Performance Measurement and Analysis of Large Filestores

BY

D. Alan Jones

Ph.D. Thesis July 1978

University of Newcastle upon Tyne

ABSTRACT

Performance measurements of two large time-sharing computer systems are presented, with emphasis on their disk filestores. Similarities of process behaviour are found in the measured systems and another system reported in the literature. Individual processes make i/o requests in sequences, or bursts. Burst lengths have a mean of two with a large variance; within a burst, file i/o requests are spatially sequential in intent and are temporally related.

Characterizations of these behaviour patterns form the basis of a methodology for filestore evaluation and design. Descriptions of spatial and temporal load are abstracted from software traces without loss of any performance factor; these descriptions are inputs to a statistical model of the processes in the environment of the filestore. The filestore is represented by a simulation queuing model. The method specifies the inputs to the composite model and describes the calibration of outputs to match observable outputs. A model is built by this method, and validated for different loads.

The model is used for three evaluation experiments. Disk request scheduling is not statistically significant; filestore layout and disk capacity are highly significant; disks with fast-access areas are shown to improve performance by taking advantage of spatial accessing patterns. The limits of performance of a novel filestore equipped with a cache store are explored to determine guidelines for this new design. Modest improvements resulting from this design are shown to produce a considerable improvement in overall system performance.

Acknowledgements.

I would like to thank Dr James Eve and Dr Hugh C. Lauer for their guidance and support at different stages during this work. I am indebted to Dr Eve and Dr Douglas R. McGregor for their criticism and encouragement during the preparation of this thesis.

I have also received much support from friends and colleagues within and without the Computing Laboratory; to them also my thanks are due.

This work was supported in part by the Science Research Council and in part by the University of Newcastle upon Tyne.

To my Parents.

Contents

		Page
1.	Introduction and Overview	1
2.	Literature Review	
	2.1 Measurement	3
	2.2 Disk Request Schedulers and Single-Disk Models	6
	2.3 Other Disk Models	8
	2.4 Software Monitoring and Event Traces	9
	2.5 Statistical Investigation Methods	10
3.	System Measurement	
	3.1 Introduction	12
	3.2 Performance Measures	12
	3.3 Data Collection	13
	3.4 Analysis of Time-Event Traces	15
	3.5 Data Analysis of Filestore Spatial References	16
	3.6 Catalogue of Trace Data	17
4.	Analysis of Collected Data	
	4.1 The Main Results	18
	4.2 I/O Bursts	19
	4.2.1 Existence of i/o bursts	19
	4.2.2 Delimiting the length of file i/o bursts	22
	4.2.3 Characteristic length of i/o bursts	23
	4.2.4 Run-time characteristics of i/o bursts	23
	4.3 Effects Limiting Work Throughput	28

4.4 Filestore Accessing Behaviour 36

4.4.1 Sequential accessing and file-i/o bursts 36

4.4.2 S/360 distribution of seek movements 41

4.5 Summary and Conclusions 44

5. A New Method of Filestore Analysis

5.1 Introduction 45

5.2 Performance Indices 45

5.3 Load Representation 46

5.3.1 Introduction 46

5.3.2 The model of i/o bursts 46

5.3.3 The model of spatial load 48

5.3.4 Run-time and Waiting time generation 54

5.3.5 Summary of inputs representing load 55

5.4 The Filestore Simulation Queuing Model 57

5.5 Validation 59

5.6 Summary 63

6. The Effects of Disk Request Scheduling and Disk Capacity on System and Filestore Performance.

6.1 Introduction 64

6.2 Disk Request Scheduling Algorithms 64

6.3 Disk Configuration 67

6.4 Experimental Plan 67

6.5 Conclusions 69

6.6 Results - Disk Scheduling algorithms 70

6.6.1 Work throughput 70

6.6.2 Mean Response Time 70

6.7 Results - Number of Disks 77

6.7.1 Work Throughput 77

6.7.2 Mean response time 77

7. Ranking of Filestore Factors for their Effect on Performance

7.1 Introduction 82

7.2 The Factors 83

7.2.1 Filestore Layout 83

7.2.2 Capacity per disk-access 84

7.2.3 Capacity per disk 85

7.2.4 Scheduling algorithm 85

7.2.5 Load variation 86

7.3 Experimental plan 87

7.4 Conclusions 89

7.5 Results - Commentary on ANOVA tables 89

8. Investigation of Two Policies for using Disk Fast-Access Areas

8.1 Introduction 97

8.2 Experimental Factors for the Fast-Access Area 98

8.2.1 The Simple Placement Policy 98

8.2.2 The Staging Policy 99

8.2.3 Size of the fast-access area 99

8.3 Other Experimental Factors 101

8.4 Experimental Plan 101

8.5 Conclusions 103

8.6 Results 106

8.6.1 Commentary on ANOVA tables 106

8.6.2 Commentary on results of Chapters 7 and 8 111

9. Potential Performance from a Novel Filestore-Controller

9.1 Introduction 112

9.2 Performance Improvement Factor 112

9.3 Experimental Plan 113

9.4 Conclusions 114

9.5 Results 115

10. Summary and Conclusions 119

Appendix A Measurements not included in the text 121

Appendix B A note on the method of statistical analysis used in Chapters 7 and 8 134

Appendix C Disk Request Scheduling Algorithms 137

Appendix D Measurement overhead costs 141

Bibliography 142

CHAPTER 1

Introduction and Overview

In a computer system the filestore is the subsystem of storage devices together with an operating strategy which deals with the storage, access and preservation of data. A filestore is characterized by its storage devices, their specifications and configuration, and an organization of files. It operates in an environment whose interface with the filestore is a sequence of access requests distributed over time and space. To the environment, the filestore is a service device whose performance can critically affect the work rate of the overall system.

This thesis presents performance measurements of real systems and a methodology for analysis and design of filestores. The measurements show certain behaviour patterns that seem to be common to several different computer systems. These observations have been incorporated into a model of the environment that reflects both temporal and spatial accessing characteristics of the individual processes that form the environment. The method of analysis combines this statistical model of the environment and simulation queuing model of the filestore. The method assumes the existence of moving-head disk devices, and would normally be used to design filestores that use such devices. However, we give an illustration of the use of this method to study the performance of a novel backing store where the assumption of disk devices is not essential. The model has been validated for different filestores. The approach taken differs from previous work principally in two aspects; it uses a more realistic characterization of load which is very different from that assumed by previous simulation studies, and it seeks to quantify the effect of filestore performance factors on overall system performance.

The thesis is laid out as follows. Chapter 2 reviews the literature on analyses of filestore and system performance. Software monitoring and methods of data analysis used for this work are given in Chapter 3. The main results of performance measurement are presented in Chapter 4. Chapter 5 describes the method of representing load and the validation of a simulation model using this representation. Chapters 6, 7 and 8 present analyses of disk filestore factors affecting design and performance. Chapter 9 presents a use of the model to evaluate a novel filestore. Finally, Chapter 10 reviews the methodology and conclusions of this research.

CHAPTER 2

Literature Review

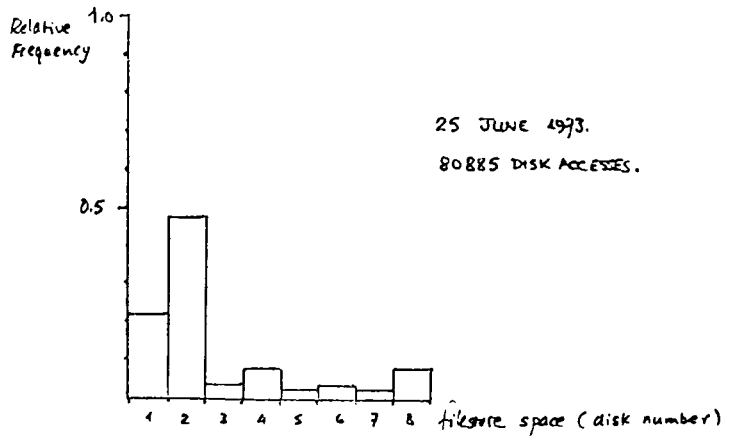
This chapter reviews the existing literature on disk filestore measurement and performance analyses.

2.1 Measurement

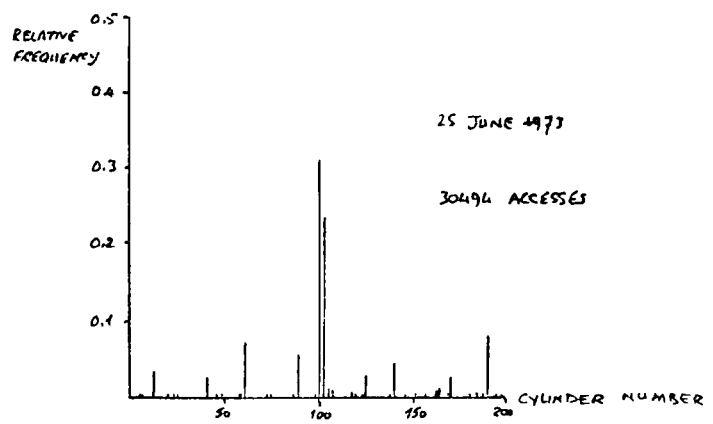
Lynch [29] and Barraclough [3] measured disk i/o activity on the Michigan Terminal System (MTS) running on the N.U.M.A.C.* IBM 360/67 computer with a filestore of eight IBM2314 disks. Their principal result is that only one-third of i/o requests cause disk arm movement (i.e. seek movement). This observation is repeated by Gray [17] and independently observed by Jalics and Lynch [26] from the TOPS-10 system on a PDP-10 computer with a filestore of eight DEC RP02 disks. For both of those systems the mean disk seek time, when there was a seek, was about 40ms which corresponds to a movement of approximately 10 cylinders (out of 200). This mean seek time had a large variance. Access rates were about 2 per second per disk.

Using a software monitoring technique, Gray measured and analyzed the disk behaviour of processes. He concentrated on spatial distribution of accesses and reported that most accesses were to the catalogue cylinders of the index-sequential style of file organization then current for MTS. Figure 2.1 shows results from Gray' work on spatial accessing. Figure 2.2 is a representation of Gray's results in a form to compare with other functions that have been used to represent spatial accessing [16,53]. Gray also found

* Northumbrian Universities Multiple Access Computer



ACCESSES PER DISK OVER AN 8-DISK FILESTORE



ACCESSES PER CYLINDER

Figure 2.1 Spatial access measurements (Gray [17]).

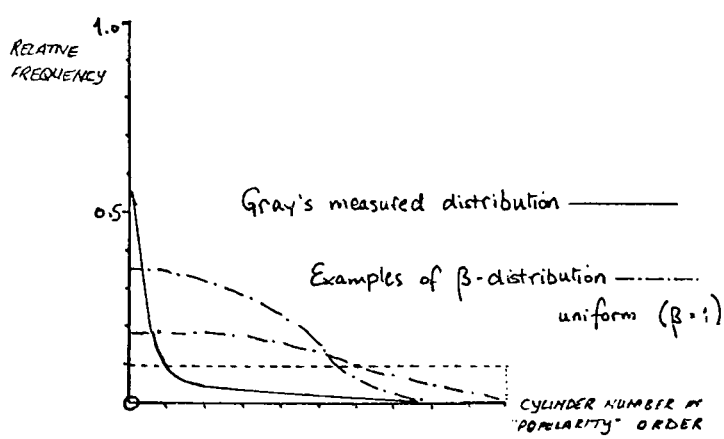


Figure 2.2 Comparison of some spatial distributions.

that a process might make sequences of i/o requests in quick succession to the same cylinder. He reported the mean sequence length as two with a large variance but made no temporal analyses. Lynch, in his earlier work using hardware monitoring [29] observed that most of the "same cylinder" requests occurred within 10ms of the previous request completion.

Jalics and Lynch [26] also used software monitoring to make a detailed study of process behaviour. In the TOPS-10 system, three of their results are of interest here because they may be compared with the length of the sequences reported by Gray;

(i) same cylinder by process - mean 3, S.D. 6, 90% less than 4
(ii) same file by process - mean 9, S.D.40, 90% less than 13
(iii) same cylinder by disk - mean 4, S.D.8, 90% less than 7.

When observing the intervals from end of terminal i/o to start of next terminal i/o, only 10% of TOPS-10 processes had disk i/o activity, and those processes had a mean of 83 disk references. "Runtime" was defined to be the time between the exit of a process from one i/o wait queue until its entry to its next i/o wait queue [25]. Jalics and Lynch reported that 55% of runtimes* between disk i/o references were less than 5ms.

These studies indicate that disk spatial references in two time-sharing systems are highly sequential. Where the organization of data does not interrupt sequentiality of reference , few seek movements are made. There is also an indication that references are correlated in time.

*The term runtime is used in this sense throughout the thesis. See in particular section 3.6

In addition to these measurement studies, the importance of sequentiality of spatial reference has been recognized by Maruyama and Smith [34] who investigated the fragmentation that occurs in a sequentially formatted filestore when files are edited or expanded. With the use of a cost function describing the excess costs due to physical disorganization, they show that the optimum reorganization point depends on sequentiality of reference and sometimes the optimum strategy is not to reorganize.

2.2 Disk Request Scheduler and Single-Disk Models

Previous studies of disk request schedulers have made assumptions about work-load that are at variance with the measurements of Lynch, Jalics and Gray. Consequently, their results do not necessarily form a basis for sound engineering in the design of filestores.

Teorey and Pinkerton [53] used an infinite-source, uniformly-accessed single disk model with exponential arrival assumptions to investigate request schedulers. In addition to the SCAN scheduler, in which the disk arm sweeps back and forward across the disk surface, Teorey introduced C-SCAN (or Circular-SCAN) in which the arm always scans in the same direction, returning to its starting extreme at the end of a scan. Their conclusion, that a combination of SCAN and C-SCAN is most efficient in terms of respnse time, does not apply to distributions of the type reported by Lynch, Jalics and Gray. Teorey [52]varied the input arrival pattern, considering batched as well as sparse arrivals as extremes on either side of Poisson arrivals. He also considered the effect of concentrating accesses on central cylinders of both single and multiple disk systems. Figure 2.2 compares the uniform pattern with the β-distributions used by Gotlieb and MacEwan (q.v.) and actual measurements made by Gray.

Gray showed that, for his address traces, the First-Come-First-Served (FCFS) scheduling algorithm was capable of servicing access rates reckoned to be in the SCAN-only region (12-20 per second for IBM 2314-style disks) by Teorey and Pinkerton. Measured filestore access rates of about 2 per second per disk compare with queuing theory and simulation investigations [1, 16, 53, 56]where access rates exceeding 10 per second per disk are considered. Scanning algorithm improvements are not apparent except under high rates of access.

Wilhelm [55] studied disk request scheduling with an infinite-source, single-disk analytical model with exponential request-inter-arrival times, but with a spatial distribution following Lynch's observation that two-thirds of requests have zero seek-time. He found that the First-Come-First-Served policy had shorter or as good response time than the Shortest-Seek-Time-First policy, and claimed this result to be anomalous. However, this result is an inevitable consequence of his model of load where disk arms rarely move. Gray had already noted that in such situations seek optimizing policies have little to optimize.

The factors of temporal and spatial accessing patterns and file organization have been recognized as relevant to filestore performance. If filestore load is simplified to exclude known patterns and effects, ensuing predictions are misleading and unsafe.

2.3 Other Disk Models

Previous studies of multiple-disk filestores have used unrealistic descriptions of load and do not contribute to filestore design in the engineering sense. Omahen [40] has reviewed queuing models for estimating response time of multiple disk filestores. He offers no recommendation, simply remarking that he expects finite-source models to be more realistic than infinite source models.

Gotlieb and MacEwan [16, 31] used the β-distribution with various parameter values to represent a non-uniform spatial access distribution. This function is compared with Gray's observed distribution in Figure 2.2. Temporal load was simply represented by a mean access rate with exponential arrival assumptions. The effect of varying the number of disks was investigated, but the mean access rate to each disk was varied over the same range for each configuration. This results in a higher load for systems with more disks, an unrealistic treatment emphasizing the "stand-alone" nature of their model. Wilhelm's general queuing model of multi-disk systems [56] repeats MacEwan's assumption of load that increases with the number of disks in the filestore.

Validation of these queuing theory models consists of building a simulation model that reflects the detail of disk operation but embeds it in the same environment as the queueing model. The load presented to the filestore remains idealized. This validation process does not build confidence in subsequent predictions of performance in real situations.

Stone and Turner [51] claim realism from their feedback-inhibited model for a system and its filestore. Validation measurements are made on a real system, but this exercise is weakened by the use of a load controlled to achieve the assumptions of their model.

Network queuing models have been reviewed by Buzen [10] who demonstrates their accuracy of prediction for indices such as CPU and device utilizations. Moore [37] has developed a network model of MTS from measurements taken at the University of Michigan. For tractable computation and system comprehension, network models are limited to a small number of user classes and resources. Diversity of user behaviour and variety of filestore configurations are difficult to model.

2.4 Software Monitoring and Event Traces

Event tracing has been developed as a measuring tool that offers much detail and insight into the operation of complex systems despite difficulties of interpretation and restricted relevance.

Pinkerton [42] implemented event-reporting code in various parts of an early version of MTS, and arranged for a collection facility to gather data on user-selected events within the system, e.g. page faulting and i/o requests. The motivation for this trace was to observe process behaviour. Gray [17] and Moore [37] used this facility to provide workload descriptions for their models.

Jalics [25] instrumented the TOPS-10 system and made many observations of user behaviour including filestore spatial and temporal accessing patterns. The measurements are at a level of detail similar to the even trace used by Parapudi and Winograd [41] who made less detailed measurements of the VMOS system on an RCA Series 70 computer.

The principles of trace-driven modelling (TDM) were presented by Cheng [12]. In TDM, an event trace is the input to a modelled variant of the real system. Validation is straightforward, in that outputs from the model can be

directly compared with real outputs. Predictions are accurate though of restricted relevance because the workload, and frequently the modelled system, is precisely specified without abstractions. New trace data, simulating changes, may be manufactured from an actual trace. Baskett, Brown and Sherman [6, 50] describe a tracing system for a CDC-6600 computer with the UT-2 operating system. Examples of collected events are operating systems calls and scheduler decisions; it is a less detailed trace than those of Pinkerton and Jalics.

Another use of event traces is to construct a more general, synthetic load. Schwetman and Brown [48] use the UT-2 trace to provide a high-level description of workload for statistically designed experiments, but with variation of only one factor. Beilner and Waldbaum [7] and Schatzoff [45] use trace data in the specification of workload for deterministic models.

Despite the disadvantages of system dependency, event traces are extremely useful. In the absence of physical laws governing how computer systems are used, measurement by event monitoring is a primary engineering tool for investigation of behaviour.

2.5 Statistical Investigation Methods

Techniques for statistical experiment on real systems have been presented by various authors [2, 33, 39, 44, 46, 54]. These methods are designs and analyses permitting variation of several chosen factors in a complex system. Classical, single-factor experiments are inadequate when the presence and possible interaction of several factors is suspected. Bard [4] quantitatively analyzed overhead in a prototype software system. He used a regression

model with weights (coefficients of the model equation) allocated to the system structures (factors) reckoned as contributing to the overhead. In [5], Bard extends this analytical method to the more comprehensive fitting of a statistical response surface for the components of the regression model. Clapson [11] has quantified resource usage by use of regression models and has established common patterns of resource behaviour in different operating systems for similar IBM computers. His analyses are of resource usage, not program behaviour; they are restricted to computers of similar architecture. Anderson and Sargent [2] performed a full factorial experiment (every factor at each of its levels observed with each level of all other factors). The resulting data were fitted by a linear regression model. They found that service time demands were very variable, and no common distribution was able to fit the data. The data were eventually fitted by composite Weibull distributions.

Improved instrumentation and increased statistical awareness are leading to new computer performance measures and methods of experiment based on what happens in real systems.

CHAPTER 3

System Measurement

3.1 Introduction

Measurements were made to provide an accurate description of what happens in a large time-sharing system. The performance of the filestore is of special interest but the load under which it operates is also of prime importance. This chapter describes how the measurements were made and how the results of interest were derived from the collected data.

The Michigan Terminal System has an event reporting and collection facility that is part of the operating system. This facility was used to monitor the temporal behaviour of the batch and terminal processes that comprise the normal workload of MTS. In addition to new data about the use of MTS on the N.U.M.A.C. IBM 370/168 computer system, Gray's [17] measurements of an earlier version of MTS on an IBM 360/67 system were available. Gray used the earlier data to study spatial accessing patterns. We have made a new analysis of these data.

3.2 Performance Measures

Filestore performance is measured by the response time for each file i/o request. For each process, the filestore response time is defined as the time interval between reports of the start and end of a file i/o wait for that process.

There are insufficient numbers of process births and deaths on the event traces to measure system performance in terms of job throughput. Instead the interaction rate between the users and the system is used. An interaction is defined as the time between completion of user input from a terminal and completion of subsequent system output to the terminal. The measured systems have batch as well as terminal users, so the start and finish of a batch job is interpreted as the completion and start of a terminal wait, i.e. one batch job completion is one interaction. We call such an interaction a work-unit. This work-unit represents an unquantified work step for one process, whether batch or terminal. The amount of work performed is not considered here. This measure is empirical and depends on the applied load and its utilization of system resources. The use of the work-unit in this thesis is an index of "work done" when presenting performance measurements, and for comparative evaluation of a simulation results. The simulations are of different filestores, but the same set of work units is used whenever comparisons are made.

3.3 Data Collection

The software trace tool on MTS is the Data Collection Facility (DCF) [27, 43]. It provides a means of gathering detailed low-level data generated by processes within the system. At about 25 locations in the system, pertinent data is recorded by the Supervisor. The event is labelled with an identifying code, the time at which it was recorded and the job number of the process that caused it. These data are buffered and transmitted to the user process that initiated DCF data gathering. The supplied data may be accumulated or analyzed on-line. Individual events are not lost, since their collection is part of the operating system. Only continuous sequences of data were used for analysis; no data has been lost.

For the work of this thesis, data accumulation only was used with subsequent analysis. In addition to a small number of obligatory and reference events, two types of events were collected.

Event A - when for any reason, a process enters a wait state (compared with ready-or-running state). The reason for the wait is given with the collected event.

Event B - when a process stops waiting.

With these two events, we were able to measure the run-time and i/o-wait-time for each process.

The version of the Data Collection Facility used by Gray was the same as the one reported by Pinkerton [43]. He evaluated the interference and overhead of collection to be 0.3% of CPU time and about 4% of main storage for buffers. The new collection facility used for this work (King [27]) was evaluated; interference and overhead was about 1% of CPU time, 4% of storage for buffers and 0.4% of channel time. Appendix D presents a description of this evaluation. This interference and overhead is very small. The software monitors used by Parapudi and Winograd [41] and Jalics and Lynch [26] were not assessed for system interference. Schwetman [4] assessed the CPU utilization of a data collection program at 13%, an almost intolerably high figure for useful deductions from the data. Screenivasan [49] gives a figure of 5% CPU overhead for his measurement study.

3.4 Data Analysis of Time-Event Traces

From Events A and B above, each process is classified into three temporal categories:

(i) not waiting

(ii) waiting but not for explicit i/o completion

(iii) waiting for explicit i/o completion.

In category (ii) are page waits*, waits for system resources and other operating system overhead. Categories (i) and (ii), when combined, form a measure of the run-time spent in a compute step. This run-time is measured for each process.

Category (iii) is divided into three types of i/o wait:

a) file i/o

b) terminal i/o

c) other i/o

File i/o is considered separately because it is our special interest. Terminal i/o waits are of interest because they delimit work units. All other waits for i/o completion are grouped into type (c) which includes i/o with non-filestore disks, magnetic tapes and slow devices such as printers and readers.

Each process is thus in one of four states at various points in its life -

*In MTS, paging and filestore traffic are independent on separate devices.

1. running [See definition of runtime on p5.]
2. in a file-i/o wait
3. in a terminal-i/o wait
4. in an other-i/o wait

3.5 Data Analysis of Filestore Spatial References

The data collected and presented by Gray gives spatial usage patterns of the filestore of MTS as it was then configured at M.U.M.A.C. Its main characteristics were:

1. Eight IBM 2314 disks (30Mbytes each)
2. Single channel linking filestore to main store
3. Sequential layout of files, with a main disk catalogue on one disk and secondary catalogues on each other disk. This organization is similar to the Indexed-Sequential Organization [24].

When measurements were made on the IBM 370/168 it was not practicable to extend the Data Collection Facility to report on filestore addresses as Gray had done. The main characteristics of the filestore on this machine at the time of measurement were:

1. Four IBM 3330,11 disks (200 Mbyte each) with Rotational Position Sensing (RPS).
2. Single channel linking filestore to main store.
3. a "randomized" layout of file in "file page frames".
   The disk space is divided into "frames" or sectors, 58 per cylinder, allocated sequentially within cylinders but not necessarily sequentially across cylinder boundaries. Brinch Hansen describes such an organization in [8] and names it the Non-Consecutive Layout.

It attempts to locate related data together while avoiding the fragmentation caused by editing and other changes. The main disk catalogue is spread over all the disks in the filestore, with a master index on one disk.

3.6 Catalogue of Trace Data

Gray's measurements, the S/360 data, were made in June 1973, and comprise two runs of 45-minutes duration (S/360,1 and S/360,2). His only use of this data was as preparation for his subsequent trace of disk accesses. The measurements made in the course of the present research, the S/370 data, were collected from December 1975 to June 1976. The traces used for presentation in this thesis are:

a) S/370,1 , 10 minutes duration, 28 June 1976

b) S/370,2 , 20 minutes duration, 29 June 1976

c) S/370,3 , 30 minutes duration, 30 June 1976.

These traces were taken during the afternoon peak loading.
Gray's traces have over 350,000 events each and the S/370 data have about 100,000 events per 10 minutes.

CHAPTER 4

Analysis of Collected Data

4.1 The Main Results

In both measured systems, processes characteristically do their i/o in bursts. In a burst, most run-times between i/o requests are short. For file-i/o, more than half the run-times between requests are less than 10ms. Indirect evidence of i/o in bursts has also been found by Jalics as discussed in Chapter 2. We investigate the properties of these bursts, their lengths and characteristic run-time distributions. This information is useful when constructing realistic models of system.

We also show that sequentiality of spatial reference exists within bursts of file-i/o. This effect complements the popularity of central catalogue areas to explain why disk arms rarely move.

4.2 I/O Bursts

4.2.1 Existence of i/o bursts

This section presents data on file-i/o bursts. Figure 4.1 shows the distribution of all run-times between requests for file i/o. For both S/360 and S/370, more than half of these runtimes are less than 10ms. Figure 4.2 shows the period 0-10 ms in detail. Both distributions peak in this region with no carry-over.

Table 4.1 shows the frequency with which a given type of i/o request is followed by any other type. A process completing an i/o wait has a greater than 90% chance of starting another wait of the same type. Jalics has reported results similar to Table 4.1 that strongly suggest the existence of bursts of i/o in the TOPS-10 system. Although he did not make any direct observation of the i/o burst effect, Jalics reported that 64% of run-times between i/o requests are less than 8ms (55% less than 5ms). His method of time measurement by logarithmic hexadecades obscured any peak in short run-time between requests. These measurements also indicate the existence of bursts of i/o in TOPS-10.

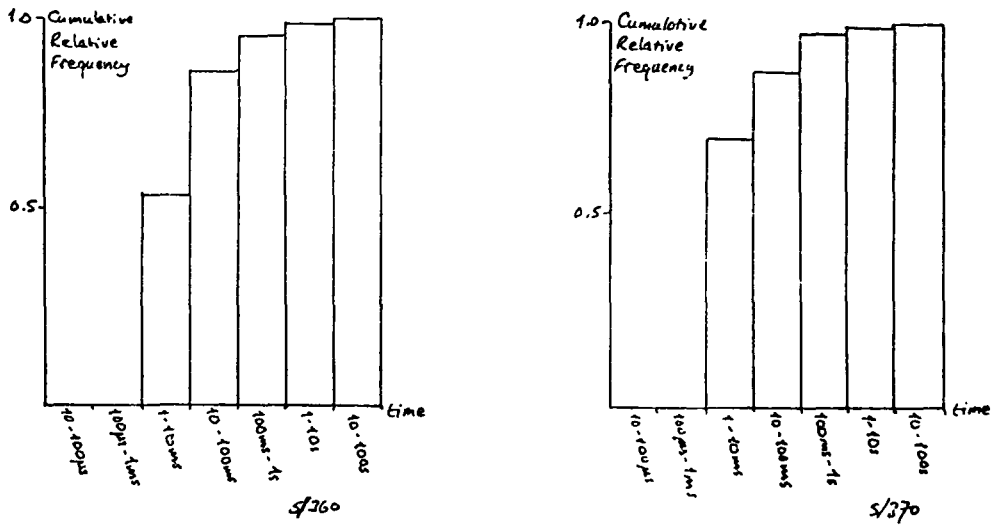


Figure 4.1 Distribution of all run-times between file-i/o requests.

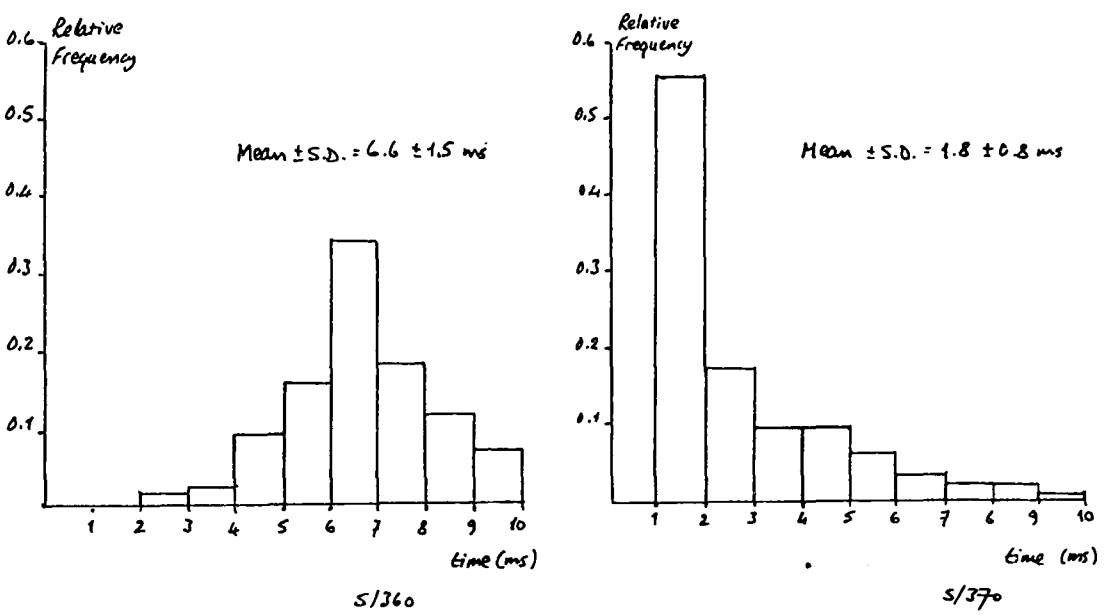


Figure 4.2 Distribution of all run-times between file-i/o requests.
(Detail 0-10ms)

from \ to	file i/o	terminal i/o	other i/o
file i/o	0.978	0.018	0.004
terminal i/o	0.042	0.886	0.072
other i/o	0.003	0.024	0.966

S/360

from \ to	file i/o	terminal i/o	other i/o
file i/o	0.965	0.016	0.019
terminal i/o	0.042	0.939	0.019
other i/o	0.032	0.017	0.951

S/370

Table 4.1 Relative Frequency of next i/o from previous i/o

4.2.2 Delimiting the lengths of file-i/o bursts

For S/360 , Lynch had already observed a high incidence of requests to a cylinder within 10ms of completion of the previous request. Gray has reported that over 50% of accesses are to the central cylinders. Some of Lynch's "repeat" requests may be due to cylinder popularity, but the frequency with which a file-i/o request is followed by another of the same (Table 4.1) and the distribution of all run-times between file requests (Figure 4.2) indicate that sequences of these "repeat" requests are generated by the same process. We wish to investigate this effect further. We shall call the maximum run-time between file-i/o requests-in-a-burst the Lynch Period, and use it to delimit the lengths of file-i/o bursts. If a process makes a file-i/o request later than the Lynch Period since completion of its previous file-i/o request (e.g. because of a disk-busy delay or a long run time), we consider that a new burst has begun. From Figure 4.2, we have chosen the Lynch Period for S/360 to be 10ms, and 4ms for S/370.

4.2.3 Characteristic length of I/O bursts

With the Lynch Period set to delimit file-i/o bursts, the lengths of these bursts are distributed hyper-exponentially, as shown in Figure 4.3. Both systems show the same distribution. When these data are presented as cumulative log plots, as in Figure 4.4, the distribution falls into two straight lines. The change of slope occurs for long bursts of 8 and over, with a mean of 2 for both systems. Simple linear regression is used to fit these data. Despite the shorter Lynch Period for S/370, there has been no significant change in this user habit between measurements separated by three years, a new version of the operating system and a new file organization.

The distributions of burst lengths for terminal-i/o and other-i/o are shown in Appendix A. All length distributions are very similar for all types of i/o burst. Terminal-i/o and other-i/o bursts are delimited by a transition of the process from that type of wait sequence to another.

4.2.4 Run-time characteristics of i/o bursts

The run-times characterized for i/o bursts are those between bursts and those between requests within a burst.

