Analysis of Long Term File Reference Patterns for Application to File Migration Algorithms

ALAN JAY SMITH, MEMBER, IEEE

Abstract—In most large computer installations files are moved between on-line disk and mass storage (tape, integrated mass storage device) either automatically by the system and/or at the direction of the user. In this paper we present and analyze long term file reference data in order to develop a basis for the construction of algorithms for file migration. Specifically, we examine the use of the on-line user (primarily text editor) data sets at the Stanford Linear Accelerator Center (SLAC) computer installation through the analysis of 13 months of file reference data. We find that most files are used very few times. Of those that are used sufficiently frequently that their reference patterns may be examined, we find that: 1) about a third show declining rates of reference during their lifetime, 2) of the remainder, very few (about 5 percent) show correlated interreference intervals, and 3) interreference intervals (in days) appear to be more skewed than would occur with the Bernoulli process. Thus, about two-thirds of all sufficiently active files appear to be referenced as a renewal process with a skewed interreference distribution. A large number of other file reference statistics (file lifetimes, interference distributions, moments, means, number of uses/file, file sizes, file rates of reference, etc.) are computed and presented. Throughout, statistical tests are described and explained. The results of our analysis of file reference patterns are applied in a companion paper to the development and comparative evaluation of file migration algorithms.

Index Terms—File migration, mass storage, memory hierarchies, replacement algorithm, time series analysis.

I. Introduction

ALMOST all computer installations (excluding hobby computers) employ a memory hierarchy much like that in Fig. 1. Each of the levels of storage from cache to mass storage is successively larger, slower, and less expensive per bit. By dynamically moving information between levels of the hierarchy, the system can usually arrange to have each level capture a vastly larger fraction of all memory references than the levels below it. In effect, the user sees a system in which the total storage capacity is the combined capacity of all levels of the memory, while the average access time is very close to that of the fastest. The success of such dynamic information movement derives from the empirically observed "principle of locality" [7], which essentially states that: 1) information in recent use is likely to be reused, and 2) information logically adjacent to recently used information is likely to be referenced soon.

Manuscript received August 8, 1980; revised March 5, 1981. This work was supported in part by the National Science Foundation under Grants MCS75-06768 and MCS77-28429 and the Department of Energy under Contracts W-7405-ENG-48 and EY-76-03-0515.

The author is with the Department of Electrical Engineering and Computer Science and the Lawrence Berkeley Laboratory, University of California, Berkeley, CA 94720.

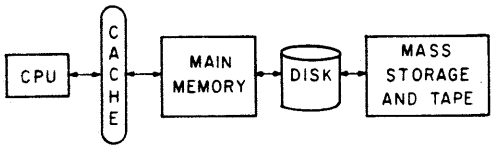

Fig. 1.

An important factor in the effectiveness of this dynamic movement among the levels of the hierarchy is the selection of algorithms for when/where to move the information. Considerable study has been devoted to the movement of information to/from cache memories [3], [9], [17], [16], [14] and main memories [6], [15], [18]. The problem of the transfer of files between mass storage and disk has been largely neglected, however, almost certainly due to the lack of suitable data. The only previous useful study of this problem was by Stritter [20] who studied the same data as is analyzed here. In a paper based on the work presented here [19], we derive and evaluate a number of file replacement algorithms. A more complete review of the literature and discussion of the problem is also provided in that article.

In this paper we study and analyze data on the use of user text editor data sets at the Stanford Linear Accelerator Center over a period of 13 months. We look at questions of how and when files are used. Some statistical analysis is performed on file reference patterns in order to determine what stochastic process models might be appropriate to represent the individual or aggregated file reference patterns. The statistical methodology is explained for the benefit of the reader. A large variety of statistics are gathered and presented, but particular attention is paid to those aspects of file reference behavior that relate to construction of effective file migration algorithms. Throughout, our emphasis is on "data derived" models and algorithms. The intent is to give the reader a good overall understanding of the file reference process, both as an aid in the development of file migration algorithms and for a general understanding of how users use the file system.

The next section of this paper describes in some detail the nature of our data and its limitations. The actual tabulation, charting, and analysis of this information is presented in Section III, which comprises the bulk of this paper. We provide an overview of our findings and discuss some of the implications of our measurements in the conclusions section. In a companion paper to this [19] the results of our analysis here are applied to the development of file migration algorithms. A general framework for such algorithms is given, and then the space of possible alogrithms is reduced to those predicted to

0098-5589/81/0700-0403$00.75

be effective. These algorithms, as well as one appearing in the literature, are evaluated in that paper using trace driven simulation.

II. Data Description

It is very important to understand both the nature of the data that we analyze and the type of system from which it was collected. As will be evident below, our information does not include some parameters of interest; thus we are unable to make some studies. The utility of our results will be constrained by the similarity between the system we discuss and the system to which the results are to be applied. In this section a relatively detailed description is given of the nature of the data and the system from which it was taken.

The data that we analyze in this paper were collected at the Stanford Linear Accelerator Center by Dr. E. P. Stritter. His analysis of these data appears in his doctoral dissertation [20]; our analysis here goes well beyond that already presented and in some cases reaches different conclusions. The measurement period was 384 days beginning August 2, 1974. Our discussion below, although phrased in the present tense, is descriptive of the system at that time; there have been changes since.

SLAC has a large computer system consisting of two 370/168-I's running VS2 release 1.6 and an IBM 360/91 running MVT release 21.8. These three machines are loosely coupled using ASP and share most of the I/O devices. The 360/91 has been in service since 1968; the two 370's were added in the early Spring of 1974. The user interface has been essentially stable for several years preceding our measurements. Most of the computing activity is concerned with the study of high energy physics and the dominant programming language is Fortran. Most of the programmers are themselves either physicists or scientists in some closely related discipline.

The vast majority of user interaction with the computer system is accomplished through Wylbur [8], which is an interactive text editor system with the capability to submit jobs into the batch job queue and later fetch the output. Each user with a Wylbur account is allocated a fixed number of 2314 Disk tracks (@7K bytes/track) which can be used to store Wylbur data sets. Most users have allocations ranging from 10-200 tracks (70K-1.4M bytes). (For a standard of reference, a 2500 line Fortran program occupies 12 tracks in standard Wylbur compressed format.) Our data are limited to these Wylbur data sets, and exclude two important classes of files: system data sets (paging data sets, VTOC's, catalogs, etc.) and all files on tape. The size of most user space allocations constrains users to keep mostly computer programs in their Wylbur data sets; input to these programs (which often consists of huge volumes of data from accelerator experiments) is usually kept on tape. Several scratch disks are used for temporary and staged (by the user) tape data sets; these disks are cleared every night and no record is available of their use.

Users are not charged for their disk space, but they are not allowed to exceed their allocation. The user is therefore inclined to ignore inactive data sets until the space is needed for some other purpose, at which time the file may or may not be copied to tape before being scratched. This particular accounting and allocation system probably affects user reference patterns. There is no automatic file migration on the SLAC system, and therefore the users have no incentive to generate spurious file references in order to prevent automatic file migration programs from moving their files to mass storage. This is different from another system that has been measured [13], and thus we believe that in that respect our data is "uncontaminated."

Information has been recorded for every Wylbur data set which was seen to exist over the 384 day measurement period indicated above. For each file, one bit is recorded for each day indicating whether that file was used by any user on that day. The date on which the file was created is available, as is the date on which the file was scratched, if it ever was. The name of the file, the user account ID and the file size (in tracks) were also recorded.

We have grouped files into three "classes," as follows. When a user session is suddenly interrupted (automatic logout, system crash), the system creates a file for the user containing his currently active Wylbur data set and saves it under the name "ACTIVE." Files with this name were considered to be one class.

Wylbur data sets can exist physically in one of two forms: a standard OS sequential data set or a partitioned data set (PDS). A PDS is usually used to hold a collection of small (less than one track), often unrelated data sets, since one track is the quantum size for file allocation. Our data do not directly distinguish PDS's from standard files but most users name their first and largest PDS "LIB" (for "library"). Those files whose first three characters were "LIB" were considered to be a class. It is also possible (but unlikely) that some files which were not PDS's were considered to be such.

All files which were not placed in the class "active" or the class "library" were placed in the class "other" or "other files." As will be noted below, the three file classes have significantly different reference patterns.

Files were also grouped according to their size. The size class is denoted "LSIZE" or "LOGSIZE" on the illustrations and is calculated from the logarithm base 2 of the file size in tracks. Thus, file size class 0 consists of files of size 1 track, class 1 of files of sizes 2 and 3 tracks, etc., up to size class 6, which is all files of at least 64 tracks. Users can be expected to treat files of different sizes rather differently because large files occupy such a large fraction of a user's allocated space.

Because SLAC is a "scientific shop" located in the midst of an academic community, there is considerable user activity at night and on weekends and holidays. Nevertheless, activity is still a great deal lighter at those times than during first shift on weekdays. We have therefore studied file reference patterns and file migration algorithms both for the entire period of observation ("all days") and then for only the working days during that period ("working days"). In the case of "working day" analysis, all file activity during weekends or holidays was mapped onto the next following working day. Thus, use of a file on both Saturday and/or Sunday would be treated as a use of the file on the following Monday. The logic behind this is the following: 1) file migration would most reasonably occur only on working days, i.e., a file migration program would run late at night on work nights and move files off line

as necessary, 2) by avoiding holidays and those days with missing data, some aspects of the analysis are simplified and/or improved, and 3) data taken only on working days should be less affected by day of the week periodicities. This paper presents data for both cases (but with the stress on "all days" rather than "working days"). The companion paper [19] considers working days only, since file migration would most reasonably occur only on working days.

III. Data Presentation and Analysis

In this Section we present the results of our study of the data discussed above. First, various terms and symbols are defined to aid in the unambiguous presentation of the material. Second, a large variety of basic descriptive statistics are given. In a number of cases statistical tests are applied to the data. These tests are briefly described, and if any aspect of the methodology requires comment, some explanation is given. Both statistical tests and simple observation are applied to data about the system activity over time, and individual file use over time. File lifetime distributions are presented. Finally, file reference patterns are analyzed carefully in order to determine how to construct file migration algorithms.

