COMPUTER SYSTEMS LABORATORY

STANFORD ELECTRONICS LABORATORIES
DEPARTMENT OF ELECTRICAL ENGINEERING
STANFORD UNIVERSITY · STANFORD, CA 94305

An Exponential Failure/Load Relationship: Results of a Multi-Computer Statistical Study

by

Ravishankar Krishnan Iyer

Steven E. Butner

Edward J. McCluskey

Technical Report #CRC-81-6

(CSL TR #214)

July 1981

Center for Reliable Computing
Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

This work was supported in part by the Air Force Office of Scientific Research under Contract F49620-79-C-0069, the National Science Foundation under grant MCS-7904864, and by CSIRO (Australia).

Center for Reliable Computing

An Exponential Failure/Load Relationship: Results of a Multi-Computer Statistical Study

by

Ravishankar Krishnan Iyer

Steven E. Butner

Edward J. McCluskey

Technical Report #CRC-81 -6

(CSL TR #214)

July 1981

Center for Reliable Computing
Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

This work was supported in part by the Air Force Office of Scientific Research under Contract F49620-79-C-0069, the National Science Foundation under grant MCS-7904864, and by CSIRO (Australia).

An Exponential Failure/Load Relationship: Results of a Multi-Computer Statistical Study

by

Ravishankar Krishnan Iyer   Steven E. Butner   Edward J. McCluskey

CRC Report #81-6
(CSL TR #214)

July 1981

Center for Reliable Computing
Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

Abstract

In this paper we present an exponential statistical model which relates computer failure rates to level of system activity. Our analysis reveals a strong statistical dependency of both hardware and software component failure rates on several common measures of utilization (specifically CPU utilization, I/O initiation, paging, and job-step initiation rates). We establish that this effect is not dominated by a specific component type, but exists across the board in the two systems studied. Our data covers three years of normal operation (including significant upgrades and reconfigurations) for two large Stanford University computer complexes. The complexes, which are composed of IBM mainframe equipment of differing models and vintage, run similar operating systems and provide the same interface and capability to their users The empirical data comes from identically-structured and maintained failure logs at the two sites along with IBM OS/VS2 operating system performance/load records. The statistically strong relationship between failures and load is evident for many equipment types, including electronic, mechanical, as well as software components. This is in opposition to the commonly-held -belief that systems which are primarily electronic in nature exhibit no such effect to any significant degree. The exponential character of our statistical model is significant not not only in its simplicity, but also due to its compatiblity with classical reliability techniques.

Keywords: statistical failure models, performance-reliability models, computer failure data.

