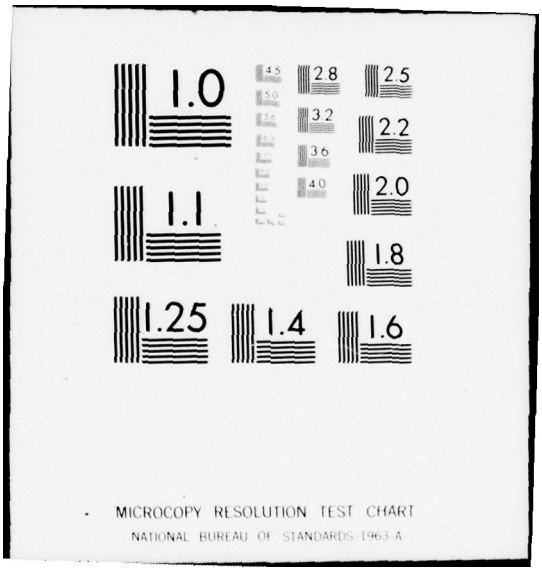
AD-A042 894 ILLINOIS UNIV AT URBANA-CHAMPAIGN CENTER FOR ADVANCED--ETC F/G 9/2
RESEARCH IN NETWORK DATA MANAGEMENT AND RESOURCE SHARING, INITI--ETC(U)
AUG 75 G G BELFORD, D A WILLCOX, J D DAY DCA100-75-C-0021
UNCLASSIFIED CAC-169 JTSA-5511 NL

1 OF 1
ADA042894

END
DATE FILMED
9-77
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Center for Advanced Computation

CAC Document Number 169
JTSA Document Number 5511

Research in
Network Data Management and
Resource Sharing

Initial Mathematical Model Report

August 20, 1975

CAC Document Number 169
JTSA Document Number 5511

Research in Network Data Management and Resource Sharing

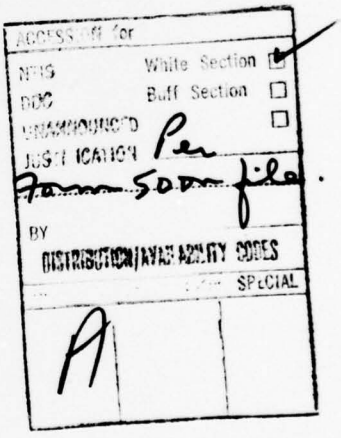
Initial Mathematical Model Report

by

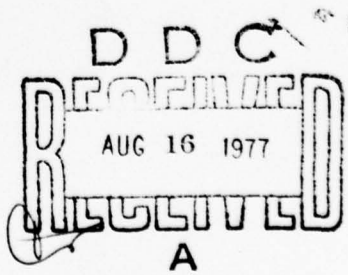
Geneva G. Belford
John D. Day
Suzanne Sluizer
David A. Willcox

Prepared for the
Joint Technical Support Activity
of the
Defense Communications Agency
Washington, D.C.

under contract
DCA100-75-C-0021



Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801



August 20, 1975

Approved for release:

Peter A. Alsberg, Principal Investigator

Table of Contents

Summary

Overview

This document is a preliminary report on one aspect of the initial phase of a proposed three-year research program on distributed data management. The work dealt with here is the development of models for data distribution. These models consist of equations for system cost, availability, and response time in terms of appropriate parameters describing system behavior, usage patterns, etc. This interim report deals with models which look at the system from a very high level. Low-level features - strategies, policies, etc. - will be built in later so that their effects on cost, response, and availability can be assessed.

Cost Model

Besides providing a tool for further research, the modeling effort has yielded some immediate insights into the advantages - and problems - of distributed data management. We have found, for example, that data distribution can be cost effective - in the sense that it may be actually cheaper to store at a remote site - for reasonable parameter values and a not excessive cost differential between sites. In addition, it appears that this result is fairly insensitive to the size of the data set to be transferred, but it does require that the data be compressed for shipment.

Availability Model

Perhaps the most interesting result of the availability study is the following. If there are two copies of the data base (located at different sites) and both are kept immediately accessible and as up to date as possible, at least one copy of the data base is available more than 99 percent of the time. (This result does not take into account

scheduled down-time or the (small) possibility that both sites are down concurrently.) If the remote copy is not a true "running spare" - i.e., is an inactive backup stored, say, on tape - the improvement in availability seems hardly enough to make such backup worthwhile. This result serves to emphasize the importance of developing techniques for on-line data base synchronization.

Response Model

The study of response time has led to some simple relations which should be useful in algorithms for determining when a site should share its query load with other sites holding a copy of the data base. Rather than being amenable to a priori study, the parameters appearing in this model are envisioned as being provided by real system monitoring and measurement, so that they are appropriate for decision making in a dynamic environment.

Report Format

In the next section we discuss the goals of the modeling program, both in the long term (as a research tool) and in the short term (i.e., for the work presented here). Following this, we briefly review work reported in the literature which seems pertinent to our effort. The major part of the document is then devoted to detailed reports on the three models: cost, availability, and response time, in that order.

Report Validity

The reader should note that the results given in this report are to be considered tentative. The models are in the process of revision and refinement, as well as more thorough testing. This is only a preliminary report; conclusions reached from the models in their present state should not be relied upon or widely disseminated.

Goals of the Modeling Program

Long-Term Goals

Developing models to describe the various aspects of distributed data management is an integral part of our research program. Model building - the development of equations to describe system behavior, costs, etc. - is an essential tool in computer science research. Using a good model, one can study alternative design options, compare decision strategies, etc.

It is important that the model effectively reflect the real world that is to be studied. For this reason, we plan to build a model which is highly modular and highly parameterized. The modularity will provide flexibility and allow us to study some aspects of the problem independently of having a detailed model of a whole system. For example, network problems of synchronization and deadlock can probably be studied through a high-level networking model which does not concern itself with details of data management. The parametrization will allow us to put into a high-level model guessed values for the effects of lower-level systems. In this way we can generate some insights into what is going on before building a complete model. Parametrization also should allow us to mimic real PWIN system behavior as closely as available measurement data will allow. This will maximize the PWIN-relevance of our research results.

The actual research areas which we plan to study in part through modeling have been described in some detail in our Research Plan (CAC Doc. No. 164, JTSA Doc. No. 5510). In the interests of brevity, we will not repeat that information here.

Goals of this Preliminary Study

In this preliminary work, we have been limited by time constraints to the construction of fairly superficial models and to the identification of some promising directions for further work. In order to get a good feel for the broad range of model components that may be useful to us, we planned a three-pronged effort, directed towards assessing the gross effects of data distribution on costs, availability, and reponse time. Our approach has been to survey the modeling literature for work which seemed relevant to our program and to begin to extend such work to study the problems of distributed data management.

In order to develop a cost model for network data distribution, we have begun with a cost model of a hierarchical storage system and extended it by including storage at a remote site as part of the hierarchy. We have then attempted to identify the major cost components which the network introduces into total cost. By carefully including these network components, we hope to have designed a basic model which we may then expand on (by providing a greater level of detail) to study such things as the cost overhead of various synchronization strategies.

To study availability, we have taken a similar approach. We have begun with a model for single-site data base recovery strategies and tried to see how the assumptions and results of that model are affected by locating the backup copy at a remote site.

The complexity of response-time models (involving, as they usually do, heavy usage of stochastic analysis and queueing theory) precluded our getting very deeply into this area in the short term. Instead, we undertook a rather superficial investigation into the conditions under which mutiple data copies and query distribution may lead to an improvement in system responsiveness.

In summary, the primary goal of this preliminary effort has been to gain an understanding of the components needed to model the major features of a distributed data management system. But we feel that the models developed, although crude, do form a solid basis for future work and have already provided us with some insights into the value of data distribution.

Models in the Literature

Introduction

The first step in our modeling program has been to look closely at relevant models reported in the literature. Actually, modeling is an extensively used tool in computer science, and models of one sort or another are found throughout the literature. For example, models are used for comparisons and evaluations of alternative system designs. Such mathematical design analysis is much less costly and time-consuming than actually building alternative systems and trying them out. Models are also heavily used in optimization studies. For example, optimal (or near optimal) file allocations can be derived from rather simple formulas for cost and response time. The simplicity of the formulas, it should be noted, is not a drawback of the model but an advantage. Working with models instead of the complex real world allows one to focus attention on only those features that one wishes to study.

The models that we review in this section are therefore generally not complex and all-encompassing, but are simple formulas which seem to have some relevance to problems of interest to us. The discussion is organized primarily according to the output of the model, and only secondarily according to its context or application. First, we consider response-time models and, under this heading, other types of models dealing with time delays (e.g. in a network) or the time it takes a process to be carried out. Thus we have collected together various models which may have some relevance to the overall response time of a distributed data base. Second, we consider very briefly the closely related concept of throughput, and techniques for modeling it.

Third, we review various models which may be useful to the study of data availability - essentially the probability that the data base is accessible when needed. Overall availability may involve such factors as network reliability, system failures, and recovery strategies - all of which have been individually modeled in some context. Finally, we look briefly at cost models - valuable for their ability to encompass all kinds of resource utilization, but not nearly so extensively studied as the other types of models.

Models for Response Time

An important quantity for the evaluation of a data management system is the expected response time, which may be defined as the average waiting time from the initiation of a data request (or from the input of a query) to the receipt of the information. Many different aspects of a data management system have an effect on the total response time. These aspects run from the low-level physical organization of data to (in a distributed environment) network delay times. In this section we briefly review some important past work on modeling these various aspects and indicate where further work appears needed to model a complete system.

Data structure modeling. At the lowest level, models have been developed to aid in choosing storage schema. A typical approach is that of Gotlieb and Tompa [1974]. They consider a number of alternative structures - trees, linked lists, etc. - and assume an expected usage pattern which involves the probabilities that the various nodes in the schema will be accessed. They then compute expected run-time costs for the alternatives. These "costs" are actually timing estimates, being computed as a simple linear combination of "the number of executions of each of three primitive instruction types: memory accesses, arithmetic

and logical instructions, and transfers of control". The expected number of executions of the different commands are obtained by simulation studies of application programs. This last point is an important one - at some stage in almost any modeling effort, data from simulations or from the measurement of real system behavior is needed. Another point to note in Gotlieb and Tompa's work is that some very important considerations in choosing storage schema appear only as constraints. For example, an upper bound on the allowable amount of storage space is used to eliminate certain schema from further scrutiny. A model which would allow for a trade-off between storage cost and access efficiency would seem to have more validity.

At what is perhaps a higher level, Shneiderman [1974] has developed a model for optimizing the structure of multilevel indexes. Again, he describes his model as a "cost" model, but he is explicitly computing access times. His approach is a very simple one. Assuming

1. a given number of levels,
2. the branching pattern of the index tree,
3. a strategy for searching the tree,
4. the costs (times) for moving from node to node in the search, and
5. an equal probability of request for all items,

he derives a straightforward algebraic formula for expected search time. As an obvious (and necessary) generalization, he suggests relaxing assumption (5). Shneiderman's basic approach, however, appears to be a useful one which may be readily incorporated into any analyses of tree structures which arise in our modeling effort.

A more ambitious effort in the use of modeling to evaluate file structures was carried out by Winkler and Dale [1971]. In their words, they study "the processing time required to evaluate Boolean functions defined on data values ... [and to] select elements from the structure satisfying the expression". They derive some rather complex algebraic formulas for expected processing time. There are over twenty input parameters describing such things as properties of the average query, file size and timing data. Specific, alternative structures are modeled in the sense that processing time formulas are developed for them. This paper merits closer study in our proposed work on data structuring.

Computer system modeling. Many of the response-time models in the literature that may be of use to us are not specifically concerned with data management but with computer systems in general. Both time-sharing systems and multiprogramming systems have been the subject of considerable analysis. Both situations are characterized by competition for shared resources. Several jobs reside in the system simultaneously and must occasionally wait for processing, I/O, etc. The natural mathematical models to describe the progress of jobs through such a system of waiting lines and processors are those of queueing theory. Indeed, queueing theory has been heavily and successfully used to develop formulas for response time in such systems.