Figure 4.5 and 4.6 show the mean run-time between bursts for all i/o types as a function of processor load. Processor load is the number of processes in the running state (i.e. not in any wait state). The system load is defined as the total number of processes in all states. Only those data points with a frequency of at least 1% are shown. The S/360 plots of Figure 4.5 show a general tendency for this mean run-time to increase with processor load. The S/370 plots of Figure 4.6 show mean run-times

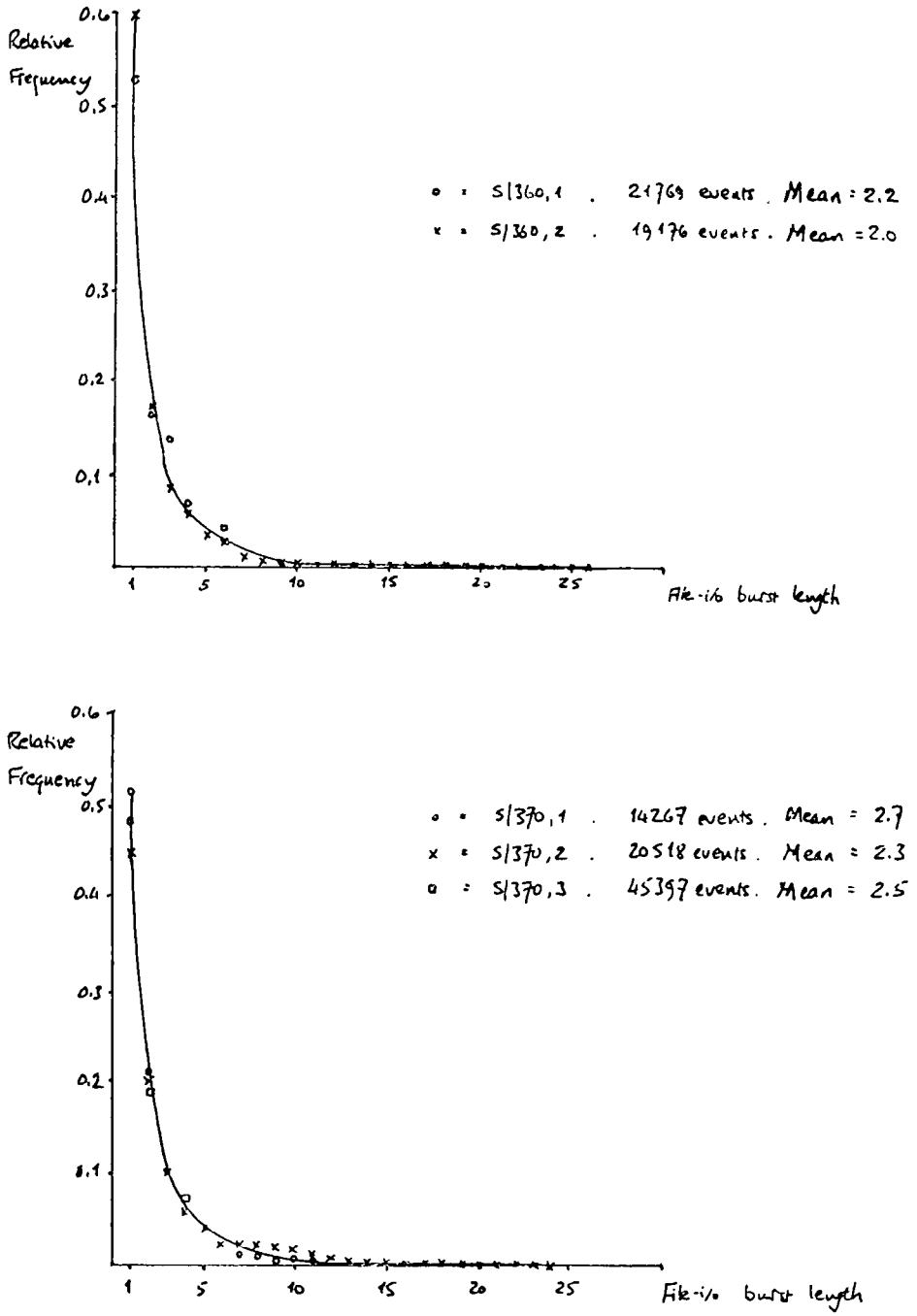
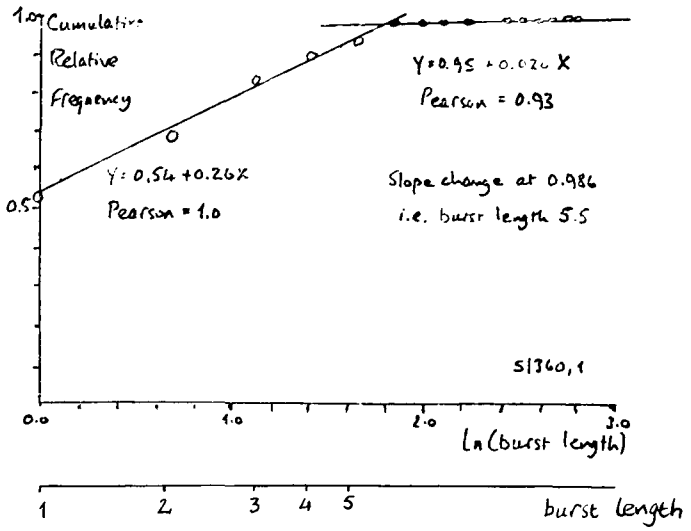
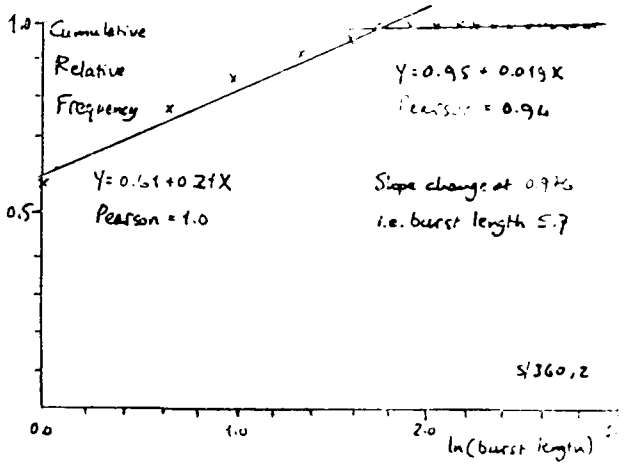
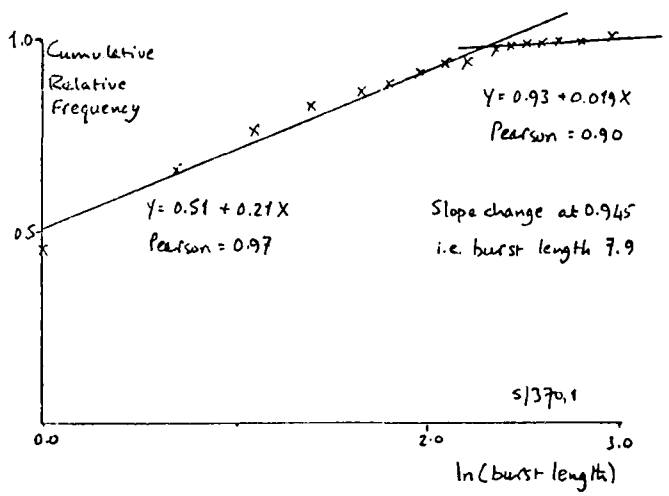
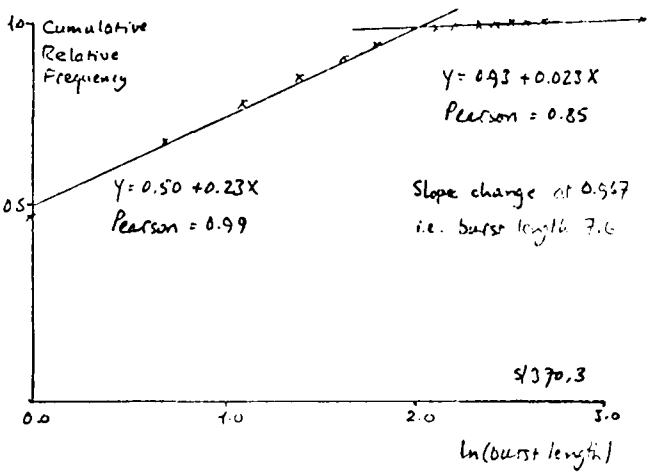


Figure 4.3 Distribution of burst length for file-i/o.









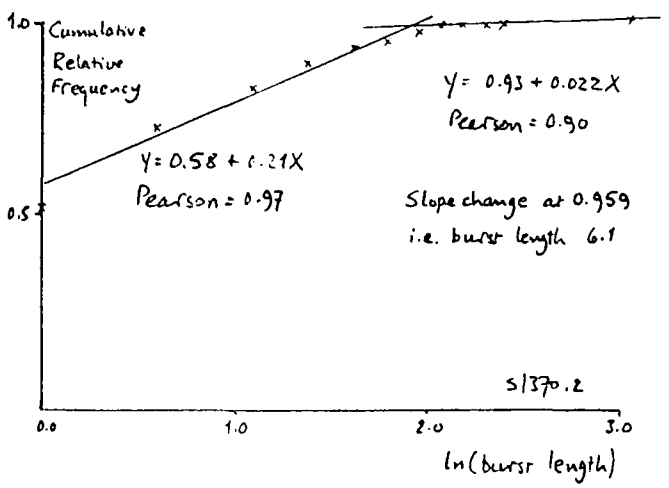


Figure 4.4 Distribution of Ln (burst length) for file-i/o.

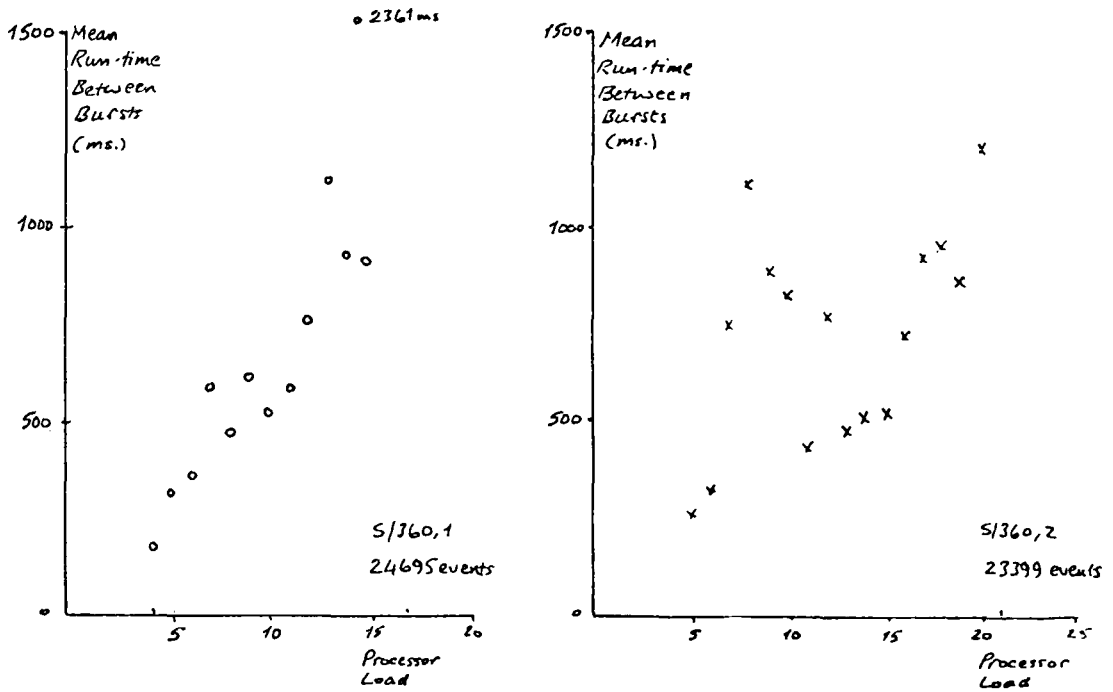


Figure 4.5 Mean run-time between bursts vs. processor load -S/360.

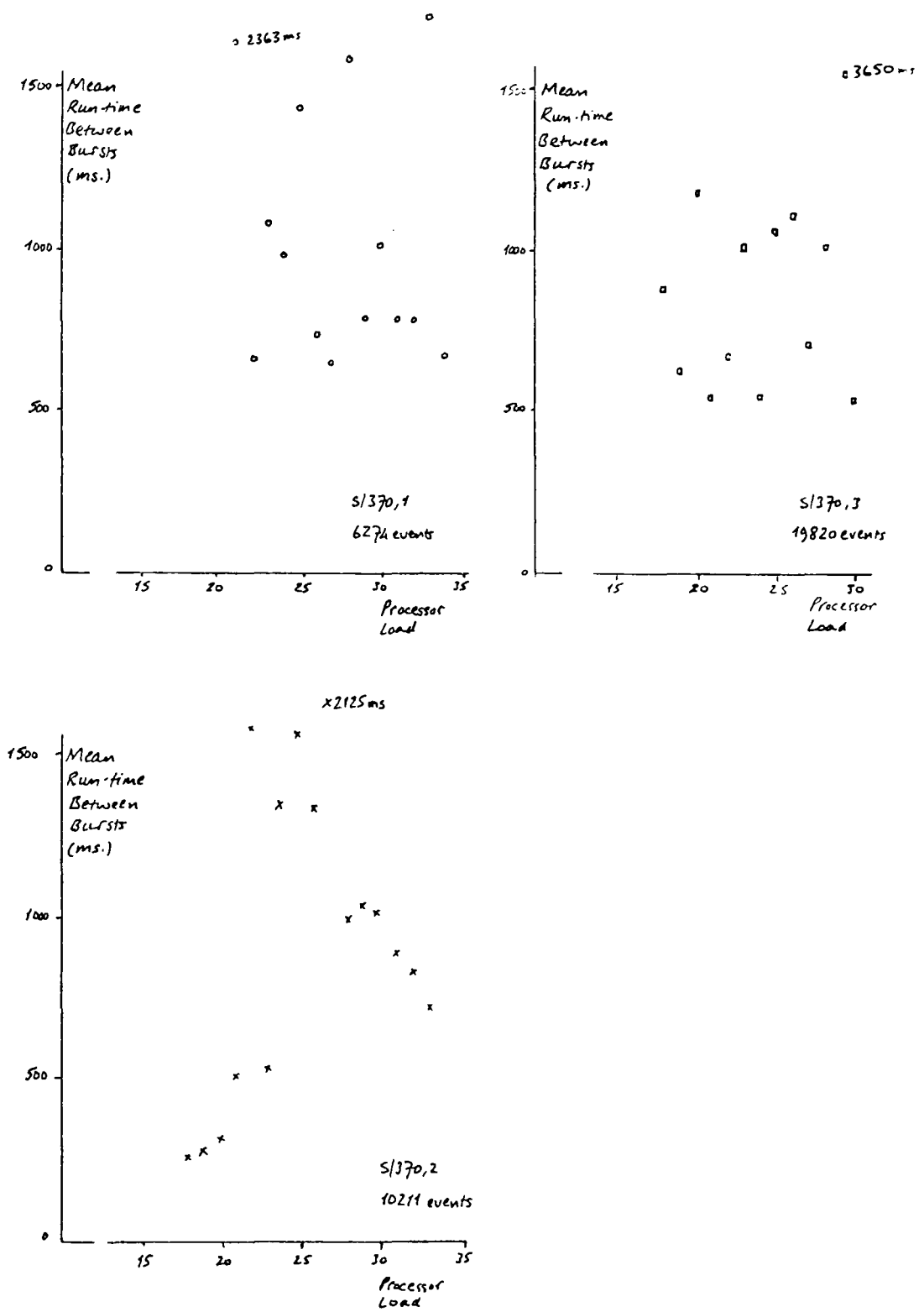


Figure 4.6 Mean run-time between bursts vs. processor load - S/370.

apparenty unrelated to load. The standard deviations of these mean run-times are very large at approximately 10^7 to 10^9 ms. Run-time between bursts is exponentially distributed with a long tail as shown in the histograms of log (run-time between bursts) in Figure 4,7.

Figure 4.8 shows the mean run-time between requests-in-a-burst for file-i/o. Simple linear regression has been used to fit these data. The S/360 plot has a positive slope indicating a CPU-contention feedback effect. However, the Pearson correlation coeffecient is significantly less than 1.0, suggesting that this effect is not a reliable indicator of how run-time varies with processor load. An explanation of the negative slope on two S/370 plots is that the S/370 was limited by a resource other than CPU. Only one trace (S/370,1) was made when the processor was sufficiently occupied to show negative feedback on run-times. From Figure 4.2 these run-times have a S/360 mean of 6.6ms (SD 1.5ms) and a S/370 mean of 1.8ms (S.D. 0.8ms).

Appendix A contains plots of all run-times between requests-in-a-burst against processor load for terminal-i/o and other-i/o bursts.

4.3 Effects Limiting Work Throughput

Figure 4.9 is a histogram of processor load measured at every wait-start and wait-finish event. At any given moment most processes are in an i/o wait; the mean processor load is 11 (out of maximum 60) for S/360, and 25 (from maximum 90) for S/370. Table 4.2 shows the relative number of each state (running, file-i/o, terminal-i/o and other-i/o). Because a running state precedes each i/o wait, there are as many running states as waiting states. Numerically, file-i/o is the most frequent wait state, but the mean terminal-i/o wait is about 3500ms (see Appendix A) so we cannot single out a particular type of wait as a limit on throughput. Generally, waiting

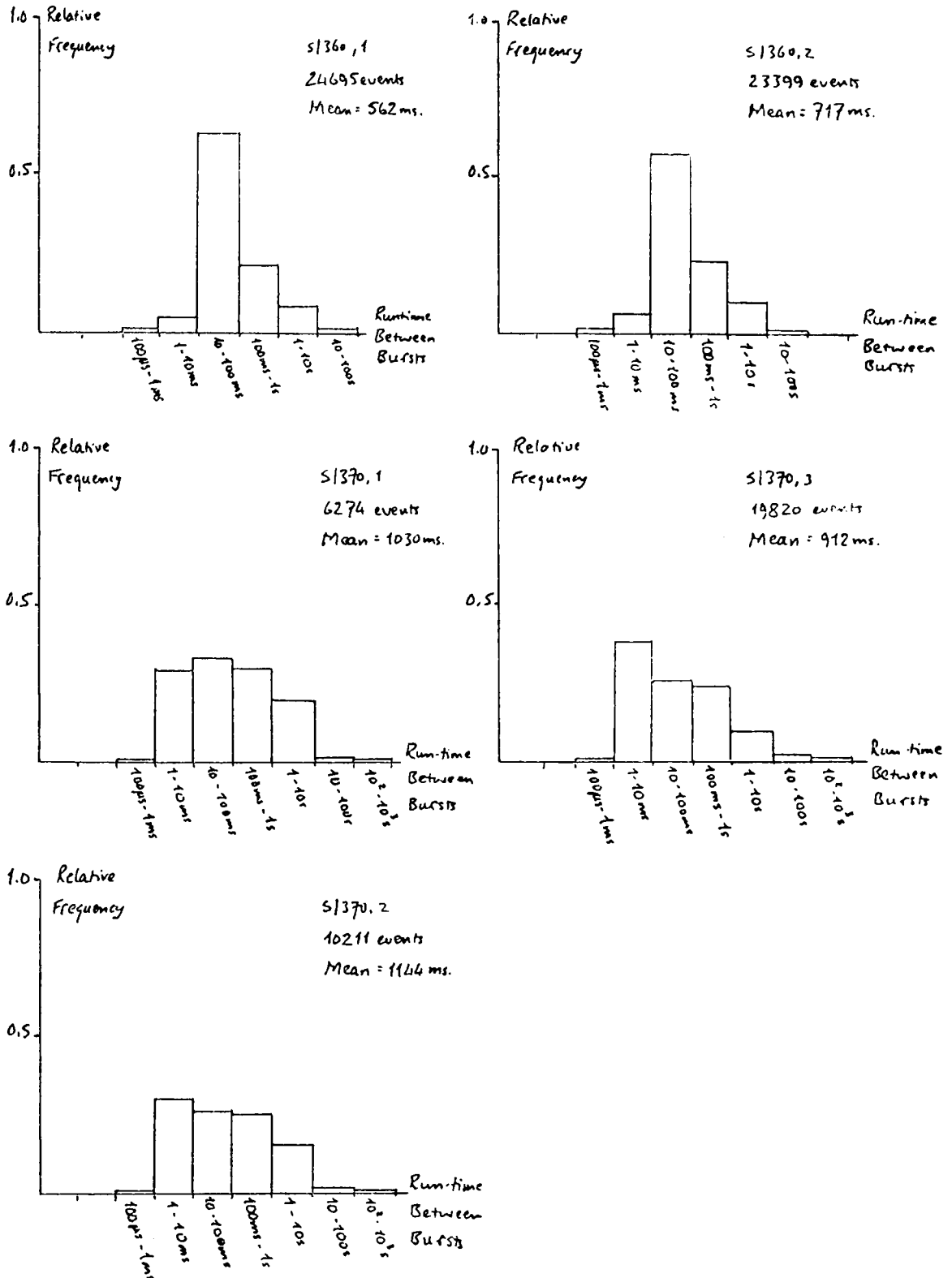


Figure 4.7 Distribution of run-time between bursts.

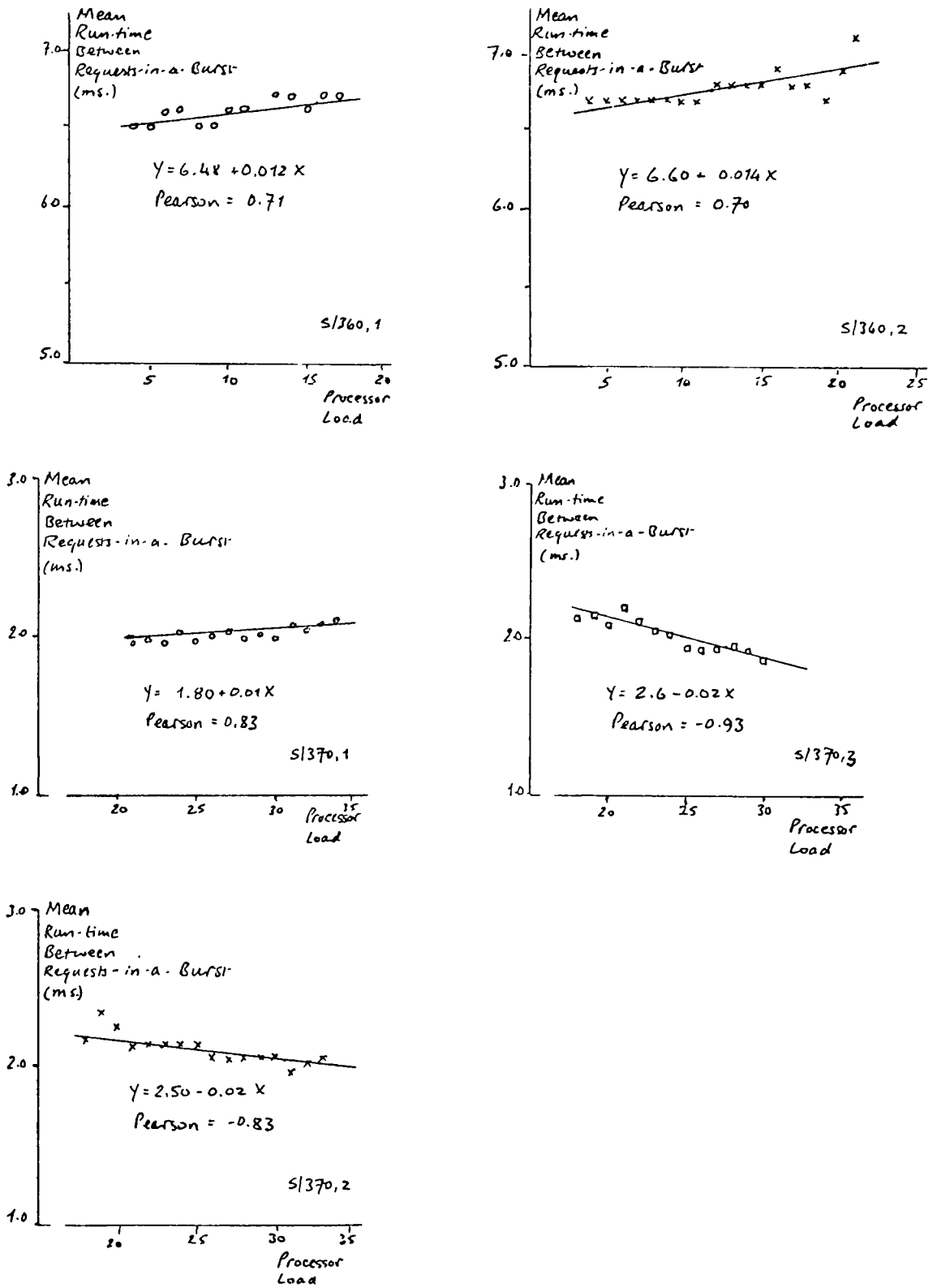
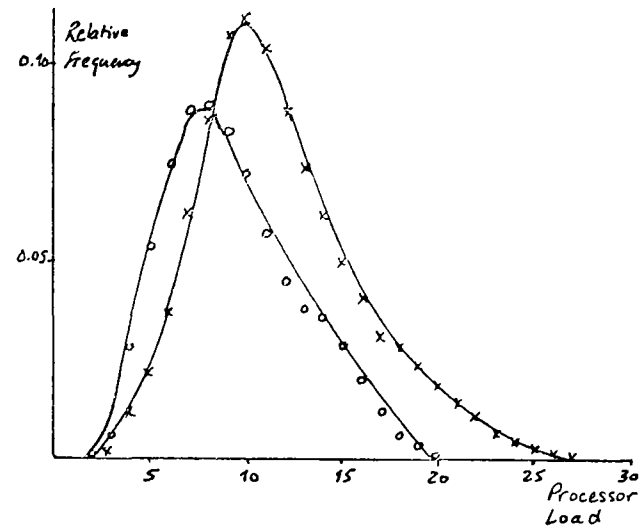


Figure 4.8 Distribution of mean run-time between requests-in-a-burst.
(for file-i/o)



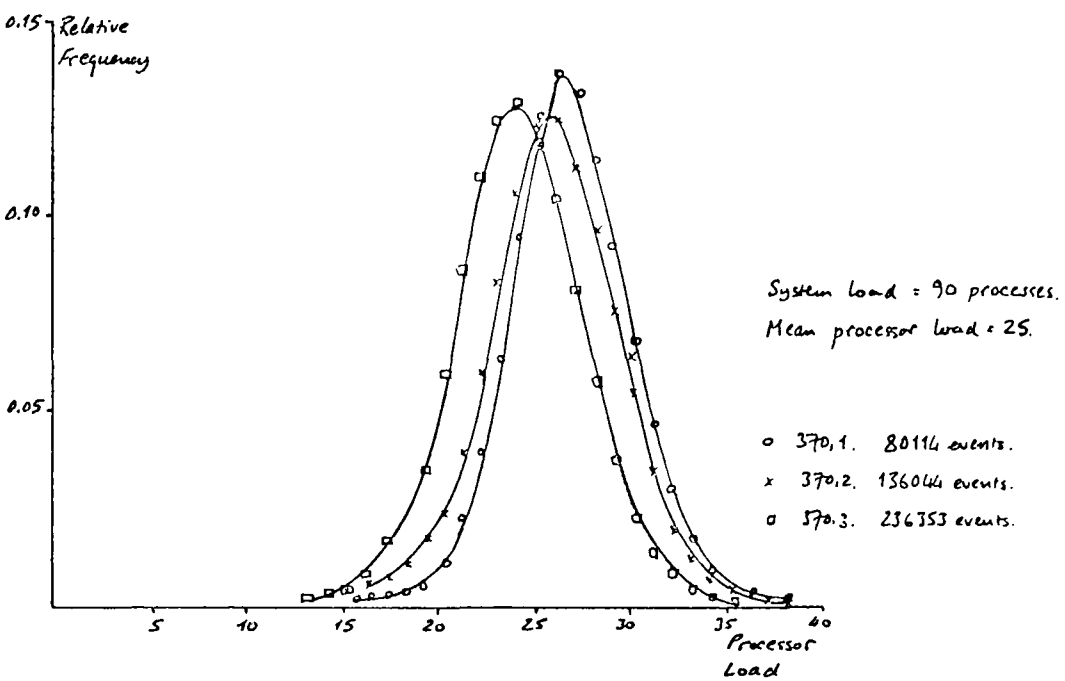


Figure 4.9 Distribution of processor load.

for i/o completion is the major systems effect that limits the rate of work throughput.

System	Running	File-i/o bursts	Terminal-i/o bursts	Other-i/o bursts
S/360,1	0.5	0.443	0.035	0.022
S/360,2	0.5	0.412	0.053	0.035
S/370,1	0.5	0.420	0.041	0.039
S/370,2	0.5	0.452	0.033	0.015
S/370,3	0.5	0.440	0.044	0.016

Table 4.2 Relative Frequency of Process States

The effect of processor load on run-time between bursts was investigated. Figure 4.10 shows a histogram of these run-times for selected processor loads. As the load increases, the model frequency decreases and the run-time distribution grows in the longer-time regions. This indication of a contention effect lead to a re-plot of Figure 4.5 with the longest 10% of run-times discarded. This is shown in Figure 4.11. The value of 10% was chosen from the discrepancy between the means and the mode of Figure 4.5 and 4.7. The reduced plots are fitted by simple linear regression models. The Pearson correlation coefficient confirms that the reduced mean run-time between bursts is highly correlated with processor load, but the variance of the unreduced mean has a larger effect since it conceals the contention result. Processor contention is not a factor limiting S/360 work throughput.

The more evently distributed run-time between bursts for the S/370 (see Figure 4.7) implied that eliminating only the longest 10% would not

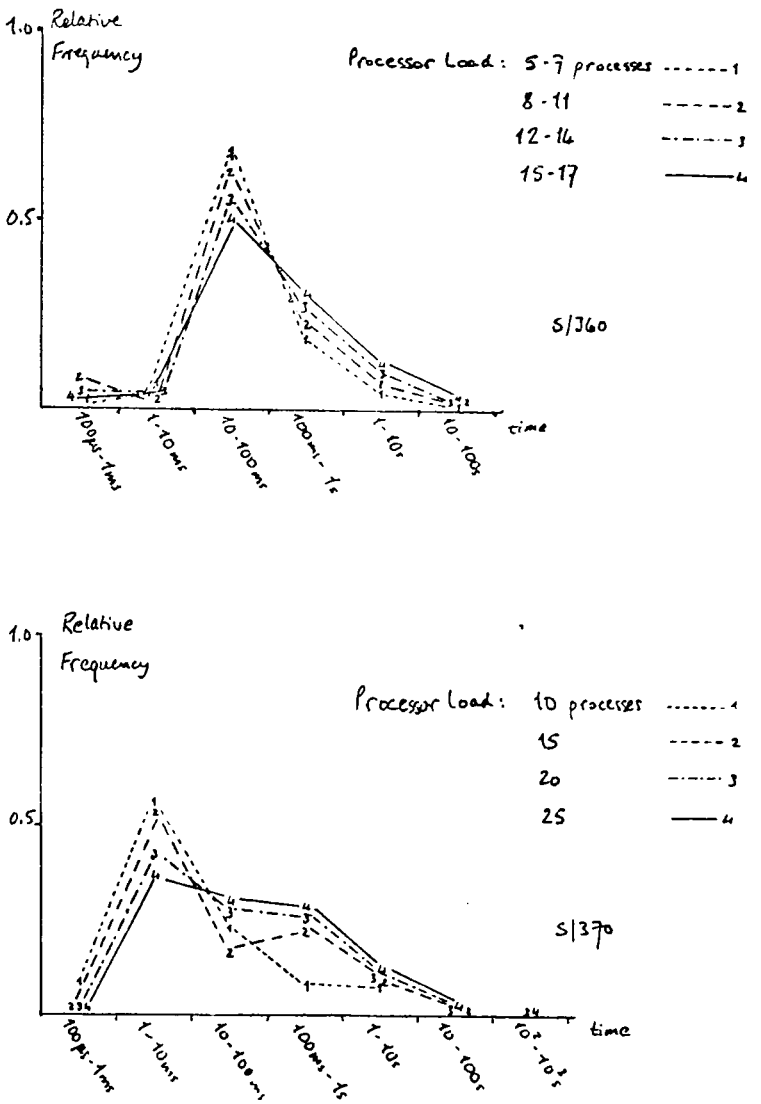


Figure 4.10 Distribution of mean run-time between bursts for various loads.

reveal any feedback contention effect. It was found that the longest 30% of the run-times had to be discarded to reduce the means to the order of the mode, as in Figure 4.12. Even then, the correlation between processor load and run-time is poor. As expected for the faster processor, CPU contention does not limit S/370 work throughput.

Independent measurements by both Jalics and Parapudi place the run-time between i/o requests with 90% means of 46ms and 46ms respectively, and with all-inclusive means about twice this magnitude. These figures agree well with the S/360 90% plots of Figure 4.11.

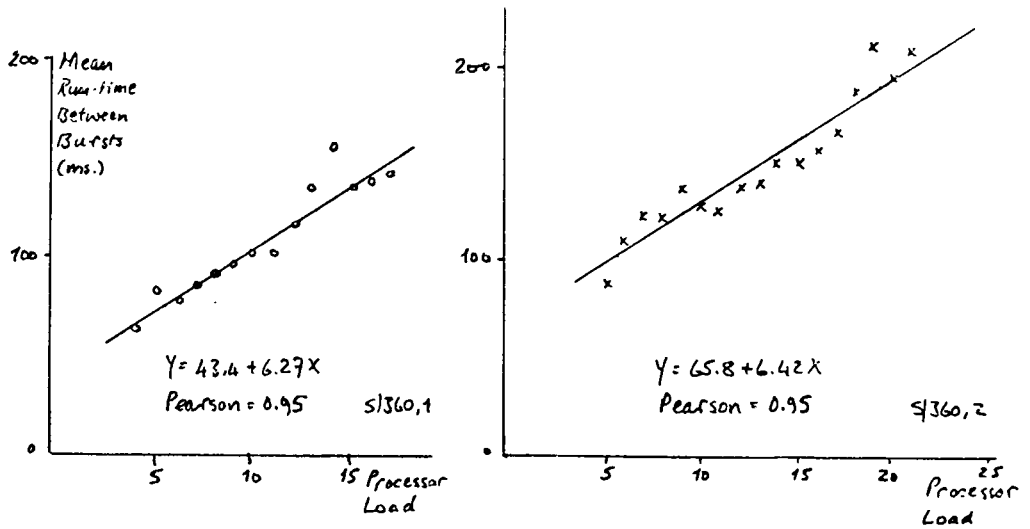


Figure 4.11 Mean run-time between bursts excluding longest 10%.

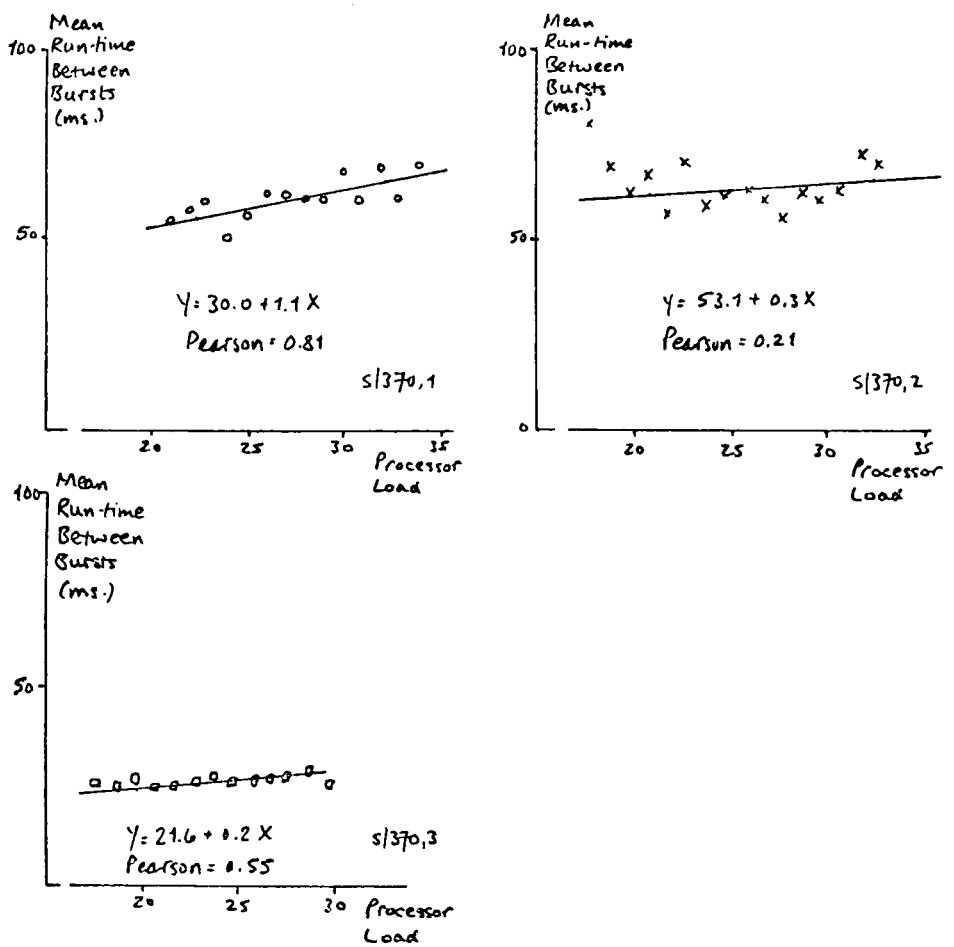


Figure 4.12 Mean run-time between burst excluding longest 30%.

