Proceedings of the 1985 Winter Simulation Conference
D. Gantz, G. Blais, S. Solomon (eds.)

PERFORMANCE ANALYSIS IN A DIFFERENTIAL FILE ENVIRONMENT

Timothy R. Hill and Ananth Srinivasan
Department of Operations and Systems Management
Graduate School of Business
Indiana University
Bloomington, Indiana 47405

ABSTRACT

An analysis of filtering characteristics for database differential file design architecture is presented. Shortcomings of previous work on this topic are discussed and a corrected analytical model is proposed. The impact of assumption violations on the analytical model is explored through the use of a computer simulation. The simulation model is described and results are presented, suggesting the sensitivity of the analytical model and providing some direction for further research.

BACKGROUND OF THE PROBLEM

In a recent paper, Mullin [1] proposed that a mathematical analysis of a problem might yield insights into the problem in a manner that is more efficient than a simulation of the same problem. The particular context of the discussion was a simulation study of the performance of database environments using a differential file access technique that was conducted by Gremillion [2]. In this paper, we attempt to point out some flaws in Mullin's analysis and correct the comparisons of performance indicators that are provided by him, between the simulation and the mathematical analysis.

A brief description of the problem of interest is provided for the reader's convenience. Consider an on-line system with a file (usually very large) containing N records that are accessed and modified on a periodic basis. When a differential file approach is used, modifications are not made in the main file itself; instead the modified records are written into a differential file which, understandably, would be considerably smaller than the main file. At selected points in time a reorganization takes place; the two files are merged making the main file current and the differential file void of any records. The advantages to using this technique are well elucidated in Severance & Lohman [3].

Since not all of the N main file records are up-to-date (when the differential file is non-empty), an incoming transaction must determine if the record that is sought is in the main file or in the differential file. The use of a Bloom filter avoids the necessity of an exhaustive search of both files when a particular record is sought. A Bloom filter consists of a collection of switches (in main memory). Every time a record is placed in the differential file a set of corresponding switches are turned on. The specific switches to be turned on are determined by applying a set of transformations on the identifier of the modified record. During retrieval, the differential file is accessed only if all the relevant switches for a record are on. This does not guarantee that the record resides in the differential file since all of the relevant switches could have been set by transformations that were applied to several previously modified records. The performance of such an environment then, is a function of the number of erroneous differential file accesses that occur.

THE MATHEMATICAL MODEL

Severance and Lohman [3] describe the problem in the following manner. The probability that a random switch will be set in the filter is given by the expression

$$1 - (1 - 1/M)^{kX}$$

where

M = size of the switch array

k = current size of the differential file

X = number of transformations functions applied to each record

The probability of all X switches being set, given the identifier of a specific record is given by the expression

$$p = [1 - (1 - 1/M)^{kX}]^X \quad (1)$$

Further, if there are N main file records in existence, the probability that a given record has not been modified since the most recent reorganization is given by the expression

$$(1 - 1/N)^k \quad (2)$$

The assumptions of this model (that were subsequently challenged by Gremillion [2] and led to the simulation study) include independent updates, uniform distribution of updates across all the main file records, independent and uniform mappings of the transformation functions over the switch array.

THE SIMULATION STUDY CONDUCTED BY GREMILLION [2]

Gremillion [2] used a simulation approach to study the problem after questioning the validity of the assumptions of the mathematical model. The relevant steps in the simulation are as follows.

1. Apply the transformation functions to the identifier of the transaction record.
2. See if any corresponding filter switches are off.
3. If not, search the differential file.
4. If the record is not there, increment the filter error counter by 1.

The measure of performance in the study was the "filtering error rate, [and it] is defined as the number of filtering errors divided by the total number of differential file accesses" [pp. 602].

The objective in Mullin's paper was to compare the performance predicted by the mathematical model (call it P_m) with the performance observed in the simulation (P_s). Given the definition of P_s, measure used in the simulation, the corresponding P_m measure used by Mullin for comparison purposes is incorrect. The P_m measure should indicate the proportion of differential file accesses that are erroneous. The measure used by Mullin, is computed by holding the size of the differential file k constant and equal to its largest value. In fact, k grows from 0 to this value. Given that erroneous accesses are less likely to occur when the k is small than when k is equal to its maximum value, we could expect P_m to be smaller than the figures reported by Mullin. By assuming that every differential file access is erroneous[1], the P figures reported by Mullin should all be equal to 1!