A classic example is Scherr's analysis of response time for time-sharing systems [Scherr, 1967]. Scherr defines response time as the mean length of time the user spends in the "working part of the interaction" - i.e., the time between when he finishes typing in his query and when the response is returned to him. The main input parameters

are the mean time per interaction that the user spends in thinking and typing, and the mean processor time per interaction. Simplifying assumptions are that the system is in a steady state (i.e., essentially that the total number n of users on line is constant) and that there is no overhead due to additional swapping as n increases. The latter assumption is questionable and leads to the result that response time increases only proportionately to n for large n. The mathematical analysis is quite simple. A Markov process describes the probability distribution for the number of users actually inside the system and the resulting set of recursive equations are readily solved. Expected response time can be immediately calculated from this probability distribution. The validity of this simple queueing model was demonstrated by comparing its predictions with real system measurements. The agreement was extremely close.

More elaborate analyses of time-sharing systems have been carried out by Kleinrock. (For a good review of this work, see [Kleinrock, 1973].) Kleinrock's analyses include various queueing disciplines (scheduling algorithms) and various probabilistic assumptions on job arrival times and processing time required. He has extended this type of model virtually to its limit, in the sense that further generalizations lead to intractable mathematical formulations.

Queueing models also play a key role in the study of multiprogramming systems. These differ from time-sharing systems primarily in that there is no assumed interlude when the user is thinking and typing. That is, a certain steady-state population of jobs is assumed to be continually moving through the system. A model which seems

especially relevant to our work is that of Arora and Gallo [1971], who are particularly interested in the optimal storage of data in a multi-level memory. They define the expected response time of a transaction as "the serial sum of the service times along with the respective waiting times at all facilities", and emphasize the importance of this statistic in evaluating data management systems. The most important parameters in their model are the I/O dependent timings, such as the access times to various memory devices and the time required to transfer a block of data from auxiliary to main memory. The model is rather detailed and complex, but has the obvious potential to be extended to the study of data distribution in a network. One need only consider some memory levels to be located remotely and take into account network delay times.

File allocation modeling. Models which have been devised to study data distribution are usually developed from higher level (and less sophisticated) analyses than those referred to above. An example is the response-time formula derived by Chu [1973] in his study of optimal file allocation in a network. Variables in his formula include line traffic between nodes (assuming it is all generated from data base access), usage rates of files by users at various sites, and average lengths of messages. An interesting feature is the result that network transmission delays increase with line traffic according to the simple factor P/(1-P), where P denotes the fraction of line capacity used by the given traffic, or traffic intensity. Indeed, Chu's expected response-time formula (for queries initiated at one given site and responded to by another) is simply

$$\text{Response Time} \simeq tP/(1 - P),$$

where t = average time to transmit a reply message. One sees that many

features are lacking in this simple model - time to transmit requests, time to access the data at the remote site, protocol overhead time, etc. In addition, there appears to be an implicit assumption that all pairs of sites are connected by a direct line used only for the query traffic.

Network delay modeling. As we noted in discussing Chu's simple model for response time from a remote site, real network delays involve many complex factors. Fortunately, much work has been done on developing realistic formulas for network delays. (For a good review see [Kleinrock, 1973].) This work has been largely done in the setting of network design and analysis. For example, queueing models have been used to compute average packet delays for given network topology, routing strategy, and network traffic (including overhead for routing, flow and error control, etc.). There seems to be no reason why such models can not be incorporated into overall models of response time for a distributed data base. Detailed modeling of network delays will provide a necessary tool for studying synchronization strategies and other network-related features of distributed data management.

Models for Throughput

Some authors argue that response time is not as important a statistic as is throughput. For example, Arora and Gallo [1971] put the case as follows: "In a multi-programming environment the response time does not measure the efficiency of the system, because of the concurrent processing of several transactions. For this reason, we introduce throughput rate as a performance measure for the multi-programming systems. It is the rate of completion of transactions per unit time." (The underlining is ours.) In his analysis of multiprogramming systems, Buzen [1971] takes the same point of view, defining "overall system

performance" as the "average number of jobs processed per unit time". A queueing theory analysis will, however, generate either response time or throughput rate with equal ease. That is, these models assume that a certain number of jobs are in the system and essentially it is the time the average job spends in the system that is computed. Thus "response time" in these models never means the absolute time it takes an otherwise empty system to do the job, but is always in the context of competition with other jobs.

In network analysis throughput has also been a useful statistic. For example, in ARPANET analyses the network throughput has been defined as the average traffic per node when average packet delay equals 0.2 seconds [Frank and Chou, 1974]. This maximum acceptable average time delay then gives meaning to the notion of throughput or "the level of traffic that the network can handle". However, it is again queueing analysis which is used to model the flow of packets through the network and to compute throughput under various conditions.

Once throughput or level of traffic flowing through a system becomes a statistic of interest, the possibility of using models analogous to those used for physical flow systems arises. The stochastic models and recursion equations of queueing theory may be replaced by the continuous models and differential equations of diffusion theory. There has recently been considerable interest in applying this type of model to queueing networks (see, for example, [Reiser and Kobayashi, 1974]), since more complex initial and boundary conditions can be imposed than are tractable in stochastic models. Of course, the close connection between throughput and average response time means that diffusion models could be very useful in response-time studies. We therefore plan to look closely into the applicability of diffusion models to our research.

Models for Availability

We here use the term availability to mean the fraction of time that a data base is available to respond to user requests or queries. In any setting, and particularly in a network, availability is a function of the reliability (or availability) of many components - host computers, network communications lines, etc. - as well as of strategies for backup and recovery. In this section we discuss some of the past modeling research that has yielded results useful to us in our concern with database availability.

File allocation modeling. One of the factors to be taken into account in distributing copies of a file to various network sites is the number of copies needed for an acceptable degree of availability. Chu [1973] takes account of this factor in the following way. First, he defines the availability of a piece of equipment (e.g., communication line or computer) as

$$\text{Availability} = \frac{F}{F + X},$$

where F is the mean time between failures and X is the mean time to repair. Then, assuming

1) all computers in the network have identical availability A,

2) all communication channels have identical availability c, and

3) the network is completely connected;

Chu obtains the following formula for the availability of the jth file:

$$A(1 - (1 - Ac)^{r_j}),$$

where r_j is the number of copies of the jth file in the network. Once A and c are known, it is a simple matter to choose r_j so as to bring the availability of a remote copy up to a satisfactory level. Overall

availability is bounded by A, the availability of the requesting computer, which is apparently assumed not to possess a copy of the file.

Although Chu's model, with its assumption of complete homogeneity of network components, may seem oversimplified, an analogous analysis can be readily carried out in the heterogeneous case to yield more complex expressions. Notice, however, that this model presents another problem. It implicitly assumes that the files are static, or are simultaneously kept up to date by some trouble-free process. In fact, the development of algorithms to keep segments of a data base identical (or nearly so) is a topic of current research. (See the chapter on Automated Backup in CAC Doc. No. 162, JTSA Doc. No. 5509.)

Network reliability modeling. Another simplification in Chu's model is the assumption that a direct communication line connects every pair of sites. This assumption allows Chu to use a single parameter to describe availability of a link from one site to another. In a general network, this availability will depend in a complex way upon network topology. Several alternate paths may exist between two given sites. Each of these paths may involve more than one "hop" and so more than one piece of subnet hardware. Indeed, in the ARPA network it has been found that the failure rate for IMP's is about the same as that for communication channels, and that IMP failures therefore have the more drastic effect on communications reliability [Frank, Kahn, and Kleinrock, 1972]. Graph theoretical techniques for computing availability from component reliabilities are, however, well known. The paper by Frank et al. contains a brief review of these techniques. No great difficulty is envisioned in applying them to any given network (such as the WIN) to obtain availabilities which may then be used in a straightforward extension of Chu's model to obtain rough estimates of file (or data base) availability.

Modeling computer system reliability. Another parameter in Chu's model that requires more detailed analysis for complete understanding is computer availability. One source of information on computer availability is direct system measurement. On a lower level, however, failures can be modeled to yield, in addition to overall figures on expected system reliability, useful insights into repair and backup strategies.

Borgerson and Freitas [1975] recently published a fairly detailed stochastic model for computer system failure. Their model is based on four distinct causes of crashes and their interrelationships. Their ultimate result is a formula giving the probability density for the event that the system crashes due to a failure. The effects of mechanisms for detecting and recovering from a failure (before the system actually crashes) are included in the analysis. Although our research is unlikely to be concerned with modeling computer systems at this level of detail, the analytical techniques of Borgerson and Freitas may well apply to reliability problems which we may wish to model (e.g. protocol resiliency).

Modeling backup and recovery strategies. This section has previously dealt with availability questions involving network and site reliabilities. On a lower level, the data base itself may "crash" or may acquire errors. It is important that strategies for returning a data base to its correct state be devised and studied.

A recent paper [Chandy et al., 1975] provides models for rollback and recovery strategies. These strategies run as follows. At certain points in time (checkpoints), a copy of the data is made and stored. A listing of subsequent data updates (i.e. an audit trail) is

then kept. When the master data base fails, it may then be recovered by beginning with the old copy from the checkpoint and using the audit trail to bring it up to date. Chandy et al. use queueing theory to model the processing of the audit trail. From the expected time to complete this process, they can compute the total recovery time. The length of the audit trail, and hence the time to recover, is a function of the time interval between checkpoints. Optimization of availability with respect to intercheckpoint time can then be carried out. Models of some complexity are developed which take into consideration the possibility of errors during recovery and the possibility of a transaction arrival rate which varies in a cyclic manner (as opposed to being constant). The results appear to be very useful for developing insights into recovery strategies, particularly for single-site systems. In a network environment, however, it may be reasonable to assume that the backup copy is stored remotely. In this case it does not make sense to assume that the data is always restored from the backup, because of the long time required to transfer a data base through the network. The strategy then is to transfer the queries to the available copy. (See the later section on the availability model.)

Models for Cost

Cost is both a very vague and ambiguous measure of system performance and a very important one. The ambiguity comes about through the difficulty of assigning dollar costs to all factors of interest. One way, of course, is to carry out experiments - i.e., to run test programs at various sites and compare the bills received. This method yields cost comparisons which are heavily dependent on the pricing policies of the various sites, as well as on site hardware and software.

Untangling all of these factors to determine what a set of cost figures really means is no easy task. On the other hand, cost is very important in that it serves as an overall measure of system resource utilization. For example, by assigning costs to them, such diverse factors as CPU time and storage used can be added together. In short, costs are a device by which one can add together apples and oranges.

Assignment of specific costs to various factors is of importance to the model user, but not necessarily to the model builder. The latter can consider costs of various resources to be simply weighting coefficients, which can be adjusted at will to reflect a specific environment. It may be, for example, that no real money changes hands. But a user may still wish to evaluate a certain system or piece of software by using a formula which weights storage (which may be in short supply) much more heavily than CPU time.

In this brief review of cost models, we will be only concerned with those which use "costs" to add together heterogeneous factors. In our search of the literature we found that several so-called "cost" models actually dealt only with time factors. Such models are therefore discussed elsewhere.

Modeling Network File Allocation. Of particular relevance to our study of distributed data management are the cost analyses developed for the network file allocation problem. A good example of such an analysis is that given by Casey [1972]. The parameters in his model are

1. the cost ("mainly for storage") of locating the file at any site k,
2. the costs of transmitting a given amount of data between two given sites (with the possibility that update and query transactions may be transmitted at different costs),

3. the amount of update traffic emanating from each site, and

4. the amount of query traffic emanating from each site.

Given values for these parameters, the cost of a particular allocation is readily computed.

Casey states that transmission costs may be "a rather complex monotonically increasing function" of traffic, but he feels that his linear model is a good first approximation. A better idea of transmission costs would require a model which goes into the transmission process in some detail and analyzes the various cost components and how they are affected by the amount of network traffic. The site costs might also profit from a detailed breakdown; note that Casey remarks that factors other than storage are being lumped into one term. It is important to realize, however, that for file allocation Casey's model is probably quite adequate. It is only when one wishes to study other aspects of data distribution - backup and recovery strategies, say - that more detail is needed.

Modeling storage hierarchies. Even before networks existed, the file allocation problem was of importance. The question arose as to where one should place a given file in a storage hierarchy - i.e., a set of memory devices of varying accessibility (core, disk, tape, etc.) connected to a single computer. A particularly comprehensive cost model for this problem has recently appeared [Lum et al., 1975]. This model differentiates between random and sequential forms of data access and includes considerations of staging, channel costs, CPU overhead, etc. Because of its completeness, we considered this model an appropriate one for extension to the network case. That is, memory devices at a remote site may simply be considered as parts of the storage hierarchy, provided

that network costs are properly taken into account. A detailed discussion of the model of Lum et al. will therefore appear below in the discussion of our cost model.