A. Definitions and Symbols

In order to be clear about the meaning of our data presentation and analysis, in this subsection we define some terms and symbols. Let:

Nfiles be the number of files observed (24 898).

Nref be the number of file-day references, i.e., the sum over all files of the number of days that file was referenced (238 871).

$N\text{ref}(i)$ be the number of days on which file i is referenced. (Reference to a file on a specific day will henceforth be called a "reference" to that file, although the file may have been referenced many times on that day.)

$I(i,j)$, $i = 1 \cdots N\text{files}$, $j = 1 \cdots 384$ is the indicator function for file i. That is, if file i is referenced on day j of the measurement period, $I(i,j) = 1$; otherwise $I(i,j) = 0$.

F is the date of the first day of measurement.

$F(i)$ is the first day (within and relative to the measurement period) on which file i exists.

L is the date of the last day of measurement.

$L(i)$ is the last day (within and relative to the measurement period) on which file i exists.

$S(i)$ is the size of file i in tracks.

$B(i)$ is the date on which file i is created (birth).

$D(i)$ is the date on which file i is scratched (death). If the file is never scratched, $D(i)$ is undefined, except that it is known to be greater than L.

$C(i,j)$, $i = 1 \cdots N\text{files}$, $j = 1 \cdots N\text{ref}(i) - 1$ is the sequence of interreference intervals for file i. That is, if file i is referenced on a total of $N\text{ref}(i)$ days, $C(i,*)$ will be the sequence of the number of days between uses. (Use on successive days yields an interreference interval of 1.)

$A(i,j)$, $i = 1 \cdots N\text{files}$, $j = 1 \cdots D(i) - B(i) + 1$ is equal to 1 if file i is referenced on day $j - 1$ after its creation (valid only during the period of measurement) and is zero otherwise. $A(i,1) = 1, A(i, D(i) - B(i) + 1) = 1$.

TABLE I
Basic Statistics

Total Days:	384
Working Days:	256
Number of files:	24,898
Number of accounts:	710
Number of file/day/uses	238,871
Files on line initially:	3977
Files on line at end:	5320
Ave. number of files used/day:	622
Ave. number of files created/day	54.48
Ave. number of files scratched/day	50.98
Ave. number of tracks referenced/day.	11,727
Ave. number of tracks allocated/day	351.7
Ave. number of tracks scratched/day:	343.5
Ave. volume of online files:	44,489 tracks
Ave. number of references/file:	10.6
Median number of referenced/file:	2
Average number of users/day:	183.5
Median number of users/day:	214
Max. number of users/day:	291
Ave. number of files used by logged on user per day:	3.41

TABLE II

Day	Total Files Used	Total Files Created	Total Files Scratched
Monday	40458	3643	3614
Tuesday	42532	3629	3254
Wednesday	43632	3843	3554
Thursday	40767	3694	3418
Friday	35714	3044	3009
Saturday	17061	1454	1304
Sunday	17004	1483	1322
Holidays	1703	131	103
Total	238871	20921	19578

B. Basic Numbers

Some of the basic numbers relating to our data are collected in Table I; we also discuss them throughout this section. A total of 24 898 different files existed over the period of observation and they belonged to 710 different accounts. The average (mean) number of accounts showing activity per day was 183.5, the median number was 214, and the maximum was 291. For an account showing activity on a given day, a mean of 3.41 files owned by that account were referenced. A total of 238 871 file-day-uses (references)($=\Sigma_{i,j} I(i,j)$) took place.

Table II shows the file activity distributed by day of the week and holidays. We note that the level of activity was about $2\frac{1}{2}$ times higher during the week than on weekends. This fact will be relevant to some of our later discussion.

C. Activity Over Time

1) Tabulation: Let: $\text{Used}(i) = \Sigma_k I(k,i)$ be the number of files referenced on day i,

$\text{New}(i)$ be the number of files created on day i,

$\text{Scr}(i)$ be the number of files scratched on day i, and

$\text{NS}(i) = \text{New}(i) - \text{Scr}(i)$ be the excess of new files over scratched files.

The four functions defined immediately above are plotted in Fig. 2. The mean values are given in Table I; day of the week averages can be computed from Table II. Immediately visible from Fig. 2 is the weekly periodicity; activity is generally high for five days and then low for two. The empirical distributions for $\text{Used}(i)$, $\text{New}(i)$ and $\text{Scr}(i)$ appear in Fig. 3, where we see that the distributions each have three modes. The rightmost one reflects workday activity, the middle one weekend/holiday activity, and the spike at 0 the days when the system was down or no data were collected (15 days). Plotting this same

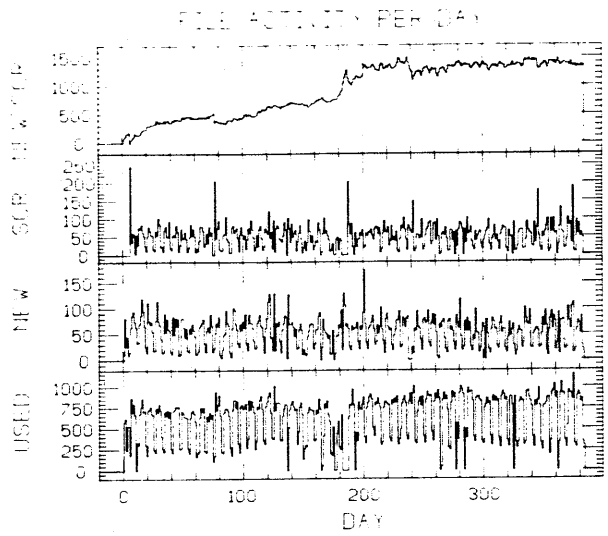

Fig. 2.

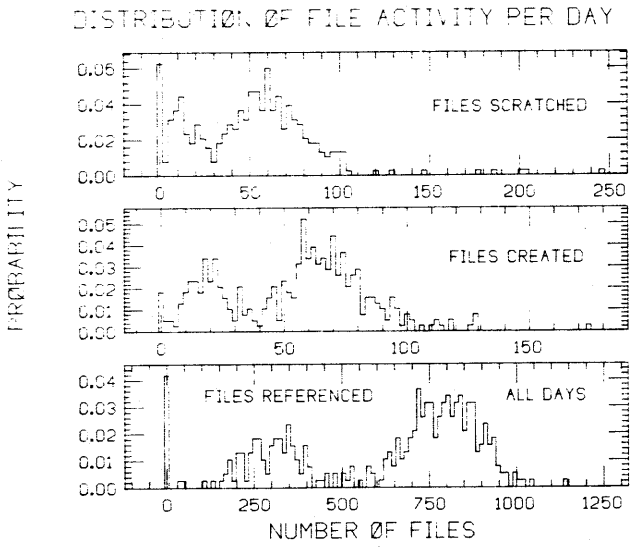

Fig. 3.

data for working days only (not shown) leaves only the single rightmost mode.

The number of tracks referenced is generally proportional to the number of files accessed. In Table I we note that the average number of tracks referenced per day is 11 727, the average number of tracks in files allocated per day is 352, and the number of tracks in files scratched is 343.

As noted, the uppermost curve in Fig. 2 is the net accumulation of files. Over the period of measurement, there is an increase of 1343 files occupying an additional 3160 tracks. It appears from these numbers and the shape of the curve that an additional disk spindle was added during the middle of the measurement period. Disk space allocations for users seem to have gone up gradually during this time. Comparing these figures to the 41 349 tracks occupied at the start of the measurement period (mean of 44 489 over the measurements), we see an increase of 7.6 percent in the total occupied file space. We note that in this system increases in the number of files and occupied disk tracks should only occur when the capacity of the system is increased. This is in contrast to the system at IBM Research, San Jose, CA [13], where the file system is open-ended and automatic file migration occurs. In that system a fairly steady accumulation of new files was noted, the rate being considerably greater than that observed here.

2) Chi Square Test: A factor affecting some uses of our data for some purposes is whether or not the level of activity on the system is stationary over time. We have already seen that the number of files and allocated tracks increased significantly over this period. We also choose to look at the time series Used(i), New(i), and Scr(i). We do this as follows.

First, we define the chi-square test for goodness of fit; this test will also be referred to later, so we make our definition general. (See [1] for further information.) Let $y(i)$ be the empirical distribution of interest and let $z(i)$ be the distribution against which it is being compared for fit. (Both $y(i)$ and $z(i)$ are discrete, $i = 1 \cdots k$.) Let n be the total number of (independent) samples used to form the empirical distribution. Then we compute

$$\chi^2 = \sum_{i=1}^{k} \frac{n(y(i) - z(i))^2}{z(i)}. \tag{1}$$

χ^2 should be distributed as the chi-square distribution with $k - 1$ degrees of freedom if $y(i)$ is indeed drawn from the distribution $z(i)$.

We let $x(i)$ represent the time series of interest. (One of the three noted above either for all days, as shown in Fig. 2, or just for working days. Thus, we have six time series.) We divide the total measurement period T into ten sections of duration $T/10$, and count the number of events $y(j)$ of interest in each section, i.e.,

$$y(j) = \sum_{i=(j-1)T/10+1}^{jT/10} x(i), \quad j = 1, \cdots, 10. \tag{2}$$

$y(j)$ was computed for all three series of events, for both all days and working days. (The all days test for stationarity is relatively worthless because of the nonuniform occurrence of holidays and days with missing data.) If the file activity were actually stationary, all values of $y(j)$, $j = 1 \cdots 10$ should be about the same.

3) Overall Activity Slightly Nonstationary: The $y(j)$ distribution was tested for uniformity ($y(j) = Y$) using the chi-square test for goodness of fit, and the hypothesis was rejected in every case at the 99.9 percent significance level. This statistical test simply confirms casual observation—from the numbers it is clear that there is a generally higher level of activity during the second half of the measurement period. The change in the rate of activity is only moderate, however, and amounts to 10-20 percent. We therefore do not believe that this is likely to have a major effect on most of the remainder of our analysis and no effort has been made to correct the data for changes in the level of overall activity.

