)	Recovery Times (in minutes)			Monthly Operating Cost	Average Filtering Error
		T1	T2	T3		
10	(25000,2,582,1,2)	10.5	0.0	1.00	$13595	38%
5	(4400,2,289,8,0)	10.5	10.2	0.20	$12854	15%
1	(3800,2,269,20,0)	10.5	24.0	0.15	$11711	14%

Table 6(a) - Differential File Architecture

w	Optimal Dump Frequency B*	Recovery Times (in minutes)		Monthly Operating Cost
		T1	T2	
10	24544	10.5	8.2	$13117
5	34710	10.5	11.6	$12090
1	77615	10.5	25.9	$10885

Table 6(b) - Conventional Update Inplace System

w	Increase in Monthly Operating Cost	Improvement in Recovery Speed	
		Single Main Data File	Duplex Main Data File
10	3.6%	38%	88%
5	6.3%	6%	12%
1	7.6%	4%	7%

Table 6(c) - Change in Operating Cost and Recovery Speed

T1 : Time to Recopy Main Data File from Its Dump
T2 : Expected Time to Repost Updates to Main Data File
T3 : Expected Time to Recover Differential File

Table 6 - Model Solutions and Performance Statistics for Different Values of w

5. SUMMARY

Backup and recovery of online databases using a differential file have been discussed and described in terms of an analytic model. Values for five decision variables (number of hashing functions, bit vector size, number of updates before reorganization, number of differential file dumps between reorganizations, and number of reorganizations before main data file backup) combine to define an optimal differential file operating strategy. We describe elsewhere [1] a complex optimization procedure which determines these values precisely, but we believe that such precision is rarely justified. In realistic situations where parameters such as system costs, access intensities, and failure rates must be approximated, the simple design procedure presented here to locate near-optimal solutions quickly is more appropriate.