The distributed data management problem is of course far more complex than the storage hierarchy problem. The model of Lum et al. (and our extension of it) assumes that all data processing (updating and responding to queries) takes place in local core. No provision exists for sending a query to a remote site for processing. Thus, although our straightforward extension of Lum's storage hierarchy model has provided some insight into data distribution, it is grossly inadequate for studying all the many facets of distributed data management. Unfortunately, the literature contains little modeling work that is readily applicable to distributed data management. Much more work needs to be done to develop models which realistically describe the distributed environment.

A Cost Model for Data Distribution

Introduction

The advantages of distributing a data base in a network environment have been discussed at length in various papers, panel discussions, and bull sessions. But it has been somewhat difficult to quantify these advantages or to investigate the various tradeoffs and determine just how great the advantages are. In this section, we will attempt to shed some light on this subject. As mentioned above, a recent paper by Lum et al. [1975] develops a cost algorithm for allocating files in a storage hierarchy. Their cost model is rather complete and lends itself well to extensions relevant to storage hierarchy problems in distributed data base systems.

For many of the cost-related questions that arise in the development of a distributed data base system (such as those concerned with the costs of queries, updates, back-up, recovery, etc.), the system can at first be viewed as a storage hierarchy. That is, to a local process or user the remote sites appear as further levels of the hierarchy. From this point of view the network is another channel with some special cost considerations. In future refinements of this model, we plan to include effects of remote processing of data. We were unable to do so in this short-term effort.

In what follows we will first review the model described in [Lum et al., 1975]. (In order to facilitate the discussion, this model will be referred to henceforth as the LSWL model.) Next we will extend the LSWL model to include a network. Then we will use the model along with some relevant data to investigate some interesting questions, and

we will draw some conclusions about the advantages of distributed data base systems. Finally, we list some ideas for refining the model.

A Review of the LSWL Model

Overview. The LSWL model primarily addresses the problem of "data staging" or "data migration". In other words, when a file or data set is not being used (i.e., is inactive) it is stored on one device (usually a slower, less expensive one). Then, when the data set is accessed, it is moved to a faster, more expensive device so that the program will waste fewer resources waiting for data. The question we are concerned with here is, given the accessing characteristics (number of reads and writes, proportion of time the file is in use, etc.), where in a given hierarchy should the data set be stored when it is inactive and where should it be moved when it is active?

The authors develop an objective function which gives the cost of accessing a data set which is stored on one device when inactive and another (possibly the same device) when active. The authors assume as a first approximation that the entire data set is moved from the inactive device to the active one.

The selection algorithm is quite straightforward. The objective function is evaluated for a given set of variables for each pair of devices in the hierarchy. The lowest cost then indicates on which pair of devices the data should be located.

Assumptions. The authors make several simplifying assumptions, most of which can be relaxed at the cost of a more complex cost function. They assume that for data sets system paging activity will not significantly affect cost. However, it would probably be necessary to relax this constraint if one wished to consider costs incurred by program activity. They further assume that transfers are direct rather than

through core and that there are no flow control problems (i.e., a fast device can always accept data from a slow device). It is also assumed that transfers are not constrained by the capacity of the device the data set is being moved to. These last two assumptions can both be dropped at the cost of a more complex equation. As we shall see, when we add a network to the hierarchy, flow control can not be ignored.

When a process or user accesses a data set, it often must wait for the access to complete. Clearly, this wait time must be figured into the total cost. However, multiprogramming systems take advantage of this wait time by letting other processes utilize the processor. To account for this the authors define an adjusted machine cost, m. For lack of a better formulation, they have defined this cost to be the percent of CPU idle time times the dollar cost associated with the CPU. There are some problems with such a definition. For example, as the load on the system increases and so does CPU utilization, queueing delays and system overhead also increase, thus increasing cost. The objective function does not account for this phenomenon.

The objective function. Now that we have reviewed the assumptions behind this analysis, let us look at the cost function itself in some detail. The reader should consult Table 1 for a key to the definition of the symbols used and Figure 1 for a summary of the objective function.

Let us assume that the data set is at level i of the hierarchy when inactive and level j when active. (For consistency we will adopt the nomenclature used by Lum et al. whereby the first subscript will be the inactive device, and the second the active one. Also the higher levels (i.e., those with faster access) of the hierarchy will have

Data Set Characteristics:

q = number of sequential block assesses.
r = number of random block accesses.
S = data set size.
s = physical block size.
τ_i = fraction of time data set is on level i.
d = number of times the data set is opened.
λ = the proportion of time to write the data set back to its original position. For read only data sets, $\lambda = 0$; for full write back at read speed $\lambda = 1$.

Storage Device Characteristics:

t_r^i = random access time for level i.

t_q^i = sequential access time for level i.

t_s^i = transmission rate for level i.

t_l^i = average revolution latency time for level i.

t_c^i = minimum access arm movement time.

n_i = unit cost of storage space at level i for the given time period.

b_i = transfer size per access when data set is being moved from a lower level i to another level (or from a higher level to level i).

B_i = largest size that can be transferred without additional access cost.

CPU and Channel Characteristics:

m = adjusted cost per unit time for computer system excluding channel
M = unadjusted computer system cost per unit time
u = cost of channel per unit time
β = number of buffers
W = computer setup time for opening a data set

Table 1

Parameters in the LSWL Model
(from [Lum et al., 1975])

higher indices.) The objective function can be considered to have three major terms:

$$f_{ij} = \begin{matrix}\text{storage}\\ \text{cost}\end{matrix} + \begin{matrix}\text{local process}\\ \text{access costs}\end{matrix} + \begin{matrix}\text{staging}\\ \text{transfer}\\ \text{costs}\end{matrix}$$

The first term is the cost of storing the data on the active and inactive devices.

$$\{\text{storage cost}\} = [\tau_i n_i + \tau_j n_j]S$$

When a data set is moved from level i to level j it is not necessarily deleted from level i; therefore it should be noted that $\tau_i + \tau_j \geq 1$.

The second term is the cost for the user or process to access the data from the active device. This term takes into account the CPU costs and transfer overhead as well as channel costs for both random and sequential accesses.

$$\begin{matrix}\text{CPU costs for}\\ \text{sequential access}\end{matrix} = mq[(t_q^j/\beta) + (s/t_s^j)]$$

$$\begin{matrix}\text{CPU costs for}\\ \text{random access}\end{matrix} = mr[t_r^j + (s/t_s^j)]$$

$$\begin{matrix}\text{random access}\\ \text{channel costs}\end{matrix} = ur[t_1^j + (s/t_s^j)]$$

$$\begin{matrix}\text{sequential access}\\ \text{channel costs}\end{matrix} = uq[(t_1^j/\beta) + (s/t_s^j)]$$

The final term computes the cost of moving the data from level i to level j and includes factors for writing the data back to level j if necessary, preparation for transfer, latency waiting for the next block, and block transmission costs.

$$\begin{matrix}\text{cost to move data from}\\ \text{level i to level j}\end{matrix} = (1 + \lambda)d\{MW + (S/b_i)[mt_1^i + (mb_i/t_s^i) + (ub_i/t_s^i)] + (mS/B_i)t_c^i\}\Gamma(i - j),$$

where $\Gamma(x)$ is 0 if $x = 0$ and is 1 otherwise.

$$f_{ij} = \{\tau_i n_i + \tau_j n_j\}S +$$ storage cost

$$mq[(t_q^j/\beta) + (s/t_s^j)] +$$ CPU cost: sequential access + transmission

$$mr[t_r^j + (s/t_s^j)] +$$ CPU cost: random access + transmission

$$\{uq[(t_l^j/\beta) + (s/t_s^j)] + ur[t_l^j + (s/t_s^j)]\} +$$ channel cost

$$(1 + \lambda)d\{MW + (S/b_i)[mt_l^i + (mb_i/t_s^i) + (ub_i/t_s^i)] + (mS/B_i)t_c^i\}\{\Gamma(i - j)\}$$ cost to move the data set between levels i and j

Figure 1

Objective Function for the LSWL Model

Network Model

Further extensions than those discussed here are necessary to model the cost of a distributed data management system in complete detail. However, the model developed here is a good first approximation and will allow investigation of the tradeoffs between storage and access economy. It will also provide an accurate model of file or data set staging in a network.

As mentioned earlier, a primary concern in extending the LSWL model to allow for a network in the hierarchy is to account for the flow control and other protocol related costs that will be incurred. The cost function used has the basic form:

$$c_{ij} = \begin{cases} f_{ij} & i > k \\ g_{ij} & i \leq k \end{cases} \qquad \text{(j always greater than k)}$$

where k is the first remote level of the hierarchy. (Here we are tacitly assuming that all staging will be done to a local device.) We

have already discussed the original objective function, f_{ij}. We will now proceed to consider the cost function that deals with the network. The reader is directed to Table 2 for a key to additional symbols and to the summary of g_{ij} in Figure 2. The network cost function can be characterized as:

g_{ij} = storage cost + cost to move from inactive remote level to highest remote level + cost to move from highest remote level to the net

+ network cost + cost to move from net to active level + process access costs

The major differences in this equation from the purely local version are the added network costs and the distinction between local and remote charging rates. Otherwise most of the terms are special cases of the original and we will not discuss them in detail.

The network costs consist of two major components: the set-up costs for using the network and the cost of the traffic sent on the network.

$$\text{network costs} = de\{(m_r + m_L)t_{nd} + (M_r + M_L)t_{np}\}\{1 + \Gamma(\lambda)\}$$
$$+ (1 + \lambda)(SKn_k/b_k)d$$
$$+ 2en_k d\{1 + \Gamma(\lambda)\}$$

The first term is the cost of setting up the transfers in terms of the number of message exchanges required (protocol negotiation), network delay and protocol processing. The other two terms are network charges for the packets actually sent. The first of these is the cost for the data sent and the second is for the messages sent for the set-up negotiation. The constant K in the first term is a "compression" factor to allow inclusion of data compression and protocol overhead in data transmission (headers, restart markers, etc.). The

e = number of message exchanges necessary to set up the transfer

t_{nd} = message round trip delay time in the network

t_{np} = CPU time for protocol overhead (on a per protocol message basis)

K = "compression" factor

t_{nr} = network CPU time to receive data

t_{nt} = network CPU time to transmit data

u_r = remote channel cost

u_L = local channel cost

m_r = adjusted remote system cost

m_L = adjusted local system cost

N = number of data set copies necessary to achieve a desirable level of reliability

n_k = network transmission cost

M_r = unadjusted remote system cost

M_L = unadjusted local system cost

b_k = network packet size

Table 2

Supplementary Parameter List for Network Model

$$g_{ij} = \{\tau_i n_i + \tau_j n_j\}S +$$ (1) storage cost

$$(1 + \lambda)d\{(SK/b_k)[m_r b_k/t_s^{\,k} + u_r b_k/t_s^{\,k}]\} +$$ (2) cost to move between highest remote level and net

$$(1 + \lambda)d\{M_L W + (S/b_i)[m_r t_L^{\,i} + (m_r b_i/t_s^{\,i}) + (u_r b_i/t_s^{\,i})] + (m_r S/B_i)t_c^{\,i}\} +$$ (3) cost to move between inactive level i and highest remote level

$$de\{(m_r + m_L)t_{nd} + (M_r + M_L)t_{np}\}\{1 + \Gamma(\lambda)\} +$$ (4) protocol set up cost

$$2en_k d\{1 + \Gamma(\lambda)\} +$$ (5) network charges for protocol messages

$$(1 + \lambda)(SKn_k/b_k)d +$$ (6) data transfer network costs

$$(M_r t_{nt} + M_L t_{nr})(S/b_k)d + \lambda(M_r t_{nr} + M_L t_{nt})(S/b_k)d +$$ (7) network software cost to send data and receive it

$$m_L q[(t_q^{\,j}/\beta) + (s/t_s^{\,j})] + m_L r[t_r^{\,j} + s/t_s^{\,j}] +$$ (8) CPU costs for random and sequential access and for retrieval from active location

$$u_L q[(t_L^{\,j}/\beta) + (s/t_s^{\,j})] + u_L r[t_L^{\,j} + (s/t_s^{\,j}) +$$ (9) channel costs for local retrieval

$$(1 + \lambda)d\{SK/b_i[(m_L b_k/t_s^{\,k}) + (u_L b_k/t_s^{\,k})]\}$$ (10) cost to move from net buffers to active device

Figure 2

Objective Function for the Network Model

transmission cost of the network, n_k, is calculated in terms of packets sent, a charging structure in use in the commercial domain. (It should be noted that the symbols with the subscript k do not refer to the properties of the highest remote level of the hierarchy but to properties of the network, such as transmission rate, packet size, etc.) Factors involving λ are included in the network costs to take account of the possibility of shipping the data back to inactive store. Notice that a transfer must be set up no matter how small an amount is sent back - hence the appearance of $\Gamma(\lambda)$ in the formula.