MODIFIED ANALYSIS

Let N_a be the total number of differential file accesses and N_b be the number of erroneous differential file accesses. Then

$$P = E(N_b/N_a)$$

Let X_i be the result of the i^{th} transaction such that

$$X_i = \begin{cases} 1 & \text{if all corresponding switches are set for that transaction record (with probability } p_i) \\ 0 & \text{otherwise (with probability } (1 - p_i)) \end{cases}$$

$$N_a = X_1 + X_2 + \ldots\ldots + X_n = \sum_{i=1}^{n} X_i$$

where n is the total number of transactions.

$$E(N_a) = E(\sum_{i=1}^{n} X_i) = \sum_{i=1}^{n} E(X_i)$$

Since X_i is an indicator random variable, $E(X_i) = p$ and hence,

$$E(N_a) = \sum_{i=1}^{n} p_i \qquad (3)$$

Let Y_i be the result of the i^{th} transaction such that

$$Y_i = \begin{cases} 1 & \text{if the transaction results in an erroneous differential file access (with probability } q_i) \\ 0 & \text{otherwise (with probability } (1 - q_i)) \end{cases}$$

By similar argument as that presented above

$$E(N_b) = \sum_{i=1}^{n} q_i \qquad (4)$$

From (1) we have an expression for p . However, k varies (and, hence, so does p) as modifications take place over time. q is the unconditional probability of an erroneous differential file access. Severance and Lohman propose that q is the product of (1) and (2). The assumption of independence (between a main file record not being modified and all of the corresponding surtiles being set) that is needed for this expression to be true, in fact, does not hold. In the absence of independence between the events of interest, it can be shown that q is given by the following expression:

$$q_i = p_i + (1 - 1/N)^k - 1$$

The rest of the paper outlines a plan for the experimental simulations and compares the predicted performance with the performance observed in the simulation.

THE SIMULATION STUDY

The development of the analytical model included assumptions which leave some doubt as to its suitability for design application. Specifically, the assumptions of independence and uniformity in both the distribution of updates within main file records and the mapping of transformation functions over the switch array are easily challenged by those familiar with actual systems. The conditions seem particularly restrictive since their violation would appear to significantly impact the analytical results. The utility of the model is thus enhanced through experimentation, exploring the model boundaries and providing a basis for some conclusions about the effects of their violations.

A computer simulation model was developed and implemented as a tool for experimentation. The approach involved the random generation of record identifiers, or keys, as input to the filtering process. As the simulated transactions were applied, the "database" was maintained, and those transactions which resulted in erroneous "differential file" accesses were counted. Performance evaluation was based on an approximation to the error rate function described by the analytical model. By partitioning the simulation into equally sized batches of N consecutive transactions, an estimate of the error rate could be calculated for each by dividing the number of errors resulting from those transactions by N. The error rate as a function of the number of transactions processed, k, was thus approximated with the frequency-based estimate for each batch, representing the average error rate over the corresponding range of transactions. In order to provide a common measure for comparative purposes, then, the analytical model was used to produce exact values for the average probabilities of the corresponding batches (given the assumptions.) The performance of the analytical model was thus evaluated in direct comparison to observed results as its assumptions were systematically violated in the simulation.

The simulation was programmed in FORTRAN, and designed to run efficiently by explicitly modeling only the essential elements of the process. Only the record identifiers, or keys, were used for the transactions, since the actual records represent unnecessary overhead. Keys were generated by transformation of psuedo-random numbers into a consecutive set of positive integer values, ranging from 1 to N, where N is the total number of unique keys being considered. This allowed the input stream to be varied easily and

[1] Mullin presents the results after setting $\alpha = 0$ where α is the proportion of modified records. If this is true, there could never be a differential file access since all the switches in the switch array would be off!

provided a high degree of control over the distribution of the hashing operation. In addition, the differential file was easily represented by an array in which the subscripts matched the corresponding keys. In this manner, the inclusion of a key in the file was quickly determined and maintained. The logic also took advantage of the fact that if the key generated for a transaction was already in the differential file, then there was no possibility of an error on that transaction. Therefore, this check was made at the beginning of each transaction, saving execution of the hashing logic when possible.