4) Serial Correlation Statistiscal Tests: It is also interesting to determine to what extent user or system wide activity is correlated from day to day. That is, if the level of activity is high on day i, is it likely to be high on day $i + 1$? The usual procedure for determining this from a nonstationary time series (such as we have) is to remove the trend from the data

and then compute the serial correlation coefficient, since correlation coefficients are statistically meaningful only in the case of a stationary time series. Detrending the data is a laborious process which did not seem worthwhile, considering that this particular question is peripheral to the major thrust of this paper (but see [12] for an example of such a procedure). We did compute the serial correlation coefficient for lags of 1, 2, and 3 and also the partial correlation coefficient of orders 2 and 3 using the original nonstationary data. This was done for general information only, since the results obtained are useful qualitatively rather than quantitatively. It is important to point out here that even though a standard statistical test may only apply exactly under certain limited circumstances, qualitative information or approximate results can frequently be obtained over a much larger set of cases. We adopt this approach, but warn the reader when the results need to be interpreted with care.

The expressions used to calculate these values are as follows. Let $s(i)$ be the serial (or auto) correlation of order i for a time series $x(t)$, $t = 1 \cdots n$, with mean X. Let $sp(i)$ be the partial autocorrelation coefficient of order i; that is, $sp(i)$ is the correlation between elements $x(t)$ and $x(t+i)$ of the time series, having removed the effect of all correlations between $x(t)$ and $x(t+j)$ and $x(t+j)$ and $x(t+i)$, for $1 \leq j < i$. Then

$$s(i) = \frac{\frac{1}{n-i}\sum_{j=1}^{n-i}(x(j)-\bar{x})(x(i+j)-\bar{x})}{\frac{1}{n}\sum_{j=1}^{n}(x(j)-\bar{x})^2} \tag{3}$$

$$sp(2) = \frac{s(2)-s(1)^2}{1-s(1)^2} \tag{4}$$

$$sp(3) = \frac{s(1)^3 - 2s(1)s(2) + s(1)s(2)^2 + s(3)(1-s(1)^2)}{s(2)(s(1)^2 - s(2)) - s(1)(s(1) - s(1)s(2)) + (1-s(1)^2)}. \tag{5}$$

These expressions are drawn from [2] and the reader is referred to that text for further discussion. We also consider the problem of estimating $s(i)$ further in Section III-F2b.

5) Serial Correlation Results: The values obtained from these expressions are presented in Table III. Interpretation of these results should be done with care, since nonstationary data tend to yield spurious positive serial correlations. That is, if there are trends in the data, then consecutive samples are likely to both be in either a high or low trend and thus similar to each other relative to the mean. Therefore, we shall largely ignore positive serial correlations in this case; negative ones are almost certainly meaningful. Looking at the working days results in Table III, it is difficult to interpret the high positive correlations for the number of files used for the reason mentioned, although it does make sense to believe that if the system is busy today, it will be busy tomorrow. Periods of high system usage generally continue for several days for a given user and this should be reflected in these positive serial correlations. The negative correlations for New and Scr activity are much more meaningful, since they would not be produced by long term trends in the data. The author believes that this indicates a basically steady underlying rate of file creation or destruction. If a large number of files are created today, this implies that very few will be created tomorrow. Similarly for scratching files. The correlations for the all days data are largely due to day of the week effects (e.g., days 1 appart are both likely to be either working or nonworking days; days 3 appart are not, etc.) and therefore are not very useful.

TABLE III
SERIAL CORRELATION COEFFICIENTS

	All Days			Working Days		
	Used	New	Scr	Used	New	Scr
s(1)	.392	.269	.106	.315	-.122	-.111
s(2)	-.092	-.114	-.108	.295	-.114	-.056
s(3)	-.252	-.230	-.112	.224	-.122	-.120
sp(2)	-.289	-.199	-.120	.218	-.132	-.070
sp(3)	-.095	-.144	-.088	.087	-.159	-.137

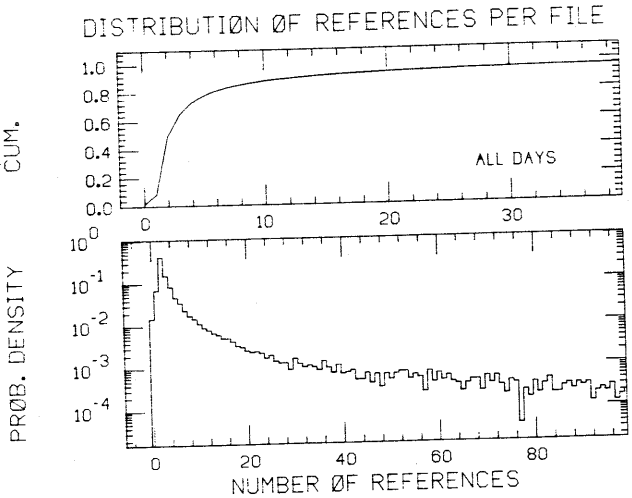

Fig. 4.

TABLE IV
NUMBER OF REFERENCES PER FILE

Number	Fraction	Cumulative
0	.015	.015
1	.069	.084
2	.411	.495
3	.152	.647
4	.084	.731
5	.047	.778
6	.035	.812
7	.023	.835
...	...	...
10	.011	.878
20	.0025	.930
50	.00048	.962
100	.00012	.980

D. Individual Files

1) Files Found to be Used Infrequently: Fig. 4 and Table IV show the empirical distribution of the number of times each file was used during the measurement period. The average file is used two or fewer times, but the average number of references per file is 10.6 (see Table I). The distribution is thus highly skewed; most files are used very little and a few are accessed a large number of times. A note of explanation is needed for these data, however. It is detected that a file has been scratched by looking for it late at night (since it existed 24 hours earlier) and not finding it. This implies that the file is always shown as not having been referenced on the day on

which it is scratched. We have assumed (by setting $I(i, L(i)) = 1$) that when a file is scratched, it must be referenced, that is, the user probably copied it to tape or at least made a listing of it. Thus, any file which was both created and scratched during the measurement period must be referenced at least twice. This does not necessarily always happen, but our feeling is that setting $I(i, L(i)) = 1$ is closer to being accurate than not doing so.

The small number of references to most files was rather surprising to the author, but there appears to be a reasonable explanation (which will also be important later). A user's perception of his file activity tends to be heavily weighted towards those files which are actively used, but in fact most files seem to be created as temporaries, e.g., to hold a modified version of a program. They are created, left unreferenced until the user needs the file space, and then destroyed. The number of files which contain information in regular use or which follow the intuitively appealing pattern of intensive use during development followed by occasional use later is relatively small.

2) File Size Distribution: Measurements of file size distributions are given in Fig. 5 and in Tables V and VI. It is possible to define the distribution of file sizes in three ways. Let $SU(i)$ be the unweighted empirical file size distribution, $SWU(i)$ be the file size distribution as weighted by use, and $SWL(i)$ be the file size distribution as weighted by lifetime. Then

$$SU(i) = \sum_{j=1}^{N\text{files}} (1 \quad \text{if } S(j) = i; 0 \text{ otherwise})/N\text{files} \qquad (6)$$

$$SWU(i) = \sum_{j=1}^{N\text{files}} (N\text{ref}(j) \quad \text{if } S(j) = i; 0 \text{ otherwise})/N\text{ref} \qquad (7)$$

$$SWL(i) = \sum_{j=1}^{N\text{files}} ((L(j) - F(j) + 1) \quad \text{if } S(j) = i; 0 \text{ otherwise}) \Big/ \sum_{j=1}^{N\text{files}} (L(j) - F(j) + 1). \qquad (8)$$

The unweighted distribution is simply that obtained by considering the file sizes of all those files in existence during the period of measurement. The weighted by use distribution is that seen by an observer picking an arbitrary file reference and observing the size of the file referenced. The weighted by lifetime distribution is that which would be observed by selecting a random file (on the disks) at a random time. Table V lists the unweighted file size distribution and Table VI gives the mean and median for all three cases. It is evident that larger files are used more heavily. This observation is intuitively reasonable. Large files are "expensive" to keep since they use a large amount of scarce disk space, and therefore if a file is not used frequently enough, it is likely to be scratched.

We noted earlier that files were classified both as to size and class. Table VII shows the average file size within each class.

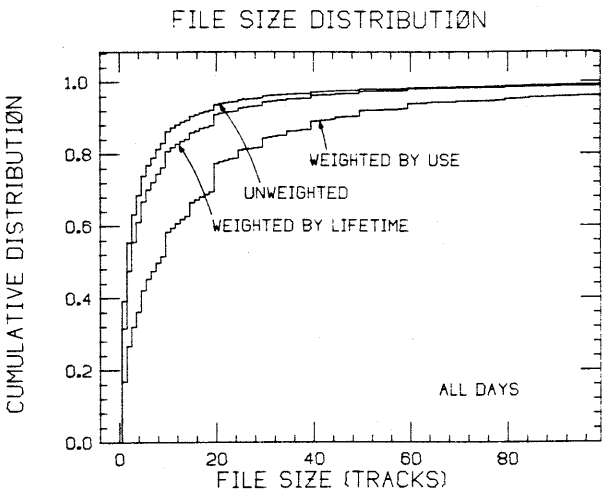

Fig. 5.

TABLE V
UNWEIGHTED FILE SIZE DISTRIBUTION

Size	Fraction
1	.391
2	.162
3	.078
4	.053
5	.053
6	.030
7	.021
...	....

TABLE VI
MEAN/MEDIAN FILE SIZE

Weighted by	Mean	Median
Unweighted	7.08	1.7
Lifetime	8.99	2.3
Use	18.85	8.3

TABLE VII

File Size Range	Average File Size Weighted By Lifetime	Use	Unweighted
1	1	1	1
2-3	2.34	2.35	2.33
4-7	5.12	5.21	5.12
8-15	10.74	11.22	10.52
16-31	22.09	22.30	21.86
32-63	44.57	45.22	44.76
>=64	117.70	118.16	114.95
Total	9.02	18.85	7.08

Interestingly, the mean file size by the three measurement methods are almost the same for each size class, but very different overall. This suggests (but does not in any sense prove) that within the particular size groupings that we have selected, files are referenced relatively uniformly with respect to size. Additional data arranged by file size and class appear in Table VIII, which we discuss below.