Example. Consider a situation in which there is a four-level hierarchy (core, drum, disk, and archive), both locally and at a remote site. Assume that values of the relevant parameters are as given in Table 3 (taken from Lum et al. [1975]) and that they are the same at both sites.

Parameter	Core	Drum	Disk	Archive	Units
t_r^i	10^{-6}	5×10^{-3}	60×10^{-3}	5	second
t_s^i	∞	10^6	3×10^5	5×10^4	byte/sec
t_q^i	0	8×10^{-3}	13×10^{-3}	25×10^{-3}	second
t_l^i	0	8×10^{-3}	12×10^{-3}	20×10^{-3}	second
t_c^i	0	0	25×10^{-3}	40×10^{-3}	second
n_i	2×10^{-2}	5×10^{-4}	3×10^{-5}	3×10^{-7}	\$/byte/ month
b_i	*	20,000	7,000	2,000	byte
B_i	*	4×10^6	140,000	10,000	byte

* Irrelevant

Table 3

Parameters for Storage Hierarchy

It does not, of course, make sense to consider inactive storage at remote core, and this case is omitted. Let the number of local buffers be two ($\beta = 2$) and assume that there is no setup time to open a data set ($W = 0$). Suppose that a data set of 10^8 bytes is active for one eight-hour shift per day, so that on a per-month basis $d = 30$ (i.e., the data set is opened once per day). Furthermore, the set is then active 1/3 of the time ($\tau_j = 1/3$), and we shall assume that $\tau_i = 1$ (i.e., that the set is permanently resident at the inactive location). Let the set be blocked into 1500-byte physical records ($s = 1500$) and suppose that $\lambda = 1$ (so that the data set is always written back at the end of each day). Finally assume that there are 90,000 sequential accesses to the active copy per month and 210,000 random accesses (i.e., $q = 90{,}000$ and $r = 210{,}000$). These values all correspond to those used by Lum et al. in their example.

Next, network parameters are needed. We have taken $b_k = 125$ bytes, the ARPANET packet size; $t_{nd} = 200$ ms and $t_s^k = 5 \times 10^3$ bytes/sec, both ARPANET figures; $t_{np} = 1$ ms, which is roughly the time for an ARPA NCP to handle one protocol command (including response); $t_{nr} = 1$ ms, an average figure which runs from about .5 ms NCP time to 2 ms if the process must be awakened; and $t_{nt} = 2$ ms, which consists of about 1 ms to get to the NCP and 0.5 to 1 ms to use it. (These estimates for t_{np}, t_{nr}, and t_{nt} were supplied to us by G. Grossman of the Center for Advanced Computation.) It should be noted that both t_{nr} and t_{nt} should be slightly larger to allow for data processing by the file transfer protocol. This is particularly true if data compression is being carried out. But for this example we initially assume $K = 1$. Also, t_{nr} and t_{nt} as given are times <u>per message</u>; we have divided by 8 to get a per-packet estimate, since a maximum of 8 packets per message is

allowed. The parameter e was set at 15. This is arrived at as follows. In the ARPANET, it requires 7 exchanges to open an FTP connection, plus from 4 to 7 commands to set parameters and 3 more to open the data connection. It should be noted that by using ARPANET data and the values supplied by Grossman we are essentially computing lower bounds on network costs. In other environments the network costs will be higher and results are likely to be quite different.

Finally, cost estimates are needed. For network transmission we assumed n_k = \$1.25 per 1000 packets, a quoted Telenet commercial rate. To begin with we have assumed that $m_L = m_r$ = \$10/hr., $M_L = M_r$ = \$100/hr., and $u_L = u_r$ = \$8/hr. Clearly under these assumptions remote storage will not be cost effective; but by adjusting the cost of the remote site relative to that locally, we should reach a point where remote storage is cheaper. The values calculated for costs c_{ij} (see Figures 1 and 2) are given in Table 4. As expected, remote storage is not economical for the assumed cost structure. The cheapest method is for the inactive data to be stored on local archive and transferred to local disk when active.

		Active Location (j)			
		Local Core	Local Drum	Local Disk	Local Archive
Inactive Location (i)	Local Core	2000			
	Local Drum	717	50.0		
	Local Disk	670	19.8	3.05	
	Local Archive	668	17.5	1.91	3.01
	Remote Drum	724	73.7	58.1	60.0
	Remote Disk	677	26.9	11.3	13.2
	Remote Archive	675	24.9	9.27	11.2

Table 4

Computed values of total costs c_{ij} for the basic example.
Entries are in thousands of dollars per month.

In an attempt to determine for what relative costs it becomes cost-effective to store remotely, we recomputed the c_{ij}'s for a decreasing sequence of values of M_r, m_r, and u_r. All other parameters were kept the same. Even when the cost ratio Z was

$$Z = \frac{m_r}{m_L} = \frac{M_r}{M_L} = \frac{u_r}{u_L} = 0.1$$

the best strategy was still to store on local archive and transfer to local disk. At this point, however, the best remote strategy (remote archive to local disk) was less than twice as expensive as local archive to local disk (compared with a factor of more than 4 in the Table 4 example). Closer examination of the individual terms computed showed that what keeps remote storage from becoming cost effective are fairly large contributions from terms (2) and (6) (cost to move from highest remote level to net and data transfer network costs, *see Figure* 2). *In* short, shipping a data base of 10^8 bytes back and forth across a network daily is just not likely to be cost effective under most conditions!

If costs for shipping through the network are, as it appears, making remote storage uneconomical, compression of the data before shipment should help. We therefore inserted a compression factor K = 0.1 (about as small as is realistic) into the model and recomputed the c_{ij} for cost ratio Z = 0.1, and all other parameters the same as for the Table 4 example. Remote storage now becomes cost effective - the best strategy is to transfer from remote archive to local disk. (See Table 5.) The reader should keep in mind throughout this discussion that the numbers and comparisons given here should not be taken too literally. The simplistic hierarchical storage model we are using does not take into account, for example, cost advantages which may occur due to remote data processing.

		Active Location (j)			
		Local Core	Local Drum	Local Disk	Local Archive
Inactive Location (i)	Local Core	2000			
	Local Drum	717	50.0		
	Local Disk	670	19.8	3.05	
	Local Archive	668	17.5	1.91	3.01
	Remote Drum	717	67.1	51.4	53.4
	Remote Disk	670	20.1	4.46	6.39
	Remote Archive	667	17.2	1.59	3.52

Table 5

Computed values of c_{ij} for K = 0.1, cost ratio Z = 0.1.
Entries are in thousands of dollars per month.

Starting at this point, we increased Z (since a ten-to-1 cost ratio is probably unrealistic) to see at what value of Z remote storage begins to become cost effective (for K = 0.1). Throughout the range of Z values, the best local strategy is archive to disk; the best remote strategy is remote archive to disk. We have graphed the costs of these, versus Z, in Figure 3. The local strategy cost is, of course, independent of K or of remote costs (and hence of changes in Z). Notice that the crossover occurs at Z = 0.3 - a value which might well occur in practice. Another interesting feature is the linearity of the curve, which may make the model more useful as input into decision algorithms.

An unexpected result was that decreasing S (the data base size) to 10^7 bytes and then to 10^6 bytes led to virtually no change in this crossover value of Z. Even at 10^5 bytes the local and remote best strategies are nearly of equal cost at Z = 0.3. But for this small a

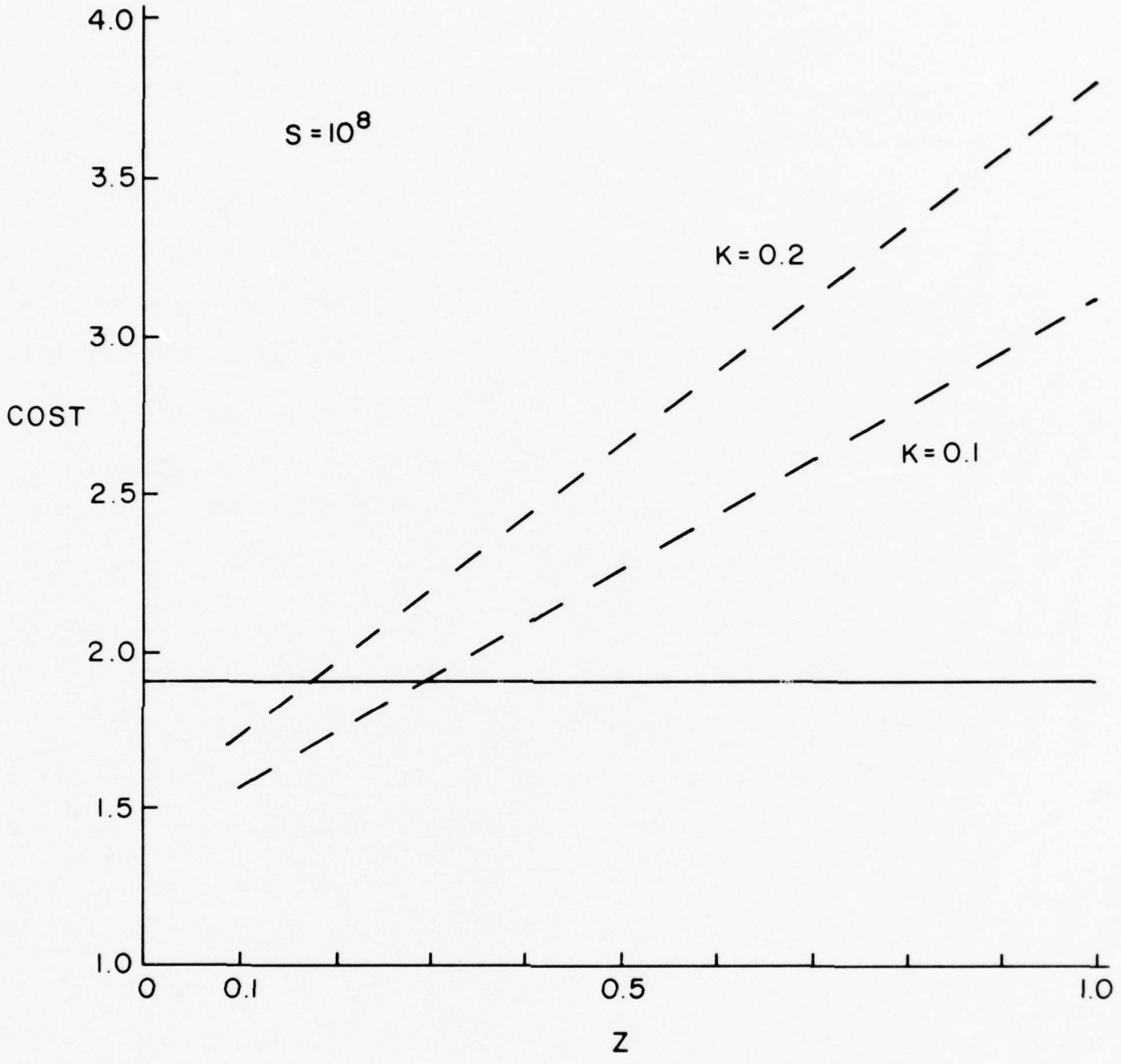


Figure 3

Comparative costs (in thousands of dollars per month).
———— Best local strategy
– – – – – Best remote strategies

data base the best active storage becomes local drum instead of local disk. For larger data bases (10^9 bytes or more), however, the main costs are those for storage, and the minimum in the matrix corresponds to permanent storage in local archive (no staging).