Table of Contents

1. **Introduction** 1

1.1 Basis and Perspective 2

2. **Sources of Data** 5

2.1 Performance and Utilization Data 5

2.2 Failure Data 6

3. **Preliminary Analysis** 7

3.1 Load Profiles 7

3.2 Failure Profiles 8

3.3 Analysis and Discussion of Preliminary Results 9

4. **Some Reliability Measures** 13

5. **Detailed Analysis** 17

5.1 Validation of Preliminary Study Results 17

5.2 Component-Level Modelling 20

6. **Views on Observed Results** 23

6.1 Discussion of Measured Results 23

6.2 Conjectures on the Model 24

6.2.1 Latent Discovery Effect 25

6.2.2 Load-Induced Hardware Failures 25

6.2.3 Load-Induced Software Failures 26

7. **Probabilistic Models** 29

8. **Concluding Remarks** 31

Acknowledgements 32

Appendix I. Preprocessing Steps 33

I.1 Raw Data 33

I.2 Required Data Attributes 33

I.3 Processing Techniques 34

i.4 The UNIMERGE Program 35

I.5 Lagged Data 36

Appendix II. SLAC and CIT Load Profiles 37

Appendix III. Component Type and Usage Groups Failure Charts 43

References 64

List of Figures

Figure 3-1: SLAC 1978 Virtual Day Load Profiles 8
Figure 3-2: SLAC 1978 Failure Profiles 9
Figure 5-1: CIT 1978 Load Profiles 18
Figure 5-2: SLAC 1978 Failure-Time CPU and Paging Profiles 19
Figure 5-3: SLAC MEDIA and CPU Failure Histograms 21
Figure 7-1: Augmented Bathtub Curve 30
Figure II-1: 1978 EXCP Profiles 37
Figure II-2: 1978 Batch CPU Profiles 38
Figure II-3: 1978 Batch Paging Profiles 38
Figure II-4: 1979/80 Job Step Profiles 39
Figure II-5: 1979/80 EXCP Profiles 39
Figure II-6: 1979/80 Batch CPU Profiles 40
Figure II-7: 1979/80 System CPU Profiles 40
Figure II-8: 1979/80 System Paging Profiles 41
Figure III-1: CIT 1979/80 Failure Histogram for Component Type 'CARD' 44
Figure III-2: SLAC 1979/80 Failure Histogram for Component Type 'CARD' 44
Figure III-3: CIT 1979/80 Failure Histogram for Component Type 'CHANNEL' 45
Figure III-4: SLAC 1979/80 Failure Histogram for Component Type 'CHANNEL' 45
Figure III-5: CIT 1979/80 Failure Histogram for Component Type 'DCTL' 45
Figure III-6: SLAC 1979/80 Failure Histogram for Component Type 'DCTL' 46
Figure III-7: CIT 1979/80 Failure Histogram for Component Type 'DISK' 46
Figure III-8: SLAC 1979/80 Failure Histogram for Component Type 'DISK' 47
Figure III-9: CIT 1979/80 Failure Histogram for Component Type 'MCTL' 47
Figure III-10: CIT 1979/80 Failure Histogram for Component Type 'MEMORY' 47
Figure III-11: SLAC 1979/80 Failure Histogram for Component Type 'MEMORY' 48
Figure III-12: CIT 1979/80 Failure Histogram for Component Type 'OPRL' 48
Figure III-13: SLAC 1979/80 Failure Histogram for Component Type 'OPRL' 48
Figure III-14: CIT 1979/80 Failure Histogram for Component Type 'PCTL' 49
Figure III-15: CIT 1979/80 Failure Histogram for Component Type 'PRINT' 49
Figure III-16: SLAC 1979/80 Failure Histogram for Component Type 'PRINT' 50
Figure III-17: CIT 1979/80 Failure Histogram for Component Type 'SWBE' 50
Figure III-18: CIT 1979/80 Failure Histogram for Component Type 'SWFE' 51
Figure III-19: CIT 1979/80 Failure Histogram for Component Type 'SWOP' 51
Figure III-20: SLAC 1979/80 Failure Histogram for Component Type 'SW' 52
Figure III-21: CIT 1979/80 Failure Histogram for Component Type 'TAPE' 52
Figure III-22: SLAC 1979/80 Failure Histogram for Component Type 'TAPE' 53
Figure III-23: CIT 1979/80 Failure Histogram for Component Type 'TCTL' 53
Figure III-24: SLAC 1979/80 Failure Histogram for Component Type 'TCTL' 53
Figure III-25: CIT 1979/80 Failure Histogram for Component Type 'TELE 54

Figure 111-26: SLAC 1979/80 Failure Histogram for component Type 'TELE 54
Figure III-27: CIT 1979/80 Failure Histogram for Component Type 'TLCT' 55
Figure III-28: CIT 1979/80 Failure Histogram for Component Type '3CPU' 55
Figure III-29: SLAC 1979/80 Failure Histogram for Component Type '8CPU' 56
Figure W-30: SLAC 1979/80 Failure Histogram for Component Type '9CPU' 56
Figure III-31: CIT 1979/80 Failure Histogram for Usage Group 'CPU' 57
Figure III-32: CIT 1979/80 Failure Histogram for Usage Group 'CPU*' 57
Figure III-33: SLAC 1979/80 Failure Histogram for Usage Group 'CPU*' 58
Figure 111-34: CIT 1979/80 Failure Histogram for Usage Group 'MDCTL' 58
Figure III-35: SLAC 1979/80 Failure Histogram for Usage Group 'MDCTL' 59
Figure III-36: CIT 1979/80 Failure Histogram for Usage Group 'MED*' 59
Figure III-37: CIT 1979/80 Failure Histogram for Usage Group 'MEDIA' 60
Figure III-38: CIT 1979/80 Failure Histogram for Usage Group 'MEMORY' 60
Figure III-39: SLAC 1979/80 Failure Histogram for Usage Group 'MEMORY' 61
Figure W-40: CIT 1979/80 Failure Histogram for Usage Group 'SOFTWARE' 61
Figure III-41: SLAC 1979/80 Failure' Histogram for Usage Group 'SOFTWARE' 62
Figure III-42: CIT 1979/80 Failure Histogram for Usage Group 'TLCHN' 62
Figure III-43: SLAC 1979/80 Failure Histogram for Usage Group 'TLCHN' 63
Figure III-44: SLAC 1979/80 Failure Histogram for Component Type 'OCTL' 63
Figure III-45: SLAC 1979/80 Failure Histogram for Component Type 'RLTM' 63

List of Tables

Table 3-1: Notation and Definitions for Simple Regression 10
Table 4-1: Comparative Reliability Measures for CIT & SLAC in 1978-80 14
Table 5-1: CIT 1978 Load-Failure Correlations 18
Table 5-2: Regression Results by Usage Groups -- 1979/80 SLAC & CIT . 22

1. Introduction

The last decade has seen an explosion in the application of computers in many key commercial, military and industrial environments. Our society has become increasingly reliant upon the error-free performance of such systems. As a consequence enhanced reliability, high performance and overall availability have ail assumed increasing importance in computer design.