4.4 Filestore Accessing Behaviour

4.4.1 Sequential accessing and file-i/o bursts.

Figure 4.13 shows the distribution of file response times. These times exclude any initial wait before the file request is accepted. We have incorporated this wait into the run-time before bursts. For S/360, about 10% of response times exceed 100ms. This is despite the observed mean number of cylinders moved of around 10. The expected response time of an IBM 2314 disk ranges from 25ms (no seek delay) to 75ms (for a seek of one third of the available cylinders, the average for uniform accessing). The bigger, faster IBM 3330, 11 disks used for the S/370 have only 5% of their response time exceeding 100ms. Expected response times for these disks are from 15ms (no seek) to 40ms (average seek).

Figure 4.14 and 4.15 show detail of these response times. For the S/360 traces, the first peak at 19ms corresponds to a rotational delay of about one-half revolution time, plus the record transmission time. This zero seek-time peak reflects the higher proportion of zero seek requests already known for S/360.

The second peak is wide, at 29-32ms. This may be explained by the "missed revolution" effect that occurs when sequentially accessing sequentially formatted records if the run-time between requests is short enough. When a process makes a "successor" request for the next record shortly after completion of i/o service for the "previous" record, the start of the next record has already passed the read/write arm. A delay of almost a full revolution is incurred, the exact delay depending on the arrival time of the successor request. We have already seen (Figure 4.2) the mean run-time between requests-in-a-burst for file-i/o is 6.6ms with S.D. 1.5ms.

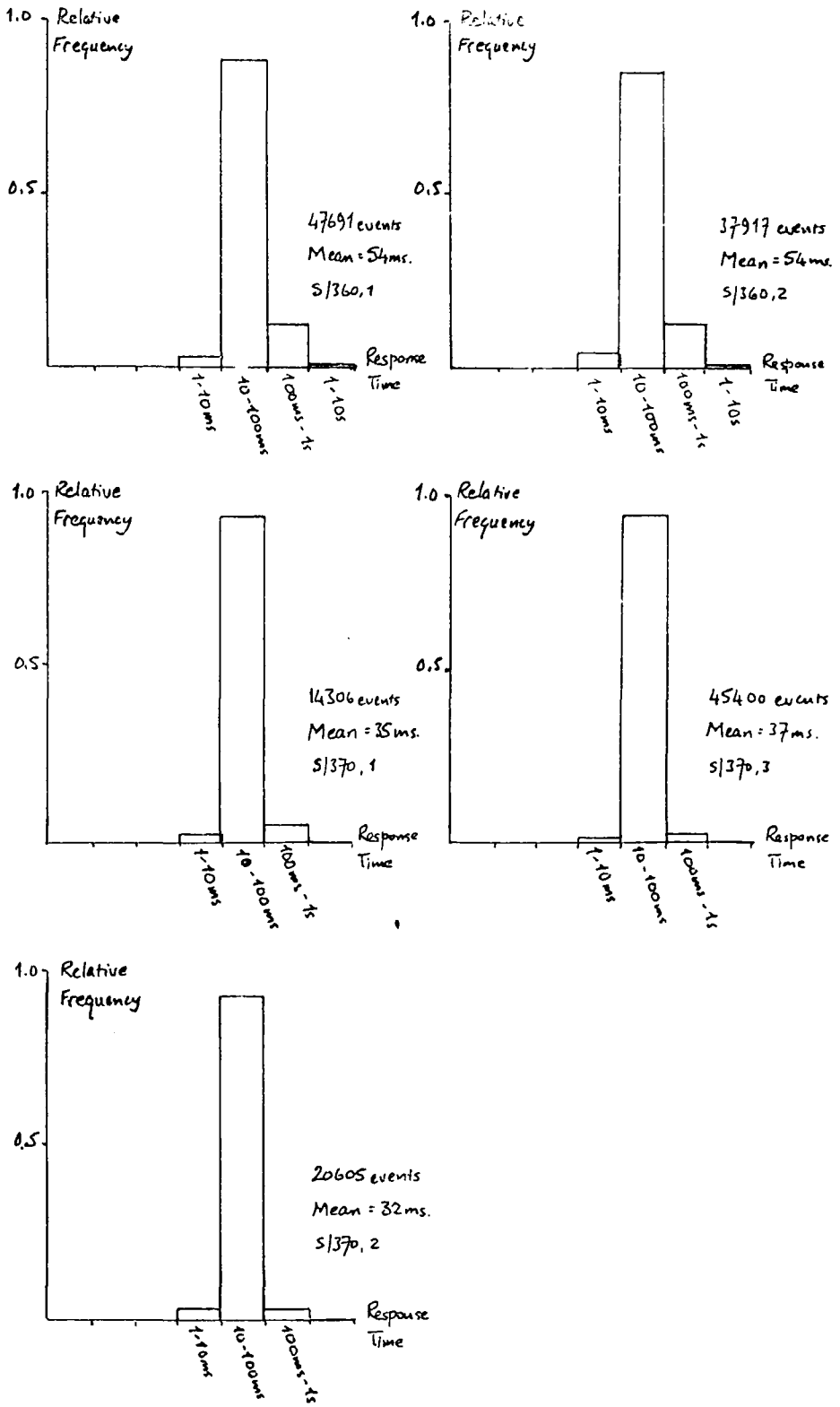


Figure 4.13 File i/o Response Time Distribution.
(excluding any initial wait)

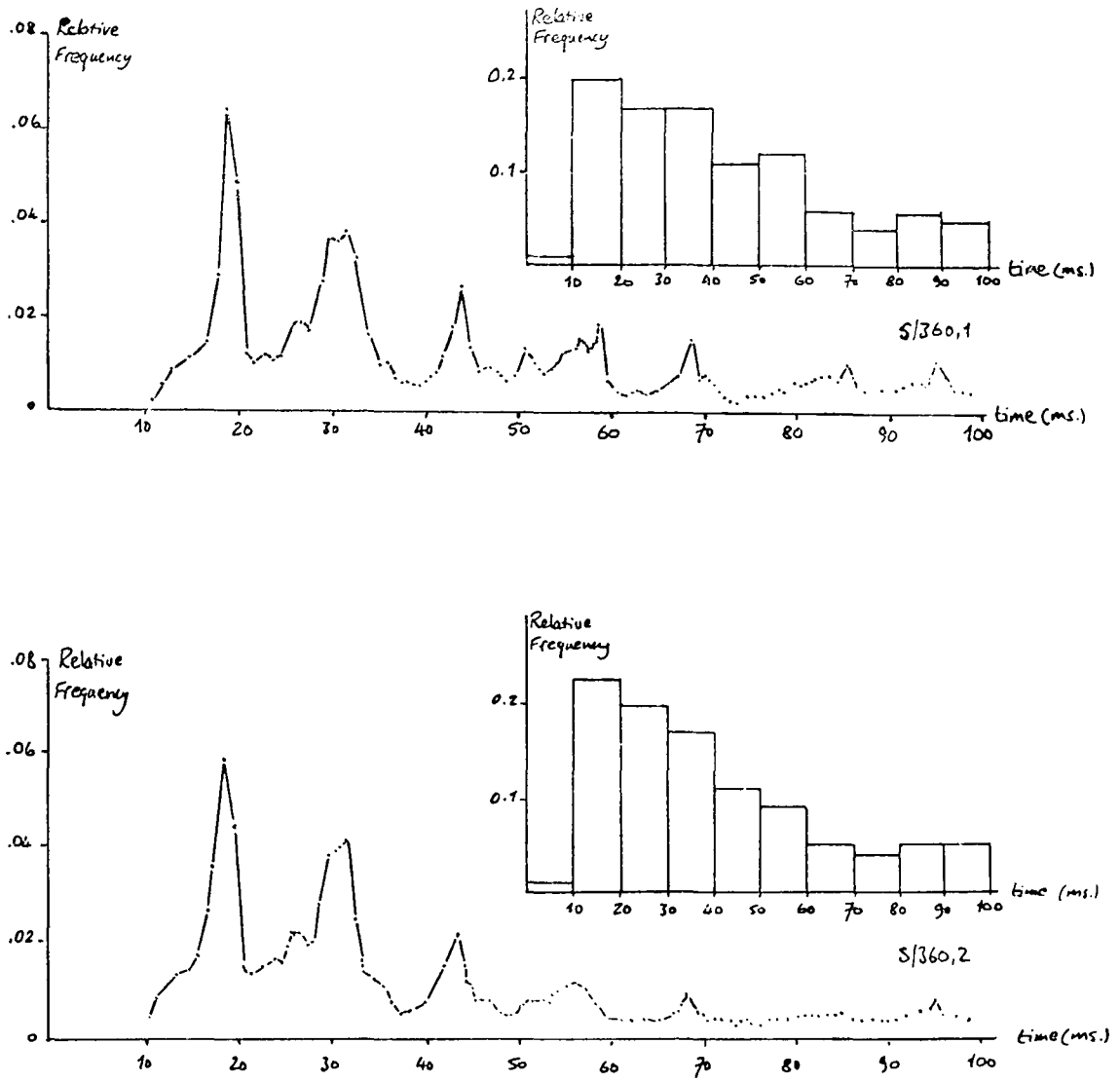


Figure 4.14 File i/o Response Time Distributions - S/360.
(Detail 0-100ms)

From this, expected rotation delay for "successor" requests

$$= (\text{rotation time} - 6.6)\ \text{ms} \pm 1.5\text{ms}$$

$$= 18.4\text{ms} \pm 1.5\text{ms}$$

$$\text{response time (mean)} = (18.4\text{ms} \pm 1.5\text{ms}) + \text{record transmission time}$$

$$= 30\text{ms} \pm 1.5\text{ms}$$

for the 11.5ms record transmission time on the S/360.This calculation fits with the second peak of Figure 4.14. Table 4.3 shows the relative proportions of services represented by the first two peaks of the response time histograms in Figure 4.14 and 4.15. The mean S/360 burst length of 2.0 (see Figure 4.3) fits well with the almost equal number of services under the first two peaks (from Table 4.3). The calculation of expected response time if accesses within bursts are spatially sequential fits well with the mean burst length and number of services under the first two peaks to support the conculsion that spatial accessing within bursts is sequential in intent.

	S/370 range	370,1	370,2	370,3	S/360 range	360,1	360,2
1st peak	0-18ms	0.52	0.49	0.45	0-24ms	0.48	0.50
2nd peak	19-24ms	0.48	0.51	0.55	25-38ms	0.52	0.50

Table 4.3 Relative number of file services under first and second Time Peaks

The S/370 response time data (Figure 4.15) show a large peak at 19-22ms. This may be explained by sequential accessing within bursts, as for S/360. To calculate the "missed revolution" characteristic delay, we proceed as for the S/360:

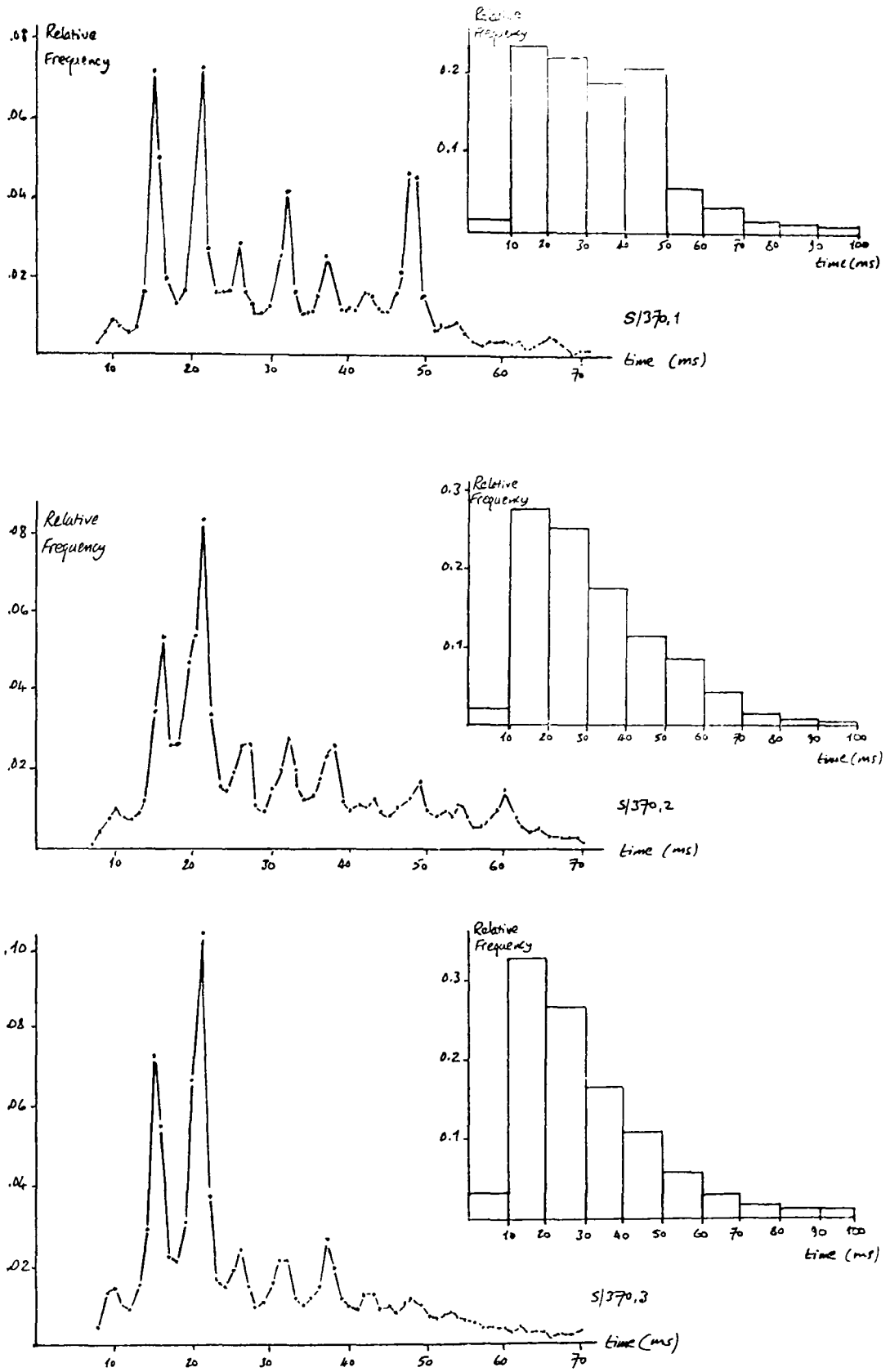


Figure 4.15 File i/o Response Time Distributions - S/370.
(Detail 0-100ms)

expected rotation delay for "successor" request

= (rotation time -1.8)ms $\pm$ 0.8ms

= 15ms $\pm$ 0.8ms

expected mean response = (15 $\pm$ 0.8) + record transmission time

= 20.5ms $\pm$ 0.8ms

This calculation fits the second response time peak. From Table 4.3, the mean burst length of 2.5 for S/370 (Figure 4.3) seems too long for the almost equal numbers under the first two peaks of Figure 4.15. However, 2.5 is the mean length for a layout in which file placement is not contiguous. If, because of the "missed revolution" time delay calculation, we say that the second peak of Figure 4.15 represents the effects of sequential accessing in bursts, we cannot expect the first two peaks to wholly reflect the mean burst length because the layout would break the sequentiality of reference for long bursts. The peaks should then indicate a shorter mean length. Our explanation fits well with the available data.

This argument strongly supports the conclusion that the Non-Consecutive layout of S/370 exhibits a large degree of spatial/sequential accessing within file-i/o bursts.

4.4.2 S/360 distribution of seek movements

A new result from Gray's S/360 trace of spatial accessing concerns the distribution of seek movements. This is shown in Figure 4.16. S/360,1 and S/360,2 have a mean of 13 cylinders moved in those 33% of occasions when a seek was made; Jalics reported a mean of 15 cylinders on 35% of occasions. From the distribution shown, small movements are common and the full range of cylinders is very rarely used.

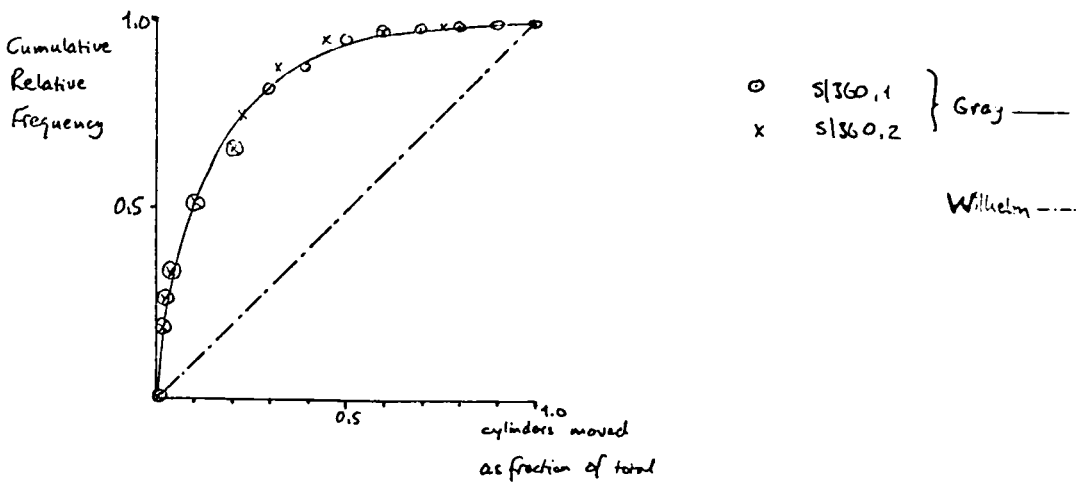


Figure 4.16 Cylinders moved - S/360 Sequential Layout.

The distribution is closely fitted by the empirical function:

$$y = \sqrt{1.5x} \quad , \quad x = \text{uniform } \{0, 2/3\}$$

The spatial distribution function used by Wilhelm [55] is also shown on Figure 4.16.

4.5 Summary and Conclusions

The main result of measurement is that i/o is performed in bursts of characteristic length and run-time distributions. This observation explains and confirms other independent observations made by Lynch, Jalics and Gray. The burst lengths are hyper-exponentially distributed with a file-i/o mean of 2 for both systems. Run-time between bursts is exponentially distributed with a mode of about 50ms for S/360 and 10ms for S/370. Run-time between requests-in-a-burst is uniform about 7ms (or 2ms for S/370) when delimited by Lynch Period, an interval within which more than 50% of these run-times fall.

Analysis of S/360 file wait times confirms Gray's findings of sequential spatial accessing with a high proportion of zero seek-time requests. This result has been extended by the new finding that sequential accessing is due in part to bursts of sequential accessing by individual processes, as well as Gray's report of the popularity among all processes of central, catalogue areas. To a lesser extent, sequential spatial accessing is also found on the S/370 filestore caused, as for S/360, by sequential accessing by individual processes. The Non-Consecutive layout partially preserves the sequential "intent" by allocating space sequentially within cylinders. A new result from Gray's data on spatial accesses is the distribution of seek movements on a sequential layout filestore.

Conclusions about the performances of the measured systems are that neither is CPU-bound and that work throughput is limited, in the system feedback sense, by i/o waits.

CHAPTER 5

A New Method of Filestore Analysis

5.1 Introduction

The method of filestore analysis comprises a simulation queuing model of the filestore within a statistical empirical model of its environment. Processes in the environment behave according to the characteristics deduced from real measurements. A simulation queuing model of the filestore facilitates trials with alternative configurations and devices used to implement the filestore.

5.2 Performance Indices

The principal outputs of the model are the two indices of performance which were deduced from software measurement. Filestore performance is measured by response time. System performance is defined as the rate of completion of terminal-i/o bursts, i.e. the rate of completion of work units as defined in Chapter 3. Whenever comparison is made using this work rate, the same set of work-units is used.

5.3 Load Representation

5.3.1 Introduction

Each process in the environment of the filestore conducts i/o in bursts with trace-derived run-time distributions between i/o-bursts and between-requests-in-a-burst for the three types of i/o. We have developed an algorithm that represents the spatial accessing habits of each process. The algorithm is parameterized to map the sequential intent of each burst on to the layout of the filestore, and is calibrated for each system. Burst lengths are well characterized, and are represented by trace-derived parameters describing their distribution. The representation techniques are statistical and use a pseudo-random number generator. Predictions are taken from several trials of the model with identical inputs save for a different random number seed.

5.3.2 The model of i/o bursts

Data describing burst lengths are derived from a software trace to recreate the load conditions prevailing during that trace. The relative probabilities for each state are inputs from the trace. The four states of each process are:

(i) running before any i/o burst

(ii) file-i/o burst

(iii) terminal-i/o burst

(iv) other-i/o burst

If an i/o burst state is selected, the burst length is generated from the distribution of that i/o type. All types of i/o have burst lengths distributed as in Figure 5.1. Generation is convenient if each part is fitted separately, with a change of origin in each case, as in Figure 5.2.

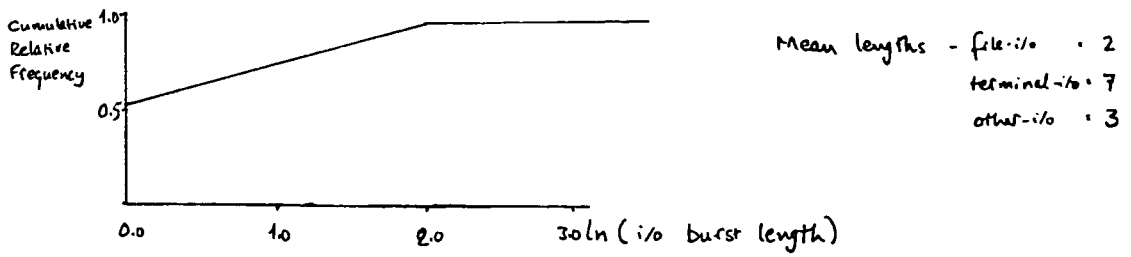


Figure 5.1 Characteristic distribution for all types of i/o burst lengt

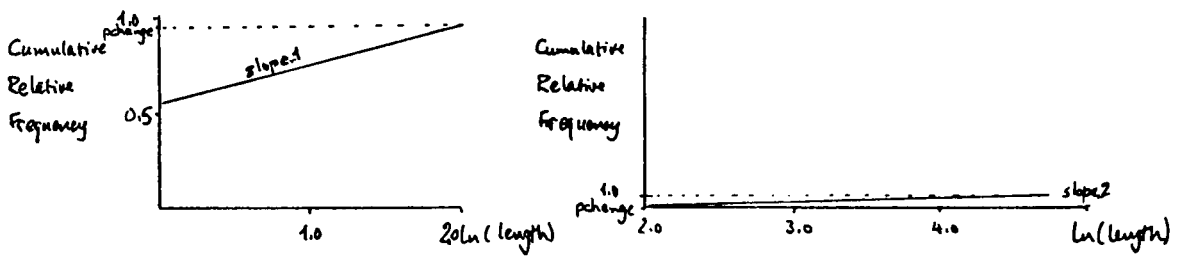


Figure 5.2 Separate fits of the burst length characteristic.

Selection of burst length is:-

1. Let p = uniform $\{0,1\}$.
2. if P $\leq$ pchange then 3. else 4.
3. length $\leftarrow$ integer (exp. (slope_1 * p)).
4. length $\leftarrow$ integer (exp. (slope_2 * (p-pchange))).

where pchange, slope_1 and slope_2 are as in Figure 5.2. The input parameter pchange describes the change of slope for each i/o burst length distribution. For each type of i/o burst these parameters are deduced from the trace. The value of pchange is chosen from the intersection of the two "best-fit" (by Pearson correlation coefficient) straight lines through the data points. Values of pchange from the traces used in this work are given in Figure 4.4 (and in Appendix A for non-file burst lengths).

5.3.3 The model of spatial load

Parameters describing addressing behaviour are calibrated for a particular layout (we have measured the Sequential and Non-Consecutive layouts) and remain set for all model runs on that layout. A filestore address is generated as a triple (disk, cylinder, sector). The disk is selected according to the distribution of accesses across the disks of the measured filestore, a trace-derived input. The cylinder is selected from -

from current cylinder i,
select catalogue cylinder with probability α
select current cylinder with probability $\beta-\alpha$
move to cylinder j with probability $1-\beta$
j is such that $|i-j|$ is drawn uniformly from $\{0, C/2\}$ and $\alpha<\beta<1.0$.

The restriction on the size of move selected is taken from the observation that 97% of all moves are less than half the available range of cylinders. An addressed sector is selected from

uniform $\{1, \text{number of sectors per cylinder}\}$.

In accordance with deductions made from the study of response time distributions (Figures 4.14 and 4.15), selection of successor file addresses depends on the layout. During a burst the disk address remains constant. Sequentiality of reference is preserved for a Sequential layout, but discontinuities are introduced for the Non-Consecutive layout. Selection is as follows:

(i) Sequential layout: the sector address is incremented modulo the number of sectors per cylinder. The cylinder address is incremented if the sector address "wraps round".

(ii) Non-Consecutive layout: successive addresses are generated sequentially or discontinuously according to probability P_s, where P_s = uniform $\{0,1\}$ and

$0 \leq P_s \leq \gamma \Rightarrow$ sequential - generate as (i) above

$\gamma < P_s \leq 1.0 \Rightarrow$ discontinuous - generate a new triple (disk, cylinder, sector)

Parameters α, β and γ are calibrated from trace-derived measurements by iteration from initial values.

For the Sequential layout, Gray reported that 50% of accesses were to the central, catalogue cylinders. This sets α initially to 0.50. The fraction of zero-seek accesses for this layout is 70%, giving an initial value for β of 0.70. Table 5.1 shows the actual fractions of zero-seek requests and the overall mean number of cylinders moved for a seek. The accessing model was iterated with various parameter values until its output most closely matched the real system output for S/360,1 (the calibration data for the Sequential layout). The final parameter values are:

$\alpha = 0.50$ $\beta = 0.83$

Performance Index	S/360, 1 Calibration	S/360,2 Validation
Observed Zero-Seek Requests Simulated Zero-Seek Requests ± S.D.	63% 63% ± 2%	68% 74% ± 2%
Observed No. of Cylinders Moved Simulated No. of Cylinders Moved ± S.D.	12 20 ± 2%	13 12 ± 2%
Observed Mean Response Time Simulated Mean Response Time ± S.D.	54ms 57ms ± 5%	54ms 63ms ± 6%

Table 5.1 Calibration and Validation Results for the Sequential Layout

Performance Index	S/370,3 Calibration	S/370,1 Validation	S/370,2 Validation
Observed Access Rate Simulated Access Rate ± S.D.	25/sec 25/sec ± 5%	24/sec 24/sec ± 6%	17/sec 16/sec ± 5%
Observed Mean Response Time Simulated Mean Response Time ± S.D.	37ms 34ms ± 9%	35ms 36ms ± 8%	32ms 35% ± 5%

Table 5.2 Calibration and Validation Results for the Non-Consecutive Layout

The data in Table 5.1 for S/360,2 are the validation results. Other inputs to the system model are derived from the S/360,2 trace as described in 5.3.4 and 5.3.2. The validation results lie close to their real values. More importantly, the accessing algorithm for individual processes accurately models gross accessing patterns on a Sequential layout. Figure 5.3 shows file response time distribution for the calibration and validation trials. Figure 5.3 should be compared with Figure 4.14 . These response times confirm the accuracy of prediction.

For the Non-Consecutive layout, no data or accessing patterns were available. From Figure 4.15, an initial value of $\alpha = 0.45$ was found from the weighted average of the fraction of accesses with no seek movements, i.e. those with a response time within 24ms (the first two major peaks). The value of $\beta = 0.83$ was retained and an initial $\gamma = 0.50$ (the parameter which introduces spatial discontinuities) was adopted. The model was iterated on the S/370,3 trace with various parameter values, and simulated outputs were compared as indicated in Table 5.2. The final results of this calibration gave

$$\alpha = 0.40 \qquad \beta = 0.83 \qquad \gamma = 0.25$$

The validation runs for S/370,1 and S/370,2 in Table 5.2 are for these parameter values. Other inputs were taken from the S/370,1 and S/370,2 traces (see sections 5.3.4 and 5.3.2). The validation sets of response time distributions are shown in Figure 5.4. The real measures are given in Figure 4.15. Each set of validation results was obtained from five independently seeded runs. The validation results lie close to their real values; they confirm that the Non-Consecutive accessing algorithm accurately predicts gross performance values from a model of individual process behaviour.

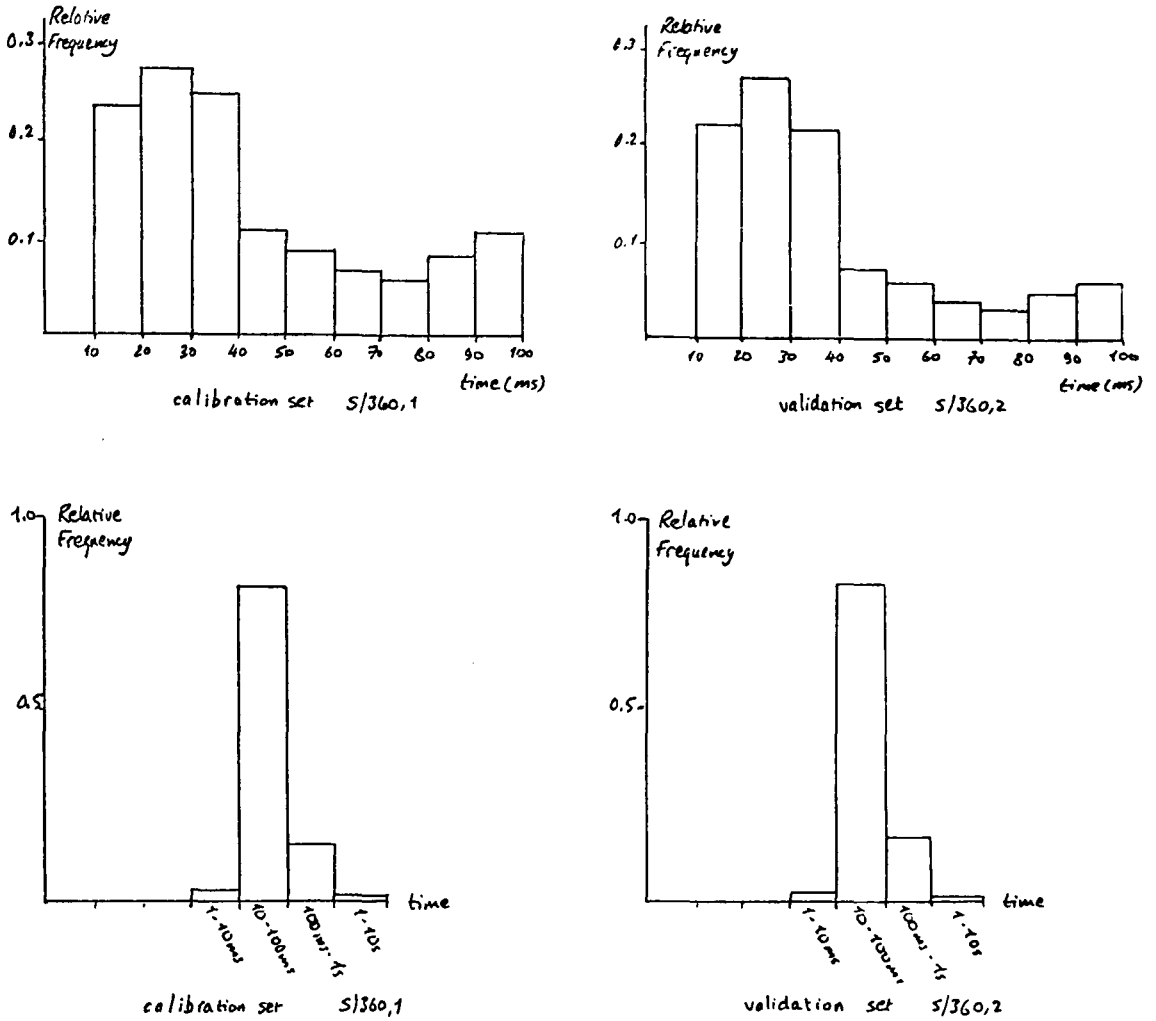


Figure 5.3 Simulated Filestore Response Time.
(S/360. α =0.50, β = 0.83)

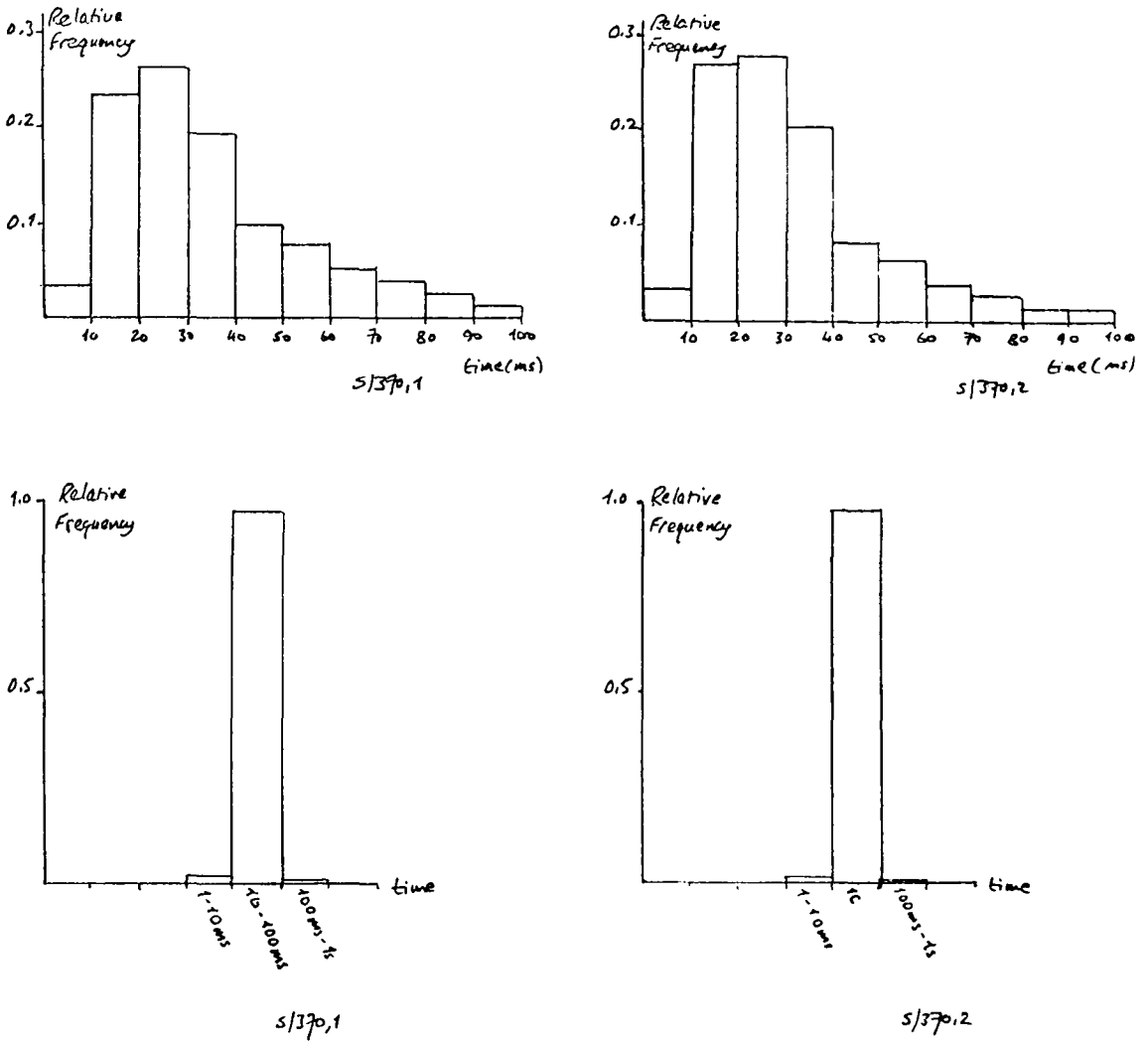


Figure 5.4 Simulated Filestore Response Time
(Validation sets - S/370 $\alpha = 0.40$, $\beta = 0.83$, $\gamma = 0.25$)

5.3.4 Runtime and waiting time generation

Means of describing runtime distributions are needed to represent the load conditions prevailing during the software trace. As shown in section 4.3, runtimes between i/o-bursts are distributed exponentially with a long tail that makes the mean misleadingly large. In the model, the runtime between i/o-bursts is generated from an exponential distribution whose mean is of the order of the mode of its histogram, and a correction factor is added for the longer values. The Gamma, Normal and Log Normal distributions were tested but the Exponential distribution based on the reduced mean was the best fit for the shorter end of the runtime range, where most runtimes lie. A technique of fitting this type of distribution with composite Weibull distributions has been described by Anderson and Sargent [2] . In this work a simpler approach has been used; the exponential distribution is used for the dominant part, and a correction factor is added for longer values. For the model, the 90% mean for S/360 and the 70% mean for S/370 were used. The correction factor is taken from a uniformly indexed table of mean values chosen to fit the longer end of the runtime distribution. These factors are simply specified and tractable in use.