Though insensitive to S, it is clear that the crossover point is sensitive to K. To investigate this feature further, we generated the second curve in Figure 3, for which the only difference in parameters is that K = 0.2. As expected, the crossover point has decreased, and to about Z = 0.17. To a good approximation, as we decrease the amount of compression, the remote costs must decrease proportionately for remote storage to remain cost effective. (A quick check showed the trend holding for K = 0.5. In this case remote storage is almost - but not quite - cost effective for Z = 0.1.)

In conclusion, we have seen that remote storage of even very large data bases may be economical, providing the data is shipped compressed and there is a sufficient differential between local and remote costs. However, it perhaps is not reasonable to assume that the whole data base is transferred.

A more realistic model would allow for transferring only a portion of the data. One approach would be to transfer a block of data only when needed. Suppose it is assumed that each access request initiates a transfer of the relevant block or blocks. This supposition, however, contradicts the whole basis of the present staging model - namely, that the data base is transferred from inactive to active storage and then accessed on a block by block basis. A compromise can be achieved by assuming that only a portion of the entire data base is staged daily, as discussed below.

Revised Model: Partial Staging

In this paragraph, we will consider the effects of altering the model to take account of the possibility that only a part of the data base is transferred from inactive to active storage. We introduce a new parameter:

S' = size of data set transferred.

Then the following changes are to be made in the equations:

In Figure 1, the storage cost becomes $\tau_i n_i S + \tau_j n_j S'$, and in the last term S is replaced by S'.

In Figure 2, the storage cost (term (1)) is changed just as in Figure 1. In addition, all other occurrences of S are changed to S'.

Figure 4 shows the results of some computations for partial staging. The parameters chosen were such that results are directly comparable to the K = 0.2 curve in Figure 3. The absolute costs have of course decreased considerably. However, the interesting feature to notice is that the crossover point is virtually unchanged from when the entire data base is transferred. This seems to be another aspect of the relative insensitivity of cost effectiveness to changes in data base size.

Application of the Model to Multi-site Usage

There is another type of strategy question that may readily be studied by using the model. Suppose users at two sites wish to use the data base, but it will be used more heavily at one (Site A) than at the other (Site B). Should Site B use Site A's copy or store its own locally? In this section we give an example of this type of problem and its solution.

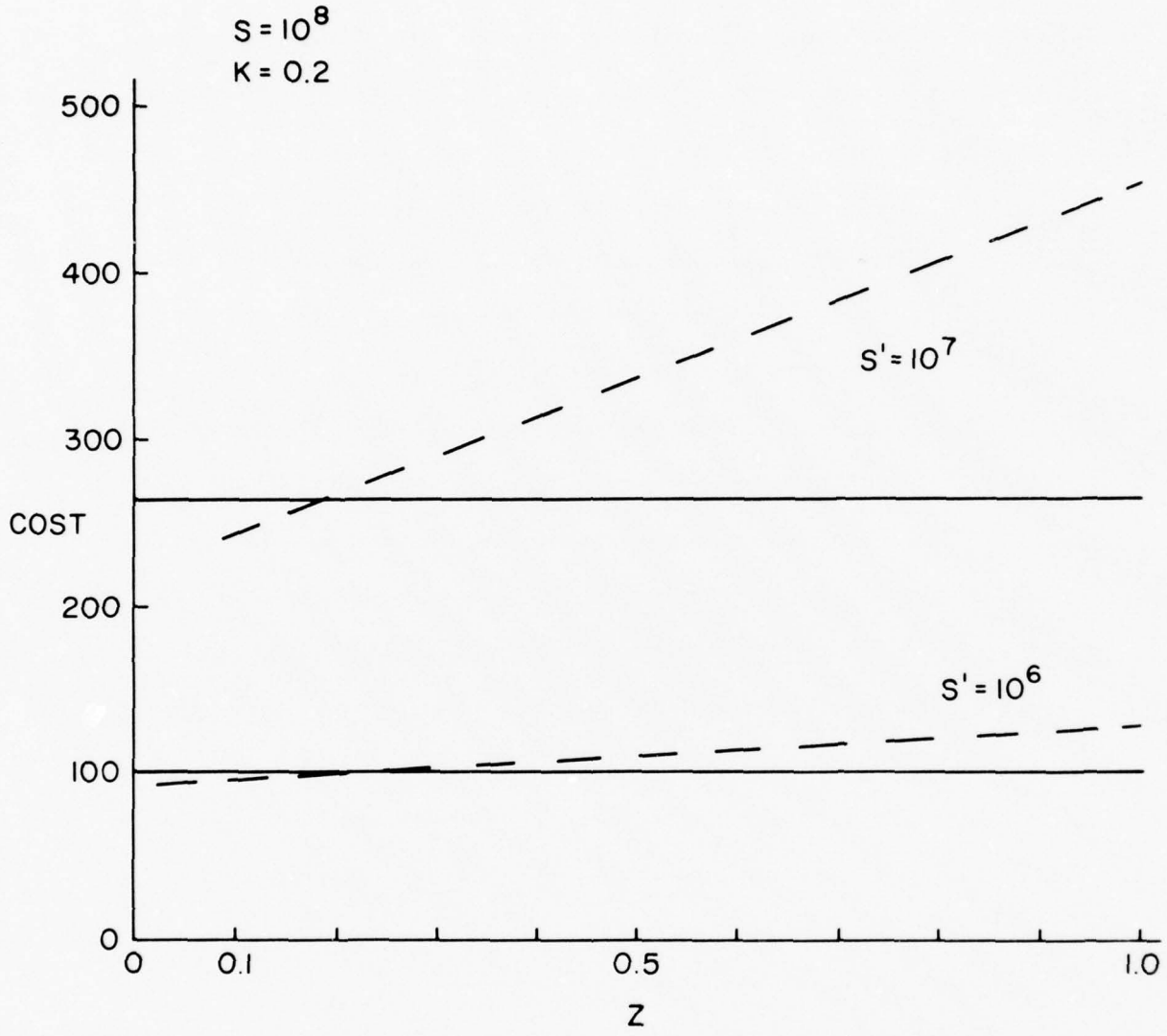


Figure 4

Comparative costs (in dollars per month).
———— Best local strategies
– – – – – – – – Best remote strategies

Suppose that the data base is 10^8 bytes in size, and suppose that costs and other parameters (except those describing usage) are the same for both sites and are those assumed in the computation of the Table 4 entries. Let the usage at Site A also be the same as was assumed in computing Table 4. Then we know that the best strategy from Site A's point of view is to store the data on local archive and stage it to local disk. We therefore assume that this is done, at a cost of $1,914 per month (from the computation for Table 4).

Now suppose that Site B only uses 10 percent of the data base (perhaps a different 10 percent on different days) and that Site B performs far fewer accesses, say again by a factor of 10. We now rerun the model to obtain the c_{ij} matrix from Site B's point of view. The parameter changes to do this are: $S = 10^7$, $q = 9000$, $r = 21{,}000$, and $\lambda = 0$. (This last change is made because we assume that Site A makes all the changes in the data; Site B just retrieves it.) The resulting matrix of c_{ij} values is shown in Table 6. (We have omitted the "local core" column here because the storage options involving core are too expensive to be interesting.)

		Active Location (j)		
Inactive Location (i)		Local Drum	Local Disk	Local Archive
	Local Drum	5001		
	Local Disk	1974	305	
	Local Archive	1712	150	301
	Remote Drum	7021	5458	5652
	Remote Disk	2329	767	960
	Remote Archive	2081	518	712

Table 6

Matrix of costs c_{ij} for Site B. (See text.)
Entries are in dollars per month.

From the table, we see that Site B's best local strategy is to store on archive and stage to disk, at a cost of $150. Furthermore we see that the best strategy involving A's archive as inactive storage is to stage the data to disk, at a cost of $518. This includes, however, a cost of $3 per month to store 10^7 bytes in A's archive and this storage cost is already assumed to be paid for by A. Thus the net cost to B is $515 per month. Finally we make the comparison:

Total cost, storage at both A and B = $2064 per month.

Total cost, storage at A only = $2429 per month.

Not surprisingly (since the cost of storage itself is so small) the first option is the cheaper. However, the increased cost of the second option is only $355 or 17 percent. This may be very much worthwhile in view of the problems that arise in trying to keep more than one copy up to date. Furthermore, this computation was carried out with K = 1 (no data compression). If the data is transferred in compressed form, say K = .25, site B's best local strategy is as before. However, the best strategy involving A's archive as inactive storage is to stage the data to disk, at a cost of $258 (subtracting off the duplicated storage costs as before). Thus the total cost of the second option is $2172 per month, which is an increase of only $108 or 5 percent.

Plans for Further Work

Clearly, much more can be learned by experimentation with the present model. By using parameters that describe specific systems and their costs, we should be able to develop cost comparisons for important real applications. In addition, we have looked carefully at the effects of varying only a few of the many parameters in the model. By varying others, we should gain further insights into costs.

We might also investigate other approaches to deciding on a "best" storage policy which might be relevant in some situations. For example, since protocol implementations reside as user-level processes in many operating systems, and since it is often useful to consider the data set as being staged in the remote system, it might be interesting to consider an alternative approach which runs as follows. The data set allocations on the remote site are determined according to the LSWL model, and the lowest-cost strategy is selected. The cost of this strategy plus the relevant network costs are then used to form the lowest level of the local hierarchy, where the cost for the local levels is computed using the LSWL model and the last level (the remote one) uses a slightly modified form. Further study is needed to determine whether this approach will yield useful data for decision making.

There are a number of refinements that could be added to the model. We list a few of these here.

1) There could be a provision for allowing some fraction of the queries to be answered locally, while the rest require remote access. (This feature may be useful in analyzing the cost effectiveness of intelligent terminals or network front-ends.)
2) The effects of the finite size of the storage devices might be included.
3) As mentioned earlier, the definition of the adjusted system cost does not appear to reflect the effects of increased load on the system. This point requires more investigation to gain a better understanding of this parameter and of how, if necessary, system loads may be inserted into the model.
4) The model developed by Lum et al. was intended to represent file migration or data staging. Thus, when a data set is

written back to the inactive device, the operation is considered to be symmetrical to the original read. If this model is to be an accurate characterization of a data management system, it will be necessary to include the cost of performing updates.

5) Since data base reliability appears to be one of the major advantages of distributing, it is very important that the model be capable of evaluating the cost of various multi-copy backup schemes with respect to the level of reliability they provide. We have therefore provided a parameter, N, to indicate how many copies exist in the network. Unfortunately, we have not yet determined how this parameter should be inserted into the model.

A Model for Distributed Data Availability

Introduction

In this section we attempt to quantify the improvement in data base availability which can be achieved by storing a backup copy at one (or more) remote sites in a network. We also discuss the practicality of certain alternative management strategies.

Availability is defined as the probability that at least one copy of the data base is up and usable as a master copy for queries and updates. Alternatively, availability can be thought of as the fraction of time that the data base is expected to be available for use.

To simplify the analysis, we will not consider various possible causes of data base failure, but will assume that the data is available when the host computer is. Furthermore, we will not take into account scheduled down time of the host computer, on the assumption that if down time is scheduled, transfer to a backup copy is automatic and immediate, and leads to no loss in availability. (The very existence of a backup copy at an alternate network site will of course improve availability considerably over the case where only one site has a copy.)

The Model

Parameters. The parameters in the model are as follows:

F = mean time between computer failures, assumed to be the same for all host computers.

X = expected time to repair computer.

L = expected time to load the data base copy at the remote site.

Y = time that the audit trail of updates has been growing (i.e., time since the copy was correct).

k = the ratio of update arrival rate to update processing rate, so that kY = time to process the audit trail.*

D = time delay between when the master fails and when the remote site determines this fact and starts to get its copy ready for use.

The equations. First, consider the case where there is a single copy of the data base. The availability of this copy is then

$$A_o = \frac{F}{F + X + kX}.$$

This is the usual formula for availability (mean time between failures divided by mean time between failures plus mean time to recover), where the mean time to recover includes repair time X plus the time kX to process the updates accumulated while repairs were made. (This formula for recovery time is that used by Chandy et al. [1975].) There is a question as to whether the term kX should be included here, since the site is technically "up" after time X. But in a network setting, it does seem appropriate to assume that updates initiated at remote sites are being logged somewhere, so that there does exist an update list to be processed. In addition, we are interested primarily in comparing A_o with availabilities computed for multi-copy strategies, where the copies are assumed to be up to date.