E. File Lifetime

1) Definition of File Lifetime: An interesting aspect of file behavior is file lifetime, that is, how long the file exists between creation and being scratched. Let $LF(i)$ be the empirical file lifetime distribution. Then

$$LF(i) = \sum_{j=1}^{N\text{files}} (1 \quad \text{if } D(j) - B(j) + 1 = i; 0 \text{ otherwise})/N\text{files}. \qquad (9)$$

TABLE VIII
ALL DAYS

Size (tracks)	Class	Number of Files	Fraction	Number of Refs.	Fraction	Number of Inter-reference Intervals	Mean Inter-reference Interval	Coef. of Variation	Fit Parameters a	b	c	Mean Lifetime	Mean Rate of Reference	Fraction Died
1	Library	12	.00048	256	.00107	244	4.30	3.34	.940	.551	.023	311	.111	.417
2-3		29	.00116	1022	.00428	944	4.65	3.16	.956	.396	.019	186	.167	.310
4-7		120	.00482	4926	.02071	4828	4.13	2.77	.930	.494	.031	319	.175	.242
8-15		249	.01000	17253	.07223	17005	2.99	2.73	.976	.472	.026	330	.277	.265
16-31		232	.00932	24591	.10295	24360	2.26	2.36	.982	.562	.034	347	.382	.272
32-63		67	.00269	9424	.03945	9357	1.77	1.61	.964	.707	.089	86	.534	.313
>=64		14	.00056	2129	.00891	2115	1.86	2.37	.991	.636	.031	422	.477	.286
1	Other	6952	.27922	33249	.13919	26421	9.58	2.56	.851	.349	.021	61	.068	.787
2-3		5180	.20805	33185	.13892	28092	8.51	2.67	.888	.317	.020	63	.078	.755
4-7		3394	.13632	31076	.13010	27733	6.36	2.88	.925	.351	.020	64	.113	.776
8-15		2421	.09724	27894	.11677	25508	5.04	2.91	.946	.373	.022	67	.143	.788
16-31		1087	.04366	18546	.07764	17471	4.05	3.00	.959	.425	.023	73	.187	.751
32-63		456	.01831	12892	.05397	12442	3.05	3.33	.982	.473	.018	81	.261	.717
>=64		385	.01546	12191	.05104	11807	2.14	2.68	.984	.608	.030	74	.374	.805
1	Active	2777	.11154	6629	.02775	3884	14.01	2.24	.831	.202	.017	34	.064	.924
2-3		774	.03109	1838	.00769	1073	12.24	2.19	.808	.256	.021	28	.073	.943
4-7		406	.01631	958	.00401	555	9.98	2.68	.932	.193	.013	23	.087	.931
8-15		183	.00735	412	.00172	231	6.97	3.48	.960	.279	.011	16	.113	.945
16-31		116	.00466	271	.00113	155	5.06	2.93	.890	.639	.030	13	.200	1.00
32-63		43	.00173	105	.00044	62	3.11	1.46	-	-	-	8	.274	.930
>=64		1	.00004	4	.00002	3	11.67	1.23	-	-	-	37	.111	1.00
1	All	9741	.39124	40134	.16802	30549	10.10	2.52	.852	.317	.020	52	.067	.825
2-3	Files	5983	.24030	36045	.15090	30159	8.52	2.66	.887	.316	.020	58	.079	.777
4-7		3920	.15744	36980	.15481	33116	6.09	2.90	.930	.353	.020	61	.117	.776
8-15		2853	.11459	45559	.19073	42744	4.23	2.97	.959	.403	.022	71	.175	.752
16-31		1435	.05764	43406	.18172	41986	3.02	2.96	.975	.490	.025	84	.263	.693
32-63		566	.02273	22421	.09386	21861	2.50	3.17	.988	.523	.019	74	.332	.686
>=64		400	.01607	14324	.05997	13925	2.10	2.65	.985	.612	.031	79	.387	.788
All	Library	723	.02904	59621	.24959	58903	2.58	2.70	.979	.529	.029	303	.321	.272
	Other	19875	.79826	169033	.70763	149474	6.23	2.94	.922	.376	.021	64	.108	.774
	Active	4300	.17270	10219	.04277	5963	12.70	2.33	.853	.209	.017	30	.070	.931
All	All	24898	1.0	238871	1.0	214340	5.41	3.08	.938	.399	.020	59	.126	.786

There is one problem with this definition. $D(i)$ is undefined for files which have not been scratched at the end of the measurement period, and from Table I we see that this is 21 percent of all files. There appear to be three ways to deal with this difficulty, but none of the three is completely satisfactory. These are as follows. 1) Consider only files which are actually scratched during the period of observation. The difficulty is that this will give an incorrect estimate unless the system is completely in steady state. This is because if files are being created more quickly than they are scratched, the file lifetime distribution will be overweighted towards short lived files; the long lived files will not have had a chance to die yet. 2) Assume that all files are destroyed on the last day of observation. This does not appear to make any sense statistically. 3) Assume that since the last day of observation is a random point in time, it happens to exactly bisect the lifetime of all files in existence on that day. This argument is closely related to that used to derive the distribution of time to the next event in a renewal process when the process is observed at a random point in time [4]. The expected time to the next event (i.e., file death) is equal to the time since the last event (file birth). This approach is satisfactory if all files have finite lifetimes and if there is a large number of such files, so that a distribution may be collected. We have selected 1) as our means of estimating the file lifetime distribution; there are arguments to be made for 3) as well.

2) File Lifetime Measures: The measured file lifetimes are presented in Fig. 6 (by size) and Fig. 7 (by class). The file lifetime appears to be only slightly related to size, but is strongly influenced by class. Active files get scratched relatively quickly (see Fig. 7) and libraries tend to never be scratched. The mean lifetime is shown in Table VIII for each size/class combination. Also shown in Table VIII is the fraction of files of each size/class that died (were ever scratched). Among the libraries, a full 73 percent were still in existence at the end of the measurement interval. For the reasons noted above, our measurements of file lifetimes may be biased, and considering the large number of files not scratched, that bias may be quite significant.

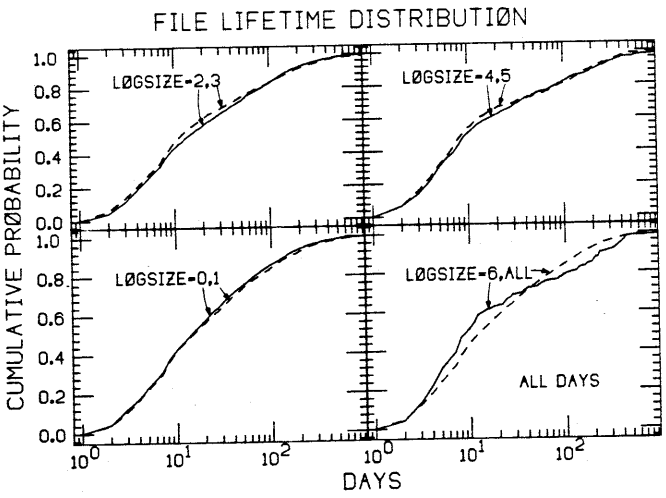

Fig. 6.

Further study of file lifetimes seems to have only slight payoff in terms of file migration, which is the ultimate aim of this study. Therefore, no attempt has yet been made to model file lifetimes. An effort in this direction is being considered for future research.

F. File Reference Patterns

1) Interreference Times:

a) Definitions and Tabulation: A very important aspect of file reference behavior from the point of view of this paper

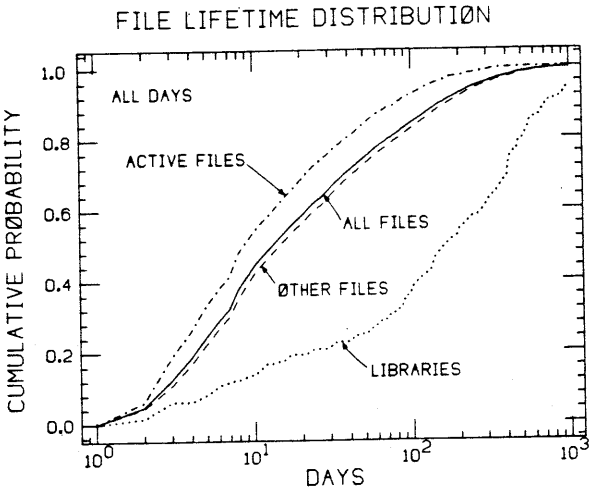

Fig. 7.

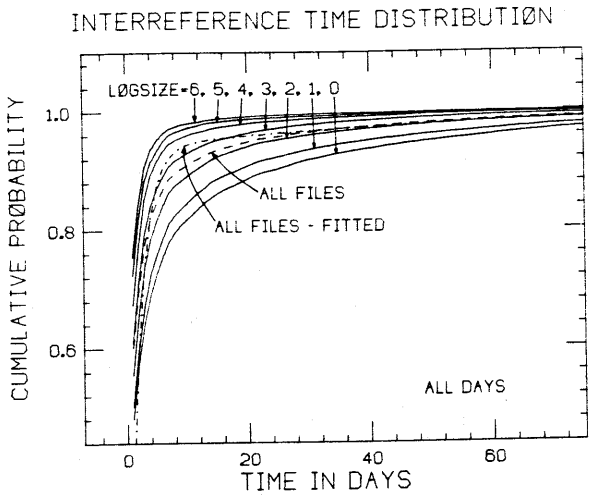

Fig. 8.