It is well known that as a system approaches high levels of utilization, degradation in performance occurs. A more interesting question, however, is whether an increased level of utilization results in a real or apparent degradation in system *reliability,* i.e. is there an inherent load versus failure relationship? This question has strategic significance; particularly since many security and defense systems are required to have maximum reliability at precisely the time of their peak load or stress. Significant as this question is there are few studies available in this area [Beaudry, 1978a] [Butner, 1980] [Castillo, 1981]. It is a wide-spread belief that since these systems are electronic in nature no such effect exists on a significant scale.

Analytic failure modelling and simulation appear un-attractive at this stage due to a lack of understanding of the processes involved. One alternative is statistical analysis. Accordingly, we embarked on a project to measure and analyse data on system load and failures. In undertaking a potentially large-scale project of this nature it is generally advisable to conduct a preliminary study on a smaller scale. Our preliminary study analysed the Stanford Linear Accelerator Center (SLAC) computer facility which is an IBM multi-processor system. The study was based on prior research at the Center For Reliable Computing at Stanford University [Beaudry, 1978a]. In our more comprehensive study we also include the IBM/3033 at the Stanford Center for Information Technology (CIT).

It is the purpose of this paper to present the results of both our preliminary study and the subsequent comprehensive statistical analysis of failures and system utilization.

1 .1 Basis and Perspective

The Stanford Linear Accelerator Center is engaged in the study of high energy elementary particle physics. A two mile long linear accelerator and associated real-time data concentration network provide a vast amount of physical data for analysis. In an average day nearly 4000 batch jobs are executed. The complex is essentially a batch system (although it does have a measurable real-time and interactive load). The system is composed of two IBM 370/168's and one IBM 360/91 in a triplex arrangement. It has an interactive "front end" text editing and job entry subsystem (WYLBUR) [SCIP, 1975a] as well as a time-sharing executive(ORVYL) [SCIP, 1975b].

The CIT system is a uniprocessor consisting of a relatively large IBM/3033 mainframe. It is a functional twin of SLAC, i.e. with WYLBUR, ORVYL, and IBM OS/VS2-(SVS) [IBM, 1972] components. Differences exist in system redundancy and system usage. In particular, CIT is oriented more toward production and general user services than SLAC. The SLAC system is considerably more complicated due to its asymmetric multiprocessor arrangement. The complex's three host processors are fully interconnected by channel-to-channel adapters (a device that connects a channel of one host to a channel of a second host allowing high-bandwidth traffic to flow between the two machines). Important system structures, including queues and communication areas reside on commonly-accessible disk volumes. All SLAC hosts can access these volumes via multi-port controllers? From a reliability point of view, the non-redundant front end components of the subject systems are a predictable source of availability problems. The redundancy of two independent, dedicated batch servers at SLAC makes that system's batch component reliable 'and highly available.

In 1976, a program for collecting reliability data on the SLAC and CIT computer systems was initiated. Under this program component and system failures have been systematically logged, causes have been meticulously tracked, and ensuing repairs noted. The data base containing this information is called the UNILOG. The availability of a

[1] An exception is the 360/91 CPU which cannot connect to the newer block multiplexer devices due to its age and design.

failure log such as this on two physically unique, but logically similar systems has proven invaluable for reliability research. The differences in equipment type and interconnection allow us to draw inferences about the reliability of varying architectures while the basic logical similarities of the two systems permit direct comparisons between them.

An early study of SLAC UNILOG data [Beaudry, 1978b] indicated that the conventionally used constant failure-rate model was not appropriate in an environment with a widely fluctuating load. It was proposed that the system be modelled as two distinct Poisson failure processes, one occurring in prime time and the other in non-prime time. A more comprehensive study was deemed necessary before such results could be considered representative. Accordingly, we conducted a preliminary statistical analysis of SLAC failure and load data. In particular we sought to determine if there was an inherent load versus failure relationship [Butner, 1980].

Before describing our <analysis in detail, it is appropriate to indicate related research -in the general area of reliability-performance modelling. Recently a theoretical definition and computation of "performability" [Meyer, 1978] [Meyer, 1980] has appeared in the literature. Performability models attempt to express the effect of failures on performance. Our goal is quite distinctly opposite, i.e. to investigate the influence of system activity on component and system reliabilities.

Some recent work on performance and reliability modelling [Castillo, 1980] has been reported. During development of performance measures for a large DEC-10A time-sharing system, it was found that the assumption of a constant system failure rate did not agree with measured data. The -authors report a four-to-one range in failure rate which they attribute to system load.

