OPERATIONS RESEARCH
Vol. 26, No. 5, September-October 1978

0030-364X/78/2605-0789 $01.25

A Performance Model for Computer Data-Base Reorganization Performed Concurrently with Usage

GARY H. SOCKUT
National Bureau of Standards, Washington, D.C.

(Received October 1976; accepted February 1978)

Characteristics of data bases, reorganization, concurrent reorganization, and data-storage devices are described briefly. We present a two-priority queuing model for concurrent reorganization and usage and derive and solve equations for steady-state expectations of user response time and the time required to perform reorganization. Numerical results are obtained, and performance characteristics of such a system are predicted quantitatively. We discuss the effect of parameter values on performance and predict that for typical values of the model's parameters, system performance will be at an acceptable level.

A *data base* [4] is a collection of data that can be used for many applications in a computer system. Its usage is managed by a *data-base management system,* as shown in Figure 1. Users can access the data base via their application programs or via interactive query facilities. The data-base management system reads and writes portions of the data base as requested by users. It also maintains structures that indicate how data items are interrelated and structures that define how data items are stored [4].

As a data base is used over a period of time, its internal storage structures may deteriorate gradually from an initially optimal state, and performance of the data base degrades with respect to response time and/or storage utilization. Hence the data base must be reorganized periodically to restore those structures to an optimal state. A data base may also be reorganized to perform logical changes (e.g., changing data from values in inches to values in centimeters) or to perform "one-shot" physical changes (e.g., changing the method by which data are encoded).

Typically, a data base (or at least the portion that is to be reorganized) is taken *offline* (i.e., is temporarily made unavailable for normal usage) for several hours overnight or over a weekend while reorganization is performed. This strategy, however, is unacceptable for an essential computer utility that is to be available 24 hours per day (e.g., a military information system) or for a data base that is so large that reorganization

could not be performed over a weekend (e.g., a census data base). In addition, many businesses prefer to have 24-hour availability of their data bases, even if this is not absolutely necessary. In such cases it is appropriate—and in the future, as these types of data bases become more common, it will be necessary—to perform reorganization concurrently with normal usage of the data base, i.e., users are granted full access to all the facilities of the data base while it is being reorganized. Reorganization algorithms and usage algorithms must be modified to accommodate concurrency (e.g., by including appropriate synchronization), but such modifications are not covered in this paper. Several such modifications and other aspects of concurrent reorganization are described in [6].

There has been research in modeling performance deterioration and in determining the time required to reorganize offline (e.g., [5]), but there

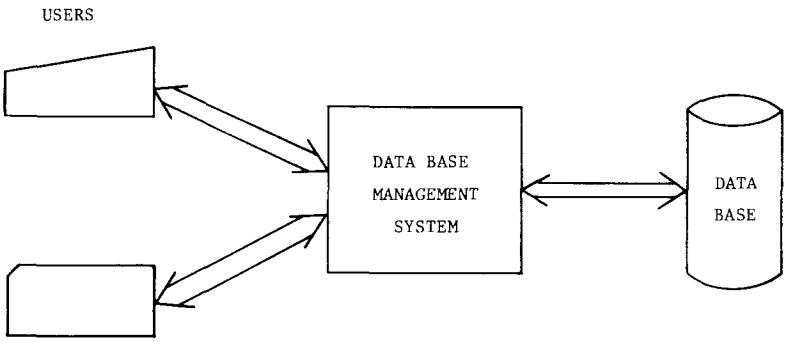

Figure 1. A data-base management system.

has been no research in modeling performance for concurrent reorganization. In this paper the performance of a data base is modeled when reorganization is performed concurrently with usage. A two-priority queuing model is used, with activity by users having high priority and data-base reorganization having low priority. The model is used to answer the following questions:

1. What will be the effect of concurrent reorganization on user performance?
2. How long will concurrent reorganization take to perform?
3. What will be the effect of changing the size and hence number of steps into which reorganization is divided?
4. What will be the effect of changing the user load on the data base system?

1. DISKS

The design of the model was influenced by the characteristics of a *disk* [1], which is the principal type of device currently used to store a data

base. As shown in Figure 2, a disk consists of a number of platters that are mounted on one axle and spin at constant angular velocity. Each platter contains two *disk surfaces* (one on the top and one on the bottom). Each disk surface contains a number of concentric *tracks* on which data are stored magnetically. Associated with each surface is a *read/write head,* which is attached to a movable *disk arm.* All the disk arms move together. The set of tracks that are accessible at a given arm position (one track for each surface on each platter) is referred to as a *cylinder.* The number of tracks per cylinder is, of course, equal to the