APPENDIX

This appendix presents the FORTRAN source code used to compute values for R*, X*, M*, D*, and B*. The package consists of a block data, a main program, and three subroutines. The block data initializes the values of the operating environment (which presumably will be modified by an analyst choosing to implement this program). The subroutines CXM, CDR, and CBR respectively return the optimal values (X*,M*), D*, and B* for a given value of R. The main program samples and evaluates R values at increments of 100 updates. That sample R* with minimum cost, and corresponding values for X*, M*, D*, and B*, are printed at the end of the run. Exhaustive enumeration or a more sophisticated search procedure for R* may be easily incorporated through modification of the main program. The remaining code would remain unchanged.

```
      BLOCK DATA
C
      COMMON /PRBPAR/ N,MUE,RHO,L0,L1,C0,D0,D1,R0,R1,
     &                U0,U1,U2,A0,A1,A2,S0,S1,S2,H1,W
C
C N   : size of the main data file
C MUE : average rate of updates to database
C RHO : average rate of retrievals from database
C L0  : rate of main data file failures
C L1  : rate of differential file failures
C C0  : system set up cost for dump and reorganization operations
C D0  : cost of dumping a main data file record
C D1  : cost of dumping a differential file record
C U0  : cost of posting an update during reorganization
C A0  : cost of accessing a record from main data file
C A1  : cost of accessing a record from differential file
C A2  : cost of unsuccessful differential file search
C S0  : cost of maintaining a bit in main memory per unit of time
C S1  : cost of storing a main data file record per unit of time
C S2  : cost of storing a differential file record per unit of time
C H1  : cost of executing a Bloom filter hashing function
C W   : weighting factor for R0, R1, U0, U1
C R0  : cost of restoring a main data file record
C R1  : cost of restoring a differential file record
C U1  : cost of reposting an update to main data file
C U2  : cost of reposting an update to differential file
C
C Assumptions:
C     Unit of time: minute
C     Unit of cost: $
C     A character: 8 bits
C
      REAL MUE,L0,L1
C.....................................................................
      DATA N,MUE,RHO,L0,L1,C0 /100000,100.,0.,0.000083,0.000083,2./
      DATA D0,D1,W,R0,R1 /0.0005,0.0005,10.,0.0005,0.0005/
      DATA U0,U1,U2 /0.01,0.01,0.002/
      DATA A0,A1,A2 /3*0.01/
      DATA S0,S1,S2,H1 /1.7E-06,3.E-07,3.E-07,1.0E-06/
      END


C
C Main line of the FORTRAN package to design
C optimal  Differential  file  architectures
C
      COMMON /PRBPAR/ N,MUE,RHO,L0,L1,C0,D0,D1,R0,R1,
     &                U0,U1,U2,A0,A1,A2,S0,S1,S2,H1,W
      NAMELIST /INPARS/ N,MUE,RHO,L0,L1,C0,D0,D1,R0,R1,
     &                  U0,U1,U2,A0,A1,A2,S0,S1,S2,H1,W
      REAL R,M,MMIN,MUE,L0,L1,REPLY,YES/'Y'/
C.....................................................................
      WRITE(6,1) N,S0,A0,U0,MUE,S1,A1,U1,RHO,S2,A2,U2,
     &           L0,R0,D0,H1,L1,R1,D1,W,C0
   50 WRITE(6,2)
      READ (5,INPARS,END=900)
      WRITE(6,1) N,S0,A0,U0,MUE,S1,A1,U1,RHO,S2,A2,U2,
     &           L0,R0,D0,H1,L1,R1,D1,W,C0
      WRITE(6,3)
      READ(5,4) REPLY
      IF (REPLY .NE. YES) GOTO 50
C
C Take samples from R values, and select the one with minimum cost
C
      CMDSTR = S1/MUE*M
      TOTMIN = 1.0E+10
      MAX = 2*N/10
      DO 100 IR = 100,MAX,100
         R = FLOAT(IR)
         CALL CXM(R,IX,M,BFLCST)
         CALL CDR(R,D,CDFDMP)
         CALL CBR(R,B,CMDDMP)
         CREORG = (U0*N*(1-EXP(-R/N))+C0)/R
         CDFSTR = S2/MUE*(R-1.)/2.
         TOTCST = BFLCST+CDFDMP+CMDDMP+CREORG+CDFSTR+CMDSTR
         IF (TOTMIN .LT. TOTCST) GOTO 100
            TOTMIN = TOTCST
            RMIN = R
            IXMIN = IX
            MMIN = M
            DMIN = D
            BMIN = B
  100    CONTINUE
C
      WRITE(6,5) RMIN,IXMIN,MMIN,BMIN,DMIN,TOTMIN
      GOTO 50
C
  900 STOP
    1 FORMAT('1 Current parameter values are:',//,
     & ' ',I9,'=N       ',F9.7,'=S  ',F9.5,'=U  ',F9.5,'=U',/,
     & ' ',F9.0,'=MUE    ',F9.7,'=S'' ',F9.5,'=U'' ',F9.5,'=U''',/,
     & ' ',F9.0,'=RHO    ',F9.7,'=S" ',F9.5,'=U" ',F9.5,'=U"',/,
     & ' ',F9.7,'=LAMDA  ',F9.5,'=R  ',F9.5,'=D  ',F9.7,'=H''',/,
     & ' ',F9.7,'=LAMDA'' ',F9.5,'=R'' ',F9.5,'=D'' ',F9.1,'=W',/,
     & ' ',F9.2,'=C0')
    2 FORMAT('0Enter any modification to parameter values.')
    3 FORMAT( '0','All OK?(Y/N)')
    4 FORMAT(A1)
    5 FORMAT('0   R*     X*  M*      B*       D*    Total Cost',/,
     &       ' --------- -- ----- ------- ------ ----------',/,
     &       ' ',F8.0,' ',I2,' ',F5.0,' ',F6.0,' ',F6.0,' ',F10.7)
      END
```

```
      SUBROUTINE CXM(R,IX,M,BFLCST)
C
C Find an efficient X-M combination at a given R (The solution
C of 13-1). M is returned in blocks of 100 bits.
C
      COMMON /PRBPAR/ N,MUE,RHO,L0,L1,C0,D0,D1,R0,R1,
     &                U0,U1,U2,A0,A1,A2,S0,S1,S2,H1,W
      REAL R,M,MUE,L0,L1,FSER(15)/1.,1.,2.,3.,5.,8.,13.,21.,34.,
     &55.,89.,144.,233.,377.,610./,CNK(45)/1.,2.,1.,3.,3.,1.,4.,6.,
     &4.,1.,5.,10.,10.,5.,1.,6.,15.,20.,15.,6.,1.,7.,21.,35.,35.,
     &21.,7.,1.,8.,28.,56.,70.,56.,28.,8.,1.,9.,36.,84.,126.,126.,
     &84.,36.,9.,1./
      COSTM(R,IX,M)=(RHO+MUE)*A2*(1-(1-EXP(-R/M*IX))/(R/M*IX))+S0*M
C.....................................................................
C
C Begin with X-M of 0-0, iterate on X,  select an efficient M for
C each case(using Fibbonacci search), and revise X-M if necessary
C
      IX = 0.
      M = 0.
      BFLCST = A2*(1.+RHO/MUE)
C
      DO 200 IXX = 1,9
         XL = 1.
         XR = A2/100.*MUE/S0
         DO 100 K=1,13
            X1 = XL + FSER(14-K)/FSER(16-K)*(XR-XL)
            X2 = XL + FSER(15-K)/FSER(16-K)*(XR-XL)
            Y1 = COSTM(R,IXX,100.*X1)
            Y2 = COSTM(R,IXX,100.*X2)
            IF (Y1 .LE. Y2) XR = X2
            IF (Y1 .GT. Y2) XL = X1
  100    CONTINUE
C
         TM = IFIX((XL+XR)/2.)*100.
         TC = COSTM(R,IXX,TM)
         TC1 = COSTM(R,IXX,TM+100.)
         IF (TC .GT. TC1) TM = TM + 100.
C
C Compute cost of this X-M, and revise optimal X*-M* if necessary
C
         TC = R
         DO 150 J = 1,IXX
  150       TC = TC + CNK(IXX*(IXX-1)/2+J)*(-1)**J*
     &           (1.-EXP(-R*J/TM*IXX))/(IXX/TM*J)
         TC = (1+RHO/MUE)*(A2*TC/R+H1*IXX) + S0/MUE*TM
         IF (TC .GE. BFLCST) GOTO 200
            IX = IXX
            M = IFIX(TM/100.+0.5)
            BFLCST = TC
  200    CONTINUE
C
      BFLCST = BFLCST+(1+RHO/MUE)*(A1+(A0-A1)*(1.-EXP(-R/N))/R*N)
      RETURN
      END



      SUBROUTINE CDR(R,D,CDFDMP)
C
C Compute the optimal number of differential file dumps
C
      COMMON /PRBPAR/ N,MUE,RHO,L0,L1,C0,D0,D1,R0,R1,
     &                U0,U1,U2,A0,A1,A2,S0,S1,S2,H1,W
      REAL MUE,L0,L1
C.....................................................................
      T1 = ((L1/MUE)*W*(U2*R-R1)/2.-D1)*R
      T2 = ((L1/MUE)*W*(R1-U2)/2.+D1)*R
      D = 0.
      IF (T1/C0 .GT. 1.) D=SQRT(T1/C0)-1.
C
      D = IFIX(D)
      CDFDMP = (C0*D+T1/(D+1.)+T2)/R
      TC = (C0*(D+1.)+T1/(D+2.)+T2)/R
      IF (CDFDMP .LE. TC) RETURN
      D = D+1.
      CDFDMP = TC
      RETURN
      END


      SUBROUTINE CBR(R,B,CMDDMP)
C
C Compute optimal main data file backup period
C
      COMMON /PRBPAR/ N,MUE,RHO,L0,L1,C0,D0,D1,R0,R1,
     &                U0,U1,U2,A0,A1,A2,S0,S1,S2,H1,W
      REAL MUE,L0,L1
C.....................................................................
      T3 = SQRT(2.*N*D0*MUE/L0/U1/W)
      B = T3/R
      IF (R .GT. T3) B=1.
C
      B = IFIX(B)
      CMDDMP = (L0/MUE)*(W*R0*N+W*U1/2.*R*(B-1.)) + N*D0/R/B
      TC = (L0/MUE)*(W*R0*N+W*U1/2.*R*B) + N*D0/R/(B-1.)
      IF (CMDDMP .LE. TC) RETURN
      B = B + 1.
      CMDDMP = TC
      RETURN
      END
```