Subsequent research by the same authors [Castillo, 1981] involves use of a doubly-stochastic Poisson process to model failures. The model assumes that the instantaneous failure rate of a system resource can be approximated by a deterministic (cyclic) function of time plus a zero-mean stationary Gaussian process, both depending on the usage of the resource considered.

Our approach presumes no model a priori, but rather starts from a substantial body of empirical data. We report the statistical trends and relationships observed in that data with the hope of discovering a descriptive modelling method. It is significant that our data comes from *two distinct systems* with differing physical equipment and workload.

At the time of our analysis (mid 1979) 1978 was the latest full year's data available. Hence our preliminary analysis at SLAC was for 1978. We begin by describing the sources of raw data used in our study.

2. Sources of Data

The raw data for our studies came from two distinct sources: the operator-maintained failure and event log (UNILOG) and the operating system's performance and accounting database (IBM/SMF).

2.1 Performance and Utilization Data

Information on system utilization came from IBM System Management Facility (SMF) records. These data are logged in real-time by the operating system software. There are approximately 50 different types of SMF data. The data contain information on the initiation, processing, and termination of jobs, on batch streams, on interactive user sessions, and on other important events.

For our preliminary study, the SMF batch job step record was used.[2] This utilization information depicts only user jobs in the *batch* system. Any load due to interactive or real-time activities is not included. Since SLAC is a heavily batch-oriented computation center, such a batch-only view was deemed adequate for our preliminary study.

From the job step record, four data elements were chosen for study:

PAGING	the sum of page-ins and page-outs for the step;
EXCPs	the count of all I/O initiations for the step;
CPU	the central processor time used for the step;
HOUR	the hour of the day during which the job started.

Job steps which were never executed and those corresponding to continuously-running system jobs (e.g. WYLBUR, ORVYL) were discarded.

In our more detailed follow-on analysis we considered system-level information as well as the batch load data. The system *wait time* record (in SMF) gave unused CPU time

[2] A "job step" is the basic unit of activity in the batch system.

and overall system paging rates. When combined with the detailed batch load data, these factors provide a better picture of overall system activity.

2.2 Failure Data

The failure data for our study came from an operator-maintained system database called the UNILOG [Butner, 1980]. It contains component failure data as well as maintenance and engineering change tracking records. Entries in the UNILOG are time-stamped events depicting individual failures or other significant happenings. Each UNILOG failure event is given a *type* which indicates a hardware failure, a software failure, a utility problem, an operator error. or an unknown. Each event is categorized as *new* or *recurrent* based upon whether or not a "repaired" component has remained in operation for at least one hour. There are approximately 1800 records for each year of activity.

All failures and other significant system events at SLAC and CIT are carefully logged by the system operators. In addition to routine training for operations personnel, an on-line tutorial on UNILOG data entry exists. The failure logging is performed via a "fill-in-the-blanks", prompting style of man-machine interface. Due to the similarity of the SLAC and CIT systems and the close cooperation of their management, the UNILOG records kept at both sites are similar in form and function.

3. Preliminary Analysis

We began our analysis by developing *profiles* of utilization and failure rate behavior. The profiles provided a visualization of significant trends and led us to hypothesize various models for subsequent statistical testing. Additionally, they enabled comparison of the merits of candidate measures of system utilization and provided a sound basis- for statistical analysis.

It is to be expected that utilization measures will be cyclic over the day. Accordingly, it is meaningful to examine failure rates over the same period. In our calculations we average the whole year's data to produce a "virtual day".

3.1 Load Profiles

The volume of load/performance data was far too large to allow direct analysis and -comparison with the corresponding failure data. For a time period spanning one year, there were fifteen magnetic tapes of utilization data with only about two thousand failure records for the same interval. It was therefore necessary to reduce the load data to a size that would allow effective, yet meaningful, 'analysis.

We reduced the utilization data significantly by computing, for each hour of the day, the average value of each load measure. Due to the volume and structure of the raw data, the averaging was performed on a month-by-month basis. Appendix I discusses the processing steps in detail. The resulting load profiles are called the "virtual day" since they represent the expected load patterns for each hour of the day. There was initially a concern that such massive averaging would not provide a meaningful indication of actual system loads. We feared that fluctuations in system load were large and that the maxima and minima were important factors affecting the failure of equipment and software. After conducting a detailed study on a month-by-month basis, it was found that the virtual day model was representative of the SLAC and CIT system loading.

Figure 3-1 gives the profiles of two important utilization measures (batch system paging and job step processing rate) for SLAC in 1978. Profiles for EXCP rate and total batch CPU time per hour were also developed (see Appendix II).