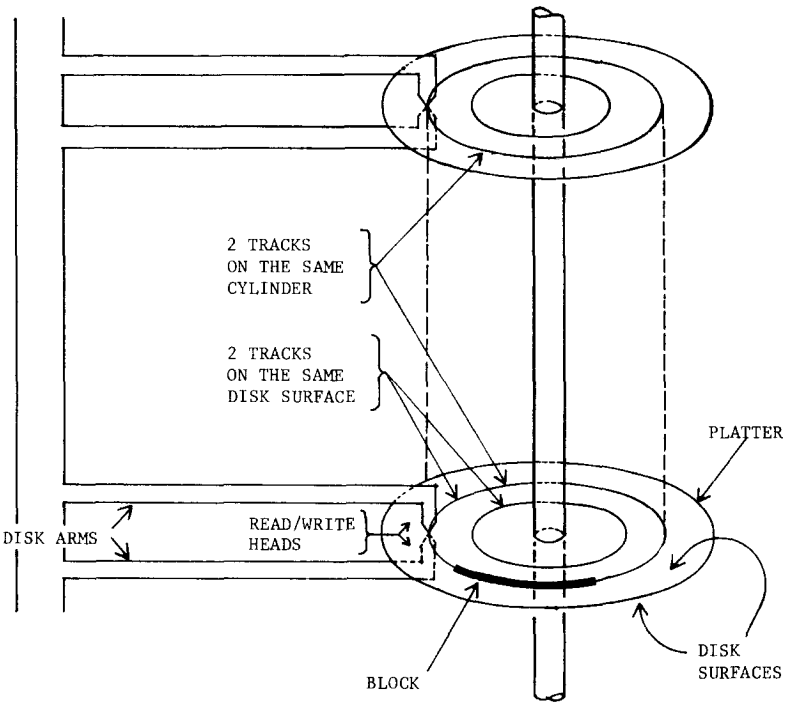

Figure 2. A disk.

number of surfaces on the disk. When it is desired to read or write a *block* (a contiguous stream of data) on a certain track, the following steps are performed:

1. The arms are moved so that the read/write heads are positioned at the cylinder that contains the track that contains the block to be read or written. This movement is called *seeking.* If the arm is already at the desired position, this step is omitted, thus saving a considerable amount of time. The time required for seeking increases (not necessarily linearly) with the distance to move.
2. Some overhead time is experienced.
3. The read/write mechanism waits until the block begins to pass

under the read/write head. This waiting time is called the *rotational latency*. The mean rotational latency is half of the disk's rotation period.

4. As the block passes under the read/write head, the actual reading or writing is performed. This time is called the *data transfer time*. The data transfer time depends upon the amount of data to be read or written (i.e., the number of blocks and the length of a block). In general, all of the blocks that are used to store a data base are of the same length.

The following numerical parameters are typical of modern disks and are used in later calculations: The rotation period is 16.7 msec. There are 19 usable tracks per cylinder. (The odd number appears because one track is reserved for use as a spare.) There are 404 cylinders (or, equivalently, 404 tracks per surface). The time to seek a distance of 1 cylinder is 8.0 msec. The *mean* seek time (assuming that consecutive disk accesses are uncorrelated with respect to cylinder position and that all cylinders are equally likely to be used) is 26.832 msec. The overhead time is 0.465 msec. Rotational latency is assumed to be uniformly distributed between 0 and the disk's rotation period and thus has mean 16.7/2=8.35 msec. There are four blocks per track. Thus the time to transfer one block is 16.7/4=4.175 msec.

This paper describes an analysis of a single-disk data base. Usually more than one disk is used to store a data base. A performance analysis of a multiple-disk data base is described in [6], but the analysis is too complicated to explain concisely here since some disk operations (e.g., seeking) can be performed concurrently by the different disks, while others (e.g., data transfer) cannot. The form of the results for a multiple-disk data base resembles that of a single-disk data base.

2. SERVICE BEHAVIOR

Figure 3 shows the two-priority non-preemptive queuing model that is to be used. The server is a disk, which operates as described above. For tractability, service behavior is assumed to be exponential, which is not true but often serves as a good approximation in computer performance models.

Users (requests for service by application programs or by query facilities) are the high-priority customers. Requests are assumed to arrive as a Poisson process with intensity λ. They form a first-come first-served queue. Such an *open* queuing model is most accurate for low utilizations and/or a large population of customers [2]. The disk's service rate for users is μ_U. Service for a user usually includes seeking. For the case of reorganization performed to improve user service time, μ_U is taken to be the *mean* service rate for users.

Reorganizers (requests for reorganization service) are the low-priority customers. There is a reorganization process that, when its service is

complete, immediately requests service again (until the entire data base has been reorganized). A reorganization service is referred to later as a reorganization *step*. When one or more users arrive, the reorganizer that is then being served (referred to later as the *last* reorganizer in a sequence of reorganizers) is allowed to complete its service before the users are served, but the next reorganization request is not served until all users waiting in the queue have been served. The service rate for reorganizers depends upon whether the previously served customer was a reorganizer or a user, as described in the next paragraph.