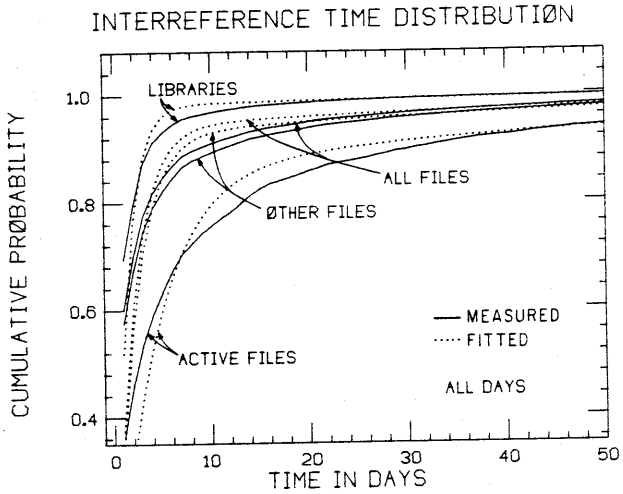

Fig. 9.

is the actual sequence of referenced-nonreferenced days, and in particular, the distribution of times between references. Let $g(i, \text{logsize}, \text{class})$ be the empirical probability mass function for the times between references to files belonging to size class "logsize" and type class "class." Let "*" denote the sum over all possible entries for that argument and let "-" denote an unspecified entry for that argument. Then

$$g(i, \text{logsize}, \text{class}) = \frac{\sum_{j \in (\text{logsize}, \text{class})} \sum_{k=1}^{N\text{refs}(j)-1} \begin{pmatrix} 1 & \text{if } C(j,k) = 1; \\ 0 & \text{otherwise} \end{pmatrix}}{\sum_{j \in (\text{logsize}, \text{class})} (N\text{refs}(j) - 1 \quad \text{if } N\text{refs}(j) \geq 1)} \tag{10}$$

$$g(i, *, \text{class}) = \frac{\sum_{j \in \text{class}} \sum_{k=1}^{N\text{refs}(j)-1} (1 \quad \text{if } C(j,k) = i; 0 \text{ otherwise})}{\sum_{j \in \text{class}} (N\text{refs}(j) - 1 \quad \text{if } N\text{refs}(j) \geq 1)} \tag{11}$$

$$g(i, \text{logsize}, *) = \frac{\sum_{j \in \text{logsize}} \sum_{k=1}^{N\text{refs}(j)-1} (1 \quad \text{if } C(j,k) = 1; 0 \text{ otherwise})}{\sum_{j \in \text{logsize}} (N\text{refs}(j) - 1 \quad \text{if } N\text{refs}(j) \geq 1)} \tag{12}$$

$$g(i, *, *) = \frac{\sum_{j=1}^{N\text{files}} \sum_{k=1}^{N\text{refs}(j)-1} (1 \quad \text{if } C(j,k) = i; 0 \text{ otherwise})}{\sum_{j=1}^{N\text{files}} (N\text{refs}(j) - 1 \quad \text{if } N\text{refs}(j) \geq 1)}. \tag{13}$$

We let $G(i, \text{logsize}, \text{class})$ be the cumulative distribution. In Fig. 8 we show as solid lines the distributions $G(i, \text{logsize}, *)$ and in Fig. 9 the distributions $G(i, *, \text{class})$. (The lines showing fitted results are discussed below.) The number of interreference intervals and the mean interreference times are given in Table VIII for each size and class combination.

b) Coefficient of Variation of File Interreference Times: Another column in Table VIII shows the coefficient of variation of the interreference time distribution, as aggregated over all of the files in a size/class combination. Let $CV(\text{logsize}, \text{class})$ be the coefficient of variation for a size/class case. Then

$$CV(-,-) = \frac{\left(\sum_i g(i,-,-) \left(i - \sum_i ig(i,-,i) \right)^2 \right)^{1/2}}{\sum_i ig(i,-,-)}. \tag{14}$$

We observe that these coefficients of variation are all over 1.0 and the mean [i.e., for $g(i, *, *)$] is 3.08. The distribution is thus moderately skewed, but this implies nothing about the reference pattern to individual files. Such a skewed distribution could equally well be obtained from either of two circumstances (among many others): 1) each file is referenced with the same skewed interarrival time distribution peculiar to its size and class, or 2) each file is referenced as a Bernoulli process, but with the rate of reference varying between different files. (The Bernoulli process is the discrete time analog of

the Poisson process, by which a file i has a constant probability $p(i)$ of being referenced each day. The interevent times are geometric.)

c) Hazard Rate for File Interreference Times: Another important statistic which has been measured is the hazard rate of the empirical interreference time distribution. The hazard rate is defined as

$$h(i,-,-) = g(i,-,-)/(1 - G(i-1,-,-)). \quad (15)$$

Examination of the hazard rate function (see Figs. 10 and 11) indicates that it declines sharply for a while and then becomes (after 20 days or so) relatively flat.

d) Fit With Two Part Geometric: A distribution that is frequently used to model discrete empirical distributions of the form we have described (moderately skewed, declining but eventually flat hazard rate) is the weighted sum of geometric distributions. We have selected a two part geometric to fit $g(i,-,-)$. Let $gf(i,-,-)$ and $Gf(i,-,-)$ be the appropriate fitted distributions. Then

$$gf(l,-,-) = ab(1-b)^{i-1} + (1-a)c(1-c)^{i-1} \quad (16)$$

for some a, b, and c to be selected in each case.

There are three methods in common use used to fit the parameters of a fitted distribution to empirical ones: least squares, maximum likelihood, and method of moments. The method of moments is by far the simplest to use, but is vulnerable to instability in the case that the number of points in the sample distribution is small. That does not appear to be a problem with our data, since there are a large number of interreference intervals in almost every size/class combination. We have therefore used the method of moments to estimate the parameters a, b, and c in each case. The method of moments involves calculating the moments (in this case, the first three) of the empirical distribution and selecting the set of values of the distribution parameters (a, b, c) that yield the same moments. This was done and the results appear in Table VIII. We note that in two cases, no set of parameters a, b, and c existed that yielded the observed moments.

In Fig. 8 the overall fitted distribution $Gf(i,*,*)$ is shown as a dot-dash line and in Fig. 9, the fitted distributions $Gf(i,*,-)$ are shown as dotted lines. As may be seen, the quality of the fit is at best fair. This observation is confirmed by the chi-square test for goodness of fit, which rejected the fit in almost all cases at the 99 percent confidence level. No attempt has yet been made to fit the observed distribution with other postulated distributions; we are considering doing so at a later time. Use of this fitted distribution is made in [19], however, and despite the fair to poor quality of the fit, surprisingly good results are obtained for its use with file migration algorithms.

2) Statistical Testing of File Reference Patterns:

a) Introduction: In the last section we obtained the distribution of times between references to a file aggregated over all files in a size/class combination. As we noted, this says very little about the reference pattern to individual files. In particular, there are three questions that should be answered

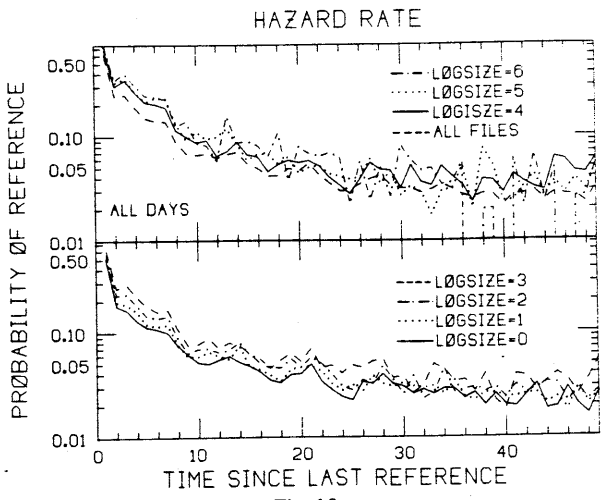

Fig. 10.

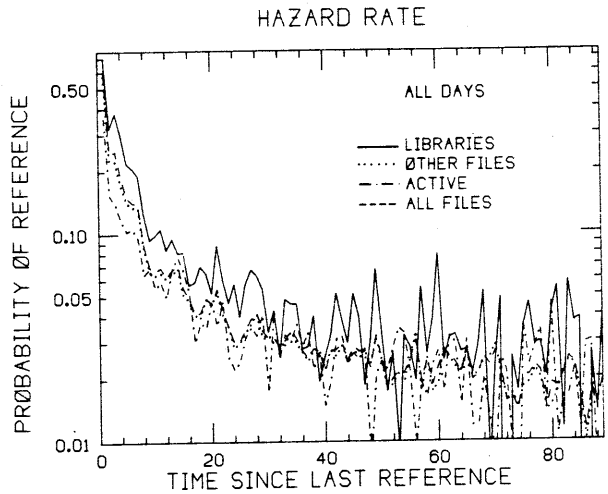

Fig. 11.

with regard to individual file reference patterns: 1) is the rate of reference to a file stationary, 2) are successive interreference intervals correlated, and 3) is the reference pattern to a file the Bernoulli process with a parameter specific to that file? (The Bernoulli process, as described below, is of interest for three reasons: it is a standard null hypothesis, its memoryless property has implications for file migration algorithms, and it has been found in previous research [20].) Testing a time series to obtain the answers to 1), 2), and 3) would be straightforward were it not for the fact that most of the sequences of interreference intervals for the files are very short. As noted earlier, 50 percent of the files were referenced two or fewer times. It makes absolutely no sense to test a time series of 2 elements (or even 3 or 4). Further, results do not appear to be available in the published literature for the distributions to be expected when applying standard statistical tests to such small samples. Our basic methodology, which we summarize below, consists essentially of restricting our tests to those file reference patterns which can reasonably be tested, and then testing them against distributions obtained through pseudorandom number driven simulation.

b) Selection of Files to Test: The only files tested were those that 1) were referenced at least six times, 2) had a value of $(Nref(i) - 1)/(L(i) - F(i))$ less than 0.95, and 3) for which

$L(i) - F(i)$ was greater than or equal to 10. The idea was to eliminate those files that either were referenced too few times, or were referenced over such a short period of time that the number of possible reference patterns was too small to allow for statistical testing. We found that this elimination procedure left us with 5334 (21.4 percent of all) files for all days and 4818 (19.4 percent of all) files in the working days case. Approximately 80 percent of all of the file references remained, however, since those files eliminated were those which were referenced very few times.

c) Generation of Null Hypothesis Distributions: We tested for monotonic trend, serial correlation, and skewness of the distribution. The null hypothesis in each case was that of a stationary, uncorrelated Bernoulli process. The statistics used are explained below. For short discrete-time time series, there appear to be very few results known, and so the means and confidence intervals for the statistical tests were determined by generating a large number of random Bernoulli processes (of appropriate rates and lengths) and applying the statistical test to those time series. That is, a large number of Bernoulli time series were generated using a pseudorandom number generator. For each of these time series, each of the statistics given below in (17)–(19) were calculated. This yielded the distribution of the statistic in the case of the null hypothesis, i.e., the Bernoulli process.