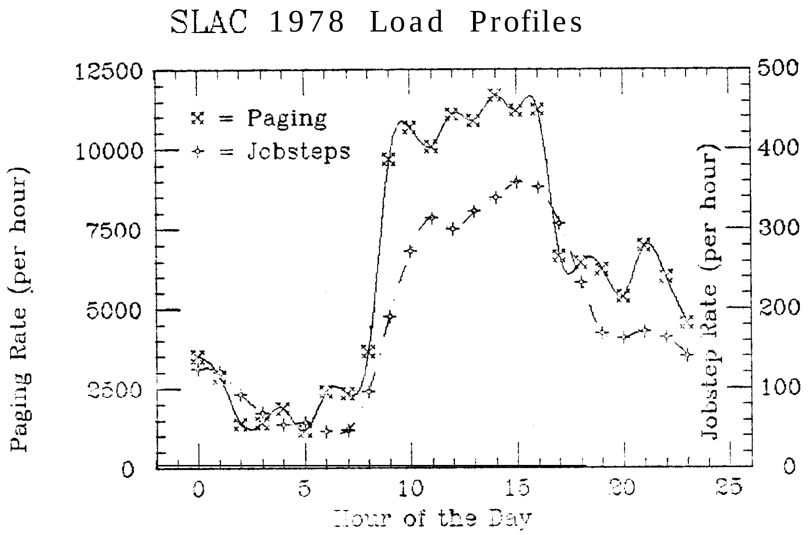


Figure 3-1: SLAC 1978 Virtual Day Load Profiles

Visual inspection of the SLAC load profiles yields an indication of the normal working day. Early morning is a low-utilization period, with activity beginning to build promptly at 8 AM. The lunch-time dip and the end of the working day are clearly visible. Evening hours are spent working off the queues of batch jobs. The higher level late-night utilization is due to the scheduled processing of large, resource-hogging jobs. This is reflected in the profiles of EXCP and CPU usage rate.

3.2 Faiiu re Profiles

In order to simplify the comparison of failure and load data, we computed the failure profiles also on a virtual day basis. This allowed the hourly load averages to be paired with 24 corresponding hourly failure rates. Separate failure profiles for hardware and software were developed based upon the UNILOG failure type codes.' Figure 3-2 gives sample failure profiles for hardware and software failures at SLAC in 1978.

It would appear from the shape of the failure profiles that the load imposed by *interactive services* has an effect on the failure rate. The failure rates are the lowest in the non-prime hours. They increase just after 8 AM as the working day begins and peak before and after lunch time. The end of the working day is also noticeable.

[3] Together, the failure types for hardware and software accounted for 95% of all failures.

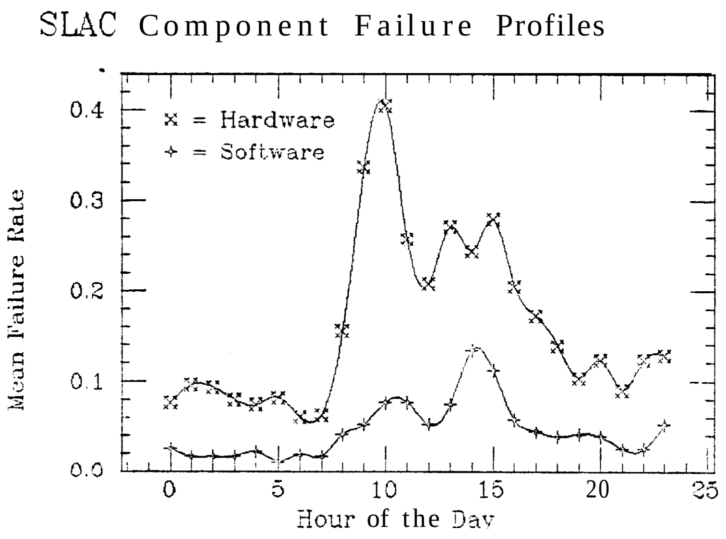


Figure 3-2: SLAC 1978 Failure Profiles

3.3 Analysis and Discussion of Preliminary Results

It is clear from the load and failure profiles that job step processing and paging rates, and, to a lesser extent EXCP counts, have some effect on failures. Regression techniques [Draper, 1966] were used in an effort to quantify this effect. Each group profile was statistically tested for correlation with one or more utilization profiles.

We computed linear, quadratic, and exponential regressions of load versus hardware and software failures. We used the same general model equation for all three types of regression; it was of the form

$$Y = \sum_{i=1}^{n} A_i X_i + B \qquad (1)$$

where Y is the failure rate (or its logarithm for exponential models), the X_i are utilization measures (and their squares in the quadratic model), and the A_i and B are coefficients. Each model was computed with one (and with multiple) utilization factor(s) as the independent variable(s). The SAS statistical analysis system [Barr, 1976] was used for testing both linear and non-linear models.