When consecutive reorganization services are performed, the service time is short since consecutive reorganization steps are usually performed

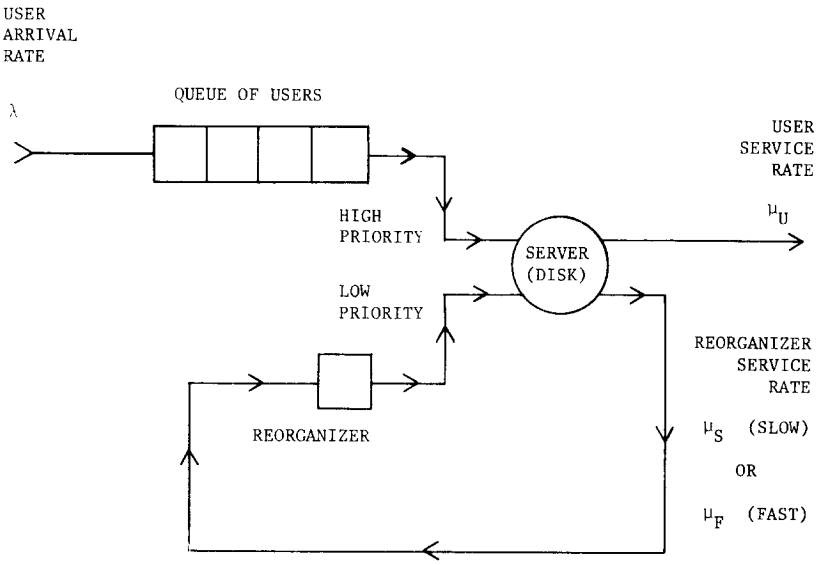

Figure 3. The queueing model.

within one cylinder (which requires no seek) before moving to the next (adjacent) cylinder (which requires only a short seek of one cylinder). However, after a user is served (which usually requires a seek), the next reorganizer must perform a seek in the opposite direction to return the disk arm to the cylinder within which reorganization had been taking place. For this reason, the first reorganization service immediately after a user has been served has service rate μ_S (slow), but subsequent (consecutive) reorganization services have rate μ_F (fast). Appropriate values are substituted for λ, μ_U, μ_F, and μ_S in Sections 6 and 7.

The model is used to describe an environment in which consecutive user requests are uncorrelated with respect to cylinder location and in which conventional rotating storage devices (disks) are used. It is assumed

that reorganization is performed within one cylinder at a time (i.e., reorganization within one cylinder is completed before reorganization within another cylinder is started). This assumption is valid for reorganization in several existing commercial data-base management systems. User-reorganizer competition for central processor time is ignored. Techniques that are used in this model could be used for more complex environments, as the above assumptions are relaxed.

3. THE SOLUTION TO THE QUEUING MODEL

Figure 4 shows the states of the queuing system and the transitions between them. Each transition is labeled with the intensity of the associated stochastic process. There are three infinitely long columns of states. The center (A) column contains states in which a user is being served. The right (B) column contains states in which the current customer is a reorganizer, but the previous customer was a user. The disk performs slow reorganization service when it is in a B state. The left (C) column contains states in which the current customer is a reorganizer and the previous customer was a reorganizer. The disk performs fast reorganization service when it is in a C state. Each column of states is indexed by the number of users at the disk (including the user being served, if a user is currently being served). For example, in state B_2 the previous customer was a user, a reorganizer is being served now (at the slow rate), and there are two users waiting in the queue. There is no state A_0.

The memorylessness of the arrival and service processes implies that the state of the queuing system is a Markov chain. The chain is clearly irreducible, as can be seen from Figure 4. The steady-state distribution, which exists if $\lambda<\mu_U$, is obtained by solving the set of flow conservation equations that are associated with the Markov chain. This analysis yields formulas for the steady-state probabilities. Since the formulas are long and are not intuitively meaningful, they are not listed here. The name of a state is used in later sections to represent the steady-state probability that the disk is in that state.

4. EXPECTED USER RESPONSE TIME

Since the steady-state probabilities have been derived, it is possible to calculate the expected number of users N at the disk and expected user response time R (i.e., the total time between a user's arrival in and departure from the system). For comparison, we first perform these calculations for a disk when there is no concurrent reorganization. This is a standard $M/M/1$ queuing system, and we have a *normal* response time R_{NORM} of $1/(\mu_U-\lambda)$. For convergence, we require $\lambda<\mu_U$.