Finally, only update, or modification, transactions were considered in the simulation. Since a retrieval transaction never alters either the differential file or the switch array, retrievals never change the probability of an erroneous differential file access on the transactions which follow it. Retrievals merely maintain the probability determined by the last previous update transaction. Thus, for a particular system configuration, the error rate function over N update transactions is identical to that over M transactions of unspecified type (updates and retrievals) if N updates are included in the M unspecified transactions (assuming that the retrievals are uniformly distributed among the transactions.) For example, if a system has an error probability of P at 100 updates, and an average of 1/3 of all applied transactions are retrievals, then the error probability is P at 150 transactions of unspecified type. Since this simple rescaling scheme may be used to adjust for the presence of retrievals, these transactions were omitted from the simulation.

Statistics were collected individually for each of the equally sized periods into which the simulation was partitioned. Thus, for a given period, the number of transactions which resulted in error were counted and divided by the total number of transactions per period. This estimate of the average error rate for that period was recorded once for each of several independent runs, or replications of the simulation experiment. An average and standard deviation of the error rate estimate were produced over the multiple runs for each period.

Several preliminary experiments were performed as a basis for the selection of the simulation parameters. As in the analytical model, the function appeared to be of interest mainly in the range from 0 to N transactions, where N is the number of unique keys being considered. It was over this range, generally, that the function rose, peaked, and began to fall off slowly. The range is usefully reported as a percentage of N, rather than absolute numbers of transactions. This generality is possible since, for systems of proportional parameters and distributions, the results are identical, regardless of the particular size, N, of the unique key set. The simulation, then, might easily be performed using a very small key set. However, since the error probability must be estimated by the percentage of observed errors in a batch, a large batch size (500 transactions) was used to better assure normality in the distribution of the estimates over thirty runs or replications. In order to describe the resulting function in sufficient detail, 24 points were estimated over the range. Each run was, therefore, partitioned into 24 batches of 500 transactions. This set the total number of transactions per run at 12,000 and, since the upper limit of the range of interest was previously reasoned to be equal to the size, N, of the unique key set, a set of 12,000 unique keys was used. For the uniform hashing loadings, then, the ratio, R, of the number of unique keys to the number of switches was easily altered from 3 to 2 by using 4,000 and 6,000 switches, respectively. This ratio indicates the number of unique keys which map to one particular switch. As the ratio increases, so does the likelihood of of an error, since a switch turned on by one key may be used to initiate a differential file search for another key which shares that switch.

The simulation was both verified an validated by the use of the analytical model. In Figure 1, the simulation results for the ideal case (all analytical model assumptions met) are plotted as cross-hairs. The exact results are represented by the solid line. The simulation appears, by inspection, to be performing the filtering process accurately.

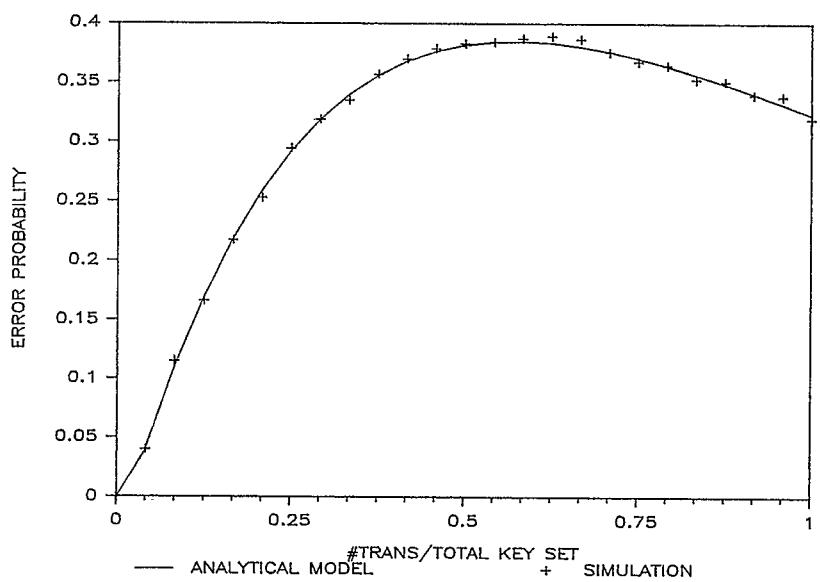

Figure 1: Simulation Validation

Figures 2 and 3 depict the effects, on the error probability function, of uniformity violations in the input stream of transaction keys. In Figure 2, the 12000 keys are hashed into 4,000 switches one time, yielding a loading ratio, R, of 3. In Figure 3, the ratio is 2 since 6,000 switches were available. In each case, the solid line represents the analytical results. The cross-hairs correspond to he simulation results with 1/4 of the unique key set 4 times as likely to occur as the remaining 3/4. The circular symbols denote the simulation results with 1/4 of the unique key set 8 times as likely to occur as the remaining 3/4.