The simple or multiple *correlation coefficient,* R^2, [Draper, 1966] provides an effective measure of the goodness of the model being tested. Quite simply, it measures the amount of variability in the data which is accounted for by the model. Values of $R^2 > 0.6$

(corresponding roughly to $R>0.75$) are typically interpreted *as strong* relationships[4] [Younger, 1979]. Table 3-l gives notation and definitions for simple regression.

Table 3-1: Notation and Definitions for Simple Regression

(i= I)

X	the independent random variable (a load measure)	
$\bar{X}$	estimate of the mean of X	
X_{ij}	j^{th} observed value of X_i	
Y	the dependent random variable (failure rate)	
$\bar{Y}$	estimate of the mean of Y	
$\hat{Y}$	predicted value of Y	
Y_j	j^{th} observed value of Y	
S_{XX}	$= \sum_{j=1}^{n} (X_{ij} - \bar{X})^2$	corrected sum of squares of X (corrected for the mean)
S_{YY}	$= \sum_{j=1}^{n} (Y_j - \bar{Y})^2$	corrected sum of squares of Y (corrected for the mean)
S_{XY}	$= \sum_{j=1}^{n} (X_{ij} - \bar{X})^2 (Y_j - \bar{Y})^2$	corrected sum of cross products (corrected for the means of X and Y)
R^2	$\dfrac{(S_{XY})^2}{S_{XX}\, S_{YY}}$	correlation coeffkient

With a *linear* model, paging rate accounts for 70% of the variability in hardware component failure rate and 69% of the software rate, i.e. the correlation coefficient, R^2, for our linear model using paging rate as the only independent variable is 0.70 for hardware and 0.69 for software. Paging, CPU, and EXCPs together correlate with hardware at a 77%

[4]More specifically, the range of $|R|$ from *0* to *1* is divided as follows: *0* = no relationship, 0.25 = moderately weak. 0.5 = moderate, 0.75 = moderately strong, 1.0 = perfect.

level. The best correlations obtained with *a non-linear exponential* regression model were 0.81 for hardware and 0.82 for software. This model was our best for *software* failures. On this basis, the single variable[5] model is

$$\text{predicted software failure rate, } \hat{Y} = 0.0134\ e^{\,Paging\ *\ 0.000160}.$$

Since the high detection rate early in the morning (i.e. the 8 o'clock phenomenon) does not exist at any other time, we found it reasonable to assume that the pre-noon and post-noon usage periods behaved differently. Accordingly, we tested separate models for the two periods. The quadratic model (using the square of paging and job-step rates) was found to give the best fit. The pre-noon failures correlated at the 0.95 level while the afternoon period was at the 0.85 level.

The model in (1) (with unknown coefficient B) is estimated using a function of the mean values of the X_i and Y. Inspection of our regression data revealed points close to the origin. This led us to consider the simpler model where the intercept, B, is zero. The explicit 'assumption that B= 0, *when appropriate in the system being modeled*, allows a different algorithm to be used for the computation of A_i [Younger, 1979]. In particular, the algorithm uses the actual values of the variables (rather than deviations from their means)! This linear zero-intercept model yields much better correlations; R^2 values were 0.92, 0.90, and 0.94 for hardware, software, and combined failures, respectively. The justification we propose in support of a zero-intercept model is that equipment and software must, in general, be exercised in order for failure detection to occur. It should be kept in mind that overall utilization is a multi-dimensional quantity, $U(X_1,X_2,...,X_n)$. The function $U(X_1,X_2,...,X_n)$ depends strongly on system configuration. When $U(X_1,X_2,...,X_n)$ is zero or nearly so, the system is inactive. In such a case it is reasonable to assume an insignificant failure rate.

[5]Paging rate (page count per hour) is used for this model since it had the best correlation coefficient with software failure rate.

[6]The intercept model (B non-zero) uses the corrected sum of cross products while the through-the-origin model uses the raw sum of cross products.

Paging was the most dominant utilization measure in our 1978 SLAC analysis. This is possible because paging indirectly measures the *concurrency* of the system. An increased level of concurrency implies an increased usage of hardware and software paths, and, thus, an increase in general exercise of the system. To understand why paging rate best indicates general exercise in our study, it is necessary to briefly discuss the internal make up of the IBM OS/VS2(SVS) operating system as modified at SLAC.

svs supports a hierarchical task structure. The batch system is composed of a number of special tasks called *initiators.* Each initiator, when idle, selects a user job according to its priority and job class.[7] An important bounding criteria of a job is region size. This parameter expresses the maximum address space size for the job. At SLAC, *the* region size estimate is used in the initiator's decision whether or not to start a job. Since each initiator runs a user job to completion, the number of active initiators corresponds to the level of multiprogramming in the batch system. The higher this level the more tasks we have sharing the machine's physical memory. More sharing increases the probability that a memory reference will require a page which is not currently in memory: paging goes up as sharing goes up. Thus, higher paging rates are indicative of higher levels of multiprogramming. Clearly. as the number of independent, concurrent activities (the level of multiprogramming) increases, the overall level of system exercise also rises. As it is this overall level we seek to measure, a metric for measuring multiprogramming works well.