Next we calculate the expected number of users N and user response

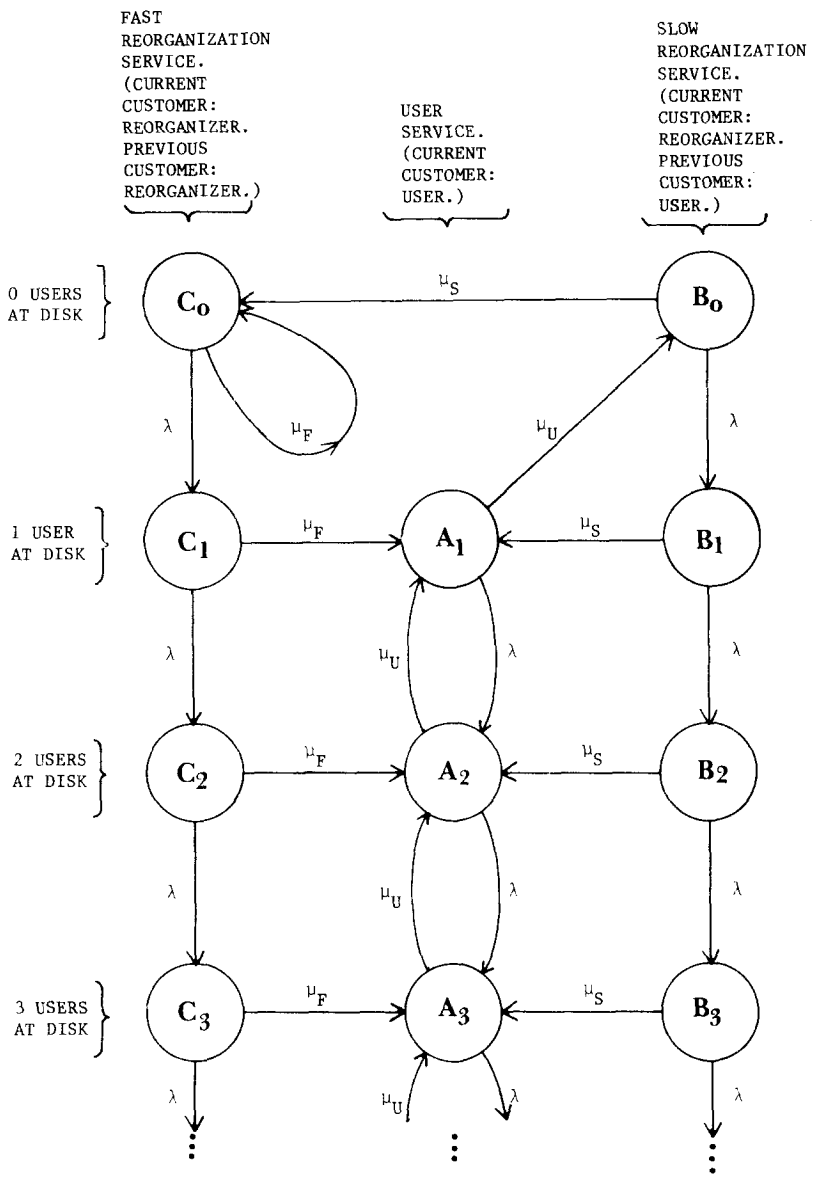

Figure 4. The state transition diagram.

time R for concurrent reorganization: $N=\sum_{i=0}^{\infty} i(A_i+B_i+C_i)$, where $A_0\equiv 0$. Using Little's result [3], we divide N by λ and simplify to obtain

$$R=\frac{((\lambda+\mu_F)\mu_S{}^3+\mu_F{}^2\lambda\mu_S+\mu_F{}^2\lambda^2)\mu_U+(\mu_F{}^2-\lambda^2)\mu_S{}^3+\mu_F{}^2\lambda\mu_S{}^2-\mu_F{}^2\lambda^3}{(\mu_U-\lambda)\mu_F\mu_S((\lambda+\mu_F)\mu_S{}^2+\mu_F\lambda\mu_S+\mu_F\lambda^2)}.$$

If $\lambda=0$, the expression for expected user response time R reduces to

$(1/\mu_U)+(1/\mu_F)$, indicating that a user must wait only for the last reorganizer's service and for its own service since there is no time spent waiting for other users to complete their service. As λ approaches μ_U, R and R_{NORM} both grow without bound since the queue grows, and the expression for the ratio (R/R_{NORM}) reduces to 1. This indicates that response time is not significantly worse than it would be without concurrent reorganization since users wait only for previous users' (and their own) services to complete, and no reorganization service is performed. Thus the response time formula is consistent with intuition at the limiting values of λ.

5. EXPECTED REORGANIZATION TIME

Another interesting quantity is the expected time required to perform one step of reorganization. If reorganization is performed *offline,* the expected number of steps of reorganization performed during a period T is $T\mu_F$ since all reorganization service is at the fast rate. Setting this expression equal to 1 and solving for T, we find that the time T for one offline reorganization step is $1/\mu_F$.

If reorganization is performed concurrently with usage, the expected number of steps of reorganization performed during a period T is $T(\mu_S$(fraction of time spent in B states)$+\mu_F$(fraction of time spent in C states)$)=T(\mu_S \sum_{i=0}^{\infty}B_i+\mu_F \sum_{i=0}^{\infty}C_i)$. Setting this to 1 and solving for T, the expected time per concurrent reorganization step, we obtain $T=1/(\mu_S \sum_{i=0}^{\infty}B_i+\mu_F \sum_{i=0}^{\infty}C_i)$, which simplifies to $T=[((\lambda+\mu_F)\mu_S{}^2+\mu_F\lambda\mu_S+\mu_F\lambda^2)\mu_U]/[\mu_F\mu_S((2\lambda+\mu_F)\mu_S+\lambda^2)(\mu_U-\lambda)]$.

If $\lambda=0$, the expression for concurrent reorganization time T reduces to $1/\mu_F$, which is the offline reorganization time since if no users arrive, all reorganization is performed at the fast rate μ_F. As λ approaches μ_U, the expression for concurrent reorganization time grows without bound since all service is performed for users and thus reorganization service requests are never satisfied. Thus the reorganization time formula is consistent with intuition at the limiting values of λ.