A similar method is used for generating runtimes between request-in-a-burst for terminal-i/o and other-i/o, since their runtimes are similarly distributed. Runtime between requests-in-a-burst for file-i/o is almost constant when bursts are delimited by the Lynch Period. It is generated as a constant runtime.

File-i/o response times are calculated in the simulation of the filestore. Terminal-i/o and other-i/o waits are generated from their measured values by uniform random selection from a small set of cumulative probabilites describing the log distribution of waiting time (see Appendix A). Once a

particular log range has been selected, generation is uniform within that range.

5.3.5 Summary of inputs representing load

5.3.5.1 State selection in a process life.

To model transitions between states, we need the relative probabilites of the four states:

1. before any i/o burst
2. file-i/o burst
3. terminal-i/o burst
4. other-i/o burst

5.3.5.2 Length of i/o burst

For each software trace, values of slope_1, slope_2 and pchange are input for each type of burst. Pchange is the change of slope point between the two best straight-line fits. Generation of length is by the algorithm given above.

5.3.5.3 Parameters describing individual accessing

For each system, values of α, β and γ calibrate the accessing algorithm for predicting the effect of file layout on the sequential intent of each burst of file i/o. Selection is by tests and calibration from available accessing data. For each trace, the relative proportions of bursts to each disk are inputs.

5.3.5.4 Realtime descriptors

For each system, the runtime between request-in-a-burst for file-i/o is found by inspection of the distribution of runtimes between file-i/o requests.

For each software trace, the reduced runtimes most nearly corresponding to the mode of distributions for

1. runtime before i/o - burst
2. runtime before requests-in-a-burst for terminal-i/o
3. runtime before requests-in-a-burst for other-i/o

represent the load prevailing during that trace. They are used as means of an exponential distribution to describe the corresponding runtime. The "long tails" are generated empirically from trace-derived data.

For each trace, i/o waiting times for terminal-i/o and other-i/o are generated from the cumulative distribution of log (waiting time).

5.4 The Filestore Simulation Queueing Model

The performance of a rotating magnetic disk can be characterized by certain service-time delays; seek-time delay to position the read/write arm at a selected cylinder, and rotation delay to await the arrival of the addressed record at the read/write position. The modelled filestore is n disks served by a single channel. Queues are maintained for each disk and the channel. Availability of the channel for command as well as data transmission is explicitly modelled. Figure 5.5 shows the time delays incurred for a filestore i/o service, and those delays that are regarded as neglegible and not modelled. Figure 5.6 shows these delays for a disk equipped with Rotational Position Sensing (RPS). In conjunction with a channel able to support the extra processing, RPS enables the freeing of the channel during rotational delay (as well as seek-time delay).

In the filestore simulation, various configurations of disks may be modelled. The number and capacity of disks may be varied, as well as the number of read/write heads. All read/write heads are fixed to the same arm, thus increasing the capacity per access, i.e. the capacity per arm position. Seek-time delay is calculated by functions approximating the modelled disks, (either IBM 2314 [20] or IBM 3330 [21]). Constant velocity when seeking is assumed. Rotational delay is calculated from disk rotation speed and the difference between the addressed sector and the current angular position.

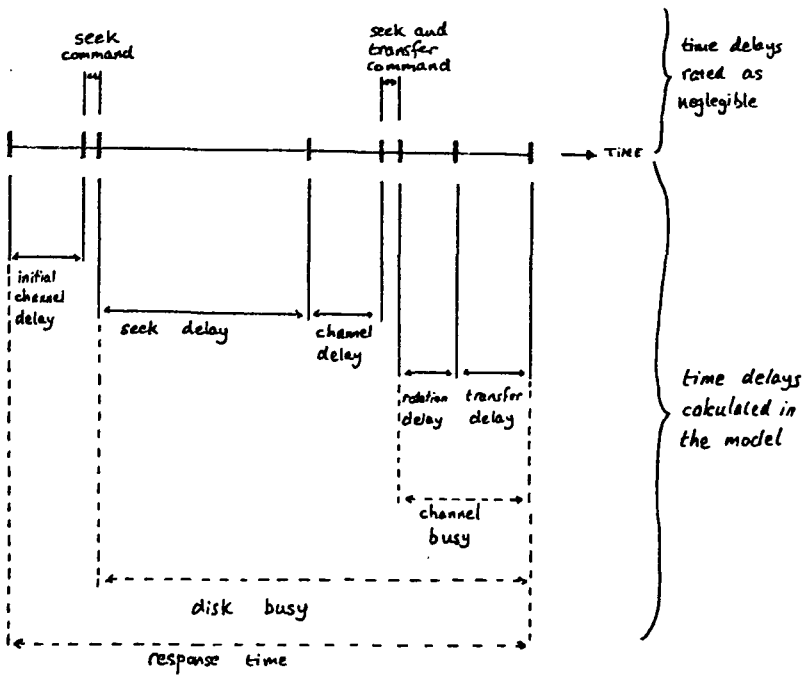


Figure 5.5 Disk service time delays - Non-RPS disk.

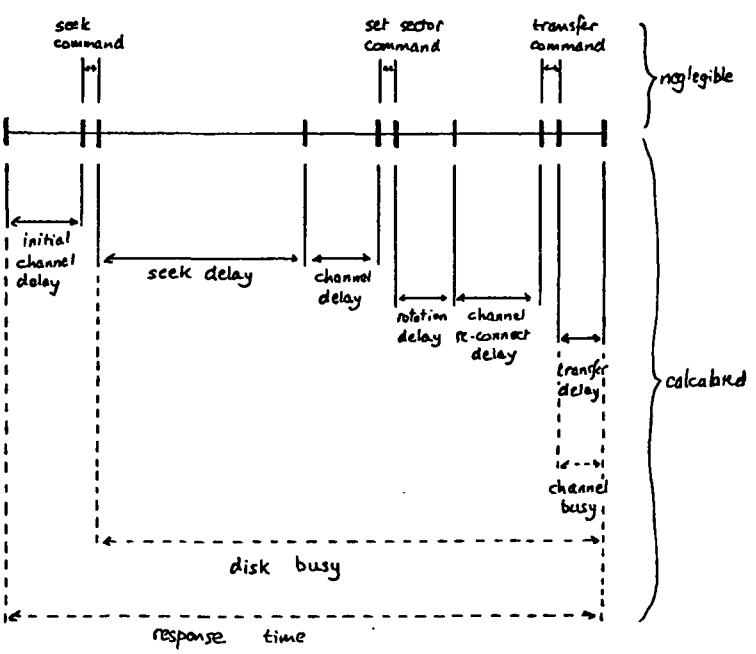


Figure 5.6 Disk service time delays - RPS-equipped disk.

5.5 Validation

Following Van Horn [19], we define validation as the process of building an acceptable level of confidence than an inference about a simulated system is a correct or valid inference for the actual system. The set of observed measures compared to model outputs in order to develop such confidence is:

1. An index of system performance - the rate per minute of work-units
2. An index of filestore performance - the filestore response time
3. An index of load - the rate per second of filestore requests.

All measurements were made after start-up transients had been eliminated. A set of five, independently - seeded runs were used to establish standard deviations and confidence limits for the calibration parameters. Further sets of five, independently seeded, runs were made for each validation trace.

From Table 5.3 the calibration predictions for load and performance lie close to the actual values. The 95% confidence limits are about six percent for these indices in the case of S/360, and range slightly higher for S/370 load and system performance indices. These confidence limite are bounds which the model is likely to exceed only on one occasion in twenty. the less stringent 90% limits are also shown. Standard deviations are all less than ten percent.

The validation results are shown in Table 5.4. The S/360,2 performance indices have 95% confidence bounds of about six percent. The standard deviations of these indices are not large. Prediction of mean response time has been discussed above in section 5.3.3.

Performance Index	S/360,1	S/370,3
Observed Work Rate	41/min	49/min
Simulated Work Rate ± S.D. 95% Confidence Bound 90% Confidence Bound	40/min ± 5% ± 6% ± 5%	48/min ± 7% ± 9% ± 7%
Observed Response Time (mean)	54ms	37ms
Simulated Response Time (mean ± S.D.) 95% Confidence Bound 90% Confidence Bound	57ms ± 5% ± 7% ± 5%	35ms ± 9% ± 11% ± 8%
Observed Access Rate	17/sec	25/sec
Simulated Access Rate ± S.D. 95% Confidence Bound 90% Confiden Bound	18/sec ± 3% ± 4% ± 3%	25/sec ± 5% ± 6% ± 4%

Table 5.3 Calibration Results

For the S/370 validation runs, the 95% confidence limits are similar to the calibration limits of about nine percent. Standard deviations for all indices are less than ten percent. All predicted means lie close to their observed values. The accurate prediction of response time is discussed in section 5.3.3 above. Wider limits than for S/360 were expected because of the absence of data on filestore accessing patterns.

These results lend confidence to the model outputs and the new methods of representing load in the environment of the filestore.

Performance Index	S/360,2	S/370,1	S/370,2
Observed Work Rate	56/min	68/min	53/min
Simulated Work Rate ± S.D.	55/min ± 2%	66/min ± 6%	47/min ± 9%
95% Confidence Bounds	± 3%	± 8%	± 11%
90% Confidence Bounds	± 2%	± 6%	± 8%
Observed Response Time (mean)	54ms	35ms	32ms
Simulated Response Time (mean ± S.D.)	63ms ± 6%	36ms ± 8%	35ms ± 5%
95% Confidence Bound	± 7%	± 9%	± 6%
90% Confidence Bound	± 5%	± 7%	± 5%
Observed Access Rate	14/sec	24/sec	17/sec
Simulated Access Rate ± S.D.	14/sec ± 5%	24/sec ± 6%	16/sec ± 5%
95% Confidence Bound	± 6%	± 8%	± 6%
90% Confidence Bound	± 5%	± 6%	± 4%

Table 5.4 Validation Results

5.6 Summary

A method of evaluating filestores has been developed from measurements of real systems. Load is accurately represented by a model of individual lives of processes in the environment of the filestore. I/O is conducted in bursts of characteristic length and with realistically simulated runtimes between i/o events. Trace derived inputs are used to describe temporal load. Filestore spatial accessing by each process is sequential in intent, as observed in real systems. Address selection within individual bursts on Sequential and Non-Consecutive layouts is accurately modelled. The model built by this method has been validated for its simulated load and two outputs of performance prediction on different sets of input data. The predictions agree well with their measured values and have acceptable standard deviations and confidence limits.

CHAPTER 6

The Effects of Disk Request Scheduling and Disk Capacity on System and Filestore Performance

6.1 Introduction

This chapter presents a simulation experiment to re-evaluate disk scheduling algorithms and to investigate the effect of disk capacity on performance. The system model developed in the previous chapter is used for this experiment. Input parameters are taken from the measurements described in Chapter 4. Simulation outputs for work-unit throughput are only compared where these units are equivalent, i.e. only within runs on the same set of outputs.

The results of Lynch, Jalics and Gray indicate that the conclusions of Teorey and Pinkerton on the ranking of disk request schedulers are not relevant to at least two time-sharing systems. The configuration problem of the number of disks used for a filestore of given capacity has arisen because of the availability of so-called "double-density" disks with twice the number of tracks per surface than their "single-density" variants. The problem is whether any performance losses incurred by fewer servers outweigh the cost savings of the halved number of disks.

6.2 Disk Request Scheduling Algorithms

The scheduling algorithms tested, with brief descriptions, are

1. First Come First Served (FCFS) - with a single waiting queue per disk.
2. Shortest Seek Time First (SSTF) - with a single waiting queue per disk.

3. SCAN with a single waiting queue per disk. The SCAN algorithm includes Merten's modification to reverse the scanning direction if there are no queued requests ahead of the currently selected position.

4. C-SCAN with a single waiting queue per disk. The disk arm returns to the start-scan position if there are no queued requests ahead of the currently selected position.

5. N-SCAN with two queues per disk, the service queue and the waiting queue. At request completion, the head of the service queue is chosen. If the service queue is empty, all waiting requests are taken and ordered by Frank's algorithm [15] approximating the optimal shortest-seek-time. This is the $N=\infty$ case.

Descriptions of the algorithms are given in Appendix C.

In addition to the above, two implementations of an algorithm suggested by Lynch are included. Lynch [29] suggested a scheduler which enqueues any request with a seek movement in the expectation of a request to the currently selected position. If within a short period the expected request has not materialized the queued requests are serviced. In Chapter 4 we have named this interval the Lynch Period. This algorithm is designed to take advantage of sequential accessing with a high percentage of zero-seek requests. The two implementations differ in their queue service method. LYNCH1 services its waiting queue in a FIFO manner, LYNCH2 uses an SSTF method. The Lynch algorithm assumes a small number of requesting processes per disk at any given moment. This is a low-use situation. For its operation on the requests to a single disk, with a single waiting queue, the LYNCH algorithm may be described at two events - "request arrival" and "queue service" :-

a) request arrival -

if ¬ disk busy and cylinders moved = 0

then service

else enqueue;

b) 1. queue service for LYNCH1 -

(i) service the longest waiting request to the currently selected position. (Such requests are enqueued because of a disk busy condition.)

(ii) if none in (i), service the head of the queue if its waiting time exceeds the Lynch Period.

(iii) if none in (ii), do nothing.

2. queue service for LYNCH2 -

(i) service the longest waiting request to the currently selected position.

(ii) if none in (i), SSTF—service the queued requests which have waited longer than the Lynch Period.

(iii) if none in (ii), do nothing.

6.3 Disk Configuration

Disk manufacturers are maintaining the economic effectiveness of their products by price adjustments and performance improvements. Only the latter concerns us here. Larger capacity is offered by increasing track density and using more precise track selection. This also reduces per-cylinder seek time, but the increased capacity may mean longer response time from fewer servers.

For the S/360 environment, a filestore capacity equal to the measured system was modelled, viz. 1600 IBM 2314-style cylinders (i.e. 240 Mbyte). The filestore is modelled with 2, 4, 8 and 16 disks each of 800, 400, 200 and 100 cylinders. For the S/370 environment, the modelled filestore has 1600 cylinders of IBM 3330-style (i.e. 800 Mbyte). This is the same capacity as the measured S/370 system. It is modelled with 2, 4 and 8 disks each of 800, 400 and 200 cylinders.

6.4 Experimental Plan

Input data from both S/360 and S/370 systems are used to re-create the filestore environment of those systems. In the one, disk performance characteristics not varied for the experiment are as for the IBM 2314. In the other, these characteristics are as for the IBM 3330 disk. When the S/360 system is simulated, a sequentially-formatted filestore is modelled using the spatial accessing algorithm developed in Chapter 5. Similarly, the Non-Consecutive layout is simulated for the S/370 system using the accessing algorithm developed to represent the spatial accessing of individual processes for that file organization.

The number of active processes in each system is varied over a small range to enable plots of performance against load for graphical evaluation. The outputs are mean filestore response time and work-unit throughput rate. Each run is triplicated and the outputs averaged; for each combination of factors the same load is presented. The simulation runs are 600 seconds long, and error bounds are taken from the 95% confidence limits of Table 5.4.

The factors and their levels for both systems are

1. Scheduling algorithm: 7 levels

2. Capacity per disk
   (i) 4 levels for S/360 - 2, 4, 8 and 16 disks in a 240 Mbyte filestore
   (ii) 3 levels for S/370 - 2, 4 and 8 disks in an 800 Mbyte filestore

6.5 Conclusions

Figure 6.1 and 6.2 for 8 and 4 disks respectively show that choice of scheduling algorithm does not affect system performance. The Figures are given with the detailed discussion of results in Section 6.6. The graphs for other numbers of disks are similar. From Figures 6.5 and 6.6 there is only a suggestion that fewer disks in the filestore adversely affects the work throughput rate. In general, the plots of results are closer together than their error tolerances.

Figure 6.3 and 6.4 show that SCAN-type schedulers do not improve response time much compared with FCFS, the improvement being of the order of the confidence limits. The LYNCH scheduler performs badly for the Non-Consecutive layout (see Figure 6.2 and 6.4). It gives very low system throughput and significantly longer response times. FCFS performs as well as any other scheduler, except at heavier loads.

Table 6.1 of sample response time variances shows that disk technology has a strong effect on scheduler performances. The characteristics of schedulers, as indicated by these variances, are generally as expected with the SCAN-type having smaller variances than the FCFS and SSTF schedulers.

6.6 Results - Disk Scheduling Algorithms

6.6.1 Work Throughput

Figure 6.1 and 6.2 show work throughput rate against load for each of the tested schedulers. In both the filestores are configured as in the real systems. For the S/360, all of SSTF, the scanning methods and the Lynch scheduler tend to perform better than FCFS. Excluding two off-trend values, each group of results lies in a 10% band of work throughput, so no particular algorithm is outstanding. For the S/370, the 95% confidence bound is $\pm 10\%$. Over the range of loads, no scheduler is significant, though FCFS is as effective as more complex algorithms. The increase in work throughput is linear with load on both systems.

The Lynch scheduler performs as well as any other in the S/360 model, but both variants show consistently low throughput rates in the S/370 model. The differences between the simulated systems are device speeds, numbers of disks and filestore layout. The Lynch Period compensates for device speeds; the algorithm ought to perform better with fewer disks (more frequent queue services); but the interrupted sequentiality in the Non-Consecutive layout will cause more requests to be enqueued. This seems the most likely explanation for the poor performance of the LYNCH schedulers on the S/370.

6.6.2 Mean Response Time

Figure 6.3 and 6.4 present the plots of mean response time with system load for each tested scheduler. FCFS increases response time in the S/360 model by a factor of 7 over the load range (from 100ms to 700ms approximately.)

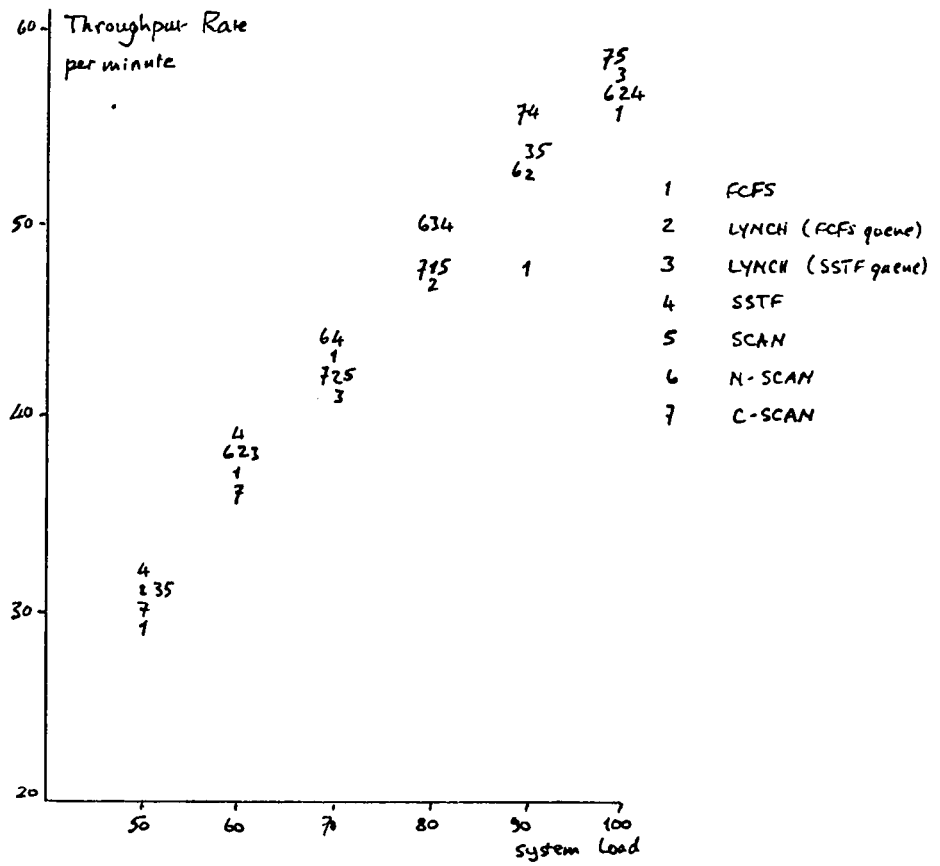


Figure 6.1 Throughput comparisons for all schedulers.
(S/360 inputs; Sequential Layout; 8 disks.)

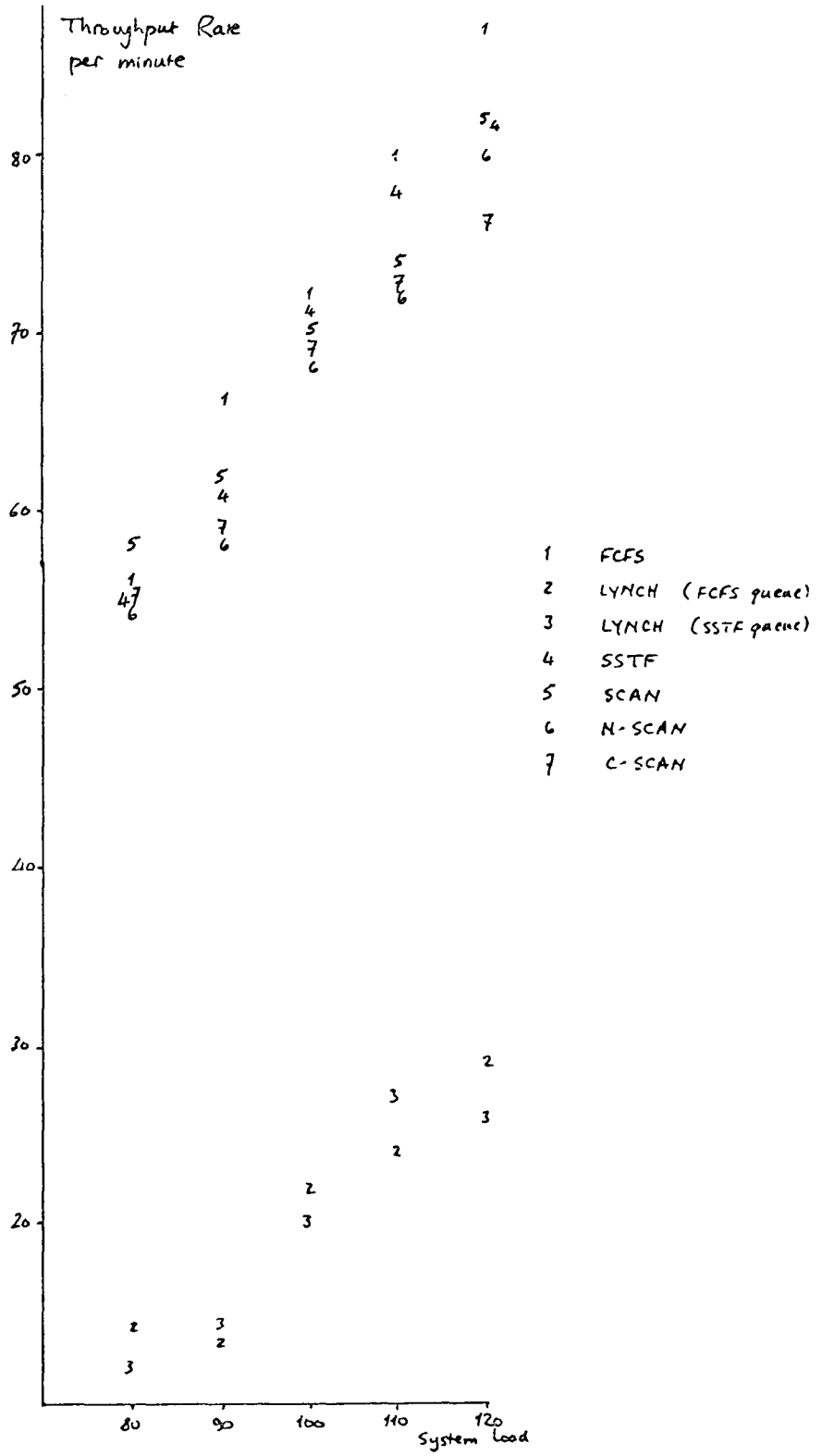


Figure 6.2 Throughput comparisons for all schedulers.
(S/370 inputs; Non-Consecutive Layout; 4 disks.)

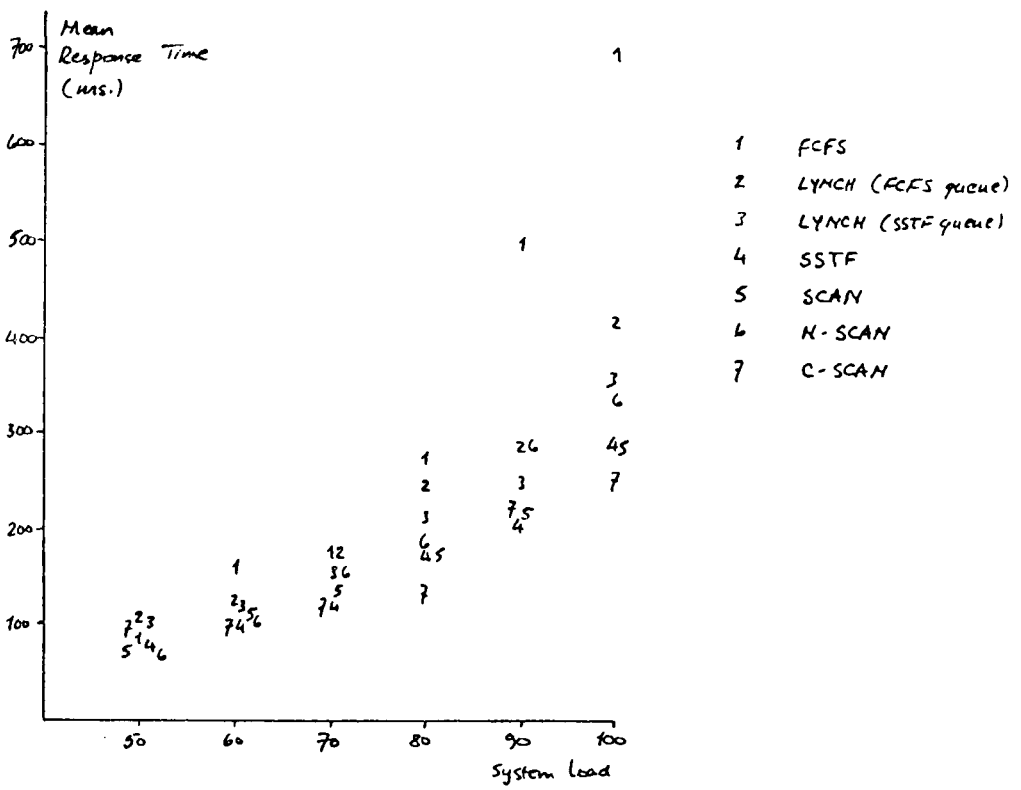


Figure 6.3 Mean response time comparisons for all schedulers.
(S/360 inputs; Sequential Layout; 8 disks.)

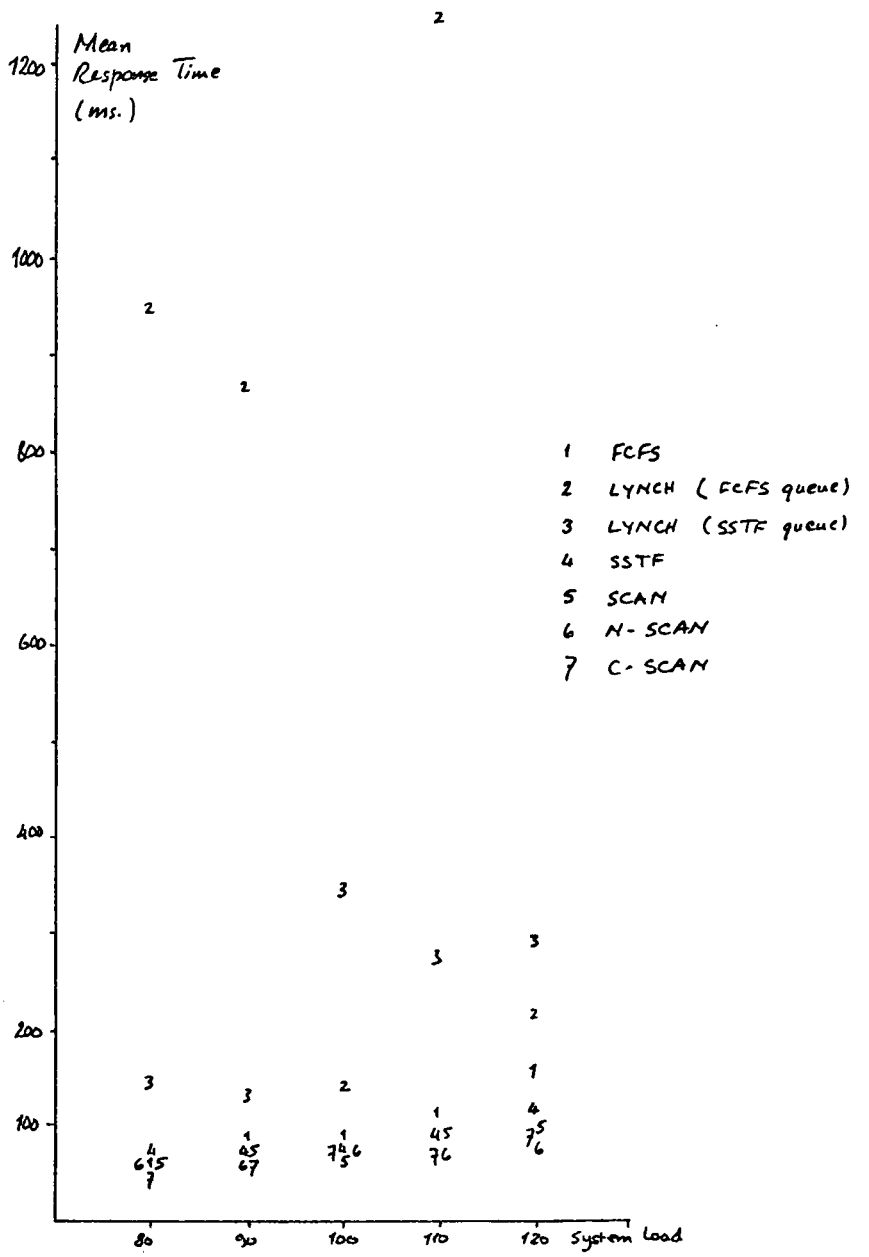


Figure 6.4 Mean response time comparisons for all schedulers. (S/370 inputs; Non-Consecutive Layout; 4 disks.)

This compares with the SCAN range of 2.5 which is almost linear with load. At the higher loads, the ±7% bound for the 95% confidence limits does not cover the gap between both Lynch and FCFS, and the other schedulers. Of the non-scan algorithms, only SSTF performs as well as the scanning algorithms. In the S/370 model the FCFS response range is much smaller at about 2, and FCFS is only worse than the others (Lynch excluded) at the highest load level where its result exceeds the ±8% bound. No significant difference in mean response time is evident amongst the other schedulers. N-SCAN performs better on the Non-Consecutive layout, where the interrupted physical sequentiality distributes the i/o requests across the disk surface, compared with its poor performance on the sequentially formatted filestore. The Lynch scheduler in both tested forms is very poor in the S/370 model and unremarkable in the S/360 model. Layout is an important factor in the viability of this scheduler.

Table 6.1 is a set of sample variances of response times taken from the results plotted in Figure 6.3 and 6.4. These variances are taken from the variances of the response times for the lowest and highest system loads. They are given to illustrate the characteristics of each scheduler. Although the Non-Consecutive layout causes longer seek movements, the faster arm-movement and rotational speed of the IBM 3330,11 disks reduces the variances below their sequentially formatted values (on slower IBM 2314 disks). For the Sequential layout, as shown in Figure 6.3, SSTF has the same order of variance as SCAN and C-SCAN, as well as a similar response time. Under these realistic conditions, SSTF does not degenerate to serving only a small area of the disk. Contrary to expectations, the variance of FCFS is larger than that of SSTF. Both LYNCH schedulers have lower variances than FCFS. For the Non-Consecutive layout (see Figure 6.4) all variances except LYNCH are small. SSTF and SCAN have similar variances, but

C-SCAN and N-SCAN values are even smaller, reflecting the designed improvement of these algorithms.