Consider Strategy 1 for transferring usage back and forth between master copy and backup copy. After the master copy is determined to have failed, the remote copy is then brought up (after a time lapse of D + L + kY) and usage is transferred to it. Meanwhile the old master is

---

* The parameter k is referred to in the literature as a "compression" factor [Chandy et al., 1975]. This is not to be confused with the usual data compression factor denoted by K in the previous section.

being repaired. Queries and updates are sent to the new master, however, until it fails, at which time the process repeats: another "new" master is identified and activated. (This may or may not be the "old" master.) This strategy is diagrammed (but not to scale) in Figure 5. Since the remote site may have been up for some time since its last failure, it is assumed that, after the data base comes up, the expected time until failure is only F/2. (Actually a smaller number may be more reasonable, since some host time has already been spent in the recovery operation.) Notice that an obvious built-in assumption can be read from the figure.

$$(1)\quad D + L + kY < X + kX$$

If this inequality is not satisifed, it theoretically does not pay to store a remote copy, since the master is expected to be repaired and updated before the remote copy can be activated. The formula for availability under Strategy 1 can then be read off Figure 5 as

$$A_1 = \frac{F}{2D + 2L + 2kY + F}.$$

Strategy 2 is to immediately replace the copy by the old master as soon as the latter has been brought back up. This scheme is diagrammed in Figure 6. Again, inequality (1) must hold in order for the diagram to be meaningful. There is an additional assumption which must be made in order for our model of either strategy to be valid. This assumption is that D + L + kY is sufficiently small compared to F that there is little likelihood of a failure of the remote host during the recovery process. In addition, Strategy 2 requires that

$$(2)\quad X + kX < \frac{F}{2}.$$

If this is violated, there is a good probability that the copy may fail before the master is ready. For reasonable values of F, however,

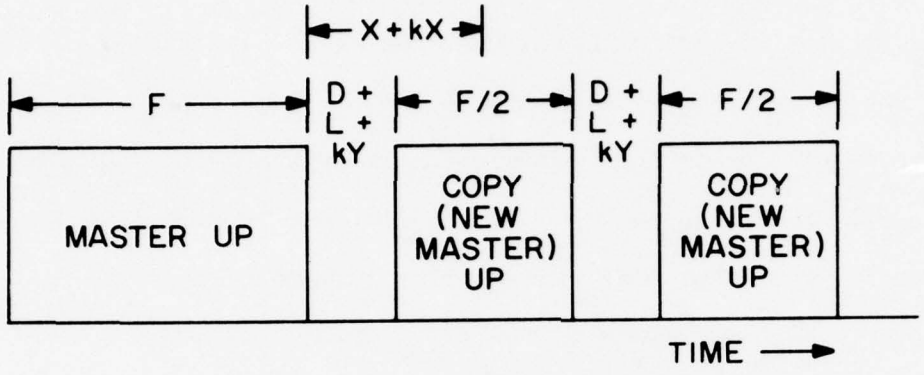


Figure 5

Diagram of Strategy 1

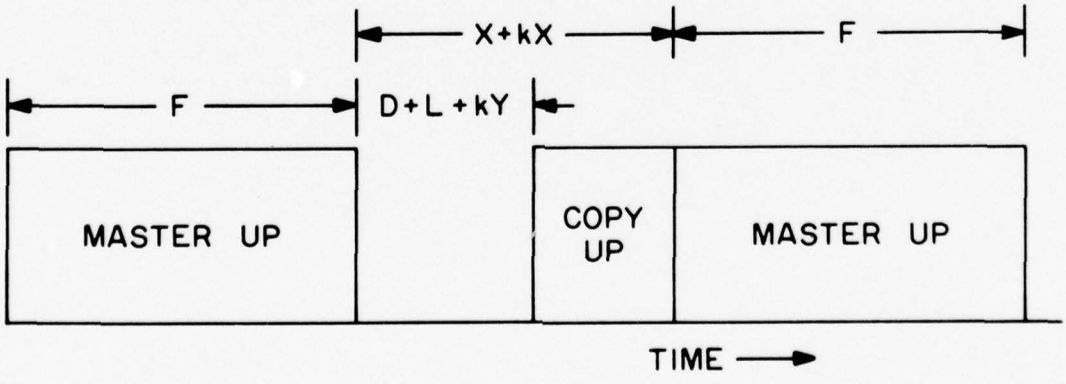


Figure 6

Diagram of Strategy 2

inequality (2) is readily satisfied; it is inequality (1) that must be carefully checked in using the model. Finally, the availability formula can be read from the diagram:

$$A_2 = 1 - \frac{D + L + kY}{F + X + kX}.$$

Keeping inequality (1) in mind and comparing formulas, we see that Strategy 1 is generally poorer than Strategy 2, and indeed A_1 is often less than A_o. We will therefore restrict consideration to Strategy 2.

Sensitivity to parameter values. In any model, it is useful to determine how sensitive the output values are to changes in the inputs. Obviously, the inputs are only known approximately or are statistical averages. If the output changes drastically for a small change in an input value, the model is rather useless for predictive or decision purposes. Chandy et al. [1975] use the elasticity E(f,y), essentially the "percentage change in f caused by a percentage change in y", to investigate the sensitivity of a function f with respect to a parameter y. Formally, E is defined by

$$E(f,y) = \frac{\partial f}{\partial y}\frac{y}{f}.$$

We have investigated the elasticity of $U = 1 - A_2$ with respect to all of the input variables. (Working with U instead of A_2 simplifies the algebra without changing the conclusion.) We find that for all parameters

$$\left|\frac{\partial U}{\partial y}\frac{y}{U}\right| < 1.$$

For example, taking y = k,

$$\frac{\partial U}{\partial k} = \frac{FY + XY - DX - LX}{(F + X + kX)^2}, \text{ and}$$

$$\left|\frac{\partial U}{\partial k}\cdot\frac{k}{U}\right| = \left|\frac{k(FY + XY - DX - LX)}{(F + X + kX)(D + L + kY)}\right|$$

$$= \left|\frac{kFY + kXY - \ldots}{kFY + kXY + \ldots}\right| < 1.$$

And for y = Y,

$$\left|\frac{\partial U}{\partial Z}\cdot\frac{Y}{U}\right| = \frac{kY}{F + X + kX} \cdot \frac{F + X + kX}{D + L + kY} = \frac{kY}{D + L + kY} < 1.$$

Similar computations show that the elasticities of U with respect to D, L, X, and F are all less than one. Elasticities of U are connected to those of A_2 through

$$\left|\frac{\partial A_2}{\partial y}\cdot\frac{y}{A_2}\right| = \left|\frac{\partial U}{\partial y}\right|\frac{y}{A_2} < \left|\frac{\partial U}{\partial y}\right|\frac{y}{U},$$

as long as $A_2 > U$. We may conclude therefore that our model is stable, being relatively insensitive to small changes in parameter values.

Experiments and Discussion

Remote journaling. In order to model a remote journaling process, we assume that the parameter Y is large; for simplicity we assume that it is equal to F. Thus we are essentially assuming that, whenever the master comes up after a failure, a copy of the up-to-date data base is shipped off to any remote site which contains a copy of the data base. (Or that the remote data base, having been used as a master copy while the master was down, already possesses an up-to-date copy at this time.)

It is interesting to note that journaling remotely by shipping the data base over the network is not feasible on a regular basis. For example, consider a data base of 4×10^7 bytes (roughly FORSTAT size). At a network throughput of 15 kilobits per second (faster than normal for the ARPANET), it would take approximately 6 hours to ship a data

base of this size. Daily backup by, say, sending tapes by courier would, however, be feasible in many situations.

The data copy at the remote site will be generally assumed to be on tape. The value L = 0.5 hr. has been assumed in the computations since it is approximately the time to read two to three tapes. The parameter D is probably on the order of one or two seconds, but we have taken it to be .01 hr. as an absolute upper bound. X = 1 seems to be a reasonable mean value for repair time. With these parameters, we get the following formula for improvement I in availability as a function of F and k.

$$I = \frac{A_2 - A_o}{A_o} = \frac{0.49 + k(1 - F)}{F}$$

It is difficult to estimate what a reasonable value of k should be. In a similar analysis, Chandy et al. [1975] take k = 1/8. Clearly the value will depend on the usage pattern for the data base; doubtless ways of measuring it for a real system could be devised. However, notice that, with k = 1/8, inequality (1) states that

$$.51 + F/8 < (1 + 1/8).$$

Hence for this large a k the time to process the audit trail is so long that the master is able to get up before the backup copy whenever F > 4.92 hrs., which is an unreasonably low value.

To get a feel for the value of remote journaling, we therefore take k = .01; i.e., we assume that there are few updates. In this case inequality (1) restricts the model to F < 50. A graph of I vs. F in this case may be seen in Figure 7. Notice that for reasonable values of F the improvement in availability is less than 5 percent which may not be enough to make remote journaling worthwhile. Values of A_o have also been plotted in the figure for reference.

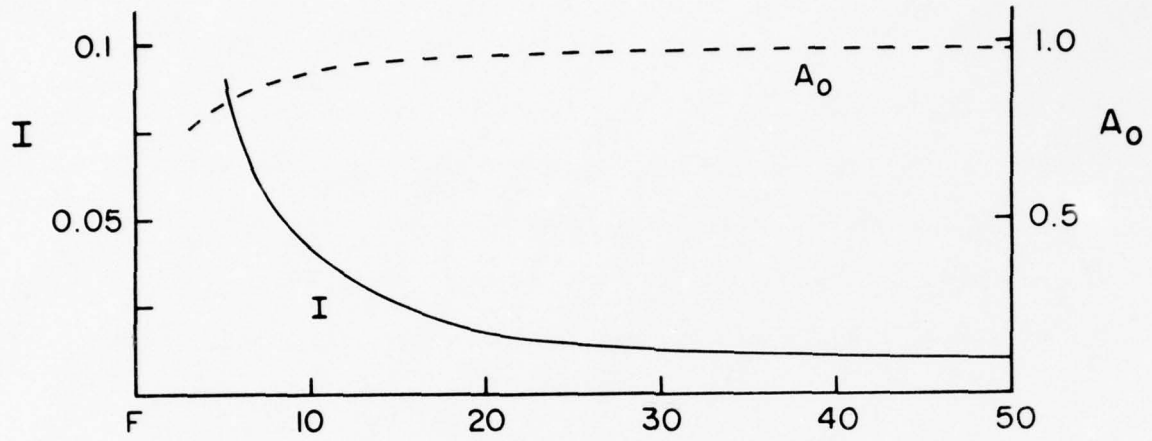


Figure 7

Single-site availability A_o and fractional improvement I through use of Strategy 2. Parameters are k = 0.01, D = .01 hr., X = 1 hr., L = 0.5 hr., and Y = F.

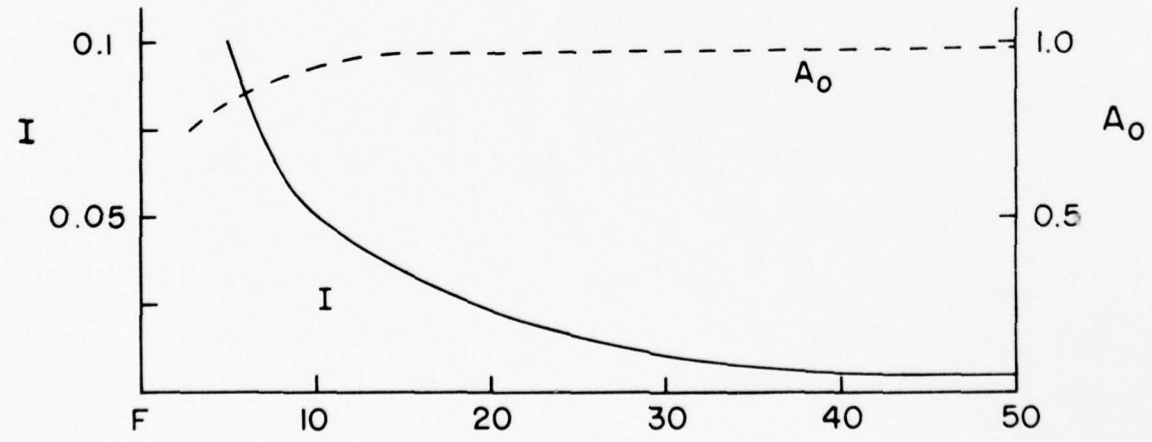


Figure 8

Same as Figure 7, except that Y = 1 hr.