6. PARAMETER VALUES

The two parameters that are varied are λ and N_B (the number of blocks that a reorganizer transfers in one reorganization step). We assume that a user always transfers exactly one block during a disk service. The three service rates can be expressed as explained below. Numerical values are as given in Section 1. As explained there, disk service consists of seek, overhead, rotational latency, and data transfer.

The mean seek time for users or slow reorganizers is 26.832 msec. since an arbitrary seek can be performed. A fast reorganizer's seek time is usually 0, as explained in Section 2. A seek is needed only if reorganization

has just been completed for one cylinder, at which time a seek is required to move to the next (adjacent) cylinder. Since there are four blocks per track and 19 tracks per cylinder, and since each block must be read and written once in a complete reorganization, the total number of blocks that must be transferred to reorganize an entire cylinder is (2)(4)(19). Since a reorganizer performs N_B of these transfers in each step, the probability that a fast reorganizer must perform a seek is $N_B/((2)(4)(19))$. The time to seek one cylinder is 8.0 msec. Thus a fast reorganizer's mean seek time is $(8.0)N_B/((2)(4)(19))$ msec., or $N_B/19$ msec.

The mean overhead is 0.465 msec. The mean rotational latency is 8.35 msec. The mean data transfer time is 4.175 msec. (i.e., one block) for users and is $(4.175)N_B$ for fast or slow reorganizers. Therefore, we have:

$$1/\mu_U = 26.832+0.465+8.35+4.175 \text{ msec.}$$
$$1/\mu_S = 26.832+0.465+8.35+(4.175)N_B \text{ msec.}$$
$$1/\mu_F = N_B/19+0.465+8.35+(4.175)N_B \text{ msec.}$$

7. RESULTS

We varied λ between 0 and μ_U (i.e., the user utilization ρ was varied between 0 and 1) and gave N_B values of 1, 2, and 4; 4 is a typical value for offline reorganization. Smaller values of N_B might be used for concurrent reorganization in order to reduce user performance degradation, as will be shown later in this section.

For each pair of values (ρ, N_B), the following results were computed:

Normal user response time (i.e., with no concurrent reorganization).

User response time (i.e., with concurrent reorganization).

Relative degradation of user response time (i.e., (time with concurrent reorganization—normal time)/normal time).

Offline reorganization time (i.e., time per step of offline reorganization).

Reorganization time (i.e., time per step of concurrent reorganization).

Relative degradation of reorganization time compared to offline reorganization time, for the same value of N_B (i.e., (concurrent reorganization time—offline time)/offline time). This figure represents the degradation in the time required to reorganize the entire data base, for a constant value of N_B.

Relative degradation of reorganization time per block compared to offline reorganization time per block for $N_B \equiv 4$ (i.e., (concurrent reorganization time /N_B-offline reorganization time (for N_B=4)/4)/(offline reorganization time (for N_B=4)/4)). This figure represents the degradation in the time required to reorganize the entire data base, given that N_B can be varied. For comparison, we assume that offline reorganization normally uses N_B=4 (which is true for at least one existing commercial data-base management system).

Figures 5 through 9 illustrate some of the results that are described above. The results might be useful to the manager of a data-base installation, who can use them together with his own criteria for desired