Response Time Variances		
Scheduling policy	Sequential Layout 8 x 2314 disks	Non-Consecutive Layout 4 x 3330 disks
FCFS	75%	17%
SSTF	38%	23%
LYNCH (fcfs Q)	52%	500%
LYNCH (sstf Q)	47%	175%
SCAN	38%	17%
C-SCAN	36%	11%
N-SCAN	41%	10%

Table 6.1 Sample Variances of Response Times.

6.7 Results - Number of Disks

6.7.1 Work Throughput

Figures 6.5 and 6.6 show work throughput against system load for each configuration of filestore, and selected schedulers. The 95% confidence bounds of $\pm 3\%$ for S/360 and $\pm 10\%$ for S/370 cover almost all the plots at each load value. The suggestion of these data is that more disks per filestore are associated with higher system performance, but no definite results emerge.

6.7.2 Mean Response Time

Figure 6.7 and 6.8 show mean response time against system load. In these figures, the confidence bounds are $\pm 7\%$ for S/360 and $\pm 8\%$ for S/370. For both systems, only the smallest disk size mitigates the response times of FCFS under heavier loads. For the other S/360 schedulers, increasing the number of disk cannot be said to significantly improve filestore performance. Although this is suggested, most of the plots fall within the confidence bound. For the S/370, despite the wider bounds the difference between performances for the least and greatest number of disks confirms that more disks per filestore improves mean response time.

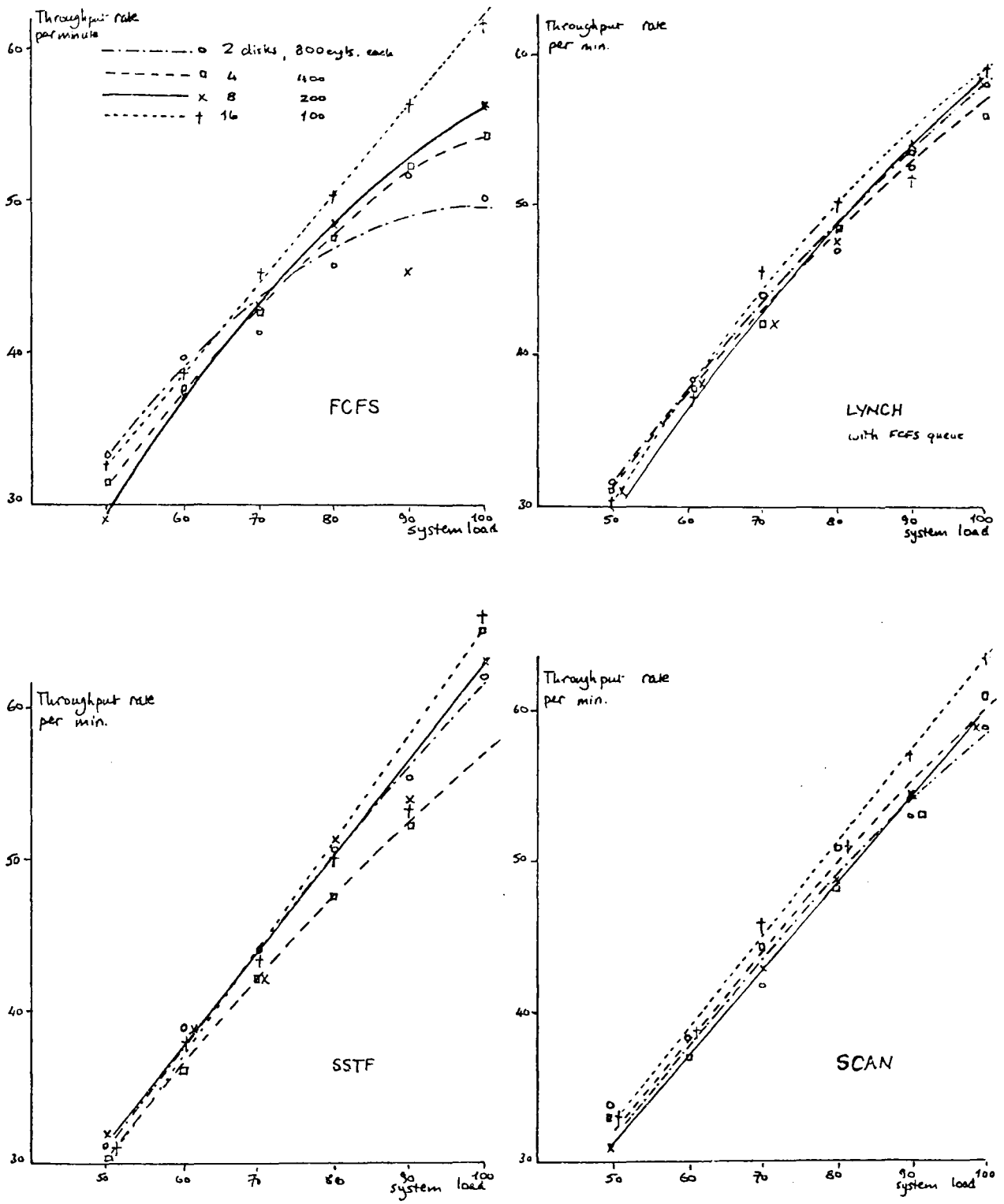


Figure 6.5 Throughput comparisons for scheduler and disk size.
(S/360 inputs; Sequential Layout.)

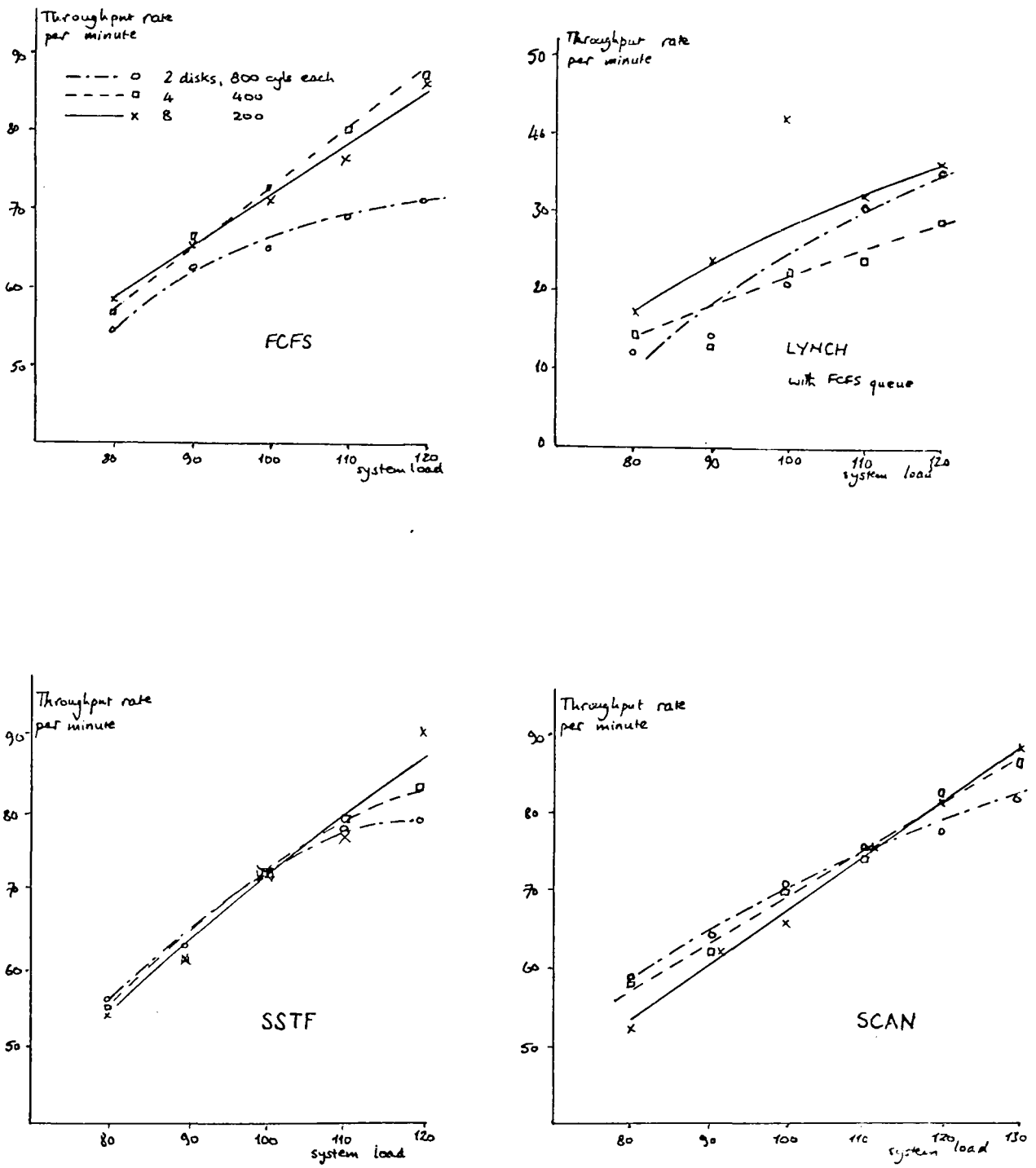


Figure 6.6 Throughput comparisons for scheduler and disk size.
(S/370 inputs; Non-Consecutive Layout.)

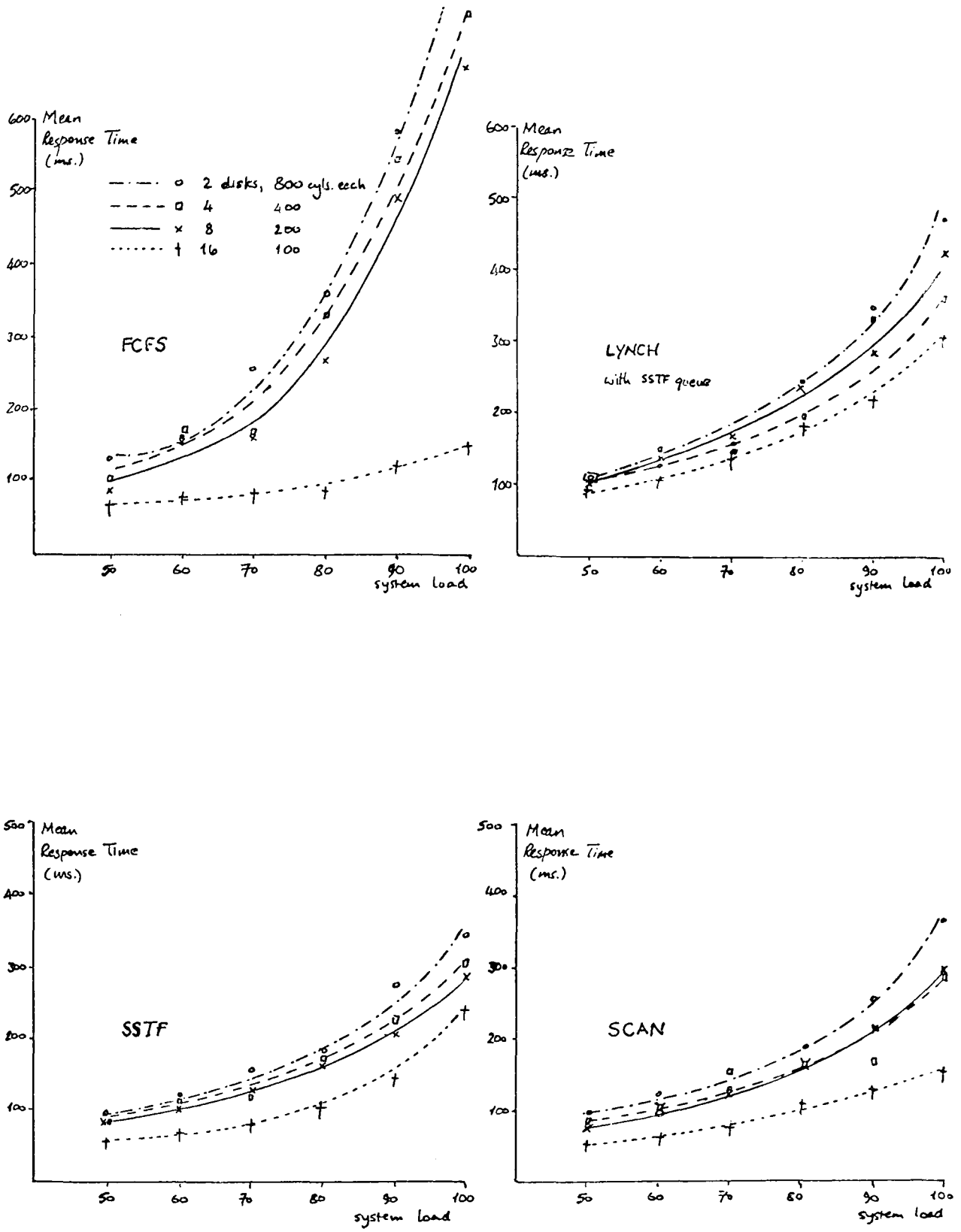


Figure 6.7 Mean response time comparisons for scheduler and disk size.
(S/360 inputs; Sequential Layout)

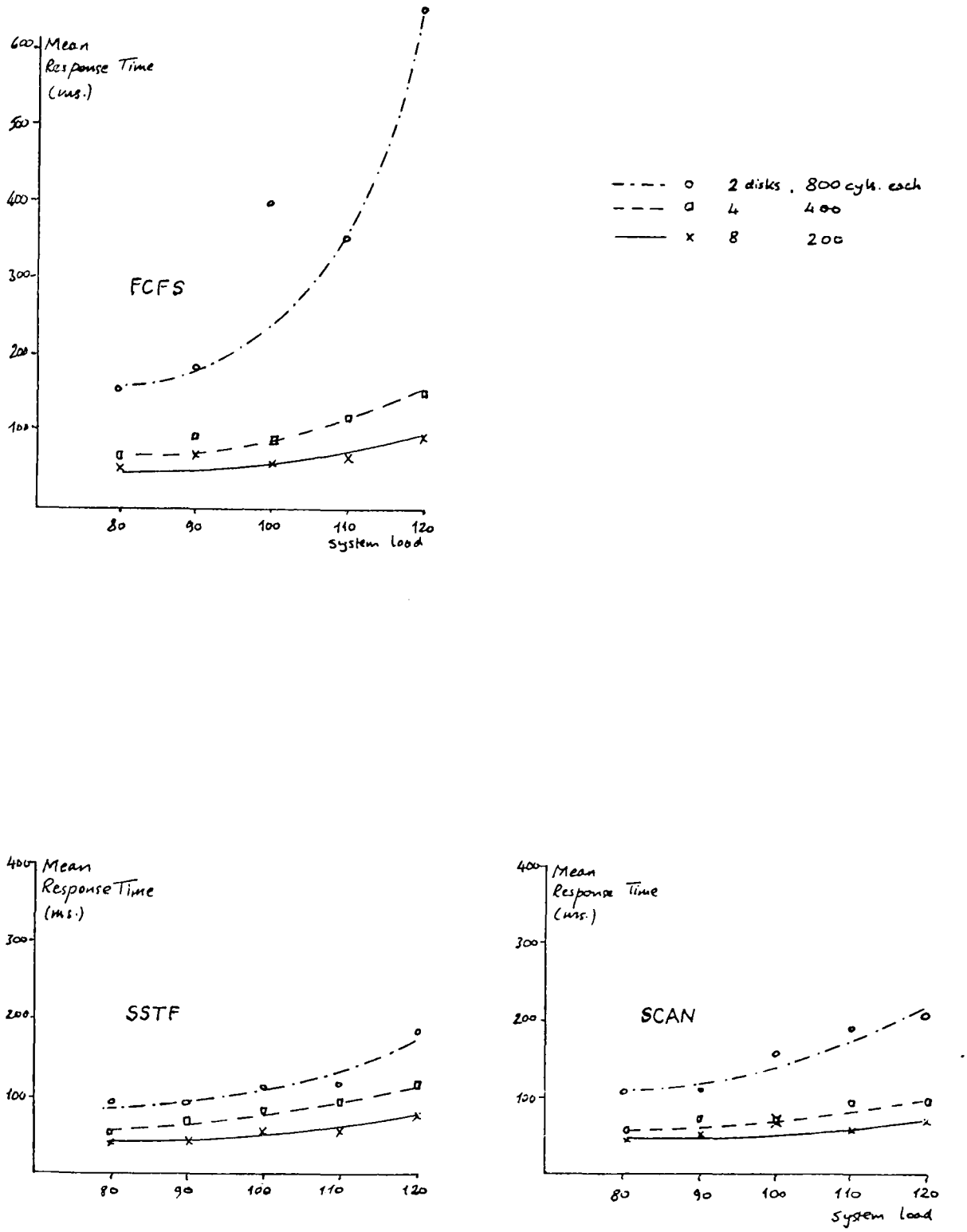


Figure 6.8 Mean response time comparisons for scheduler and disk size. (S/370 inputs; Non-Consecutive Layout.)

CHAPTER 7

Ranking of filestore factors for their effect on performance

7.1 Introduction

The investigation of request schedulers and disk capacity for two different systems showed that neither scheduling algorithm nor capacity per disk was a sufficiently influential factor in system or filestore performance to offer a simple rationale for design. Furthermore, these investigations do not permit any statement to be made about the ranking of these factors and others (e.g. layout) that seem to be important. In this chapter, we investigate the following set of filestore factors.

a) filestore layout
b) capacity per disk access
c) capacity per disk
d) scheduling algorithm

The experiments are designed for the application of Analysis of Variance techniques (ANOVA) to the results in order to produce a ranking of factors alone and in interaction with each other. This design is further described in section 7.3, and the statistical method is described in Appendix B. These simulation experiments are performed on the system model developed in Chapter 5.

7.2 The Factors

7.2.1 Filestore layout

In addition to modelling the Non-Consecutive layout, the Sequential layout is used as a second level of this factor. A third level, the Precessed Layout, is also simulated.

The Precessed Layout allows the introduction of a gap between logically consecutive records without any penalty of lost space [30] . This method of placement is designed to mitigate the rotational delay incurred during sequential accessing of a sequential layout - the "missed revolution" effect discussed in Chapter 4. The Precessed placement policy used in this experiment is shown in Figure 7.1. The filestore has 3 records per track, simulating the S/370 layout.

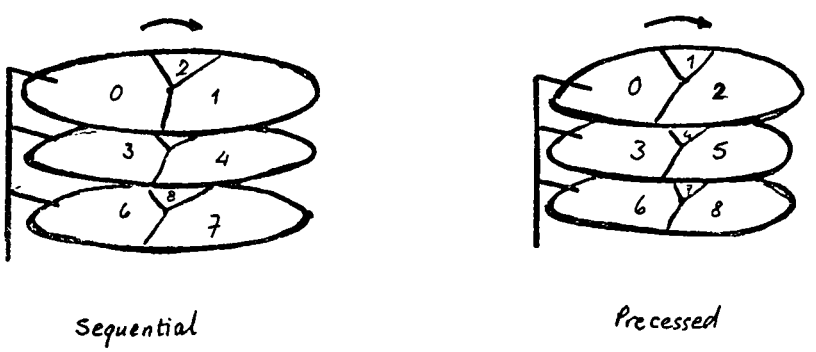


Figure 7.1 Sequential and Precessed Layouts

In this Precessed implementation, the gap between logically sequential records is effectively one-third of the revolution time. For the IBM 3330 disk in this system model, this time of 5.5ms exceeds the Lynch Period

which is fixed at 4ms. Thus the gap between logically sequential records exceeds the maximum runtime between requests in a file-i/o burst, and eliminates the "lost revolutions" originating from an inter-record gap of less than the Lynch Period. This practical scheme is simulated by the sequential accessing behaviour algorithm for individual processes, but with the sectors on each surface renumbered from (0, 1, 2) to (0, 2, 1) for the calculation of rotational delay.

7.2.2 Capacity per disk access

Three levels of capacity per disk access are represented by 1, 2 or 4 read/write heads per surface. These heads are placed on the same arm. The effect of the arrangement of Figure 7.2 is to create

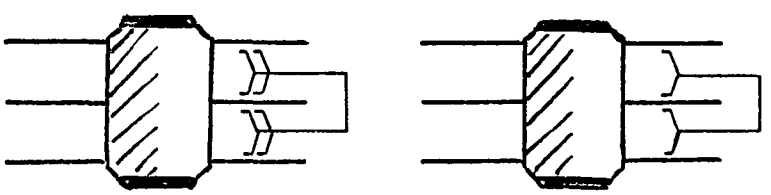


Figure 7.2 Illustration of capacity per disk-access

'logical' cylinders of twice the number of cylinders compared with a single 'physical' cylinder. When in position, all data on this 'logical' cylinder is accessible without further head movement. This is implemented in the model by mapping addresses generated from the range of physical cylinders on to the range of logical cylinders. When necessary, the heads are moved in unison. Little reduction in seek time may be expected since the number of cylinders traversed is only reduced by the number of heads in the assembly;

the likely benefit is an increased chance of selecting an address without incurring a seek delay.

7.2.3 Capacity per disk

Three levels of capacity per disk (i.e. track density) are simulated in a filestore size of 800 Mbytes and composition of IBM 3330-type disks. This is achieved by setting the number of cylinders per disk (equivalent to tracks per surface) to 200 (8 disks), 400 (4 disks) and 800 (2 disks). The minimum and maximum seek delays were as for the standard IBM 3330. The previous chapter indicated that fewer disks in the filestore were associated with longer service delays, but the effect on system performance was not significant by the method of analysis used.

7.2.4 Scheduling Algorithm

From the indications of performance given in Chapter 6, three schedulers are selected:

i) FCFS

ii) SCAN

iii) C-SCAN

FCFS seems well-suited to a Sequential Layout, and by implication a Precessed Layout also. Although mean response time increases with load, the indications are that the magnitude of this increase for 3330-style disks is modest if there are sufficient disks. SCAN and C-SCAN are two scanning variations which seem suited to a Non-Consecutive layout, even with its limited-length bursts of sequential accessing. Their

advantage is likely to be in situations of higher load with fewer, large-capacity disks.

7.2.5 Load variation

The simulated system load is varied from 80 to 130 processes in steps of 10. This small range is to facilitate graphical presentations of results, and to represent the load variation conditions encountered in the real system.

7.3 Experimental Plan

The experimental plan is an orthogonal design, i.e. all combinations of levels are used in the model runs, and an equal number of observations of each output variable are made for each factor arrangement. These results, when the model runs are independently seeded, are amenable to Analysis of Variance techniques for determining factor effects and interactions. This technique and its requirements of the experimental plan are described in Appendix B. The ANOVA analysis tests the Null Hypothesis that the factor by itself or in combination with other factors has no effect on the resultant output. The outputs with the factor at one value, or level, are compared with those outputs at another level. In simulation experiments, the normal fluctuation of the output is estimated by making several observations of the output. The Null Hypothesis is true if the outputs at factor level 1 do not differ from those at factor level 2 by more than the expected fluctuation. In statistical terms, this test is the F-ratio (see Appendix B) and the probability of the Null Hypothesis is expressed in accepted steps of 5%, 1% and 0.1%, these values being calculated from the F-ratio. That is, a probability of 1% implies that there is 1 chance in 100 that the Null Hypothesis is satisfied. These probability levels are expressed as * ("significant", 5%), ** ("very significant", 1%) and *** ("highly significant", 0.1%) representing increasing confidence in the existence of the indicated effect.

The factors and their levels are:

1. Layout - 3 levels ; Sequential, Precessed and Non-Consecutive.
2. Capacity per disk - 3 levels ; 2,4 and 8 disks in a 1600 Mbyte filestor
3. Capacity per disk access - 3 levels ; 1,2 and 4 heads per surface

4. Scheduling algorithm - 3 levels ; FCFS, SCAN and C-SCAN
5. Load - 6 levels ; 80 to 130 processes in steps of 10.
6. Replication - 3 levels ;

For each level of load, there are 81 combinations of the other factors excluding replication. This experimental set consists of 81 x 6 x 3 runs, i.e. 1458 runs. These results are averaged over each replication and used for graphical illustration of statistically indicated effects. The statistical analyses use the set of results for two load levels, a medium load (90 processes) and a heavy load (130 processes). Each analyzed set comprises 243 runs of the model. The analyzed outputs are work-unit throughput and filestore mean response time.

7.4 Conclusions

The ANOVA data show that layout is highly significant in system and filestore performance. Figure 7.3 and 7.4 show that the Non-Consecutive layout affects performance adversely; the Sequential and Precessed layouts show less sensitivity to disk capacity and have half the mean response time. System and filestore performance is penalized by large capacity disks, but the absolute effects are very small. The Non-Consecutive layout is most adversely affected.

Request scheduler and capacity per access (i.e. number of heads per surface) are not significant.

7.5 Results - Commentary on ANOVA tables

Tables 7.1 and 7.2 show the Analysis of Variance (ANOVA) evaluations of each factor for its effect on the observed performance indices at two load levels. For both indices, layout is highly significant, with a lesser rating for work throughput at the lower load. Disk capacity is also highly significant for filestore performance, and at the higher load for work throughput. The significant 2nd and 3rd order interactions of Table 7.1 are probably due to the strength of the single factors layout and disk capacity; these interactions are investigated further below. In Table 7.2, the 2nd order interaction at both loads between layout and disk capacity is highly significant. The other interactions lessen at the higher load.

Table 7.3 shows the ANOVA results for system performance when layout is treated separately. No significant factor emerges except disk capacity

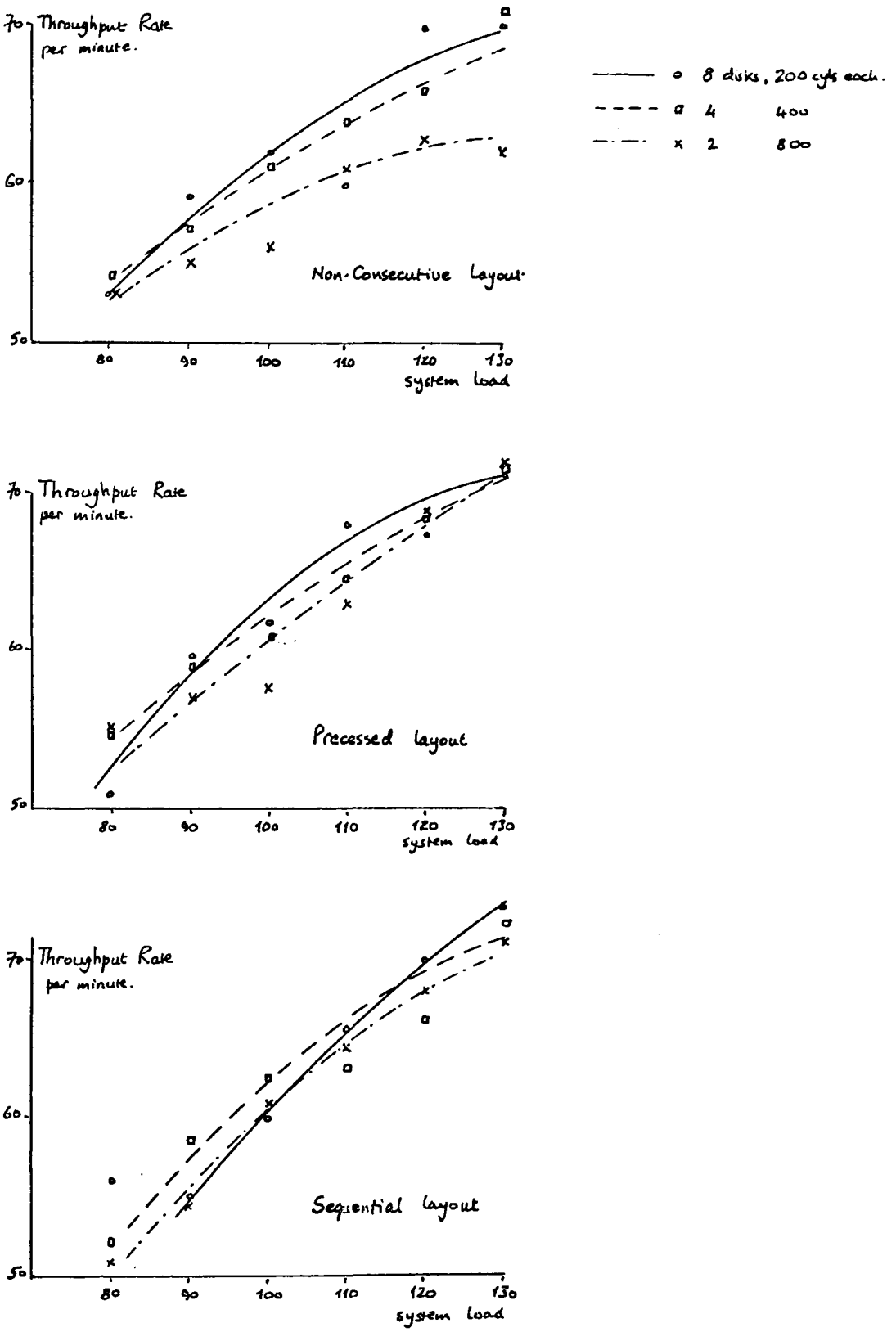


Figure 7.3 Throughput comparisons for layout and disk size.

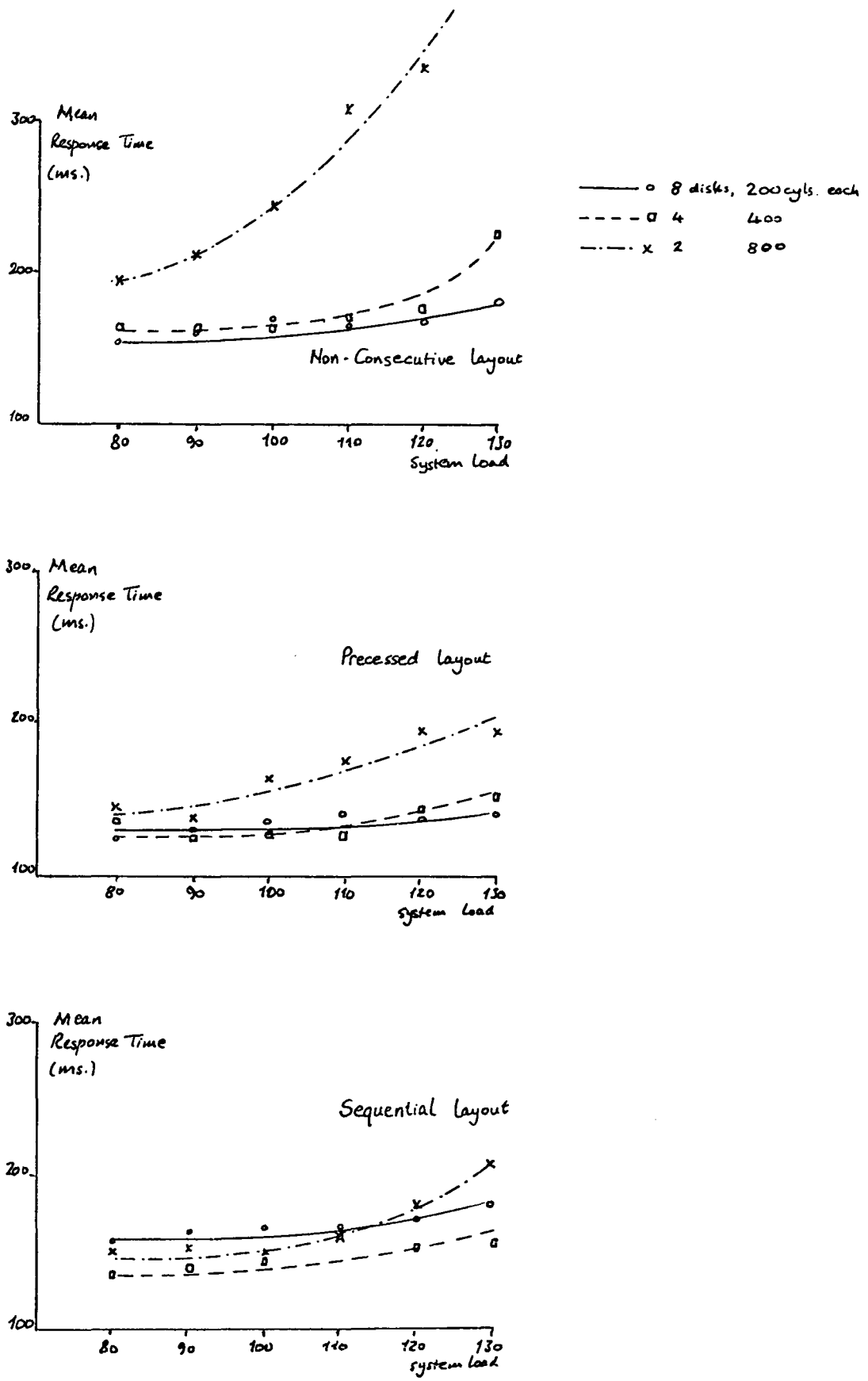


Figure 7.4 Mean response time comparisons for layout and disk size.

Source of Variation	Degrees of Freedom	F-ratio at load=90	Significance at load=90	F-Ratio at load=130	Significance at load=130
A	2	0.38		0.95	
B	2	0.03		8.95	***
A x B	4	2.09		0.97	
C	2	0.39		0.89	
A x C	4	1.00		1.83	
B x C	4	1.37		2.07	
A x B x C	8	2.64	**	2.41	*
D	2	4.89	**	23.6	***
A x D	4	1.46		1.63	
B x D	4	0.79		2.87	*
A x B x D	8	0.50		0.88	
C x D	4	1.41		0.56	
A x C x D	8	0.41		1.29	
B x C x D	8	1.14		0.78	
AxBxCxD	16	0.43		1.43	
R	2				
E	160				

A = Scheduler

B = Capacity/disk (track density)

C = Capacity/Access (number of heads)

D = Layout

R = Replication

E = Error (all interactions with R)

* "significant"

** "very significant"

*** "highly significant"

Table 7.1 Results of ANOVA for Work throughput rate

Source of Variation	Degree of Freedom	F-ratio at load=90	Significance at load=90	F-ratio at load=130	Significance at load=130
A	2	13.0	***	6.53	**
B	2	41.4	***	109	***
A x B	4	8.51	***	2.60	*
C	2	0.51		1.21	
A x C	4	1.15		1.01	
B x C	4	0.85		0.97	
AxBxC	8	0.61		2.09	*
D	2	77.6	***	93.4	***
A x D	4	3.75	**	3.27	*
B x D	4	7.82	***	18.8	***
AxBxD	8	3.63	***	0.77	
C x D	4	0.53		0.08	
AxCxD	8	0.46		2.30	*
BxCxD	8	0.36		1.20	
AxBxCxD	16	1.13		1.30	
R	2				
E	160				

A = Scheduler

B = Capacity/disk (track density)

C = Capacity /Access (heads/surface)

D = Layout

R = Replication

E = Error (all interactions with R)

Table 7.2 Results of ANOVA for Filestore Mean Response Time

at the higher load. The other factors involved with layout in Table 7.1 are not significant These results are illustrated in Figure 7.3. Precessed and Sequential Layouts differ little in their effect, but both have a work throughput higher by a few percent than the Non-Consecutive layout when it is most efficiently configured (see below). The absolute benefits of the smaller disks are small, but most marked for a Non-Consecutive layout; at medium-to-heavy loads, the improvement in system throughput is 10 - 15% for small disks (200 cylinders) and about 5% for medium disks (400 cylinders). The Non-Consecutive layout is most severely affected by large capacity disks.