As a final comment on the remote journaling strategy described here, we note that availability may actually decrease as F increases. For example, suppose X = 2, k = 0.25, L = 0.5 and D = 0. Then A_2 = .7692 for F = 4 and A_2 = .7647 when F = 6. Differentiating A_2 (for Y = F) with respect to F shows that this decrease will occur whenever

$$k(k + 1)X > D + L.$$

Intuitively, this phenomenon occurs because for large k the effect of the lengthening audit trail to be processed outweighs that of the increasing reliability of the host computer.

Frequently updated remote journal. The lack of effectiveness of the remote journaling strategy described in the last section seemed to be caused by the necessity of processing an extremely long audit trail. Suppose, then, that we drop the assumption that Y = F and assume instead that the remote copy is periodically brought up to date. As an example, we might assume this updating to take place every two hours, so that the average length of audit trail to process to bring up the remote copy is 1 hour. With all other parameters as specified for Figure 7, but with Y = 1,

$$I = .49/F.$$

This result, which is graphed in Figure 8 is independent of k (because of the cancelling of kX and kY terms), as long as k and F are such that the model is valid. Unfortunately, the improvement is still generally less than 5 percent.

Indeed, the curves in Figures 7 and 8 are nearly identical. To see why this should be so, consider more closely the formula for I.

$$I = \frac{A_2 - A_o}{A_o} = \frac{X + kX - D - L - kY}{F}$$

As long as k is small (or when X = Y as above) it is clear that

$$I \approx \frac{X - L}{F}.$$

Running spares. Here we assume that the backup copy is stored on disk for virtually instantaneous access and is kept almost up to date. Reasonable parameters for this case might be L = 0, Y = .1 hr., and (for comparison with the results above) X = 1, k = .01. Then we have

$$I = \frac{0.999}{F}.$$

We will not bother to graph this; this curve looks just like the earlier ones, only the values of I are approximately doubled. In this case, improvements of 5 to 10 percent are seen for F between 10 and 20 - certainly enough to make the strategy worthwhile. In fact, what happens in this case is that, under our assumptions, availabilities are brought up to very nearly unity. To see this, note that

$$A_2 = 1 - \frac{.01 + kY}{F + (1 + k)X}$$

and for our example kY = 0.001. Increasing k will cause somewhat smaller values of A_2, but A_2 will be over 99 percent for a wide range of reasonable parameter values.

Effect of varying Y. We have looked at three separate cases which differ from one another in large part in the widely differing values for the parameter Y. To better understand the effect of this parameter, we select typical values of the other parameters (X = 1, L = 0.5, D = 0.01, F = 20) and consider A_2 as a function of Y for several different values of k. When k = .01, we have

$$A_2 = 1 - \frac{0.51 + 0.01Y}{21.01}.$$

The small coefficient of Y in this case makes the effect of Y minimal. As Y ranges between 0 and 20, A_2 decreases linearly from 0.976 to 0.966. Now suppose that k is increased to 0.05. In this case as Y goes from 0 to 20, A_2 decreases from 0.976 to 0.953 - still not a very dramatic change! To a large extent what makes the "running-spare" approach so worthwhile is not the small value of Y but the instantaneous access ($L \simeq 0$).

Conclusions and Plans for Future Work

We have presented here a model for data availability which, while superficial, does seem to reflect the realities of various strategies for backup. We have seen that remote journaling, in the sense of storing a copy in archival storage (e.g. tape) at a remote site, leads to very little in the way of availability improvement - perhaps 5 percent at best. On the other hand, the running spares strategy, in which the remote copy is nearly up to date and almost immediately accessible, brings availability up to over 99 percent and appears to be worthwhile. It should be noted, however, that the running spares strategy is bound to be relatively expensive. Furthermore, before this strategy can be effectively used, many of the problems of multi-copy management must be solved. For example, updating must be synchronized in order for the backup copy to be effectively kept up to date.

One point to notice about the model is the importance of the parameter k. We found that remote journaling was theoretically of no value unless k was fairly small. The parameter k is essentially a proportionality factor, determining how long it takes a processor to "catch up" when there has been a backlog of updates accumulating. The

value of k will depend on many factors - the rate at which updates are generated, the complexity of the updating procedure, the processor speed, etc. Some of these factors and how they enter into k are amenable to theoretical study; others require system measurement.

Another feature of the "catch-up" time deserves some thought. Is kY an adequate expression for this? Or should one then add on k•kY to take account of the updates that have been entered while the first set was being processed, and so forth? Adding on these terms would add little complexity to the model; but it seems hardly worthwhile as long as k is so uncertain. That is, k as an effective proportionality constant can be assumed to include the effects of the higher order terms.

Finally, further work on this model should include some careful statistical analysis of a number of questions. What is the probability that a host will fail during the recovery process? What is the probability that a "new" master copy will fail before the old one has been repaired? (In both of these cases more than two copies would be advantageous.) How many copies are needed to achieve a given level of availability? Is there some "optimum" number of copies? In short, there are a number of interesting questions which can be addressed if the parameters in the model are considered to be random variables instead of simple average values.

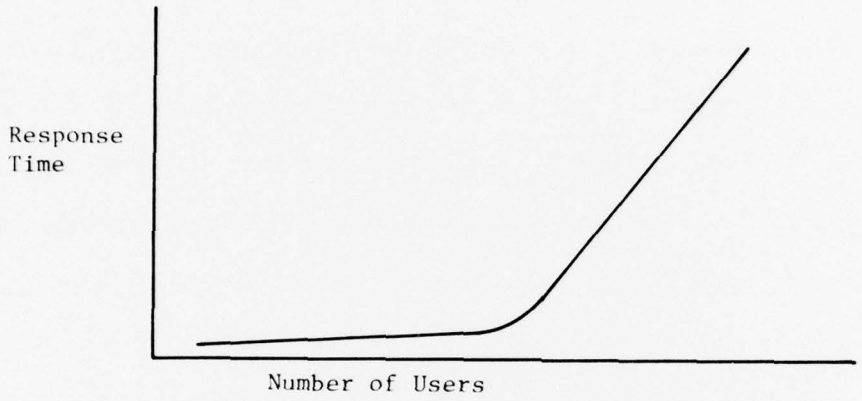
A Response-Time Model for Distributed Data

Introduction

The hypothesis to be studied in this section is that, because of disparity in site loads, sending queries to a remote site may improve response time, in spite of network delays. (Response time we define here to be the length of time between when the user inputs a query and when he receives the response.) We assume that the data base is equally available (stored on disk and kept up to date) at all of the alternate sites. We also will ignore such things as the effect of increased availability on real response time. (That is, if there is only a single copy and that goes down for several hours, the response time during that period is clearly very poor. But this effect is hard to include in our model in its present primitive state.)

The problems which arise in trying to develop a model of this type are extremely difficult. First of all, the question of how machine "load" is to be defined and measured has never been satisfactorily resolved. We are forced to simply assume that there is such a quantity and that it increases in proportion to the number of jobs in the system. Second, it is uncertain as to how response time is affected by system load. The most relevant work that we have been able to find in the literature is in Scherr's monograph [Scherr, 1967] on time-shared systems. Scherr carried out both theoretical and experimental studies of response time as a function of the number of users on the system. He found that, for a small number of users, the response time is nearly constant, showing only a slow rise as the number of users increases.

At a certain point (defined as the "saturation" level) the response curve takes a sharp upward turn, rising linearly with number of users thereafter. This general shape is pictured below.



Since Scherr assumes that the users are keeping busy, it seems to be a valid assumption that response time will also increase linearly with load, when the load is reasonably heavy. That is, we assume that the region of this curve that is pertinent to our study of the advantages of data distribution is the steep linear rise.

The Model

Parameters. The parameters in the model are as follows:

N = number of computers in the network which possess a copy of the given data base.

G_i = that part of the load at computer i which is not related to data base use.

V = number of updates per unit time.

H = number of queries per unit time.

v = load induced by an update rate $V = 1$ (so that Vv is the total load due to updates).

h = load induced by a query rate $H = 1$ (so that Hh is the total load due to queries).

a, ℓ = parameters describing the linear increase in response time as load increases. That is, at a single site,

Response time = $a(\text{load} - \ell)$.

We assume these parameters are the same for all sites.

T_n = increase in response time due to network delays and overhead of sending a query to a remote site.

The equations. Suppose, for simplicity, that all queries are entered at a single site (at computer 1, say) that possesses a copy of the data base. If the site opts to respond to the entire query load itself, then its total load is

$$Vv + Hh + G_1.$$

The single-site response time R_s is then given by

$$R_s = a(Vv + Hh + G_1 - \ell).$$

Now if the site decides to distribute the queries equally among the N sites which have a copy, the load on computer i is

$$Vv + Hh/N + G_i.$$

Notice that all sites are assumed to have equal update loads, since all sites have the responsibility of keeping their copies as up to date as possible. The response time for a query answered locally is then

$$R_1 = a(Vv + Hh/N + G_1 - \ell),$$

while the response time for a query answered at remote site i is

$$R_i = a(Vv + Hh/N + G_i - \ell) + T_n,$$

where $i \neq 1$.

The average response time $\bar{R}$ is then

$$\bar{R} = \frac{1}{N}(R_1 + R_2 + \ldots + R_N).$$

The quantity of interest is the ratio

$$R = \frac{\bar{R}}{R_s}.$$

If $R < 1$, response time is improved by distributing the queries. We therefore would like to obtain some idea of the conditions under which $R < 1$.

Use of the Model

For simplicity, consider the case $N = 2$. Then

$$R = \frac{R_1 + R_2}{2R_s} = 1 + \frac{G_2 - G_1 - Hh + T_n/a}{2(Vv + Hh + G_1 - \ell)}.$$

The denominator of the second term is always positive, by our assumption that loads are large enough that response time is described by the steeply rising line. Therefore the sign of the numerator determines whether R is greater than or less than one. That is, we have the result:

Distribution of the queries improves response time if and only if

$$aHh + a(G_1 - G_2) > T_n.$$

Now the parameter a is the rate of increase of response time with respect to load - the slope of the response-time curve. Thus the left side of the above inequality is just an increase in response time due to the query load and the load differential between sites 1 and 2. It is intuitively reasonable that when this quantity becomes greater than T_n (the increase in response time due to network delays and overhead), it pays to distribute. For general N the inequality becomes hardly more complex:

Distribution improves response time if

and only if

(I) $$aHh + a(G_1 - \bar{G}) > T_n,$$

where $\bar{G}$ is the average load at the remote sites;

i.e., $\bar{G} = (G_2 + G_3 + \ldots + G_N)/(N - 1)$.

An interesting point to notice is that, if the query load is sufficiently large, distributing the queries may improve response even if the local site is less heavily loaded than the remote sites.

Determination of the parameter values to use in this model poses a difficult problem. As was noted earlier, the concept of load is not well defined. Values for the G_i are difficult to come by. It may be possible, however, to make simple assumptions. For example, one could assume that all sites are approximately equally loaded. In this case, inequality (I) becomes

(I') $$aHh > T_n.$$

At this point we have quantities which undoubtedly can be measured. Even though we don't know what "load" is and would find it hard to determine a and h individually, the term aHh can be determined as follows. Measure the response time $R(H_1)$ and $R(H_2)$ for two different query rates H_1 and H_2. Then, assuming that the system is sufficiently heavily loaded so that these points fall on the steep linear rise of the response-time curve (this point can be checked by further measurements),

$$ah \approx \frac{R(H_1) - R(H_2)}{H_1 - H_2}.$$

Once we have a good estimate for ah, we can estimate aHh for arbitrary H. Notice that this same approach will yield estimates of the left side

of the inequality above even if we are not measuring a true query rate H, but only some parameter H' proportional to H. If the network is homogeneous, T_n can simply be measured by sending off some queries and comparing the response time to that for locally handled queries. A data management system can then automatically monitor query rate and response times and use inequality (I') to decide when queries should be distributed.