We do note one problem with our statistical procedure. Because of the short length of most of the time series we deal with, the values of the statistics (17)–(19) used will vary widely even in the case that the null hypothesis is valid. Thus, it will be difficult to determine that our sample time series are not Bernoulli, and the test procedure may not actually be very powerful. An alternative approach is to find some way of aggregating the data from several time series, but we leave this for future research.

d) Testing for Trend–Test and Results: Lewis and Shedler [11] (see also [5]) give a statistic for testing a continuous-time time series for being a stationary uncorrelated Poisson process against the alternative of a Poisson process with monotonic trend. As adapted for discrete-time series, it is

$$Tr = \frac{z - T/2}{T/\sqrt{12N\text{ref}(i)}} \quad \text{where } T = L(i) - F(i) + 1,$$

$$Z = \sum_{j=F(i)}^{L(i)} I(i,j)(j - F(i))/N\text{ref}(i). \tag{17}$$

This statistic is simple to calculate and seems to be equally suitable for testing discrete time series.

As shown in Table IX, approximately 35 percent of the file reference patterns displayed significant trend (at the 99 percent confidence level); the vast majority of such cases were of declining trend. This is as one might expect–a file is used less after it has been around for a while. We comment that a skewed but stationary distribution will display a larger variance in Tr than the Bernoulli process; thus the 99 percent confidence level was used rather than the more usual 95 percent in order to minimize false rejection.

TABLE IX
TREND TEST

Fraction That Reject (99% Confidence Level) For Increasing/Decreasing Trend		
	All Days	Working Days
Unweighted	.052/.282	.051/.274
Weighted by Lifetime	.087/.323	.077/.304

TABLE X
SERIAL CORRELATION TEST

Fraction That Reject (99% Confidence Interval) For Excessive Correlation	
All Days	.042
Working Days	.066

e) Testing for Serial Correlation–Test and Results: The usual estimator (and usually the most powerful; see [11]) for serial correlation is given in (3); that estimator, however, is an approximation that is acceptable only for long time series. The correct estimator is (from [10])

$$S(k) = \frac{\frac{1}{n-k}\sum_{i=1}^{n-k}\left(x_i - \frac{1}{n-k}\sum_{i=1}^{n-k} x_i\right)\left(x_{i+k} - \frac{1}{n-k}\sum_{i=1}^{n-k} x_{i+k}\right)}{\left(\frac{1}{n-k}\sum_{i=1}^{n-k}\left(x_i - \frac{1}{n-k}\sum_{i=1}^{n-k} x_i\right)^2 \frac{1}{n-k}\sum_{i=1}^{n-k}\left(x_{i+k} - \frac{1}{n-k}\sum_{i=1}^{n-k} x_{i+k}\right)^2\right)^{1/2}}. \tag{18}$$

We warn the reader that this estimator is biased to the approximate extent of $1/n$, where n is the number of elements in the time series. Thus, knowledge of the actual mean and distribution of the estimator is essential.

The serial correlation was estimated using (18) for all files showing no significant trend. The fraction of files which displayed significant serial correlation is shown in Table X, where we see that only about 5 percent of all files rejected the hypothesis of no correlation at the 99 percent confidence level. (Our earlier comment about skewed distributions holds here also.) In almost all cases, those significant correlations found were positive ones, not negative ones. Most correlation coefficient values observed were small (65 percent were less than 0.25) so that independent of their significance, their predictive power is low.

An additional test was made to look for possible serial correlations in interreference intervals. Fig. 12 shows an $x - y$ plot of successive interreference intervals taken over all interreference intervals for all files. Each entry is the (truncated) logarithm base 2 (in hexadecimal) of the number of points at that location. There is no pattern apparent to the author in this figure.

The cross correlation (between x and y) was computed over all of the points in this figure and a value of 0.295 (0.307 for working days) was obtained. This is a high and significant value, but it presumably reflects the fact that we have lumped all files together, rather than actual correlations within indi-

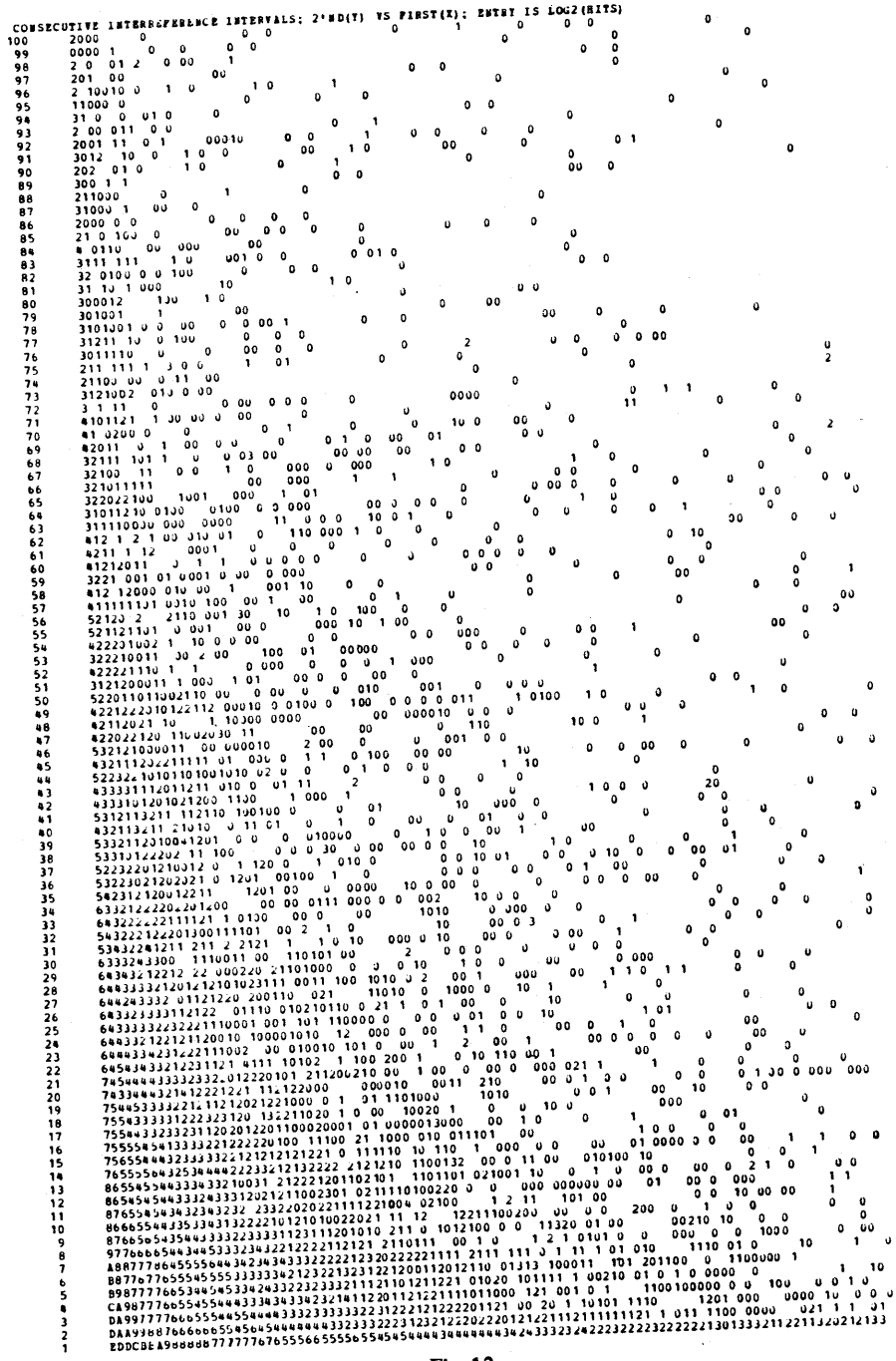

Fig. 12.

vidual file reference processes. That is, files referenced at a given rate are likely to continue being referenced at a similar rate; this rate may be characteristic of the file or its size or its class.

f) Clustering of References to Files Found: We also collected data to look at the extent of clustering in the reference pattern to a file. For every period of 15 days during which a file existed, the number of references to that file was counted. The distribution of the number of references to a file in 15 working days is given in Table XI. As can be seen, there is substantial clustering; if a file is used a few times, it is just as likely to have been used on many more days. This confirms one's intuitive idea of file use–if a file is in active use (development or production), it is likely to be referenced with high probability each day over a several day period.

g) Testing for the Bernoulli Process: Testing an interarrival process against a known statistical distribution is usually done with the chi-square goodness of fit test [see (1)] using the empirical interarrival distribution. This is only feasible, however, if the number of events in the arrival process is quite large (at least 25 or 50 in this case). We choose instead to measure the coefficient of variation of the interarrival intervals for each file and then determine whether that coefficient of variation is an acceptable one under the hypothesis of a Bernoulli process.

The estimator for the coefficient of variation used is

$$\frac{\left(\frac{1}{N\text{ref}(i)-2}\sum_{j=1}^{N\text{ref}(i)-1}\left(C(i,j)-\left(\sum_{j=1}^{N\text{ref}(i)-1}C(i,j)/(N\text{ref}(i)-1)\right)\right)^2\right)^{1/2}}{\sum_{j=1}^{N\text{ref}(i)-1}C(i,j)/(N\text{ref}(i)-1)}. \quad (19)$$

We again warn the reader that this estimator does not approach its asymptotic distribution until the time series becomes quite long. We found (see Table XII) that about 40 percent of all files tested (those with no trend) showed unacceptable coefficients of variation at the 99 percent confidence level. In almost every case, this was because the coefficient of variation was too large (i.e., the distribution was skewed). Thus, our earlier caution in the trend and correlation tests regarding skewed distributions appears to have been appropriate.