For filestore performance with the layouts treated separately (Table 7.4), disk capacity is highly significant at both load levels. The 2nd order interaction between scheduler and disk capacity diminishes at the higher load. No other main effects or interactions are significant. Figure 7.4 shows mean response time against system load for all levels of factors disk capacity and layout. With Precessed and Sequential layouts, the mean response time is half that of the Non-Consecutive layout when efficiently configured. With all layouts, disk capacity is also highly significant. For the Non-Consecutive layout, the smaller capacities are better. The large disks (800 cylinders) give response times which increase rapidly with load; the medium (400 cylinder) and small (200 cylinder) disks are equal until heavier loads when the small disks give a faster response. The Precessed layout performs equally well with the Sequential layout for small and medium disks. Only at heavier loads does the smaller disk show a few percent improvement for the Precessed Layout.

	Source of Variation	Degree of Freedom	F-ratio at load=90	Significance at load=90	F-Ratio at load=130	Significance at load=130
I	A	2	2.69		0.71	
	B	2	0.68		3.11	*
	A x B	4	0.91		0.41	
	C	2	0.24		0.23	
	A x C	4	0.66		0.26	
	B x C	4	1.20		0.87	
	AxBxC	8	1.25		1.38	
	R	2				
	E	52				
II	A	2	0.61		1.17	
	B	2	1.00		3.29	*
	A x B	4	0.99		1.15	
	C	2	1.24		0.15	
	A x C	4	0.82		0.76	
	B x C	4	2.34		1.89	
	AxBxC	8	0.53		1.47	
	R	2				
	E	52				
III	A	2	0.13		2.26	
	B	2	0.08		8.03	***
	A x B	4	1.25		1.16	
	C	2	1.79		1.52	
	A x C	4	0.44		3.19	*
	B x C	4	0.46		0.95	
	AxBxC	8	1.71		2.42	*
	R	2				
	E	52				

Scheduler = = A
Capacity/disk = B
Capacity/Access = C
Replication = R
Error = E

Separate analyses of each level of layout:

I = Sequential
II = Precessed
III= Non-Consecutive

Table 7.3 ANOVA Results for Work throughput rate

	Source of Variation	Degree of Freedom	F-ratio at load=90	Significance at load=90	F-ratio at load=130	Significance at load=130
I	A	2	6.47	**	0.04	
	B	2	15.7	***	45.6	***
	A x B	4	2.53	*	0.35	
	C	2	1.61		0.83	
	A x C	4	1.19		0.31	
	B x C	4	1.38		2.08	
	AxBxC	8	0.68		0.30	
	R	2				
	E	52				
II	A	2	0.28		3.89	*
	B	2	13.4		37.2	***
	A x B	4	1.14	***	1.26	
	C	2	1.08		0.75	
	A x C	4	2.41		0.91	
	B x C	4	0.27		0.34	
	AxBxC	8	1.30		0.72	
	R	2				
	E	52				
III	A	2	7.76	**	5.01	*
	B	2	21.8	***	50.3	***
	A x B	4	6.77	***	1.53	
	C	2	0.16		0.37	
	A x C	4	0.48		2.19	
	B x C	4	0.28		1.05	
	AxBxC	8	1.10		1.83	
	R	2				
	E	52				

Table 7.4 ANOVA Results for Filestore Mean Response Time

A = Scheduler
B = Capacity/disk
C = Capacity/Access
R = Replication
E = Error

Separate analyses for each level of layout:

I = Sequential
II = Precessed
III = Non-Consecutive

CHAPTER 8

Investigation of Two Policies for using Disk Fast-Access Areas

8.1 Introduction

A disk configuration factor not considered in Chapter 7 is the availability of fixed heads over a small group of cylinders [22, 23] . This enables fast, zero seek-time accessing of data on this area. In this chapter, the system model is used to evaluate two strategies for using a fast-access area. The evaluated strategies are

(i) the Simple-Placement Policy

(ii) the Staging Policy

Both policies take advantage of known accessing patterns to anticipate sequential referencing. The Simple-Placement policy is to place frequently accessed data (catalogues and other files) on the fast-access area. The Staging Policy uses the fast-access area as a "staging" area, i.e. long sequences of records are buffered on the fast-access area in response to a demand for one record of the sequence.

These strategies are both suitable for the Sequential layout filestore. Both are also suitable for the Non-Consecutive layout if a buffer area not larger than one cylinder is staged since that layout is organized sequentially within cylinders but not necessarily so across cylinder boundaries.

8.2 Experimental Factors for the Fast-Access Area

8.2.1 The Simple-Placement Policy

Measurements of the S/360 filestore showed that 50% of the accesses to each disk were to the 1% of surface area containing the catalogue.

Inferences about accessing patterns on the S/370 filestore suggest that 40% of its accesses are made without incurring any seek delay. These are the bases for the Simple-Placement Policy, which is to place the catalogue (or other disk index) on the fast-access area. Contemporary disks with this facility typically have 0.5% of their area under fixed heads. Our interest is a disk with a fast-access area of at least 1% of its surface.

To model the Simple-Placement Policy, the request generating algorithm is modified to select the fast-access area when generating a request to the 'central' area. This parameter is α, as described in Chapter 5. A variable, additional fraction of requests is also directed to this area to simulate the placement of other frequently-accessed data. The address generating algorithm becomes:

from current cylinder i,

select fast-access area with probability $\alpha + S$

select current cylinder with probability $(\beta + S) - (\alpha + S)$

move to cylinder j with probability $1 - (\beta + S)$

Cylinder j is selected such that $|i - j|$ is according to layout type, $(\alpha + S) < (\beta + S) < 1.0$; S is the additional fraction of requests satisfied from the fast-access area (see discussion in section 8.2.3) and any move is selected according to layout, as described in section 5.5.1 and 5.5.2 .

8.2.2 The Staging Policy

The alternative strategy of Staging is to use the fast-access area as a buffer between slower-access storage and the main store. Sequentiality of reference on both layouts indicates that one or more successor requests will be made during a file-i/o burst. The Staging Policy transfers complete cylinders of data to and from the staging area. A cylinder is staged-in when one of its records is referenced (and the cylinder is not already in the fast-access area). We assume that all records on the addressed cylinder may be staged-in for the cost of a seek delay plus one rotation. The cylinder to be staged-out is selected by the Least Recently Used algorithm. In the absence of any other data, we assume a probability of 0.5 that this cylinder has been written to, and so requires saving to its permanent residence area. The "write home" cost is the seek delay to the cylinder position plus one rotation.

8.2.3 Size for the fast-access area.

Various sizes of the fast-access area are simulated by varying the number of requests it receives under the Simple-Placement policy, and by varying the number of fast-access cylinders in the associative table used to implement the Staging Policy. The levels of the size factor are chosen to keep these interpretations of size equivalent. Gray reported 50% of accesses to 1% by number of the cylinders of the sequentially organized filestore, and his measurements show peaks of accesses on certain cylinders (see Figure 2.1). However, without good data on file referencing and file sizes, we analyze the uniform-access situation in which the fraction of requests satisfied from the fast-access area is linear with the size of the area. This sets a lower bound on the number of requests so satisfied, since it is likely that much higher "hit" rates are possible.

For both Sequential and Non-Consecutive layouts, the spatial accessing algorithms fix α to be the fraction of requests to central cylinders. We assume that this fraction at least is satisfied from the fast-access area under the Simple Placement policy. In addition, up to 10% of other accesses are considered to be satisfied from this area. For the Sequential layout, up to 70% of all accesses may be satisfied; for the Non-Consecutive layout, up to 50%. The smallest non-zero area is simulated as satisfying an additional 2.5% of all requests over the value of α, as shown in Table 8.1.

	Simple-Placement	Staging
level 1	0 of accesses	0% of cylinders
2	α + 0.025 of accesses	1% of cylinders
3	α + 0.050 of accesses	2% of cylinders
4	α + 0.100 of accesses	4% of cylinders

Table 8.1 Interpretations of fast-access area size.

We equate the first non-zero level of size under Simple-Placement with 1% of available cylinders, the representation simulated under the Staging policy. The maximum size under Staging is 4% of available cylinders, and this corresponds to the conservative "hit" rate of (α + 0.1) under the Simple-Placement policy.

8.3 Other Experimental Factors

Layout schemes are expected to influence the choice of strategy, so this factor is also retained though at two levels only - Sequential and Non-Consecutive. Increased capacity per disk-access (i.e. more heads per surface) was found to have no significance in the configuration specified for the previous experiment. However, this factor is retained to test for any interaction with the fast-access factors of policy and size. The number of disks in the filestore is fixed to eliminate variation from this source. Request scheduler is not significant, so FCFS is selected for ease of implementation. As in the previous experiment, the system load is varied over a small range to enable graphical presentation of statistical effects.

8.4 The Experimental Plan

The experimental plan for ANOVA called for an orthogonal design of factor levels i.e. all combinations of levels are employed and and equal number of observations are made for each set of levels. Each combination of levels is replicated three times with independent seeding of each run of the simulation. Further description and references about the experimental technique and method of analysis are given in Appendix B.

The factors and their levels are:

1. Layout - 2 levels; Sequential and Non-Consecutive.
2. Use policy - 2 levels; Simple Placement and Staging
3. Fast-access area size - 4 levels; 0%, 1%, 2% and 4% of available cylinders.
4. Capacity per disk access - 3 levels; 1, 2 and 4 heads per surface
5. Load - 6 levels; 80 to 130 processes in steps of 10
6. Replication - 3 levels;

For each level of system load there are 48 combinations of the other levels excluding replication. This experimental set consists of 48 x 6 x 3 runs, i.e. 864 runs. These results are averaged over each replication and used for graphical illustration of statistically indicated effects. The statistical analyses use the sets of results for two load levels, a medium load (90 process) and a heavy load (130 processes). Each analyzed set comprises 144 runs of the model. The analyzed outputs are work-unit throughput rate and filestore mean response time.

8.5 Conclusions

Figure 8.1 and 8.2 show that Simple-Placement and Staging policies perform equally on a Sequential layout, but that Simple-Placement provides a faster response from the filestore and more system throughput on the Non-Consecutive layout. Size of fast-access area is not consistently significant (as indicated in ANOVA Tables 8.4 and 8.5). Figure 8.3 shows the weak interaction of size with service policy on the Non-Consecutive layout.

The Non-Consecutive layout is again a poor choice for filestore performance (Figure 8.2); for this layout, the Simple-Placement policy is to be preferred. However, the Sequential layout gives shorter response times and a slight throughput improvement under both fast-access area strategies. The choice of strategy for a Sequential layout is between performing data placement manually or implementing Staging as a self-regulating policy.

Informal comparisons with the results of Chapter 7 show that a fast-access area is an almost unqualified benefit; system throughput is increased on both layouts by 20 - 40% (Figures 8.1 and 7.3). For the Sequential layout, mean response time is halved and increases linearly with load even with a FCFS request scheduler which is normally associated with increasingly long response times as load increases.

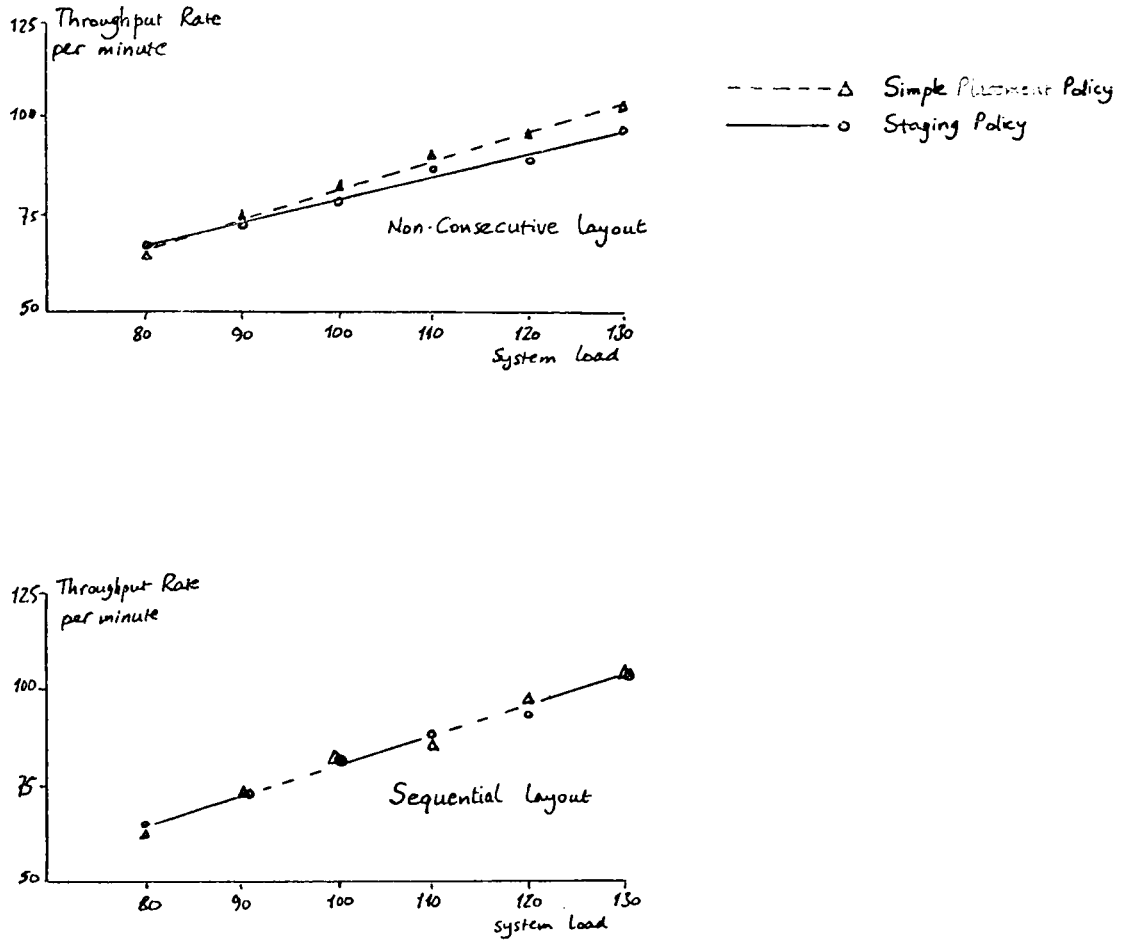


Figure 8.1 Throughput comparisons for layout and service policy.

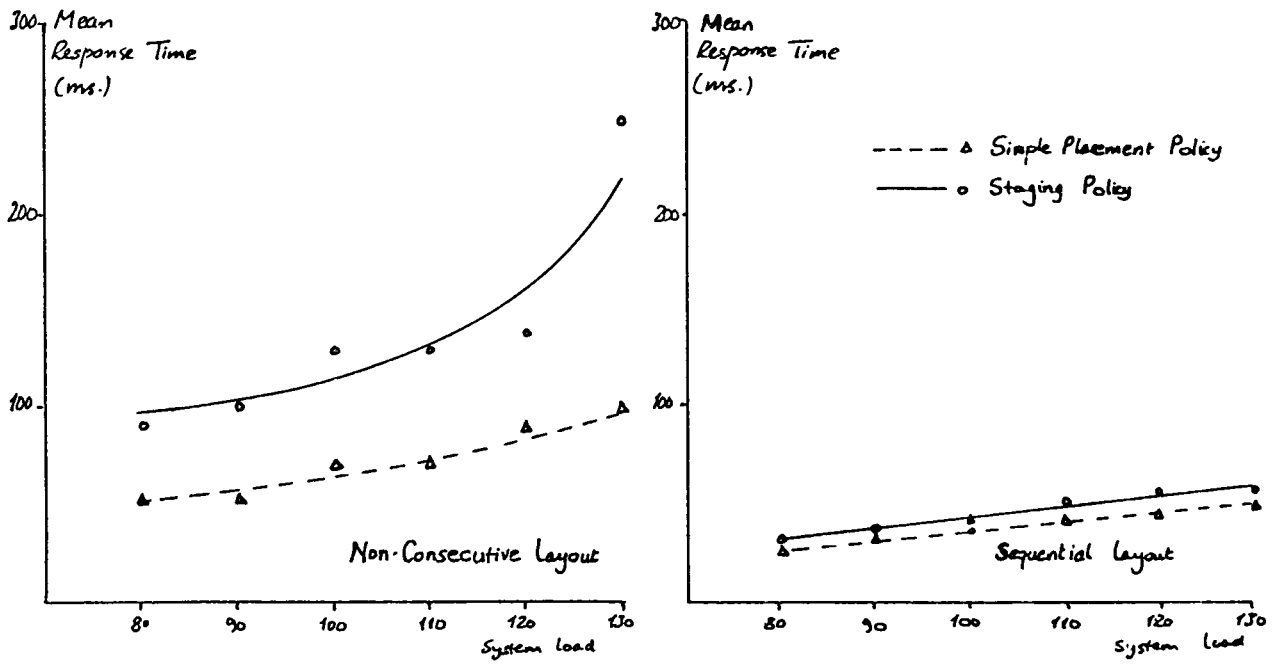


Figure 8.2 Mean response time comparisons for layout and service policy.

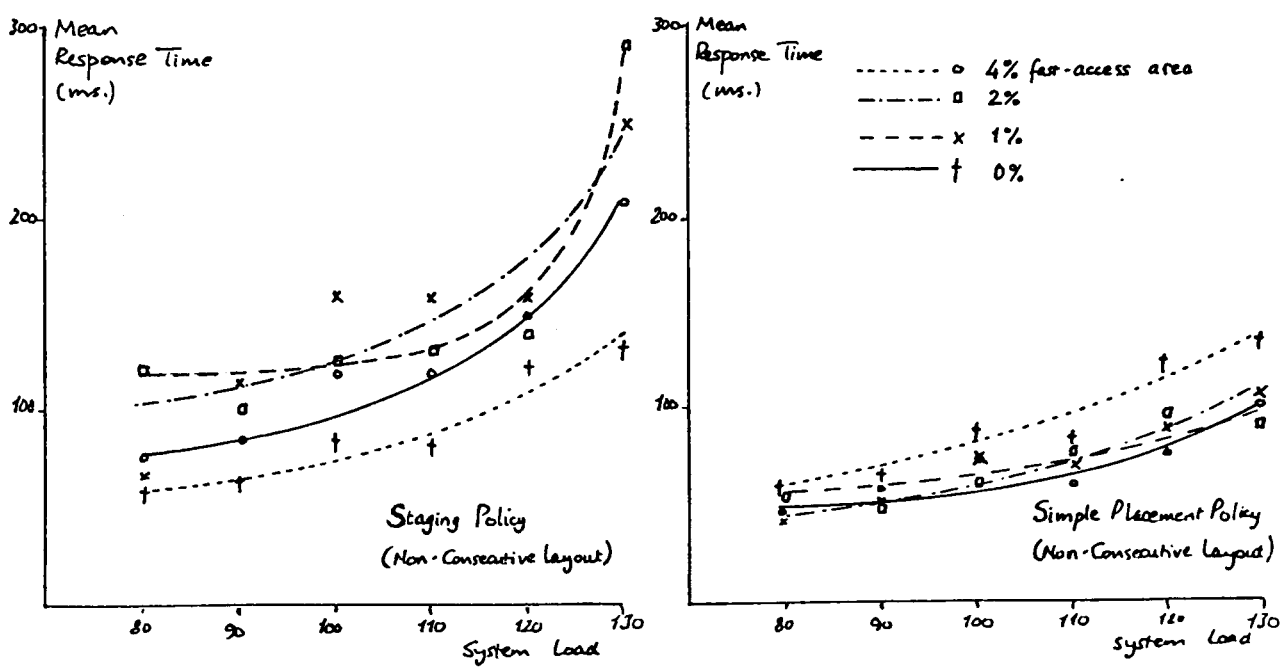


Figure 8.3 Effect of size and service policy for the Non-Consecutive Layout.

8.6 Results

8.6.1 Commentary on ANOVA tables

Tables 8.2 and 8.3 show the ANOVA analyses of each factor at two system load levels. Layout dominates in its influence on system and filestore performances. The work throughput (Table 8.2) is only sensitive to layout at the higher load level, but mean response time (Table 8.3) is affected by layout and service strategy at both load levels. The significant interactions all contain the strong factors.

The analyses are repeated in Tables 8.4 and 8.5 where layout is treated separately. The former layout-dominated result for work throughput now shows that the service strategy is highly significant at the higher load for the Non-Consecutive layout. There is an absence of other effects, and Figure 8.1 confirms the equivalent nature of both service policies for system throughput, indicating that the Staging policy adjusts well to the spatially contiguous bursts of the Sequential layout. Size of fast access area and capacity per disk access are not significant.

Source of Variation	Degree of Freedom	F-ratio at load=90	Significance at load=90	F-ratio at load=130	Significance at load=130
A	1	0.00		14.1	***
B	2	0.48		2.09	
A x B	2	0.73		0.25	
C	3	2.04		0.90	
A x C	3	0.84		1.35	
B x C	6	1.18		1.17	
AxBxC	6	1.02		0.36	
D	1	0.19		4.36	*
A x D	1	0.00		4.20	*
B x D	2	0.32		1.03	
AxBxD	2	0.32		0.54	
C x D	3	0.53		0.63	
AxCxD	3	0.24		2.05	
BxCxD	6	1.67		0.39	
AxBxCxD	6	0.66		0.63	
R	2				
E	94				

Table 8.2 Results of ANOVA for Work Throughput Rate

A = Layout

B = Capacity/access (heads/surface)

C = Fast - area size

D = Fast - area service policy

R = Replication

E = Error (all interactions with R)

Source of Variation	Degrees of Freedom	F-ratio load=90	Significance load=90	F-ratio load=130	Significance load=130
A	1	69.7	***	130.	***
B	2	3.76	*	0.09	
A x B	2	6.63	**	1.14	
C	3	0.75		1.94	
A x C	3	2.94	*	1.50	
B x C	6	2.12		1.48	
AxBxC	6	1.01		1.00	
D	1	23.2	***	40.2	***
A x D	1	16.0	***	27.4	***
B x D	2	0.48		1.36	
AxBxD	2	0.78		0.64	
C x D	3	3.62	*	5.28	**
AxCxD	3	2.75	*	4.17	**
BxCxD	6	0.98		0.70	
AxBxCxD	6	1.26		0.46	
R	2				
E	94				

Table 8.3 Results of Anova for Mean Response Time

A = Layout

B = Capacity/access (heads/surface)

C = Fast area size

D = Fast area service policy

R = Replication

E = Error (all interactions with R)

	Source of Variation	Degrees of Freedom	F-ratio load=90	Significance load=90	F-ratio load=130	Significance load=130
I	A	2	0.91		1.88	
	B	3	0.52		0.66	
	A x B	6	0.98		0.85	
	C	1	0.07		0.00	
	A x C	2	0.49		1.57	
	B x C	3	0.43		0.61	
	AxBxC	6	0.65		0.58	
	R	2				
	E	46				
II	A	2	0.04		0.52	
	B	3	2.99	**	1.59	
	A x B	6	1.25		0.70	
	C	1	0.13		8.44	***
	A x C	2	0.00		0.05	
	B x C	3	0.30		2.06	
	AxBxC	6	2.00		0.45	
	R	2				
	E	46				

Table 8.4 Results of ANOVA for Work Throughput Rate

A = Capacity/Access (heads/surface)

B = Fast-area size

C = Fast-area service policy

R = Replication

E = Error (all interactions with R)

Separate analyses for each level of layout:-

I = Sequential

II - Non-Consecutive

	Source of Variation	Degrees of Freedom	F-ratio load=90	Significance load=90	F-ratio load=130	Significance load=130
I	A	2	2.30		4.38	*
	B	3	8.18	***	0.16	
	A x B	6	8.82	***	1.13	
	C	1	1.30		2.96	
	A x C	2	1.32		0.68	
	B x C	3	0.20		0.83	
	AxBxC	6	0.28		0.44	
	R	2				
	E	46				
II	A	2	5.61		0.19	
	B	3	0.90		1.96	
	A x B	6	0.49		1.30	
	C	1	22.2	***	38.9	***
	A x C	2	0.53		1.08	
	B x C	3	3.62	*	5.39	**
	AxBxC	6	1.25		0.62	
	R	2				
	E	46				

Table 8.5 Results of ANOVA for Mean Response Time

A = Capacity/Access (head/surface)

B = Fast-area size

C = Fast-area service algorithm

R = Replication

E = Error (all interactions with R)

Separate analyses for each level of layout:

I = Sequential

II = Non-Consecutive

Table 8.5 shows that service policy has a highly significant effect on filestore performance for a Non-Consecutive layout. Figure 8.2 shows that the Staging policy on this layout has much longer mean response times than the Simple-Placement policy. For the Sequential layout, Table 8.5 shows that size is highly significant at the low load when it also interacts strongly with capacity per disk-access. Figure 8.3 explores this result, showing that increased size merely mitigates the poor performance of Staging on the Non-Consecutive layout.

For the Sequential layout, the mean response time increases linearly with load, the Staging policy being consistently a few percent slower than the Simple-Placement policy (Figure 8.2). Mean response time for the Sequential layout is at least half that for the Non-Consecutive layout.

8.6.2 Commentary on Results of Chapters 7 and 8.

Figures 7.3 and 7.4 of Chapter 7 and 8.1 and 8.2 of Chapter 8 show filestore performance under various levels of significant factors with and without a fast-access area. The Simple-Placement policy and a fast-access area increase work throughput by 20% to 40%. The reduction in mean response time for a Sequential layout is unexpectedly large; the very long service times expected under FCFS at higher loads are eliminated. Mean response time for a Non-Consecutive layout does not reflect the throughput improvement. It is probable that a reduction in the variance rather than the mean takes place. Overall, the Sequential layout and FCFS filestore is twice as fast with a fast-access area.

CHAPTER 9

Potential Performance from a Novel Filestore-Controller

9.1 Introduction

The filestore is adapted with a special form of disk controller equipped with a cache store. The purpose of the controller is to reduce the average access time of the filestore by using a rapid-access store, or cache, under a Simple-Placement policy. This extension of filestore capability was foreshadowed in Chapter 8 where a fast-access area attached to each disk resulted in considerable improvement of filestore and system performance under certain conditions. In [32] , McGregor, Thomson, Dawson and Jones describe their microprocessor implementation of the controller. This author's contribution is an assessment of the improved filestore.

The increased effectiveness of a cache-equipped filestore is represented as a performance improvement which is an input to the model of the filestore and its environment. Monitoring the behaviour of the work-unit throughput rate and response time achieves the aims of assessment. These aims are:

i) to explore the limits of system performance (work-unit throughput) with a fast filestore.

ii) to determine the necessary filestore performance improvement to release system performance from any filestore constraints.

9.2 Performance Improvement Factor

The Performance Improvement Factor (PIF) is defined in relation to the basic time-delay characteristics of a disk filestore. These characteristics

are seek-time, rotational delay and record transmission time. A Performance Improvement Factor of 1 represents the currently available filestore without any fast-access cache to shorten its service times. The performance gained by use of a cache store is represented by reducing the basic time delay characteristics. Halving these times is represented by a PIF of 2, and halving again by a PIF of 4, and so on. It is convenient to continue using the disk filestore model, but for the modelling of Performance Improvement Factor this is not essential.

9.3 Experimental Plan

Input data from both S/360 and S/370 recreates the filestore environment of these systems. The layouts and configurations are as described in Chapter 5, viz .

S/360 - Sequential layout ; eight IBM 2314 disks (240 Mbytes)

S/370 - Non-Consecutive layout ; four IBM 3330,11 disks (800 Mbytes)

Five sets of duplicated runs are made for each modelled system. Each set has its filestore performance improved by a factor of 2 over the previous run. Record transfer speed is increased until the limit of channel speed. The choice of load is determined by the search for maximum work throughput. The system load is increased to overcome the limiting effect of non-file-i/o on file load.

This model extrapolates from the well characterized situations used for the more precise estimates presented in Chapters 7 and 8. The aim of the experiment is to explore the effect of the new controller on existing systems.

9.4 Conclusions

The potential benefits from a simple extension of filestore capabilities are considerable. The filestore equipped with a fast-access area as in Chapter 8 has a Performance Improvement Factor of 2, and that with immediately realizable technology. A cache-equipped filestore with a PIF of 4 reduces mean response time by an order of magnitude and increases system performance up to four-fold before non-file-store limiting conditions are reached. A PIF of 6 or over has spare work capacity at lower load but the mean response time is reduced by an order of magnitude. The work throughput could be increased by a factor of ten and filestore performance would still be faster than contemporary systems.

The limits of work throughput are much higher than achieved with conventional multi-user general-purpose systems, even with modest PIFs of 6 or 8. However, the effect of processor contention ~~on~~ by the system load necessary to achieve the high throughput is unknown. Also, for concurrent running of many processes, a very large main store is required. The obvious application of the new filestore is to very large data bases and existing time-sharing systems where reduced response times would be the principal benefit. A possible new application is the use of such a filestore as a node in a small network of general-purpose systems or in a large network of single-user systems.

9.5 Results

Figures 9.1 and 9.2 show the predictions of work throughput and mean response time against the Performance Improvement Factor. In both systems, very large system loads are able to run concurrently, though at the expense of very long mean response times for low PIFs. The Figures are marked to show the transition from work throughput limited by the applied load to that limited by the performance of the filestore. This is calculated as follows:

a) for Figure 9.1 , at PIF = 8

(i) ratio of system load 100/60 = 1.67

ratio of work rates 75/45 = 1.67

⇒ work rate improvement is due to increased load.

(ii) ratio of system load 300/200 = 1.50

ratio of work rates 185/140 = 1.30

less improvement than expected from increased load therefore

⇒ the filestore is limiting the work rate.

b) for Figure 9.2 , at PIF = 8

(i) ratio of system load 200/100 = 2.0

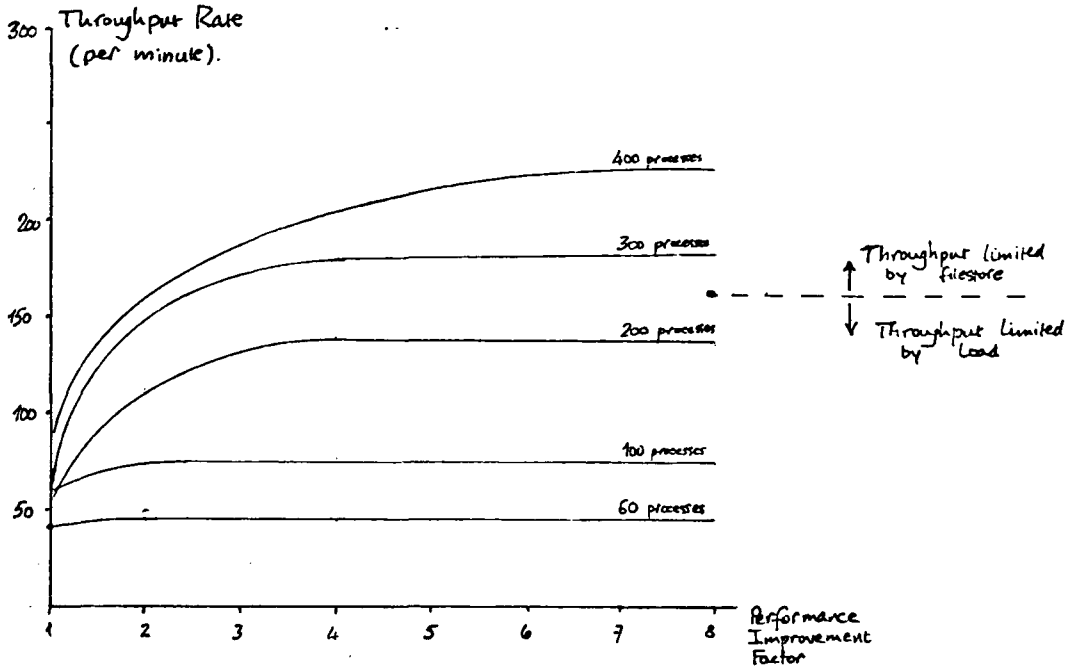
ratio of work rates 155/80 = 2.0

⇒ work rate improvement is due to increased load.

(ii) ratio of system load 600/400 = 1.50

ratio of work rates 380/285 = 1.30

⇒ filestore is limiting the work rate increase.



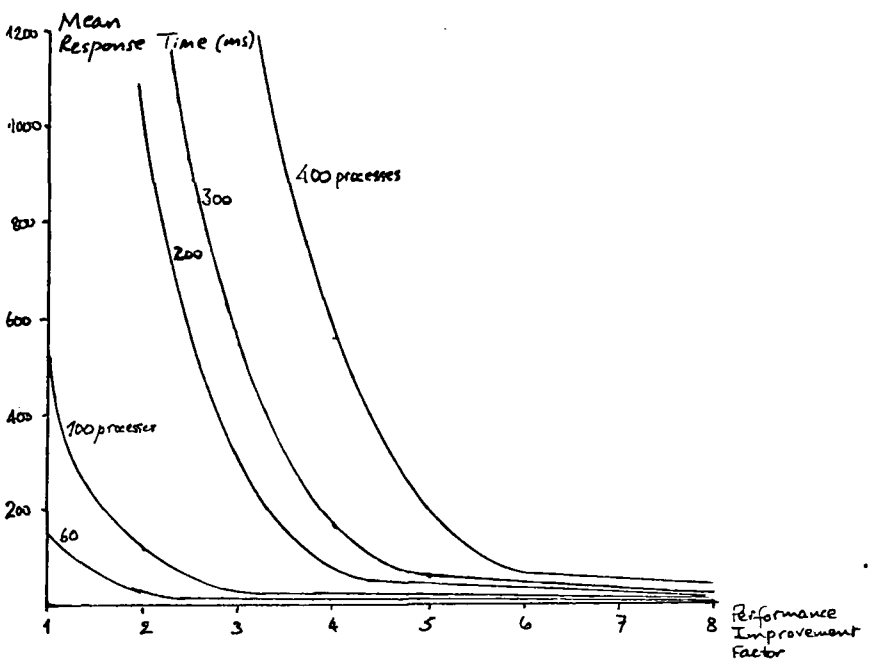
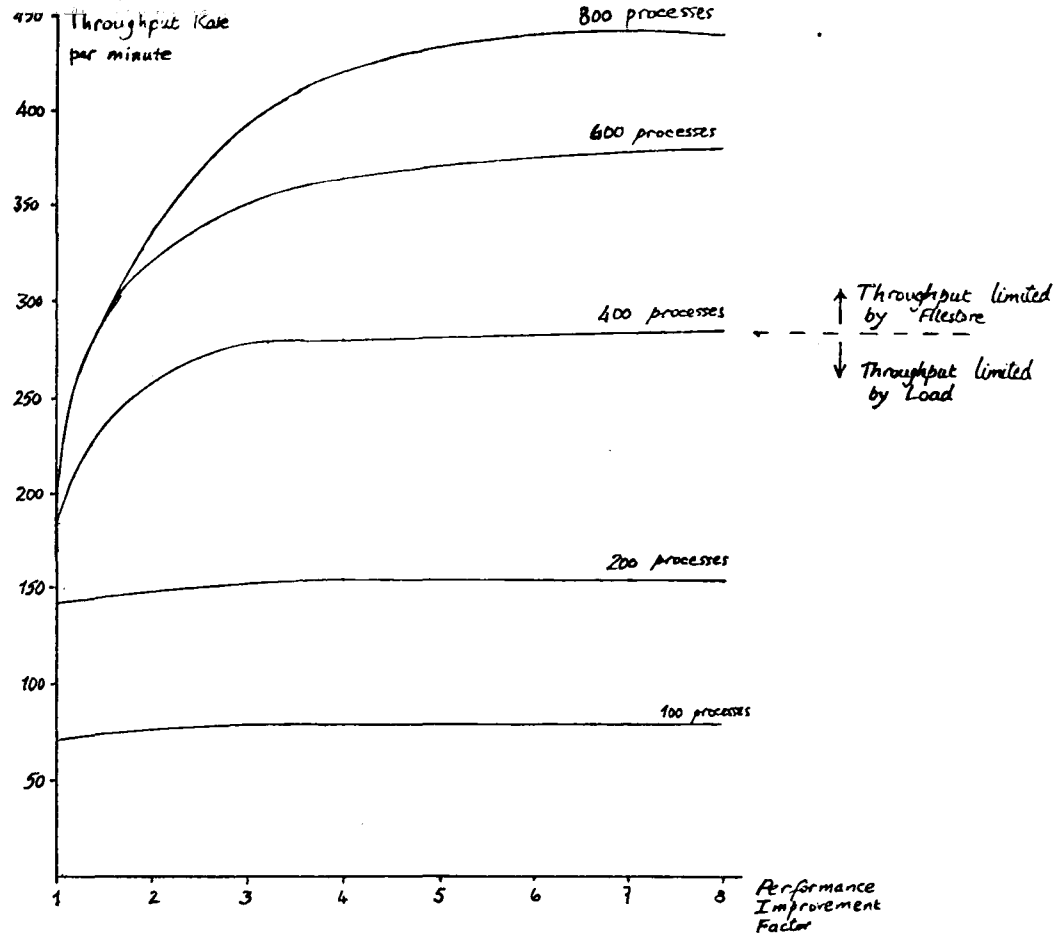


Figure 9.1 Performance limits of cache-equipped filestore.
(S/360 inputs; Sequential Layout; 8 disks.)