Generalizations

Unequal distribution of queries. Suppose that the queries, instead of being divided equally among N sites, are divided arbitrarily, a fraction w_i being handled by the ith site. Then

$$\sum_{i=1}^{N} w_i = 1$$

and the appropriate quantity to take for the average load $\bar{G}$ at the remote sites is the weighted average

$$\bar{G} = \sum_{i=2}^{N} w_i G_i/(1 - w_1).$$

Inequality (I) then becomes

$$\text{(II)}\quad aHh(1 - \sum_{i=1}^{N} w_i^2)/(1 - w_1) + a(G_1 - \bar{G}) > T_n.$$

Once the concept of distributing the query load unequally among the various sites is introduced, it becomes of interest to study optimization of the distribution. What we mean by optimization is the determination of a set of weights $w_1, w_2, \ldots, w_N$ such that $\bar{R}$ is a minimum. Let us consider how this problem can be solved for N = 2. In this case $w_2 = 1 - w_1$, and we can write $\bar{R}$ in terms of the single variable w_1, the fraction of query load to be handled locally. In detail,

$$\begin{aligned}\bar{R} &= w_1 R_1 + (1 - w_1) R_2 \\ &= a\{w_1(Vv + w_1 Hh + G_1 - \ell) \\ &\quad + (1 - w_1)(Vv + (1 - w_1)Hh + G_2 - \ell)\} \\ &\quad + (1 - w_1)T_n \\ &= aVv + aHh(w_1^2 + (1 - w_1)^2) + w_1 G_1 a \\ &\quad + (1 - w_1)G_2 a - a\ell + (1 - w_1)T_n.\end{aligned}$$

Then

$$\frac{\partial \bar{R}}{\partial w_1} = aHh(4w_1 - 2) + a(G_1 - G_2) - T_n.$$

If we set this derivative equal to zero, we find that there is a prospective extremum at

$$w_1 = \frac{T_n - a(G_1 - G_2) + 2aHh}{4aHh},$$

$$w_2 = \frac{a(G_1 - G_2) - T_n + 2aHh}{4aHh}.$$

Since the second derivative (4aHh) is always positive, this extremum is in fact a minimum, as desired. We must, however, examine another constraint - that the weights w_1 and w_2 must both be positive. We can rewrite w_1 and w_2 as

$$w_1 = \frac{1}{2} + \frac{T_n - a(G_1 - G_2)}{4aHh},$$

$$w_2 = \frac{1}{2} - \frac{T_n - a(G_1 - G_2)}{4aHh}.$$

The weights w_1 and w_2 can be seen to be positive under a wide range of conditions; for example, if $G_1 = G_2$ and inequality (I') holds.

Some interesting conclusions can immediately be read from the equations for w_1 and w_2. First, we note that if the loads are equal

($G_1 = G_2$) the local site should always handle more than half of the queries. Only when $T_n = a(G_1 - G_2)$, so that the network delay equals the increase in response time due to load differential, should the query loads be equalized. And only when T_n is less than $a(G_1 - G_2)$ should the local site send off more queries than it keeps.

It must again be emphasized that careful measurements are required for these relationships to be useful for real decision making. It is easy to estimate that T_n, aHh, and $a(G_1 - G_2)$ may all, under reasonable assumptions, be on the order of one to two seconds. This information is not at all helpful for developing long-term strategies, but merely demonstrates that the optimum decision on query sharing should be done dynamically and only after monitoring current system usage and response.

The above analysis for optimum distribution strategy was done for the N = 2 case. The general case can be handled similarly, but is more complicated because of the multi-variable minimization. Setting the derivatives to zero and solving yields the following equations for $i \neq 1$.

$$w_i = \frac{1}{2}\left(1 - \sum_{j \neq i,1} w_j\right) - \frac{T_n - a(G_1 - G_i)}{4aHh}$$

Clearly this reduces to the simple formula found above for N = 2, i = 2. But in this case we have a set of simultaneous, linear, algebraic equations in $w_2, \ldots, w_N$ to solve. It is a simple matter to show that this set of equations has a unique solution, readily obtainable by computation, and that this solution does minimize $\bar{R}$. Again, it is necessary to check that the weights w_i that are computed are all positive, in order that the solution be meaningful.

Usage from various sites. All of the analysis so far has been under the assumption that the query load all originates at a single site. Suppose instead that each site i generates some fraction f_i of the total query load H. Site i then distributes its query load with a strategy described by weights $w(i)_1$, $w(i)_2$, ..., $w(i)_N$. The net rate of input of queries that a site i must respond to is then given by

$$H_i = \sum_j f_j H w(j)_i,$$

so that site i's response time (i.e. time to respond to a query) is

$$R_i = a(Vv + H_i h + G_i - \ell).$$

From the point of view of site j, the average response time seen is computed as

$$\bar{R}_j = \sum_j w(j)_i R_i + (1 - w(j)_j) T_n,$$

since a network delay of T_n is observed for the fraction of queries answered remotely. Now to get an average response time for queries originated throughout the network, we must take another weighted average:

$$\bar{R} = \sum_j f_j \bar{R}_j.$$

Combining the preceding four equations, we get an equation for $\bar{R}$ in terms of the N^2 variables $w(j)_i$. As above, we can carry out an optimization analysis or compare various strategies. (For example, the strategy where each site handles its own queries is described by $w(j)_i = 1$ when $i = j$ and $w(j)_i = 0$ otherwise.) We will not go into further details on this generalization in this brief report.

Proposed further generalizations. We list here several other ways in which assumptions may be relaxed and the model made more flexible.

1) We have assumed that T_n is a constant. To be realistic, T_n should depend upon the two sites between which the messages (query and response) are traveling. Thus we need to insert into the model a set of values $T_n(i,j)$. In addition, the values of $T_n(i,j)$ may vary depending upon what routes are taken - but it is undoubtedly adequate to take average values. Finally, the values of $T_n(i,j)$ will vary with the amount of network traffic. In particular, if we assume that query traffic forms a non-negligible percent of net traffic, $T_n(i,j)$ will be some function of H. Theoretical analysis (e.g. by queueing theory) can probably be used to determine this function, which will depend on network parameters as well as on H and the distribution strategy.

2) We have assumed that the parameters a and ℓ, which describe response time as a function of "load," are the same for all sites. This assumption is not true in a heterogeneous network, or for a network of dissimilarly configured "homogeneous" hosts (e.g. the PWIN). It should be noted, however, that the parameter ℓ did not enter into any of the decision relations, except in the assumption that "load" must be large enough compared to ℓ so that the linear expression for response time holds. In addition, we have seen that a is measured only as it occurs combined with other factors. That is, differences in a may be taken into account by varying the H_i. (See preceding section.) Thus the practical impact on the model of allowing a and ℓ to vary from site to site is probably minimal.

3) We have assumed that all sites have an equal load associated with updating the data base. This will not in general be true. If, say, the updates all originate at Site 1, the other sites will all incur network overhead in processing the updates. On the other hand,

Site 1 may do much preprocessing to make the update task simpler at the remote sites. Details of this sort could be built into the model. Results can not be too different, however, since site differences in the terms Vv can always be subsumed in the G_i.

4) We have assumed that N sites in the network have up-to-date copies of the data base and the problem is to determine a strategy for distributing the queries as they are entered into the system. A somewhat different, but very similar, model is needed to study the problem of setting a policy for distributing the data base - i.e., for determining which sites should have copies. In this case a careful analysis of the effects of updates will be essential. The sites at which updates originate will find their loads increased by the necessity of distributing the updates to other sites having a copy of the data base. Remote sites holding a copy of the data base will all have increased loads due to the processing of updates and associated network overhead. Strategies for synchronization and aspects of multi-copy management will affect loads and hence response times. These effects will, of course, implicitly enter into the query distribution model, but there we assumed that response times were obtained by direct system measurement, so that lower-level details were not modeled, but included in measured parameters. To determine a distribution policy a priori requires modeling these lower-level effects and, if possible, optimizing the lower-level strategies (e.g. for synchronization) so that the policy is based on the best up-to-date technology.

5) The four proposed generalizations listed above involve relatively straightforward extensions of the approach described in this report. In addition, however, the approach itself needs investigation.

That is, we have assumed that response time is a simple linear function of "load," and that "load" can be described as a simple linear combination of updates, queries, etc. It should be possible to examine these assumptions - both by measurement and by stochastic queueing analysis. A careful examination of this type is expected to yield some refinements in the relations studied here.

References

Arora, S.R. and Gallo, A.
1971 "The Optimal Organization of Multiprogrammed Multi-level Memory," ACM SIGOPS Workshop on System Performance Evaluation, pp. 104-141.

Borgerson, B.R. and Freitas, R.F.
1975 "A Reliability Model for Gracefully Degrading and Standby-Sparing Systems," IEEE Trans. Computers, C-24, pp. 517-525.

Buzen, J.
1971 "Analysis of System Bottlenecks Using a Queueing Network Model," ACM SIGOPS Workshop on System Performance Evaluation, pp. 82-103.

Casey, R.G.
1972 "Allocation of Copies of a File in an Information Network," Proc. AFIPS Spring Joint Computer Conference, AFIPS Press, Montvale, N.J., pp. 617-625.

Chandy, K.M.; Browne, J.C.; Dissly, C.W.; and Uhrig, W.R.
1975 "Analytic Models for Rollback and Recovery Strategies in Data Base Systems," IEEE Trans. Software Engineering, SE-1, pp. 100-110.

Chu, W.W.
1973 "Optimal File Allocation in a Computer Network," Computer-Communications Networks, N. Abramson and F. Kuo, eds., Prentice-Hall, pp. 82-94.

Frank, H. and Chou, W.
1974 "Network Properties of the ARPA Computer Network," Networks 4, pp. 213-239.

Frank, H.; Kahn, R.E.; and Kleinrock, L.
1972 "Computer Communication Network Design - Experience with Theory and Practice," Proc. AFIPS National Computer Conference, AFIPS Press, Montvale, N.J., pp. 255-270.

Gotlieb, C.C. and Tompa, F.W.
1974 "Choosing a Storage Schema," Acta Informatica 3, pp. 297-319.

Kleinrock, L.
1973 "*Scheduling, Queueing, and Delays in Time-shared Systems* and Computer Networks," Computer-Communication Networks, N. Abramson and F. Kuo, eds., Prentice-Hall, pp. 95-141.

Lum, V.Y.; Senko, M.E.; Wang, C.P.; and Ling, H.
1975 "A Cost Oriented Algorithm for Data Set Allocation in Storage Hierarchies," CAM 18, pp. 318-322.

Reiser, M. and Kobayashi, H.
1974 "Accuracy of the Diffusion Approximation for Some Queueing Systems," IBM J. Res. Develop., March 1974, pp. 110-124.

Scherr, A.L.
1967 An Analysis of Time-shared Computer Systems, MIT Press.

Shneiderman, B.
1974 "A Model for Optimizing Indexed File Structures," International J. of Comp. and Inform. Sci. 3, pp. 93-103.

Winkler, A.J. and Dale, A.G.
1971 "File Structure Determination," ACM Proc. Symp. Inform. Stor. and Retr., J. Minker and S. Rosenfeld, eds., pp. 133-146.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: CAC Document Number 169; JTSA Document Number 5511
2. GOVT ACCESSION NO.:
3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): Research in Network Data Management and Resource Sharing - Initial Mathematical Model Report.
5. TYPE OF REPORT & PERIOD COVERED: Research Report - Interim
6. PERFORMING ORG. REPORT NUMBER: CAC-#169
7. AUTHOR(s): G.G. Belford, J.D. Day, S. Sluizer, and D.A. Willcox
8. CONTRACT OR GRANT NUMBER(s): DCA100-75-C-0021
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Center for Advanced Computation, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: Joint Technical Support Activity, 11440 Isaac Newton Square, North, Reston, Virginia 22090
12. REPORT DATE: August 20, 1975
13. NUMBER OF PAGES: 72
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):
15. SECURITY CLASS. (of this report): UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:
16. DISTRIBUTION STATEMENT (of this Report): Copies may be requested from the address in (11) above.
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report): No restriction on distribution
18. SUPPLEMENTARY NOTES: None
19. KEY WORDS (Continue on reverse side if necessary and identify by block number):
information system modeling
data base availability
data base backup
hierarchical storage
network modeling
distributed data management
information system response time
20. ABSTRACT (Continue on reverse side if necessary and identify by block number):
This is a preliminary report on a project to build mathematical models for the study of distributed data management. Under simplistic assumptions, equations for cost, availability, and response time are developed and studied.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)