It also happens that 96 percent of all coefficients of variation are below 2.0, whereas the mean coefficient of variation when computed over all interreference intervals for all files is 3.08. Similar high coefficients of variation are observed for individual size/class combinations (see Table VIII). Thus, the interreference distribution for a size/class combination may not be a very good estimate of the distribution for an individual file.

h) Conclusions and Summary of Tests of File Reference Patterns: From our tests in this Section we can see that files are frequently used less as they age, and that the interreference time distribution is more skewed than the Bernoulli. There is little if any significant serial correlation within the reference process to one file. A file migration algorithm would therefore find it useful to condition on the time since the last reference and perhaps the age of the file; conditioning on previous interreference intervals does not appear to be helpful.

It is worth mentioning that different results were reported by Stritter [20] who analyzed the same data. His descriptions of the tests he ran are vague, but he found that the number of files displaying either trend or large coefficients of variation to be small enough that he was willing to describe the file reference process as "Poisson." It appears that he tested only a subset of the file reference processes (selection process unspecified); thus, from his description of his results (which are not quantified) it is not clear if there is actually a conflict with our results or not. We believe that our methodology has been more thorough and that the results presented here are correct.

3) Rates of Reference to Files by Size and Age: Another interesting statistic is the measured rates of reference to files. The rate of reference to a file i, $R(i)$, is just $N\text{ref}(i)/(L(i) - F(i) + 1)$. That is, it is just the number of times the file is referenced over the period of observation divided by the lifetime of the file during the period of observation.

We have computed the cumulative distribution of $R(i)$ in four ways. $RU(x)$ is the unweighted distribution

$$RU(x) = \sum_{i=1}^{N\text{files}} (1 \quad \text{if } R(i) \leqslant x)/N\text{files}. \quad (20)$$

This distribution can be weighted by file lifetime as

$$RL(x) = \sum_{i=1}^{N\text{files}} (L(i) - F(i) + 1 \quad \text{if } R(i) \leqslant x;$$

$$0 \text{ otherwise}) \Big/ \sum_{i=1}^{N\text{files}} (L(i) - F(i) + 1) \quad (21)$$

TABLE XI
DISTRIBUTION OF NUMBER OF TIMES A FILE IS USED IN 15 WORKING DAYS

Number	Probability	Cumulative
0	.600	.600
1	.112	.713
2	.060	.773
3	.037	.809
4	.025	.834
5	.020	.854
6	.017	.871
7	.015	.886
8	.013	.899
9	.012	.911
10	.011	.922
11	.012	.934
12	.012	.946
13	.013	.959
14	.014	.973
15	.027	1.00

or by use as

$$RUS(x) = \sum_{i=1}^{N\text{files}} (N\text{ref}(i) \quad \text{if } R(i) \leqslant x;$$

$$0 \text{ otherwise})/N\text{ref} \quad (22)$$

or by file size as

$$RS(x) = \sum_{i=1}^{N\text{files}} (S(i) \quad \text{if } R(i) \leqslant x;$$

$$0 \text{ otherwise}) \Big/ \sum_{i=1}^{N\text{files}} S(i). \quad (23)$$

The measured values for $RU(x)$, $RL(x)$, $RUS(x)$, and $RS(x)$ all appear in Fig. 13. The mean and median rates of reference are given in Table XIII.

It is also interesting to compute the rate of reference to a file as a function of its age. The age of a file is defined to be the number of days since it was created. We compute the mean rate of reference $RA(i,-,-)$ to a file as a function of its age i by considering those files which were created during the period of observation.

$$RA(i,-,-) = \sum_{j\in(-,-)} A(j,i) \Big/ \sum_{j\in(-,-)} (1 \quad \text{if } (L(j) - F(j) + 1) \geqslant i; 0 \text{ otherwise}). \quad (24)$$

Note that $RA(1,-,-) = 1$. The function $RA(i,-,-)$ is rather irregular for moderately large values of i, so we have smoothed it by using a five day rectangular window. The smoothed function $RA(i,-,-)$ is shown in Figs. 14 and 15. The up-down regular oscillation of RA is due to the interaction of the smoothing window with weekends. Since most files are created during working days, a file of age 19 is less likely to be referenced than a file of age 21. (Since a file of age 21 is very likely to be in existence on a working day.) A smoothing window of a multiple of 7 days would have removed this oscillation, but was judged to be too wide.

Also shown in Figs. 14 and 15 is the value of $RAf(i,-,-)$ obtained from our fitted two part geometric distribution of (16). We assume that the file is referenced as a renewal process

TABLE XII
COEFFICIENT OF VARIATION TEST

Fraction That Reject (99% Confidence Interval) The Bernoulli Process	All Days	Working Days
Unweighted	.354	.367
Weighted by Use	.494	.470

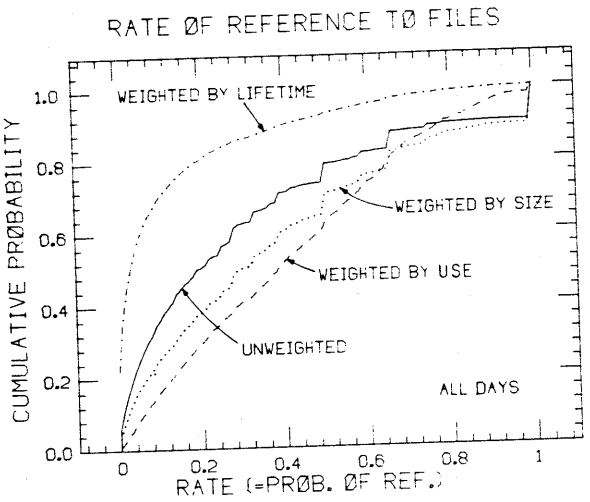

Fig. 13.

TABLE XIII
MEAN/MEDIAN RATE OF REFERENCE

Weighted By	Mean	Median
Unweighted	.306	.19
Use	.410	.386
Lifetime	.121	.030
Size	.380	.296

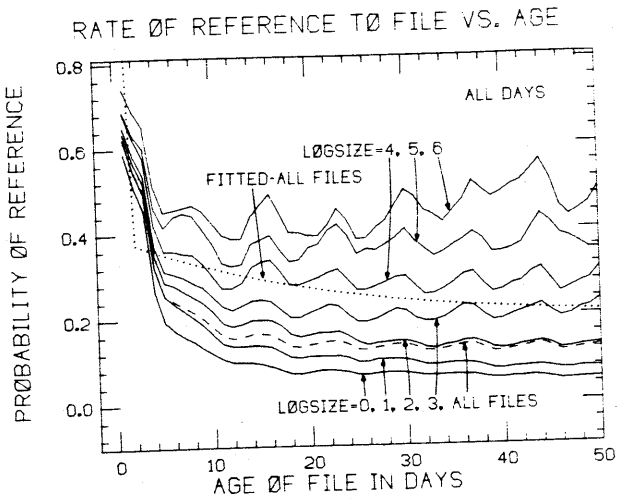

Fig. 14.

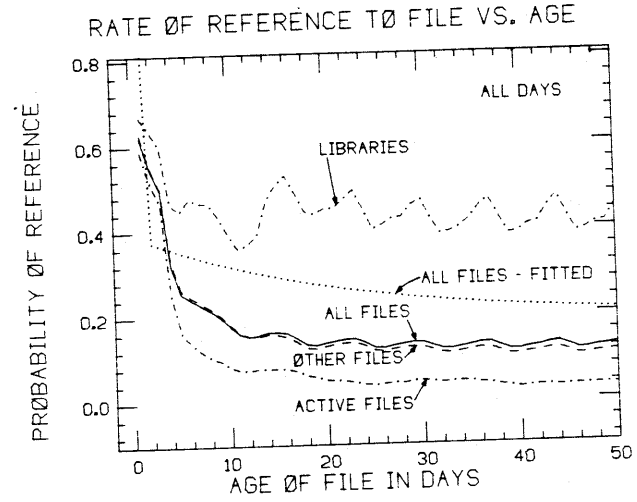

Fig. 15.

with the $gf(i,-,-)$ interarrival distribution, and that the file was referenced on the day of its birth. Then from the renewal equation [4] one has

$$RAf(i,-,-) = \sum_{k=1}^{i-1} RAf(k,-,-)g(i-k,-,-), i>1,$$

where $RAf(1,-,-) = 1$. (25)

We note that because a file is referenced on the day it is created, the predicted rate of reference declines with age. The fact that the process is stationary (e.g., a renewal process) does not imply that the rate of reference need be constant as a function of age.

This declining rate of reference to a file with age is consistent with our earlier determination that about 35 percent of all tested files showed a declining rate of reference during their lifetime.

4) Expected Time to Next Reference Increases with Time Since Last Use: In performing file migration, one would generally like to remove that file with the largest space-time product to the next reference((size of file) x (time to next reference)). (Criteria for file migration are discussed in detail in the companion paper to this.) One can estimate the time to the next reference by conditioning on the time since the last reference, among other things. Let $E(i,-,-)$ be the expected time to the next reference, given that the file has not been referenced for i days. If there have been i days elapsed since the last reference ($i=0$ means that the reference was that day), then by conditioning only on the time since last reference

$$E(i,-,-) = \sum_{i=k+1}^{\infty} g(k,-,-)(k-i)/(1-G(i,-,-)). \quad (26)$$

In Figs. 16 and 17 we show the expected time to next reference, both as calculated from the empirical distributions $g(i,-,-)$ and from the fitted distributions $gf(i,-,-)$. The dotted lines in each case are the fitted distributions.

The important observation to be made from Figs. 16 and 17 is that the expected time to next reference is a generally increasing function of the time since the last reference. Thus, one is more likely to want to migrate a file which has not been used for a long period of time than a file which has been recently referenced. This issue is considered further in [19].