As indicated by inspection, there does not appear to be an important effect from the change in the loading ratio, R, on the pattern of results. The actual values are lower with R equal to 2, producing a peak of only about .26 versus the peak of .38 in the case of R equal to 3. Of greater interest is the general effect of non-uniformity on the observed results. The introduction of non-uniformity into the input skews the error probability, producing a more quickly rising error rate as the input non-uniformity increases. The skewed peaks are somewhat smaller than the exact results dropping from .38 to .32. In an actual system, reorganization (re-initialization) would probably be performed as the error rate increased to an upper limit of acceptability, prior to the peak. Thus, system designers will find that as user activity tends to focus on a subset of the unique key set, the analytical model will tend to underestimate the error rate. For example, consider a period of update transaction processing during which the number of transactions processed is equal to 1/24 of the number of unique keys in the key set. In Figure 3, the average error rate observed during the fourth such period after reorganization is underestimated by the analytical model as about 22% while the observed results are as high as 30%. Users of such a system would realize degradation of system performance, characterized by extended response times.

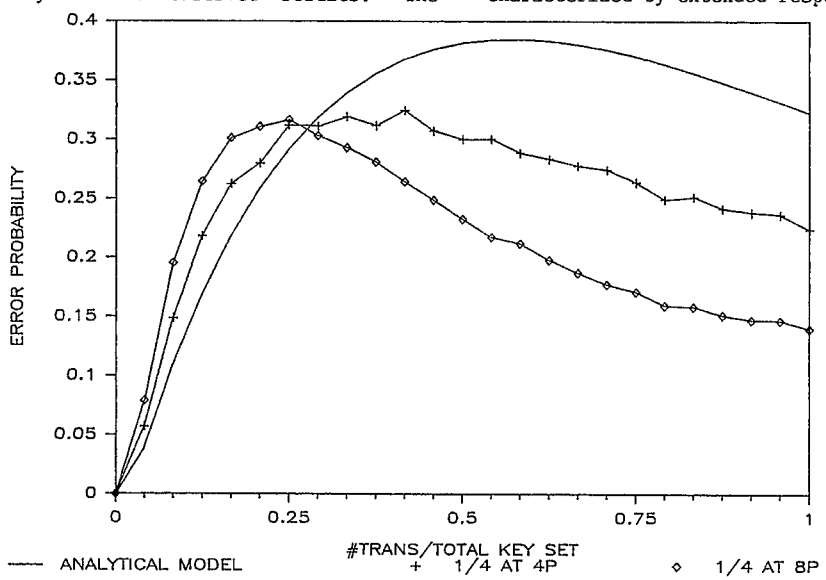

Figure 2: Effects of Key Distribution with R = 3

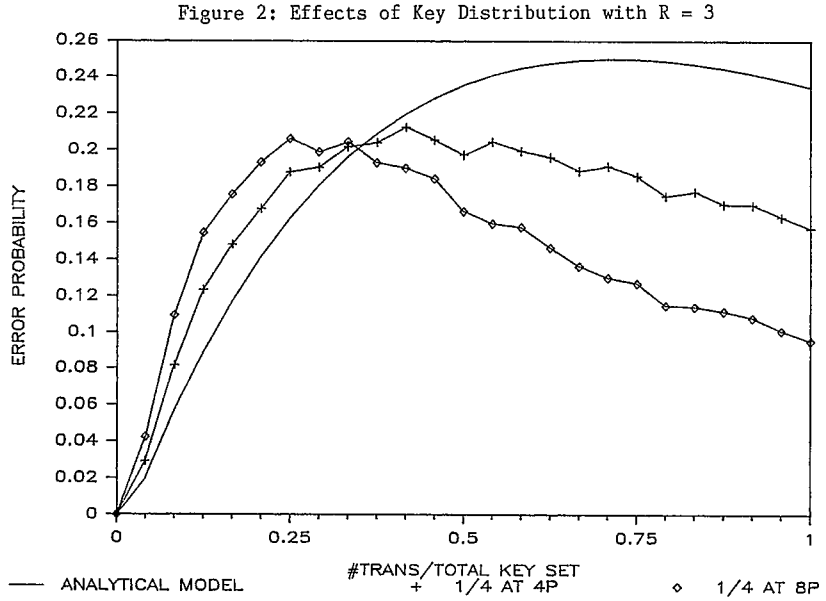

Figure 3: Effects of Key Distribution with R = 2

In Figure 4, the effects of non-uniformity in the distribution of keys into the switch array is investigated. In this case, the simulation was configured to generate uniform input of keys, but the hashing transformation mapped 1/2 of the keys with a loading ratio of 4 keys per switch, while the remaining 1/2 were mapped with 3 keys per switch. The cross-hairs represent the results of the simulation. The solid line shows the exact results for a uniform loading of 3 keys per switch, while the circles denote the exact results for 4 keys per switch. Inspection indicates that the observed results are bounded by the exact results on each side. A conservative estimate of the error rate could thus be derived from the analytical model for a system with uniform input by using the largest loading ratio among the switches.