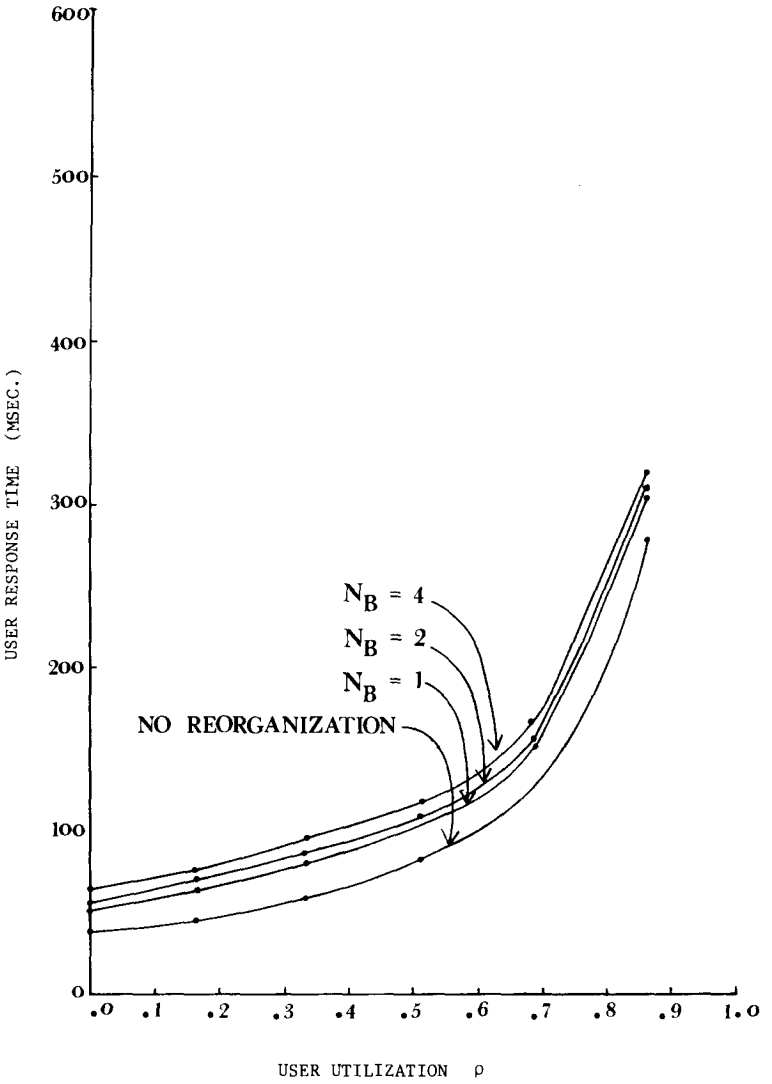

Figure 5

maximum response time and reorganization time to determine what types of storage devices to use, how much work should be performed in one step of reorganization, and what maximum user utilization should be allowed in the system.

Figure 5 shows user response time (in milliseconds) versus user utilization. The three uppermost curves represent user response time with concurrent reorganization for N_B=4, 2, and 1. The bottom curve represents user response time with *no concurrent reorganization* and thus is

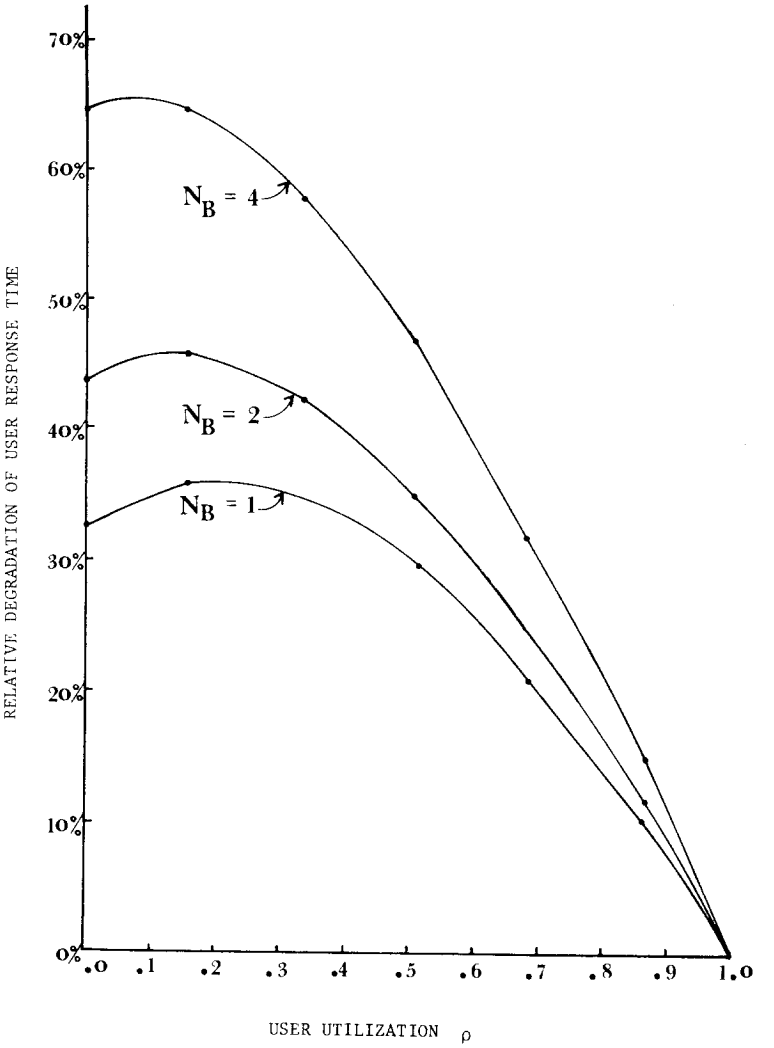

Figure 6

a lower bound on the response time with concurrent reorganization. The curves are all of the same form. Response time with or without concurrent reorganization increases monotonically as utilization increases, and response time grows without bound as user utilization approaches 1 since the queue grows.

Figure 6 shows the relative degradation of user response time versus user utilization. The three curves represent N_B=4, 2, and 1. Relative degradation of user response time generally decreases to 0 as user utilization increases to 1. This is because a greater fraction of time is spent serving users, and thus response time is closer to normal response time since most waiting time is spent waiting for other users rather than waiting for the last reorganizer. Response time increases as N_B increases