Note that while the paging rate at SLAC is strongly linked with failures, this should not be taken to mean that reduced paging (due to reconfiguration) will decrease failure rates. Any reconfiguration requires re-evaluation of $U(X_1,X_2,...,X_n)$. In support of this hypothesis we later show that in the last year of our study period, due to a differing job mix, CPU speed, physical memory size, and very conservative paging policy the CIT system nearly eliminated paging. We find, indeed. that another metric replaces paging rate as the best load measure correlating with CIT failures. Unfortunately, our data is not detailed enough to isolate the contribution of each system activity measure in order to find the function $U(X_1,X_2,...,X_n)$.

[7] Job classes are used to partition the work load. Typical classes indicate express, small, medium, or large jobs. They further reflect estimated CPU time, region size, and I/O line limits.

4. Some Reliability Measures

Since all failures occur within some specific component, we classify all failures as *component* failures and logically attach the failure log record to that component. The failures that cause an unscheduled interruption in system service form a subset termed *system failures.* Each UNILOG failure is categorized as either a component-only or system failure, *as* well *as new* or *recurrent.* With these categorizations we can calculate both component and system-level reliabilities.

We define *the fault tolerance level* of a system, FT, as the probability that a component failure will *not* lead to a system failure. A simple but coarse estimate of this probability is

$$FT = 1 - \frac{\text{System failures during interval } t}{\text{All failures during interval } t} \quad .$$

For SLAC in 1978, the estimated fault tolerance level was found to be 0.759.

In a similar fashion an estimate of the effectiveness of the maintenance procedures (which *we* define to be the *maintenance effectivity)* is given by the ratio:

$$E_m = \frac{\text{MTBF with } \textit{new \& recurrent} \text{ failures}}{\text{MTBF with } \textit{new} \text{ failures}}$$

From UNILOG data, we estimate the ratio for SLAC in 1978 to be 0.867.

Comparative reliability measures for the two systems are tabulated in Table 4-l. The table is broken into four sections corresponding to major categories of failures *(all* failures, *all new* failures, *all system* failures, and *new system* failures). Each such section has three rows of values. one for each of the hardware-only, software-only, and all failure types. Note that there are other failure types besides *hardware* and *software, (e.g. utility, operator induced unknown).* This explains why the percentages for hardware and software do not add up to 100%.

By making comparisons within the basic reliability data of Table 4-1, we find some

Table 4-1: Comparative Reliability Measures for CIT & SLAC in 1978-80

All failures, *new & recurrent*

	SLAC 1978			SLAC 1979-80			CIT 1978			CIT 1979-80		
Type	No.	MTBF	%	No.	MTBF	%	No.	MTBF	%	No.	MTBF	%
All	1797	4.87	100%	2064	5.66	100%	1124	7.80	100%	1607	7.28	100%
H/W	1318	6.64	73.3%	1395	8.38	67.6%	780	11.17	69.4%	1002	11.68	62.3%
S/W	388	22.6	21.6%	641	18.26	31.1%	244	35.55	21.7%	302	38.38	18.8%

All failures, *new* only

	SLAC 1978			SLAC 1979-80			CIT 1978			CIT 1979-80		
Type	No.	MTBF	%	No.	MTBF	%	No.	MTBF	%	No.	MTBF	%
Ail	1610	5.43	100%	1806	6.48	100%	882	9.94	100%	1275	9.17	100%
H/W	1189	7.36	73.8%	1190	4.33	65.9%	612	14.16	69.4%	781	14.98	61.2%
S/W	339	25.8	21.0%	589	19.87	32.6%	195	44.53	22.1%	242	47.94	19.0%

System failures, *new & recurrent*

	SLAC 1978			SLAC 1979-80			CIT 1978			CIT 1979-80		
Type	No.	MTBF	%	No.	MTBF	%	No.	MTBF	%	No.	MTBF	%
All	433	20.2	100%	505	23.19	100%	392	22.40	100%	653	17.70	100%
H/W	154	56.5	35.4%	129	90.28	25.5%	176	47.93	44.9%	322	36.15	49.2%
S/W	237	36.6	54.8%	354	33.10	70.1%	192	45.23	49.0%	248	47.77	37.9%