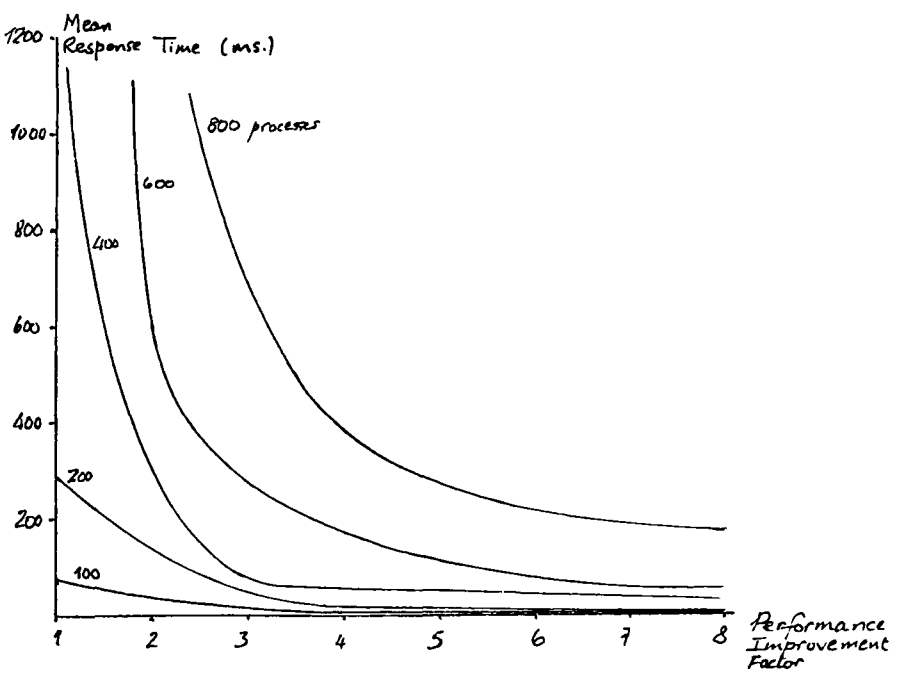


Figure 9.2 Performance limits of cache-equipped filestore.
(S/370 inputs; Non-Consecutive Layout; 4 disks.)

If the real systems could support more load, more work throughput is available at the expense of filestore response time. From Figure 9.2, a modest PIF of 4 enables a four-fold increase in system load and work rate, with mean response time remaining at the levels of current, unimproved systems. A PIF of 6 or over offers a very fast response time of about 5ms at current system loads of 60 to 100 processes, and with a considerable reserve of work capability. This reserve could be realized up to the limit of the filestore with mean response times (at 40ms, including waiting) still less than current systems (at 60 to 150ms).

Comparisons with Figures 8.1, 8.2 and 9.2 show that the fast-access equipped, sequentially formatted filestore has a Performance Improvement Factor of 2. Its improvement over the equivalent filestore without a fast-access area (Figures 7.3 and 7.4) is a factor of 1.5 in work throughput, and a factor of 2 in mean response time.

CHAPTER 10

Summary and Conclusions

Measurements of two large time-sharing systems have been presented. These data are separated by three years, and are from two different versions of the Michigan Terminal System. Many similarities exist between user behaviour patterns on these traces and patterns reported by Jalics and Lynch. Processes perform i/o in bursts of short mean length but very large variance. These burst lengths have a characteristic distribution that is well fitted by two straight lines. Run-time per process between i/o events has also been characterized. For file-i/o bursts, the spatial intention of a burst from an individual process is sequential. The filestore layout determines whether the sequential intent is mapped on to sequential physical addresses.

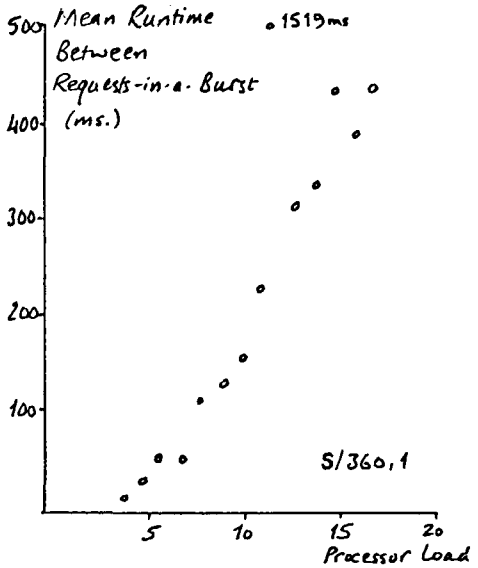
A method for evaluating a filestore in its operating system environment has been developed. The measured patterns of individual process behaviour enable the specification of temporal and spatial load in the environment. Load representation by these behaviour patterns differs considerably from previous simulation models, especially with respect to filestore accessing habits in time and space. A simulation model incorporating this load representation has been established and validated by checking its predictions against actual system measurements for a range of different situations. The methodology encapsulates studies of many sets of trace data. It is flexible in its application to other systems. It requires software trace facilities for accurate measurement of process activity. Future systems may be modelled by extrapolation of time costs, since it is reasonable to assume the future validity of these behaviour patterns.

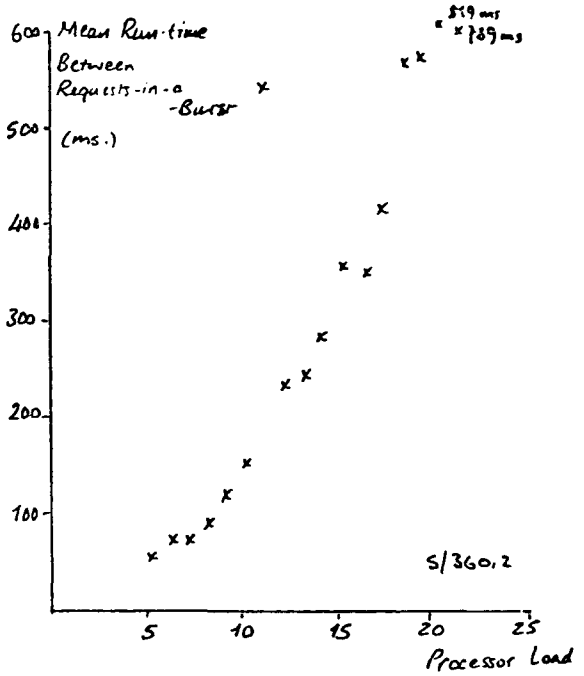
The model has been used to analyze various disk filestore configurations within contemporary environments. It has also been successfully used as a design tool in the investigation of new systems architecture. In contemporary systems, the absolute effects of disk request scheduling are negligible. Statistically, file organization and disk capacity are the significant factors most affecting current system performance. Disks with fast-access areas can greatly improve performance by taking advantage of disk accessing patterns. A new design, equipping the filestore with a cache store, offers considerable performance improvements even with current-technology disks.

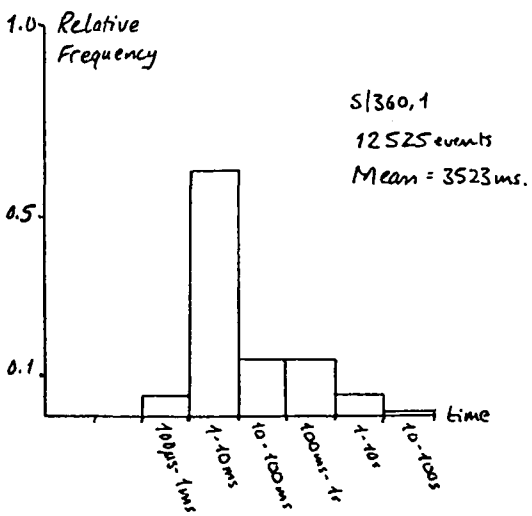
This work contributes by measurement and by the abstraction of salient characteristics to the development of the engineering study of computer systems. The proposed methodology contributes to the improvement of computer system models for reliable predictions. Further work is required to establish the engineering rules and limits of this methodology.

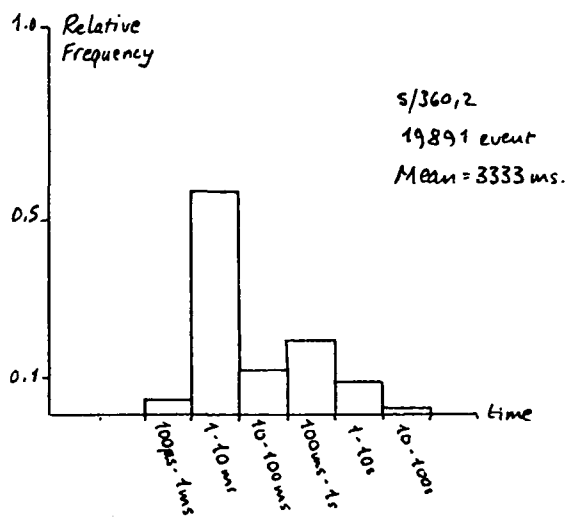
Appendix A Measurements not given in the text.

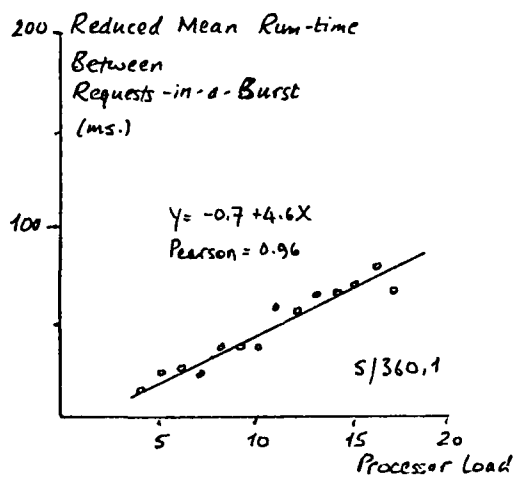
1. Run-time distributions.
2. Burst length distributions.
3. Waiting time distributions.











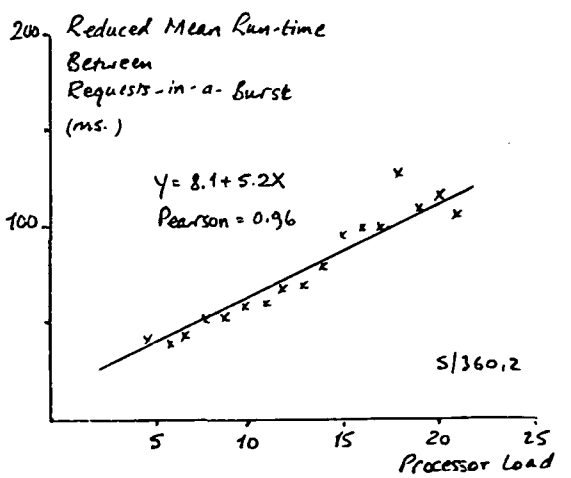


Figure A1 Run-time measures for terminal-i/o. S/360

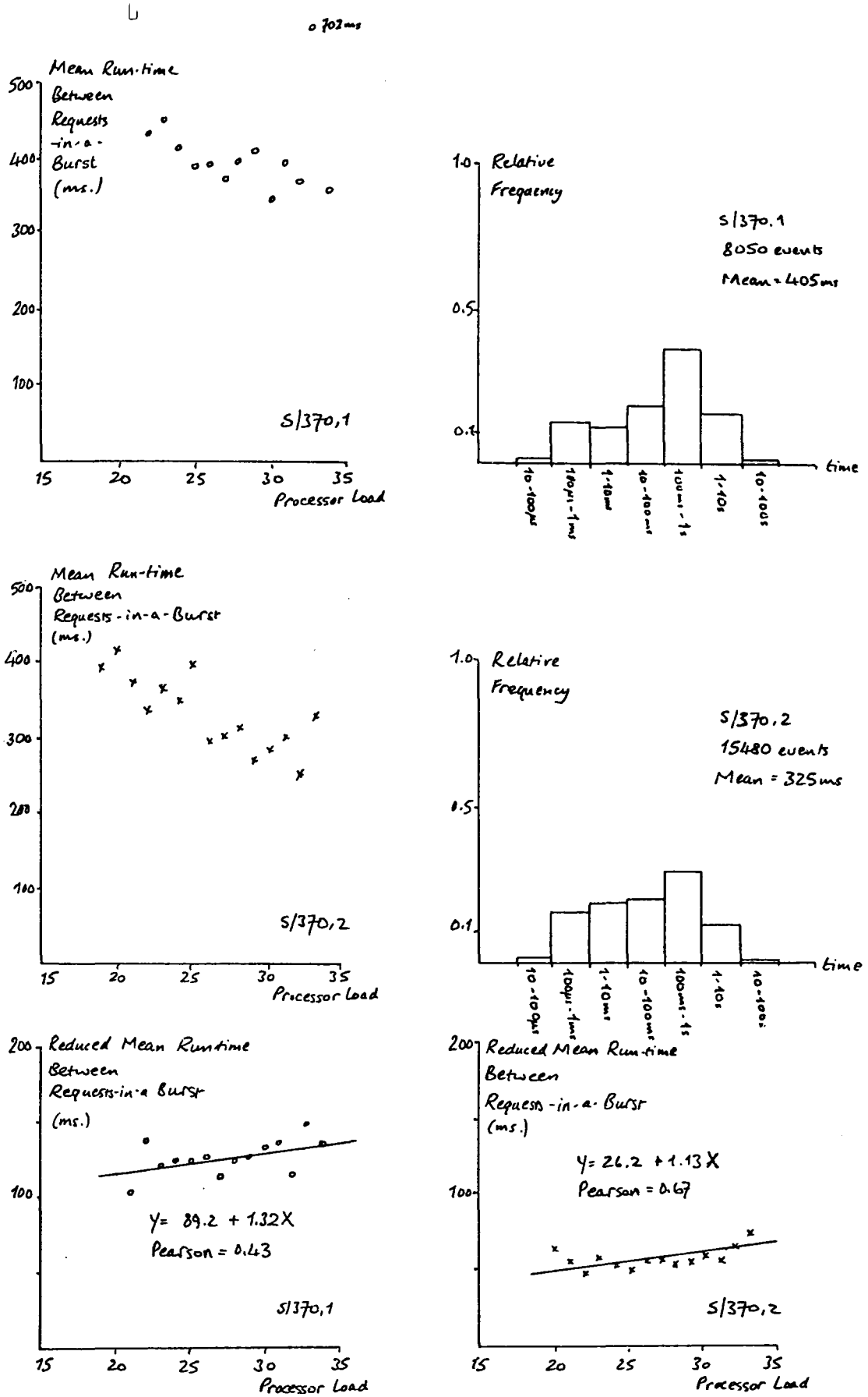
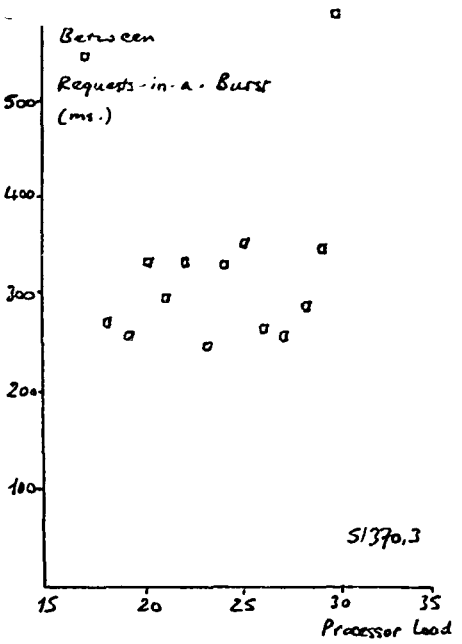
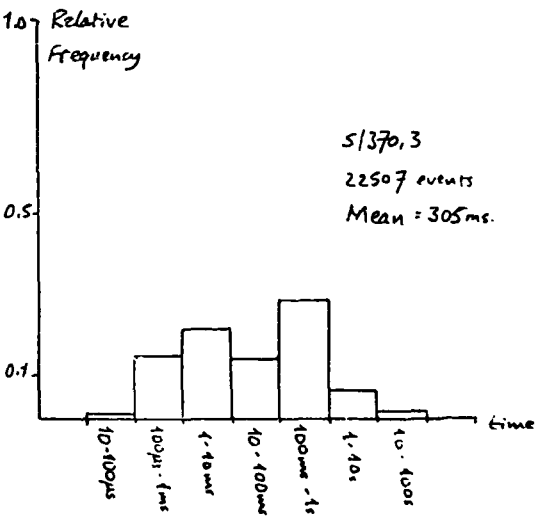


Figure A2. Run-time measures for terminal-i/o. S/370.





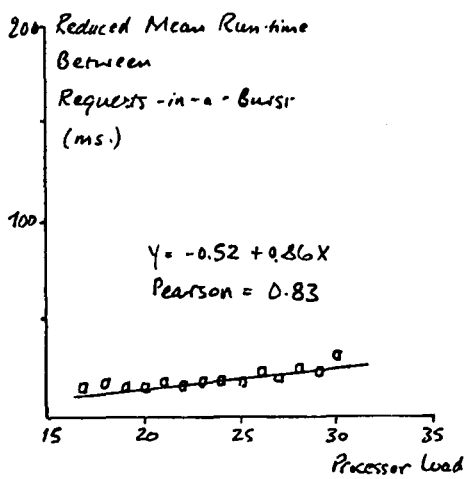


Figure A2. continued.

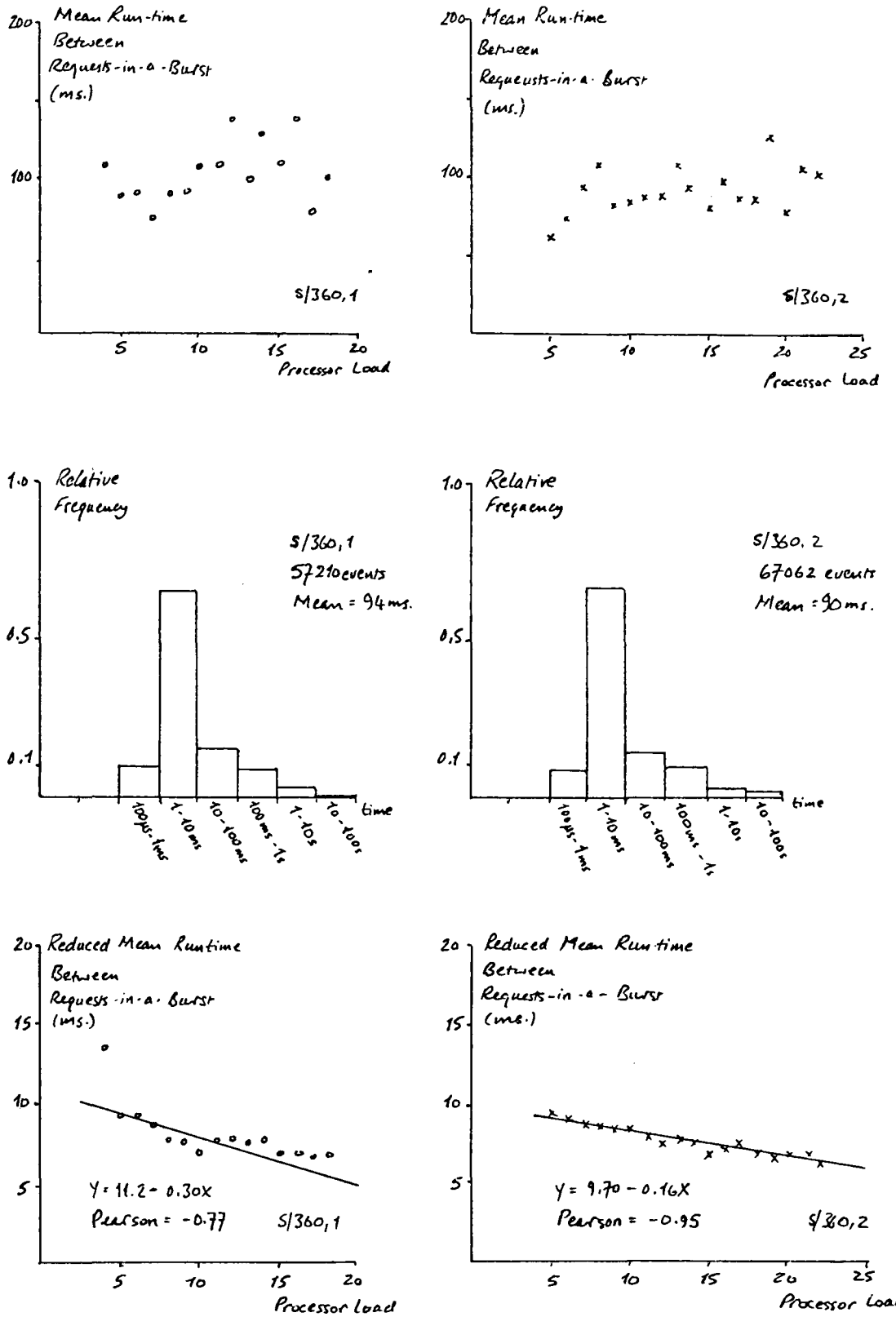
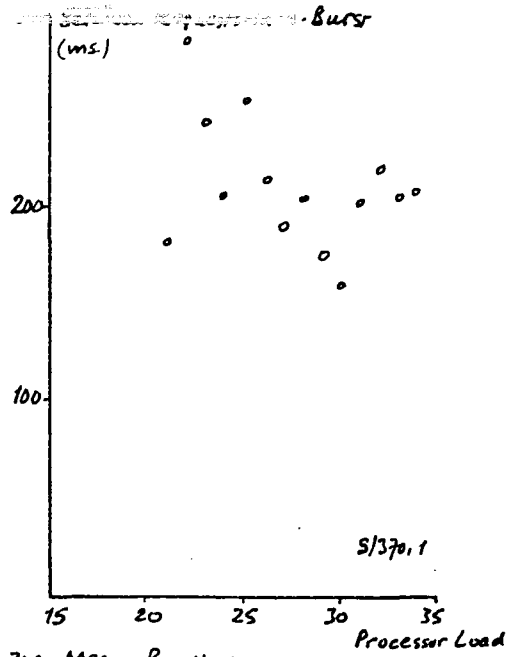
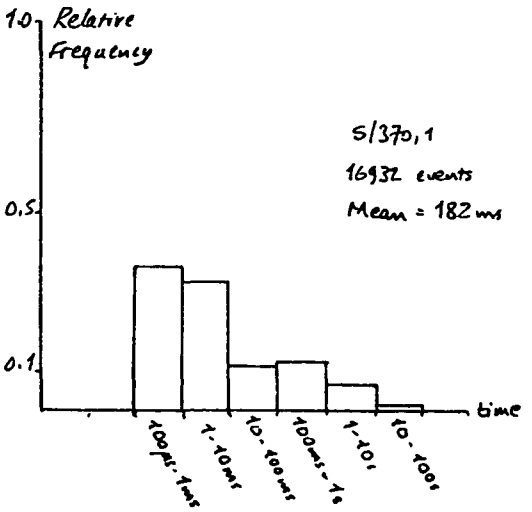
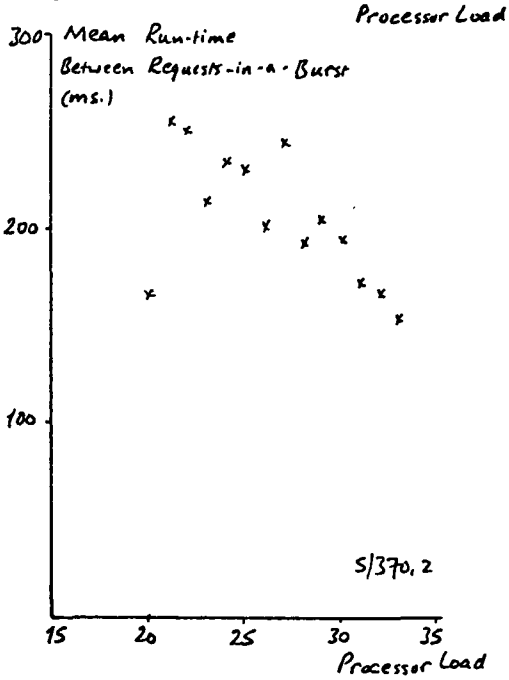
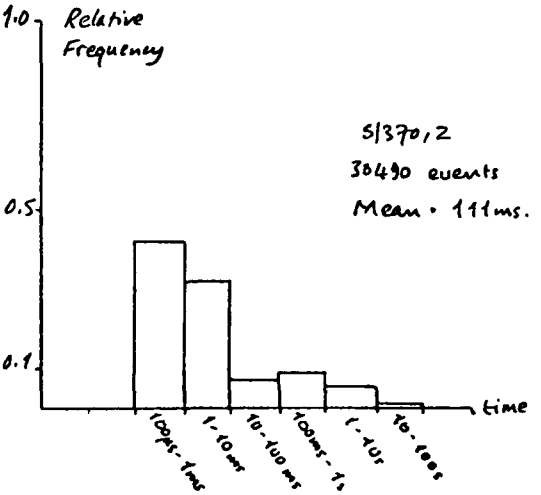


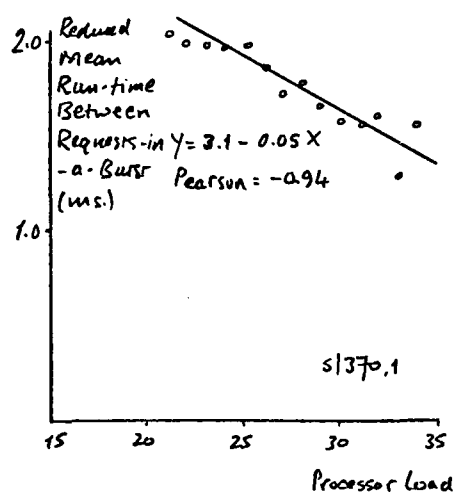
Figure A3. Run-time measures for other-i/o. S/360











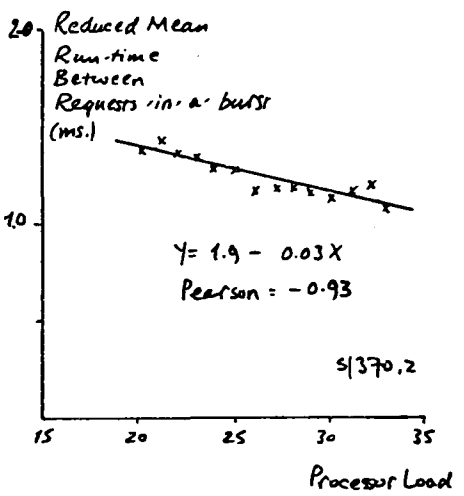


Figure A4. Run-time measures for other-i/o. S/370.

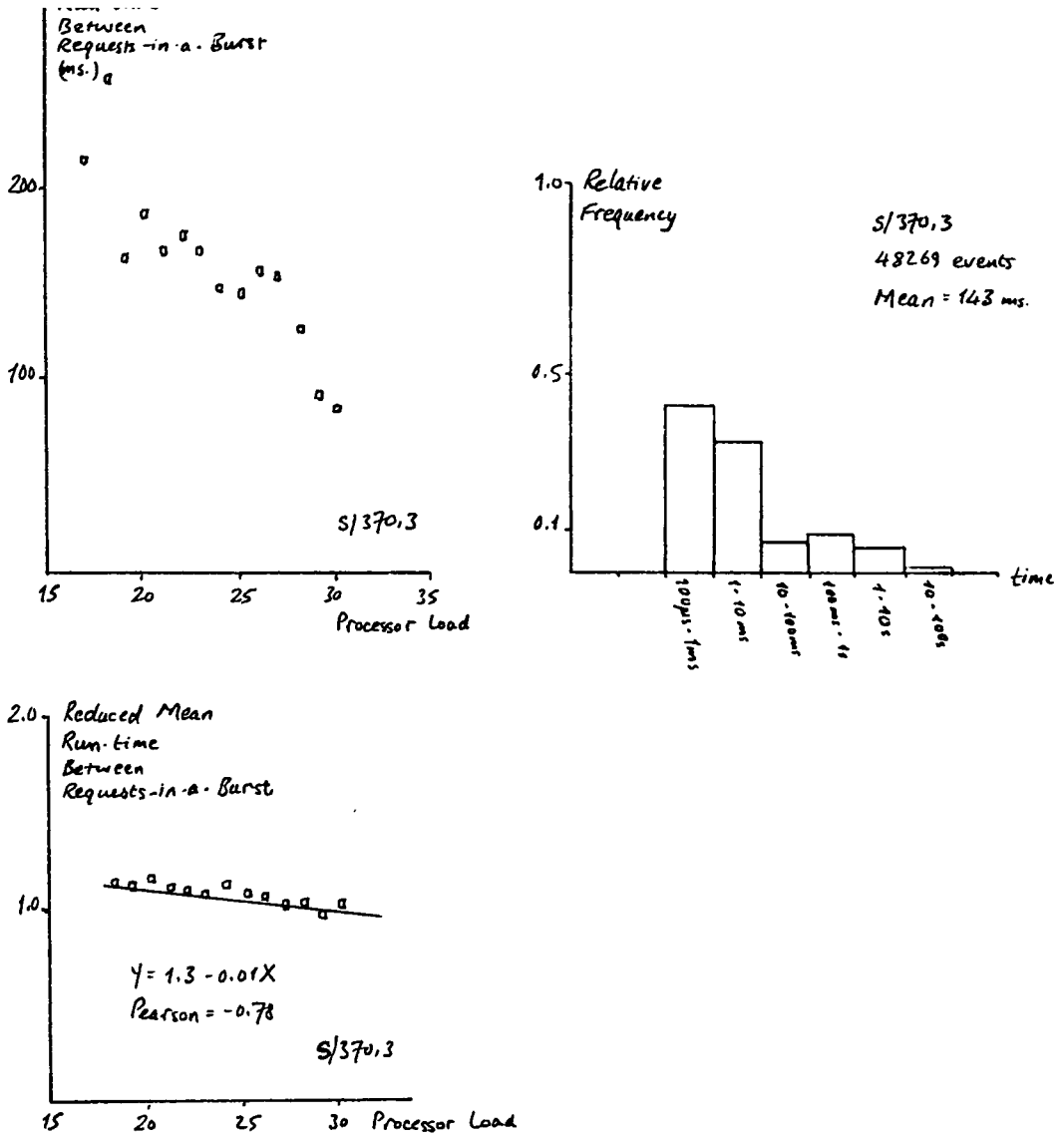


Figure A4 continued.