REFERENCES

1. Aghili, H. Differential File Architecture: Analysis and Applications. Ph.D. thesis, The University of Michigan, Ann Arbor, Michigan, to appear.

2. Aghili, H., and Severance, D.G. The Use of Differential Files in an 80/20 Update Environment. Unpublished working paper.

3. Becker, J.R. "EXPLORE: A Computer Code for Solving Non-linear Continuous Optimization Problems." Technical report No. 73-6, IOE Department. The University of Michigan, Ann Arbor, Michigan, 1973.

4. Beightler, C.S.; Phillips, D.T.; and Wild, D.J. Foundations of Optimization. Englewoods Cliffs, N.J.: Prentice-Hall, 1979.

5. Bloom, B.H. "Space/Time Trade-offs in Hash Coding with Allowable Errors." Communication of ACM 13,7 (July 1970), pp. 422-426.

6. Canning, R.G. "Recovery in Data Base Systems." EDP Analyzer 14,11 (November 1976).

7. Lohman, G.M. Optimal Data Storage and Organization in Computerized Information Processing Systems Subject to Failure. Ph.D. thesis, Cornell University, Ithaca, N.Y., January 1977.

8. Lohman, G.M., and Muckstadt, J.A. "Optimal Policy for Batch Operations: Backup, Checkpointing, Reorganization, and Updating." ACM Transactions on Database Systems 2,3 (September 1977), pp. 209-222.

9. Powell, M.J.D. "A FORTRAN Subroutine for Solving Systems of Nonlinear Algebraic Equations." Report No. R-5947 A.E.R.E. Harwell, Didcot, Berkshire, England, 1968.

10. Powell, M.J.D. "A Hybrid Method for Nonlinear Equations." Report No. T.P. 364, A.E.R.E. Harwell, Didcot, Berkshire, England, 1969.

11. Sayani, H.H. "Restart and Recovery in Transaction Oriented Information Processing System." Proc. 1974 ACM SIGMOD Workshop on Data Description, Access, and Control (May 1974), pp. 351-366.

12. Severance, D.G., and Carlis, J.V. "A Practical Approach to Selecting Record Access Paths." Computing Surveys 9,4 (December 1977), pp. 259-272.

13. Severance, D.G., and Duhne, R.A. "Practitioner's Guide to Addressing Algorithms." Communication of ACM 19,6 (June 1976), pp. 315-326.

14. Severance, D.G., and Lohman, G.M. "Differential Files: Their Application to the Maintenance of Large Databases." ACM Transactions on Database Systems 1,3 (September 1976), pp. 256-267.