REFERENCES

1. Mullin, J. K., "A Second Look at Bloom Filters," Communications of the ACM, Vol. 26, No. 8, August 1983, pp. 570-571.

2. Gremillion, L. L., "Designing a Bloom Filter for Differential File Access," Communications of the ACM, Vol. 25, No. 2, September, 1982, pp. 600-604.

3. Severance, D. G., and G. M. Lohman, "Differential Files: Their Application to the Maintenance of Large Databases, " ACM Transactions on Database Systems," Vol. 1, No. 3, September, 1976, pp.256-267.

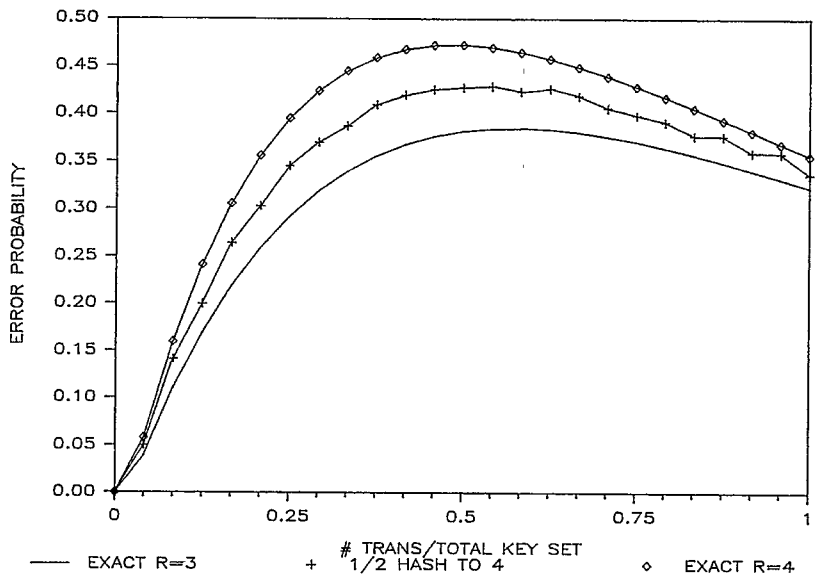

Figure 4: Effects of Hashing Distribution

CONCLUSIONS AND IMPLICATIONS

The analytical model developed corrects the previous model and provides a sound basis for research aimed at refining the differential file system design process. The experimental results provide some indication of the applicability of the model to actual systems. Violations of the assumptions have an impact on the results which increases with the degree of violation and may be realistically significant. The simulation modeling features and the pattern of results suggest directions for further experimentation by the system manager tuning an actual system as well as the researcher attempting to generalize the analytical model. Subsequent research efforts might focus on a general measure of the quality of a given hashing scheme based on the switch loading scheme in relation to the input distribution. Such a measure should account for the number of hashing transformations and their degree of independence, their ability to differentiate between keys, and their tendency to work best for the most heavily accessed subset of the unique key set. The effects of multiple hashing transformations and combinations of assumption violations could then be easily explored and reported in very general terms for direct use in system design.

ANANTH SRINIVASAN
Ananth Srinivasan is an Assistant Professor of Operations and Systems Management in the School of Business, Indiana University. He received his Ph.D in MIS from the University of Pittsburgh in 1983. He is a member of SIM, ACM, AIDS, and TIMS.

Department of Operations and Systems Management
Graduate School of Business
Indiana University
Tenth and Fee Lane
Bloomington, Indiana 47405

TIMOTHY R. HILL
Timothy R. Hill is a doctoral student in the department of Operations and Systems Management, Indiana University. He completed an M.S. in Industrial Engineering from Purdue University in August, 1985.
(812)-335-8449

Department of Operations and Systems Management
Graduate School of Business
Indiana University
Tenth and Fee Lane
Bloomington, Indiana 47405
(812)-335-0900