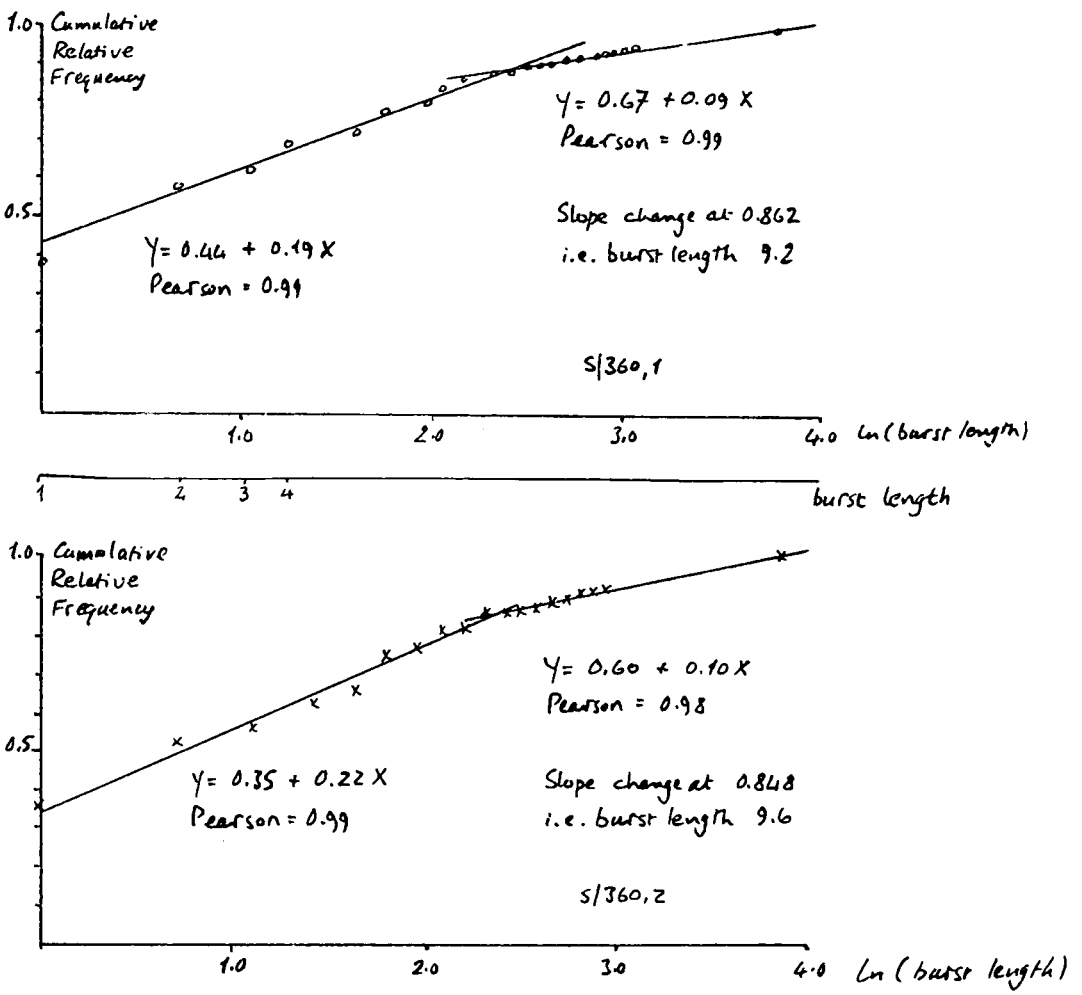
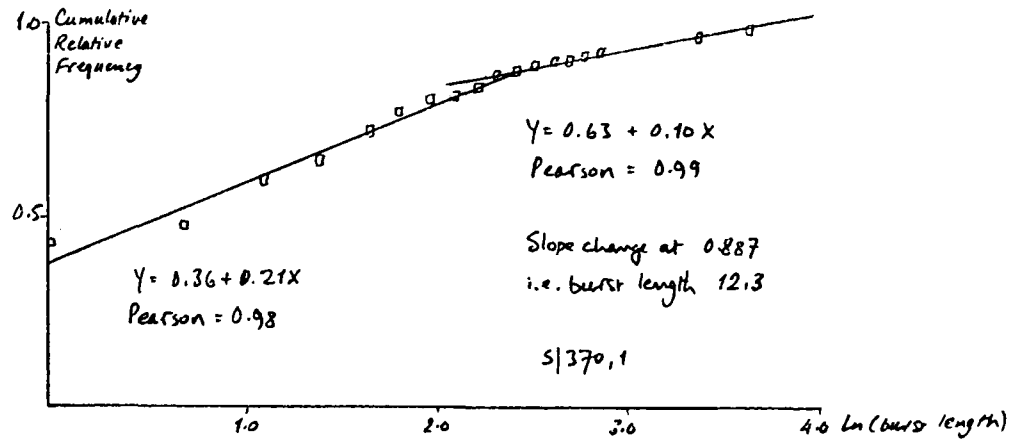
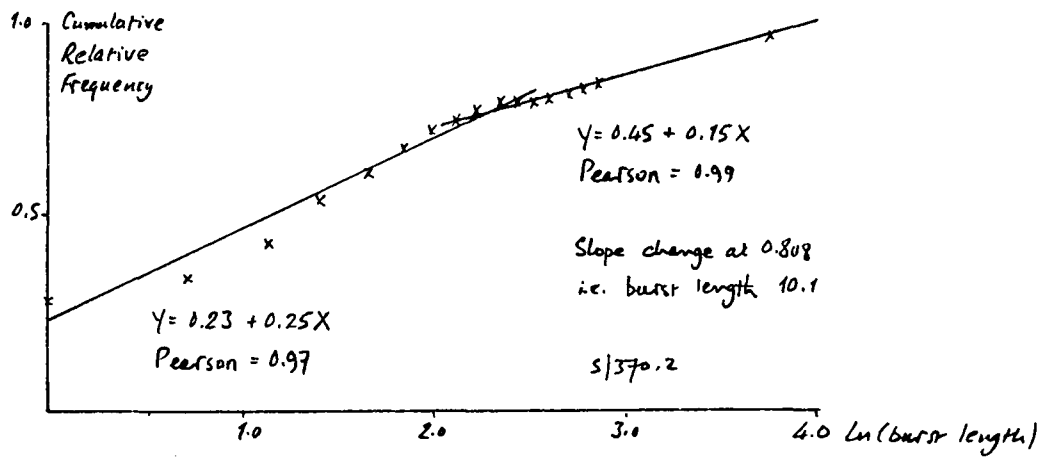


Figure A5. Measures of burst length for terminal-i/o. S/360.





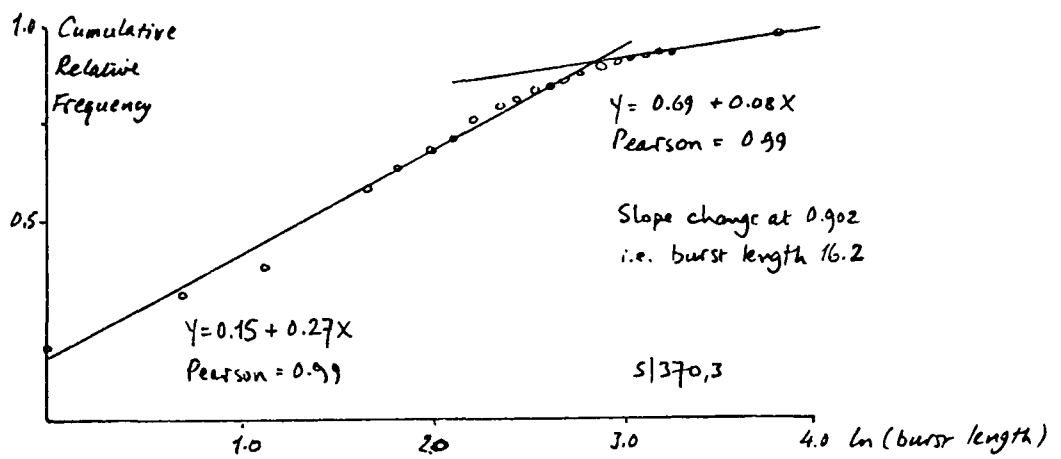
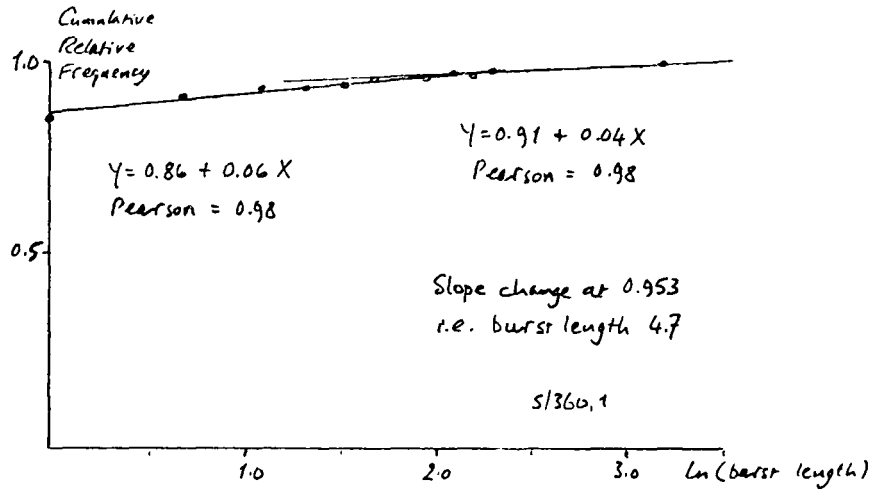


Figure A6. Measures of burst length for terminal-i/o. S/370



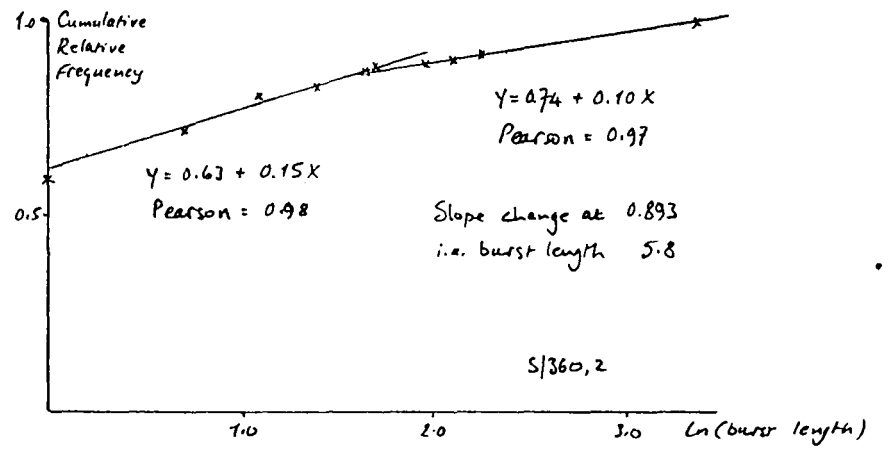
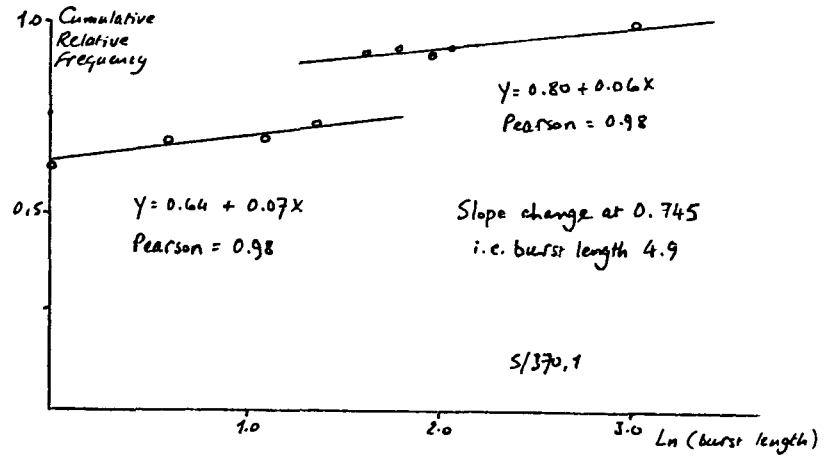
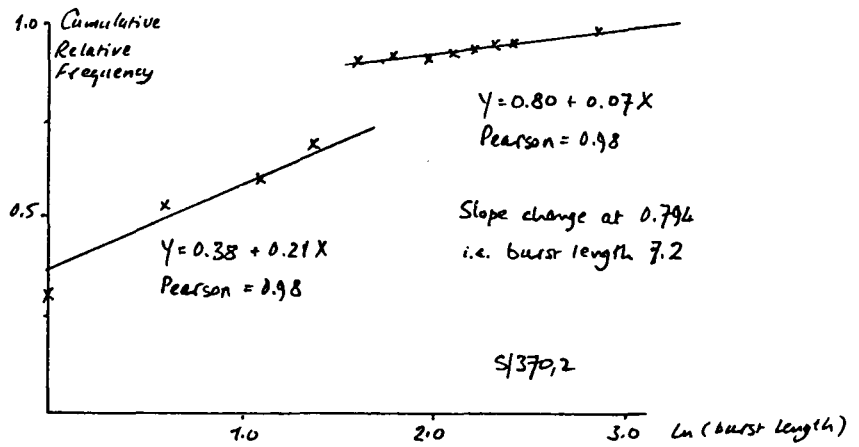


Figure A7. Measures of burst length for other-i/o. S/360.





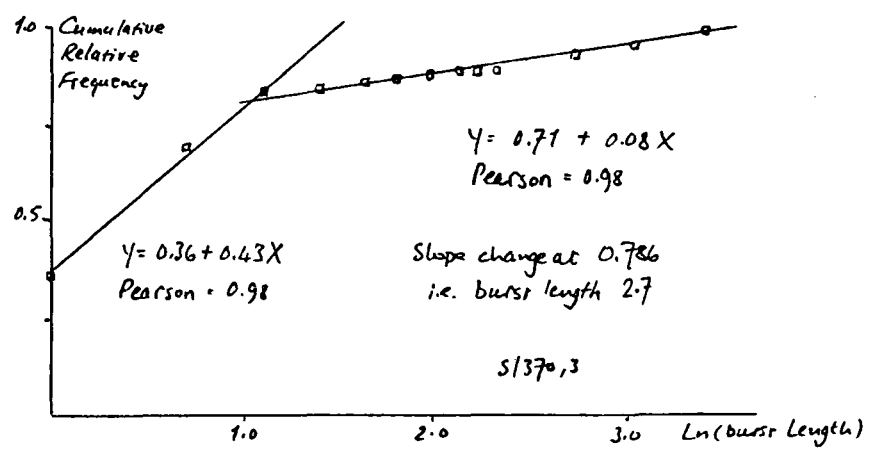


Figure A8. Measures of burst length for other-i/o S/370.

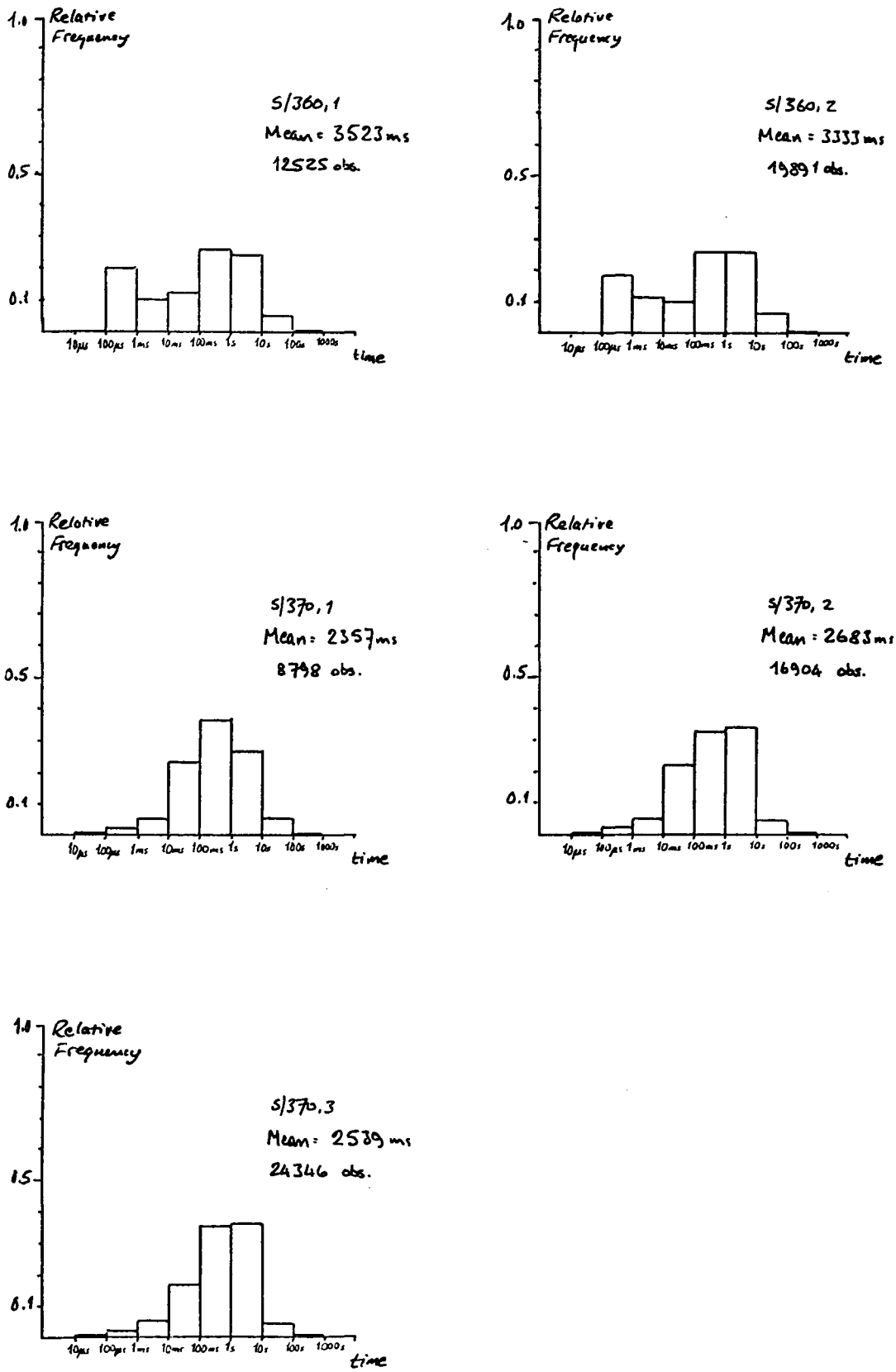
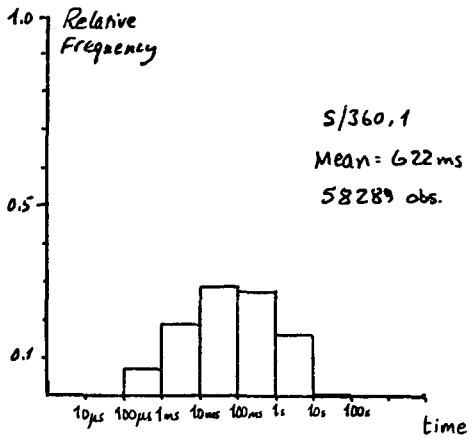
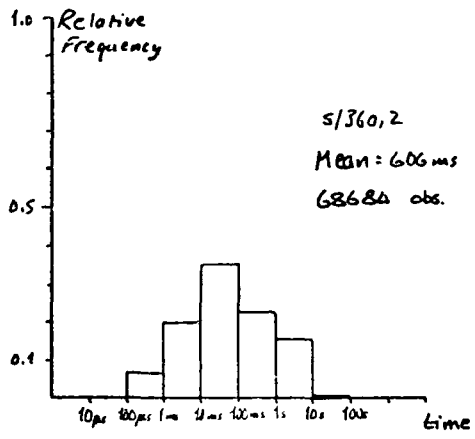
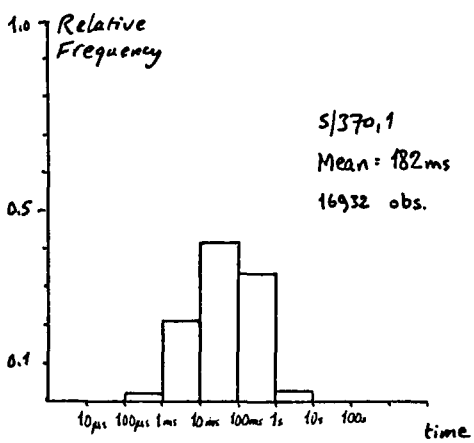
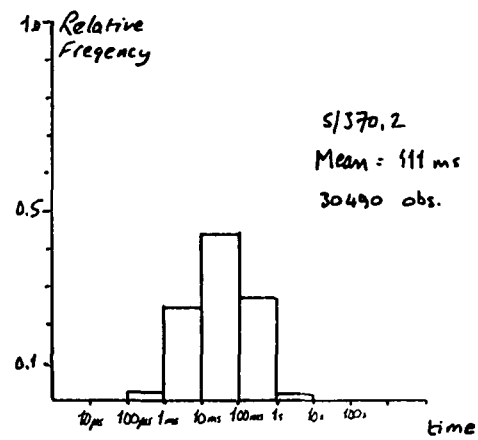


Figure A9. Terminal-i/o service time distributions.









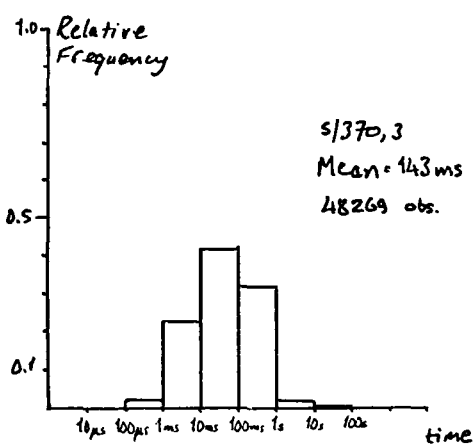


Figure A10. Other-i/o service time distributions.

Appendix B

A Note on the method of Statistical Analysis used in Chapters 7 and 8.

The experimental design used in Chapters 7 and 8 uses qualitative factors at several levels, and two quantitative outputs. This system may be analyzed for each output by the method of Analysis of Variance (ANOVA) because the design is orthogonal, i.e. all combinations of levels are employed and an equal number of observations of the output variable are made for each factor arrangement.

Routines from the program library NAG (Nottingham Algorithms Group [38]) were used for this analysis. The method decomposes the variance in the output observable from its several values in order to test whether those output values with ~~that~~ a factor at, say, level 1 differ from those values with that factor at another level, say level 2. For the simulation technique used, replications of the output observation at each factor level are required to establish this variation. The test of significance is whether the difference in value between factor levels is more than is expected because of random, or error, differences within each factor level.

The applied test of significance is:-

are the sums of squares of the output observable x due to a particular factor A $\left(SS_A = var_A(x)\right)$ from a population where no such effect exists?

This Null Hypothesis, if true, has the property that the ratio

$$\frac{\text{mean square of variance attributable to A}}{\text{mean square of variance unattributable}}$$

follows the F-distribution.

Further

$$\text{Mean square of variance}_A = \frac{\text{Sum of Sqq of x due to A}}{\text{degree of freedom of A}}$$

where the concept "degrees of freedom" is generally the number of observations of x less the number of constraints imposed on the system. This concept is treated explicitly in the analyses of Chapters 7 and 8.

The effect of factor A is statistically significant if the calculated ratio exceeds the value of the F-distribution. Tables of the variance ratio at which a factor is considered to be significant may be found in [14]. The probability levels usually chosen are 5%, 1% and 0.1%, i.e. for the 5% level there is a 1 in 20 chance that the observed effect satisfies the Null Hypothesis and belongs to the population of values which the output may normally assume. Similarly, the 1% and 0.1% levels express increasing confidence that the value observed does indeed come from a different population, i.e. that the factor has an effect on the observable. Practical texts [9, 18, 36] advise that a result about the 5% level is worth further investigation, and for final results and conclusions a higher level is preferred.

In the analyses of Chapters 7 and 8, the level of significance of a variance ratio is indicated by *(5%) meaning "significant"; ** (1%) "very significant"; and *** (0.1%) "highly significant".

The demands made upon experimental design for useful application of this statistical method are:

1. orthogonality of design
2. replication with independent random number generator seeding (for simulation experiments)
3. a reasonable expectation of cause and effect among the factors and the output observable.

Condition 1 is met by design. Condition 2 is met by preparing a pool of acceptable seed values for the mixed congruential pseudo-random number generator used [28] and recording final values as seeds for succesor runs. The pool of acceptable seeds was chosen by screening candidates with the Chi-Squared and Kolmogorov-Smirnoff tests for randomness and long life cycle [28] . Condition 3 is a safeguard that the physics of the situation lends itself to interpretation by this method of analysis.

Appendix C

Disk Request Scheduling Algorithms

C.1 Scheduling Algorithms - Literature

Denning [13] used a single disk model with exponential arrival assumptions to analyze the Shortest Seek Time First (SSTF) scheduler which he adopted from the Shortest Access Time First (SATF) policy developed for fixed-head rotating storage devices. He introduced SCAN, a method of service in which the disk arm sweeps backwards and forwards across the disk. This algorithm guarantees service under heavy load where SSTF discriminates against requests for extreme addresses. SCAN is also capable of a higher throughput of requests than First Come First Served (FCFS) under heavy load.

Variations of SCAN have been proposed by Frank [15], Merten [35] and Teorey [52]. Frank introduced the N-step SCAN policy (FSCAN or N-SCAN) in which N requests are taken from the queue of waiting requests and serviced during a scan. This policy is designed to reduce response time variance with little effect on request throughput. Merten removed the constraint of sweeping to a disk extremity before reversing direction. This policy is sometimes called LOOK, although SCAN now often implies Merten's modification. Teorey introduced C-SCAN, a circular scan. At an extremity the disk arm recommences sweeping at the starting extreme.

These investigations conclude that the SCAN algorithm and its derivatives N-SCAN and C-SCAN increase the variance of response time in return for a reduced average compared with FCFS. FCFS provides the most rapid service at light loads but response time increases with load more rapidly than the

the other algorithms. SSTF seems inherently unsafe since at heavy loads it shows a high request throughput but degenerates to serving only a fraction of available cylinders.

C.2 Algorithmic descriptions of Request Schedulers.

The schedulers described are FCFS, SSTF, SCAN, C-SCAN and N-SCAN. The Lynch scheduler is described in Chapter 5. These algorithms are described in terms of a single disk with two relevant events - Request Arrival and Queue Service. Request Arrival is self-explanatory. Queue Service occurs after a service completion when waiting requests are serviced. Extension to multiple disks involves another queue for the access path, and several queues for the several disks.

C.2.1 FCFS - One waiting queue.

a) at Request Arrival :

```
if ¬ disk_busy then service
                else enqueue;
```

b) at Queue Service:

```
serve request at head of queue;
```

C.2.2 SSTF - One waiting queue

a) at Request Arrival:

```
if ¬ disk_busy then service
                else enqueue;
```

b) at Queue Service:

```
service request closest to current arm position;
```

C.2.3 SCAN (including Merten's modification). One waiting queue.

a) at Request Arrival:

if ¬ disk_busy then service

else enqueue;

b) at Queue Service:

Service request least ahead in current service direction.

If none, reverse direction and select a request by the same rule;

C.2.4 C-SCAN - (including Merten's modification)

One waiting queue.

a) at Request arrival:

if ¬ disk_busy then service

else enqueue;

b) at Queue Service:

Service request least ahead in the service direction.

If none, retract to starting exteme and select request by the same rule;

C.2.5 F-SCAN - Two waiting queues, the service-queue and the waiting-queue.

a) at Request Arrival:

if ¬ disk_busy then service

else enqueue on waiting-queue;

b) at Queue - Service:

```
if service-queue ¬ = empty
then service request at head of queue
else
begin
    copy the first N requests from the waiting-queue to the
    service queue;
    order the service-queue by Frank's empirical shortest-
    seek-time algorithm, viz. arrange the requests for a scan
    first in the direction for which the farthest request
    distance is a minimum, and second backwards until the
    remaining requests have been serviced;
end
```

Appendix D Measurement Overhead Costs

The DCF version used by Gray was substantially the same as the facility described by Pinkerton [43]. The version used for this research had been implemented by King [27]. Pinkerton evaluated the DCF-induced interference to his observed system to be very small, namely 0.3% CPU time and about 4% of main storage for buffers. King's DCF version was used to monitor itself by including in the reporting set another event to enable explicit CPU time accounting per process. This event constitutes more than half the events gathered for interference evaluation. Table D.1 shows the results of two typical assessments. All assessments showed less than 2% CPU time attributable to DCF; this includes a high overhead introduced to make the assessment. The newer DCF uses more buffer areas, but the percentage remains the same at 4%. Channel utilization for accumulation of data is a calculated 0.4%. The measurement overhead costs are very small.

ELAPSED TIME (secs.)	CPU%			
	DCF	MTS SUPERVISOR MODE	MTS USER MODE	IDLE
121	1.8	2.3	95.9	0.0
121	1.4	1.8	96.8	0.0

Table D.1 DCF Interference Measurements

BIBLIOGRAPHY

1. Abate, J., Dubner, H. and Weinberg, S.B. , "Queuing Analysis of the IBM 2314 Disk Storage Facility." J. ACM 15, 4 (1968) 577 - 589.

2. Anderson, H.A. and Sargent, R.G. "A Statistical Evaluation of the Scheduler of an Experimental Interactive Computing System." in Statistical Computer Performance Evaluation (ed. Freiberger) 73 - 98, Academic Press, New York (1972)

3. Barraclough, E.D. "Performance Measurements of the Michigan Terminal System as Operated in Newcastle upon Tyne, 1971." MRM 16, Computing Laboratory, Univ of Newcastle upon Tyne, 1971.

4. Bard, Y. "CP67 Measurement and Analysis - Overhead and Thruput." in ACM/SIGOPS Workshop on System Performance Evaluation. ACM (1971) 246 - 260.

5. - "An Experimental Approach to System Tuning." in ACM/SIGMETRICS 1976 Proc. Intl. Symp. on Computer Peformance, ACM, NY (1976) 296-315.

6. Baskett, F., Browne, J.C. and Sherman, S. "Trace Driven Modeling and Analysis of CPU Scheduling in a Multi-Programming System" in ACM Workshop on System Performance Evaluation, ACM (1971) 173 - 199.

7. Beilner, H. and Waldbaum, G. "Statistical Methodology for Calibrating a Trace-Driven Simulator of a Batch Computer System," IBM Research Report RC 3855, 1972.

8. Brinch - Hansen, P. "Disk Scheduling at Compile Time." Software, Practice and Experience 6, 2 (1976) 201-205.

9. Brownlee, K.A. "Industrial Experimentation." Chemical Publishing Co. N.Y. 1953.

10. Buzen, J.P. "Cost Effective Analytic Tools for Computer Performance Evaluation". Proc. IEEE Compcon, Fall 1975.

11. Clapson, P.J. "Towards a Performance Science: a Comparative Analysis of Computing Systems." Comp. J. 20, 4 (1977) 308 - 336.

12. Cheng, P.S. "Trace Driven System Modelling." IBM Syst. J. 9, 4 (1969) 280 - 289.

13. Denning, P.J. "Effects of Scheduling on File Memory Operations." Proc. AFIPS SJCC 30 (1967) 9 - 21.

14. Fisher, R.A. and Yates, F. "Statistical Tables for Biological, Agrucultural and Medical Research" Oliver and Boyd, London (1948).

15. Frank, H. "Analysis and Optimization of Disk Storage Devices for Time-Sharing Systems." J. ACM 16, 4 (1969) 602 - 620.

16. Gotlieb, C.C. and MacEwan, G.H. "Performance of Movable-Head Disk Storage Devices." J. ACM 20, 4 (1973) 604-623.

17. Gray, J. "Moveable Head Disks in M.T.S." M.Sc. Dissertation, University of Newcastle upon Tyne (1973).

18. Hartley, A.O. "Analysis of Variance" in Ralston and Wilf (eds) "Mathematical Methods for Digital Computers." Wiley, NY 1960.

19. van Horn, R. "Validation." in Naylor (ed) "The Design of Computer Simulation Experiments." Duke University Press (1969) 232-251.

20. IBM Corporation, "IBM 2314 Direct Access Storage Facility" A26-3599-2 IBM Co, N.Y. (1965).

21. - , "Reference Manual for IBM 3330 Series Disk Storage." GA26-1615-3, IBM Co. Calif. (1974).

22. - , "Reference Manual for IBM 3340/3344 Disk Storage." GA26-1619-4, IBM Co. Calif. (1975).

23. - , "Reference Manual for IBM 3350 Disk Storage" GA26-1638-1, IBM Co, Calif. (1976).

24. - , "OS ISAM Logic" BY28-6618, IBM Co., NY.

25. Jalics, P.J. "Measurement of the PDP 10 TOPS-10 Time-Sharing Operating System". Ph.D. Thesis, Report TR 1122, Computing and Information Science Dept., Case Western Reserve Univ. (1973).

26. - , and Lynch, W.C. "Selected Measurements of the PDP10 TOPS-10 Time-Sharing Operating System." Proc. IFIP 1974, 2 (Software) 242-246.

27. King, P.J.B. "Users Guide to the M.T.S. Data Collection Facility" Computing Laboratory, University of Newcastle upon Tyne (1976).

28. Knuth, D.E. "The Art of Computer Programming - Volume 2, Seminumerical Algorithms." Addison-Wesley, Mass. (1971)

29. Lynch. W.C. "Do Disk Arms Move?." ACM SIGMETRICS Performance Evaluation Review 2 (1973).

30. - , "Disk Data Arrangement for Fast File Transfer."Report TR 1134, Computing and Information Science Dept,, Case Western Reserve University (1973).

31. MacEwan, G.H. "Performance of Disk Storage Devices in Computer Systems." Ph.D. Thesis, University of Toronto, 1971.

32. McGregor, D.R.,Thomson, G.R.,Dawson. W.N. and Jones, D.A."A Design for an Improved Backing-Store." Dept. of Computer Science, University of Strathclyde, 1977.

33. Margolin, B.H., Parmlee, R.P. and Schatzoff, M. "Analysis of Free-Storage Algorithms." IBM Syst. J. 10, 4 (1971) 283 - 304.

34. Maryuama, K. and Smith, S.E. "Optimal Reorganization of Distributed Space Disk Files "C. ACM 19, 11 (1976) 634 - 642.

35. Merten, A.G. "Some Quantitative Techniques for File Organization" TR 15, Ph.D. Thesis Univ. of Wisconsin (1970).

36. Mihram, G.A. "Simulation - Statistical Foundations and Methodology" Academic Press, London (1972).

37. Moore, C.G. "Network Models for Large-Scale Time-Sharing Systems". Ph.D. Thesis, Univ. of Michigan (1971).

38. "NAG Fortran Library Manual MK6". Volume 3, NAG, 1977.

39. Naylor, T.H. and Burdick, D.S. "Design of Computer Simulation Experiments for Industrial Systems" C. ACM 9, 5 (1966) 329 - 339.

40. Omahen, K. "Estimating Response Time for Auxiliary Memory Configurations" Proc. Intl. Conf. on Very Large Data Bases, ed. D.S. Kerr, ACM (1975) 473 - 485.

41. Parapudi, M. and Winograd, J. "Interactive Task Behaviour in a Time Sharing Environment." Proc. ACM Annual Conf. 1972 II 680 - 692.

42. Pinkerton, T.B. "Performance Monitoring in a Time-Sharing System" C. ACM 12, 11 (1969) 608 - 610.

43. - , "The MTS Data Collection Facility." Concomp Memo 18, University of Michigan (1968).

44. Reddy, Y.V. "Experimental Evaluation of a Multi-Programmed Computer System" C.ACM 18, (1975) 108 - 111.

45. Scahtzoff, M. "Some Approaches to the Modelling and Evaluation of Programs and Systems". in State of the Art Report No. 18, "Computer Systems Measurement"., 597 - 620, Infotech, London (1974).

46. - , Tsao, R. and Wiig, R. "An Experimental Comparison of Time Sharing and Batch Processing." C.ACM 10, 5 (1967) 261 - 270.

47. Schwetman, H.D. "Gathering and Analyzing Data from a Computer System - A Case Study." Proc. ACM Conf. 1975, 112 - 117.

48. , and Brown, J.C. "An Experimental Study of Computer System Performance." Proc. ACM Conf. 1972 (2) 693 - 703.

49. Screenivasan, S. "Application of Accounting Data in Evaluating Computer System Performance." Software, Practice and Experience 6, 2 (1976) 201 - 205.

50. Sherman, S.W. "Trace-Driven Modeling Studies of the Performance of Computer Systems." Ph.D. Thesis, University of Texas at Austin, 1972.

51. Stone, D.L. and Turner, R. "Disk Thruput Estimation." Proc. ACM Conf. 1972 (2) 704 - 711.

52. Terey, T.J. "The Role of Disk Scheduling in Multiprogrammed Computer Systems."" Ph.D. Thesis, TR35, University of Wisconsin, Madison, 1974.

53. - and Pinkerton, T.B. "A Comparative Analysis of Disk Scheduling Policies." C.ACM 15, 3 (1973) 177 - 184.

54. Tsao, R.F., Comeau, L.W. and Margolin, B.H. "A Multi-Factor Paging Experiment: the Experiment and the Conclusions." in Statistical Computer Performance Evaluation (ed. Freiberger) Academic Press, N.Y. (1972).

55. Wilhelm, N.C. "An Anomaly in Disk Scheduling: A Comparison of FCFS and SSTS Seek Scheduling Using an Empirical Model for Disk Accesses." C.ACM 19, 1 (1976) 13 - 17.

56. , "A General Model for the Performance of Disk Systems". J. ACM 24, 1 (1977) 14 - 31.