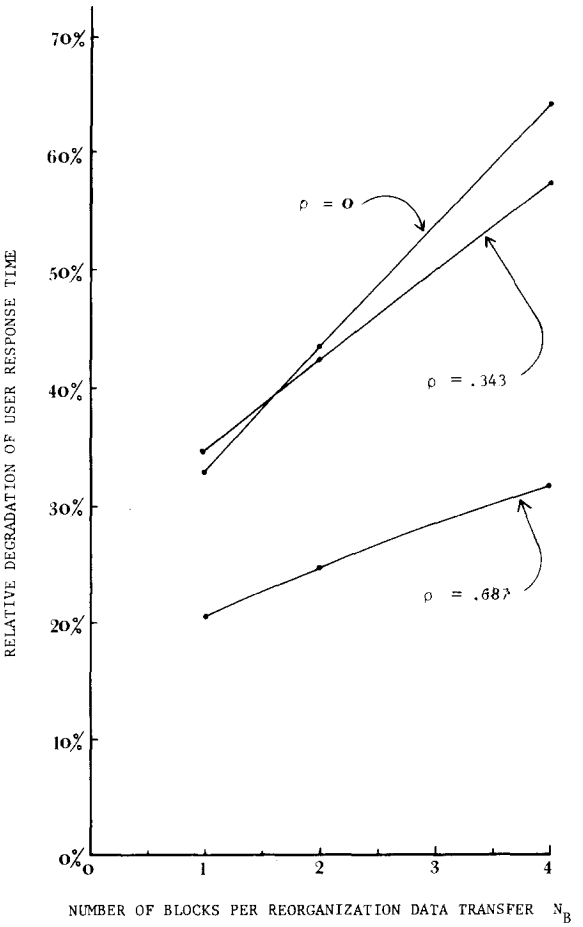

Figure 7

since users must wait longer for the last reorganizer to complete its service, but for high user utilization, reorganizers are served so rarely that N_B has little effect on relative response time degradation. For N_B=1 and ρ=0.5 (a reasonable figure), degradation is 30%, which can be considered reasonable performance, especially if the computer system's principal bottlenecks are in other areas such as terminal input/output rather than (or as well as) in disk activity.

Figure 7 shows the relative degradation of user response time versus N_B, for ρ=0, 0.343, and 0.687. Degradation increases almost linearly with N_B since users must wait longer for the last reorganizer to complete its service. The intersection of the two topmost curves is due to the different points of peaking in Figure 6 and should be ignored.

Figure 8 shows the relative degradation of *reorganization* time (for a

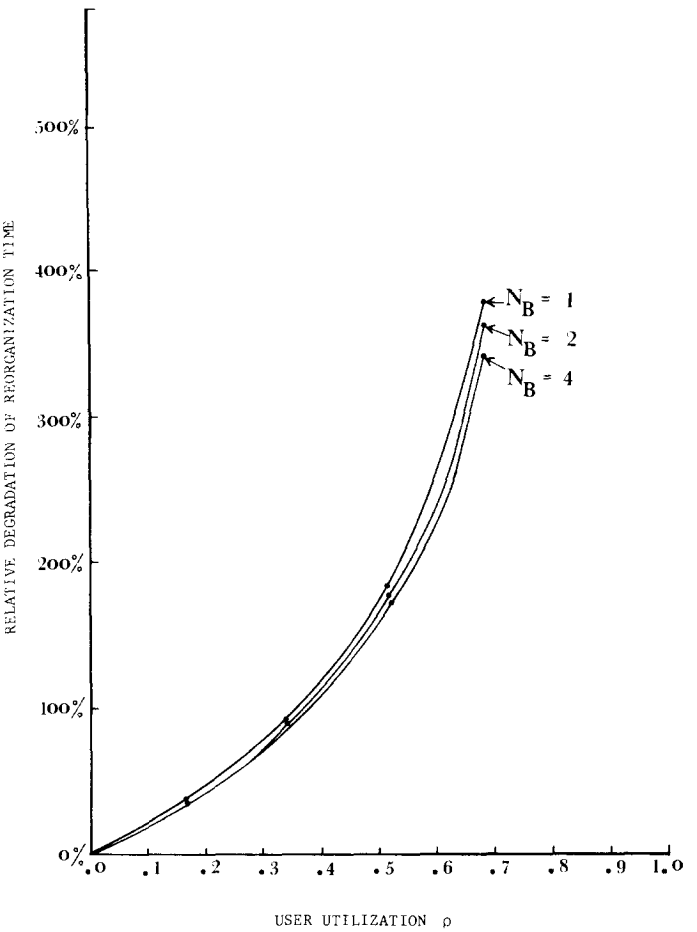

Figure 8

constant N_B) versus *user utilization.* The three curves represent N_B=1, 2, and 4. The curves are all of the same form. For ρ=0 (which is offline reorganization), the reorganization time degradation is always 0 since all reorganization service is performed at the fast rate. Reorganization time increases without bound as user utilization increases to 1 since the disk devotes more time to users and less time to reorganizers, and more of the remaining reorganization services are slow services.