IV. SUMMARY, CONCLUSIONS, AND PLANS FOR FURTHER RESEARCH

We have observed that most files are referenced very few times. Of those files susceptible of statistical testing, we found that about one-third showed a declining rate of reference. Almost half of the files showing no trend displayed reference patterns which ruled out a Bernoulli process model for the reference process; almost no serial correlation was detected

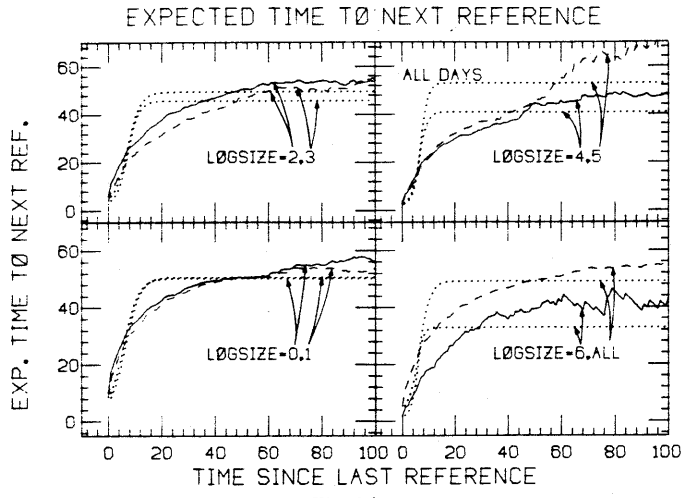

Fig. 16.

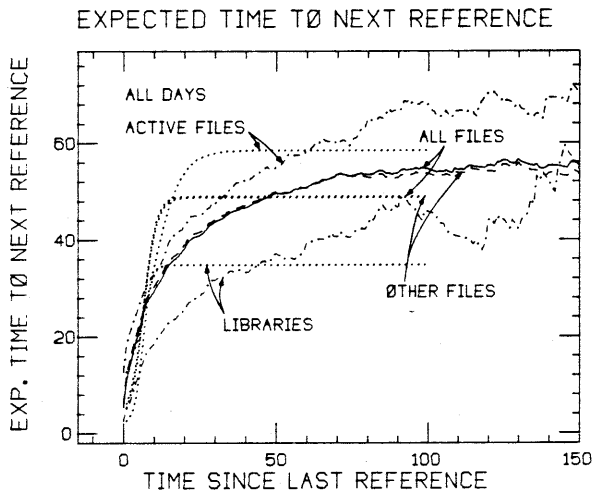

Fig. 17.

between successive interreference intervals. Thus, for those files displaying no trend, a renewal process model with a moderately skewed interreference time distribution would appear to be appropriate. Agreeing with the observed frequent decline in the rate of reference to a file with its age is the overall decrease (when aggregated over all files) in the rate of reference to a file with age. We also note that we were not able to fit the interreference time distribution with a two part geometric. Our interpretation of the results enumerated in this paragraph is that a Markov chain model for the file reference process, if one exists, is likely to require a minimum of three or four states (including a state which represents file death). A very simple renewal process model or even a two state Semi-Markov Process model [11] seems insufficient to represent the observed properties of the data. Further research to develop a satisfactory model for the file reference process is planned.

The goal of our data analysis in this paper has been twofold: 1) describe and characterize to whatever extent reasonable the file behavior patterns observed, and 2) develop a basis for the specification of file migration algorithms. For a file migration algorithm, it appears to be useful to use: the time since last reference, the file size, the file class, and the file age in order to predict the time to next reference for a file. The lack of serial correlation suggests that conditioning on previous file interreference intervals will not be useful.

In [19] we use the data analysis in this paper as a basis for the construction and evaluation of a number of file migration algorithms. There we find that the size of the file and the time since it was last used are most important in determining when to migrate that file. Algorithms are generated that specify how long to keep a file after it is last used based on the file size, age, type, etc. These algorithms include both those based on the analysis given here and those obtained from the literature. The algorithms are evaluated using trace driven simulation.

For our description and analysis of file reference data in this paper to be useful in the sense of being applicable to other installations, there must be some reason to think that file behavior will be similar across users and computer systems. Until data is gathered and analyzed for other systems, it is, of course, impossible to say, but we believe the following. Users use text editor files in much the same manner everywhere, subject to the distortions induced by the accounting or file migration algorithms. User access patterns to nontext editor files, such as data files or databases, are still unknown and it is not reasonable to claim that our results here will apply.

Acknowledgment

The author would like to extend many thanks to Dr. E. Stritter for collecting the data used in this paper and for making it available to the author for this research.

References

[1] P. J. Bickel and K. A. Doksum, *Mathematical Statistics: Basic Ideas and Selected Topics.* San Francisco: Holden-Day, 1977.
[2] G.E.P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, revised ed. San Francisco: Holden-Day, 1976.
[3] C. J. Conti, "Concepts for buffer storage," *IEEE Comput. Group News*, vol. 2, pp. 9–13, Aug. 1969.
[4] D. R. Cox, *Renewal Theory.* London: Methuen, 1962.
[5] D.R.Cox and P.A.W. Lewis, *The Statistical Analysis of Series of Events.* London: Methuen, 1966.
[6] P. J. Denning, "Virtual memory," *Comput. Surveys*, vol. 2, pp. 153–189, Sept. 1970.
[7] —, "On modeling program behavior," in *Proc. Spring Joint Comput. Conf.*, 1972, pp. 937–944.
[8] R. Fajman and J. Borgelt, "WYLBUR: An interactive text editing and remote data entry system," *Commun. Ass. Comput. Mach.*, vol. 16, pp. 314–322, May 1973.
[9] K. R. Kaplan and R. O. Winder, "Cache based computer systems," *Computer*, vol. 6, pp. 30–36, Mar. 1973.
[10] M. Kendall and A. Stuart, *The Advanced Theory of Statistics, Volume 3, Design and Analysis and Time Series*, 3rd ed. New York: Hafner, 1976.
[11] P.A.W. Lewis and G. S. Shedler, "Empirically derived micromodels for sequences of page exceptions," *IBM J. Res. Develop.*, vol. 17, pp. 86–100, Mar. 1973.
[12] —, "Statistical analysis of non-stationary series of events in a data base system," *IBM J. Res. Develop.*, vol. 20, pp. 465–482, Sept. 1976.
[13] R. Revelle, "An empirical study of file reference patterns," IBM Res. Rep. RJ 1557, Apr. 1975.
[14] A. J. Smith, "Characterizing the storage process and its effect on the update of main memory by write-through," *J. Ass. Comput. Mach.*, vol. 26, pp. 6–27, Jan. 1979.
[15] —, "A modified working set paging algorithm," *IEEE Trans. Comput.*, vol. C-25, pp. 907–914, Sept. 1976.
[16] —, "Sequential program prefetching in memory hierarchies," *Computer*, vol. 11, pp. 7–21, Dec. 1978.

[17] —, "A comparative study of set associative memory mapping algorithms and their use for cache and main memory," *IEEE Trans. Software Eng.*, vol. SE-4, pp. 121-130, Mar. 1978.
[18] —, "Bibliography on paging and related topics," *Oper. Syst. Rev.*, vol. 12, pp. 39-56, Oct. 1978.
[19] —, "Long term file migration algorithms," *Commun. Ass. Comput. Mach.*, 1981, to be published.
[20] E. P. Stritter, "File migration," Ph.D. dissertation, Stanford Univ. Comput. Sci. Rep. STAN-CS-77-594, Jan. 1977.

Alan Jay Smith (S'73-M'74) for a photograph and biography, see p. 146 of the January 1981 issue of this TRANSACTIONS.

Proofs of Networks of Processes

JAYADEV MISRA, MEMBER, IEEE, AND K. MANI CHANDY

Abstract—We present a proof method for networks of processes in which component processes communicate exclusively through messages. We show how to construct proofs of invariant properties which hold at all times during network computation, and terminal properties which hold upon termination of network computation, if network computation terminates. The proof method is based upon specifying a process by a pair of assertions, analogous to pre- and post-conditions in sequential program proving. The correctness of network specification is proven by applying inference rules to the specifications of component processes. Several examples are proved using this technique.

Index Terms—Communication networks, distributed systems, message passing systems, program proofs.

I. Introduction

We propose a proof technique for networks of processes in which component processes communicate exclusively through messages, as in Hoare [8]. The technique is based upon specification of a process h by a pair of assertions r and s, analogous to pre- and post-conditions in sequential program proving. The specification is denoted by $r|h|s$, which means: 1) s holds initially in h and 2) if r holds at all times prior to any message transmission of h, then s holds at all times prior to and immediately following that message transmission, where a message transmission of process h could be either h sending or h receiving a message.

The proof technique is built around a few inference rules. These rules allow us to deduce the specification of a network from the specifications of its component processes. The advantages of such a proof technique are the following.

1) A network specification is obtained solely from component process specifications and not from the details of process implementation.

Manuscript received May 23, 1979; revised August 18, 1980. This work was supported by the National Science Foundation under Grant MCS79-25383 and ARPA Grant Systems Performance Modeling Part II N00039-78-G-0080.

The authors are with the Department of Computer Sciences, University of Texas, Austin, TX 78712.

2) The proof technique supports the hierarchical decomposition of networks. Starting with R, S for network H, we construct r_i, s_i of component processes h_i's, such that the component process specifications yield the desired network specifications. The h_i's may in turn be networks themselves, in which case decomposition of h_i's into component processes proceeds hierarchically in the same manner.

We give several examples which demonstrate the power and convenience of using r, s to specify a process. Our inference rules are built upon Hoare's theory of *traces* [9].

II. A Model of a Network of Processes

We are not concerned with the definition of an entire programming language in this paper. We are concerned only with proving properties about message communication among processes. We consider the message communication mechanism proposed by Hoare [8]. The example programs will be written in Hoare's *CSP* with the following minor differences. *CSP* uses an explicit process addressing mechanism in message communication. For instance, process A may have commands of the form $B?x$ to receive a message from process B and put its content in local variable x; similarly $B!x$ denotes transmission of a message to process B, where the content of the message is the value of x. For autonomous proofs it is preferable to avoid explicit process naming. Hence, we will only allow process A to communicate using commands of the form $C?x$ or $C!x$, where C denotes a channel (see Section II-B). As Hoare has noted, addressing via channels is semantically equivalent to explicit process addressing.

We expect the reader to know *CSP* because our model is derived from it. We briefly summarize below concepts related to message transmission that we use in this paper.

A. Process

A process communicates only by sending or receiving *messages*. A process is either a *sequential process*, i.e., a sequential program with commands for message transmission, or a *network of processes*, as described next.

0098-5589/81/0700-0417$00.75