Figure 9 shows the relative degradation of *reorganization time per block* versus N_B, for user utilizations of 0.687, 0.343, and 0 (offline). As N_B increases, the time to reorganize the entire data base decreases since

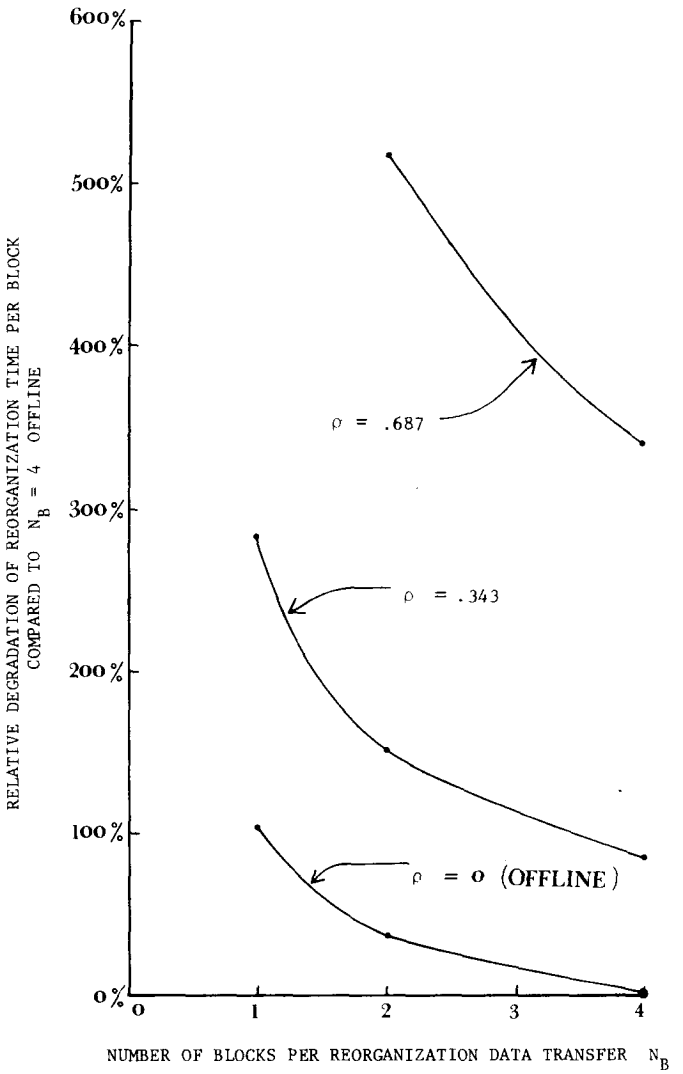

Figure 9

there is less total time spent in overhead, in rotational latency, and (for concurrent reorganization) in seeking. For N_B=1 and ρ=0.343 (a reasonable figure at night), reorganization time degradation is less than 300%, which is probably a reasonable performance level, considering the avail-

able amount of time (e.g., for one particular data base, offline reorganization now requires 30 hours and is performed four times per year).

8. CONCLUSIONS

Our major conclusions are:

1. Results generally agree with intuition.
2. As user utilization increases, user response time and reorganization time increase. Therefore, if the user load varies (e.g., it is low at night), then reorganization should probably be performed mainly during relatively slack periods.
3. As N_B (the amount of work performed in 1 reorganization step) increases, response time increases and the time to reorganize the entire data base decreases. Therefore, if user performance is more important than reorganization performance (as is probably the case), a wise policy might be to use small N_B (if we have the freedom to vary it).
4. For typical values of ρ and N_B, both user response time and reorganization time have values that can be considered reasonable in many situations. Thus, for environments to which the model is applicable, concurrent reorganization can be feasible from a response time point of view and from a reorganization time point of view.
5. This research has produced a first step toward a thorough performance analysis of concurrent reorganization.

9. DIRECTIONS FOR FUTURE RESEARCH

A more general treatment of concurrent reorganization performance modeling is given in [6], which covers multiple disks, multiple data transfers per disk service, multiple seeks per disk service, and use of the computer's central processing unit.

Future research might include more parameters (e.g., synchronization interference between users and reorganizers), different reorganization scheduling policies (e.g., waiting a specified length of time between consecutive reorganization services, in order to reduce the probability of delaying a user), and unconventional storage devices with behavior different from that of disks.

ACKNOWLEDGMENTS

I would like to thank Jeffrey P. Buzen, Robert P. Goldberg, and Peter P.-S. Chen for assisting in the development of the model and its parameterization and for reviewing an earlier draft of this paper. The referees also suggested improvements. This work was supported in part by the Advanced Research Projects Agency under Contract N00039-76-C-0168.

REFERENCES

1. J. P. BUZEN, "*I/O* Subsystem Architecture," *Proc. Inst. Elect. Electron. Engrs.* **63,** 871–879 (1975).
2. J. P. BUZEN AND P. S. GOLDBERG, "Guidelines for the Use of Infinite Source Queuing Models in the Analysis of Computer System Performance," *Proc. Nat. Comput. Conf.* **43** (Amer. Fed. Info. Proc. Societies), 371–374 (1974).
3. J. D. C. LITTLE, "A Proof for the Queuing Formula $L=\lambda W$," *Opns. Res.* **9,** 383–387 (1961).
4. J. MARTIN, *Computer Data-Base Organization,* Prentice-Hall, Englewood Cliffs, N. J., 1975.
5. B. SHNEIDERMAN, "Optimum Data Base Reorganization Points," *Comm. ACM,* **16,** 362–365 (1973).
6. G. H. SOCKUT, "Data Base Performance Under Concurrent Reorganization and Usage," Ph.D. thesis, Division of Applied Sciences, Harvard University, November 1977.