System failures, *new* only

	SLAC 1978			SLAC 1979-80			CIT 1978			CIT 1979-80		
Type	No.	MTBF	%	No.	MTBF	%	No.	MTBF	%	No.	MTBF	%
All	376	23.3	100%	445	26.30	100%	297	29.59	100%	509	22.78	100%
H/W	133	65.5	35.4%	106	110.1	23.8%	119	71.08	40.0%	233	50.00	45.7%
S/W	206	42.0	54.5%	318	36.86	71.5%	157	55.37	52.9%	202	57.47	39.6%

important trends. Because of age and quantity of equipment, the SLAC components together are more prone to failure than those at CIT. In spite of this fact, the SLAC system is *more* reliable than CIT (MTBF: 23.2 hours vs. 17.7 at CIT). Over the three years studied, the SLAC system as a whole has grown more reliable (by approximately 15% based upon MTBF) while the CIT system has become slightly less reliable (6%).

An important fact is that between the 1978 and the 1979180 study periods the CIT system was extensively upgraded. The major component, an IBM 3033, is of a newer, more reliable technology than SLAC's 168s and 360/91. The peripherals, particularly the disks, are also newer at CIT.* The results of this upgrade show an increase in hardware *component-level* reliability by approximately 5% but a net decrease of nearly 25% in *system* reliability for failures attributed to hardware. Thus, while the new components exhibit marginally better failure rates, the structure of the new system seems to be less fault tolerant. The FT measures for the two systems are:

$$\text{SLAC } FT_{hardware}\text{: } 0.89 \text{ (197X)} \Rightarrow 0.91 \text{ (1979/80)}$$

$$\text{CIT } FT_{hardware}\text{: } 0.77 \text{ (1978)} \Rightarrow 0.68 \text{ (1979/80)}$$

We attribute this to the fact that a tight linkage exists between peripheral equipment and the central processor at CIT. The SLAC system, on the other hand, is much more loosely connected. A great deal of the computational load is served by the background batch machines, those processors currently configured as pure batch stream servers. Many unit record device failures (i.e. line printers, plotters, card readers) and failures in software resident in the front-end processor simply do not affect the work flow in these background machines. In like manner, failures within these machines do not affect the rest of the SLAC system. This accounts for the system's relatively high fault tolerance (FT) measure.

The software components of our systems tend to be critical resources upon which the overall system reliability depends. CIT's software is measurably more reliable (358 total failures versus 641 at SLAC in the same interval). However, since the CIT system is a

[8]We allowed four months of operation with the new equipment before starting to gather failure data. Presumably. most of the initial problems with the new system were ironed-out during that period.

uniprocessor, it is much less tolerant of software failures than SLAC. This is clearly indicated by the FT measures for the two systems.

SLAC $\mathbf{FT}_{\mathit{software}}$: 0.39 (1978) $\Rightarrow$ 0.45 (1979/80)

CIT $\mathbf{FT}_{\mathit{software}}$: 0.21 (1978) $\Rightarrow$ 0.29 (1979180)

The *system* is more likely to fail due to a software component failure at CIT than at SLAC.

5. Detailed Analysis

In order to put our results in a more general perspective, we studied another system. In particular we wished to determine whether the strong correlations exist only at SLAC or if they may be evident on other large computer complexes. Additionally, it was important to test the assumption that the "virtual day" load profiles were representative of the load *at time offailure.* Finally, our preliminary study showed that all failures taken together are correlated with load. The question 'arises whether our results are biased due to *dominant* effects of one component type or if the relationship with load is more general and across the board. For example, we wish to know whether CPU and memory failures are also load dependent.

In our main study we analyzed a second system (CIT) for 1978 and subsequently performed detailed analyses for the two systems for a different period (a 16 month period starting 1 September, 1979). A 16 month period was needed in order to get a statistically significant amount of failure data on a component by component basis. Additionally, the combined effects of a differing sample size and sampling interval tend to make our statistics more reliable.

5.1 Validation of Preliminary Study Results

Since our SLAC study was for 1978, we thought it appropriate to use data for the same year in our CIT study. Our objectives were

- to test whether or not strong correlations exist between failures and load;
- to determine the best single measure of computer system utilization;
- to offer feasible explanations for any significant differences found with respect to the SLAC results.

During 1978, the CIT system used an IBM 370/168 host. In order to make a fair comparison with our preliminary 'analysis, we used the programs from our SLAC study to produce failure and load profiles. As in our preliminary study, we computed linear zero-intercept regression models. Table 5-l gives the correlation results. The very strong correlations found were quite encouraging.

Table 5-1: CIT 1978 Load-Failure Correlations

Dependent Variable	Independent Variable(s)	Correlation Coefficient, R^2
Hardware Failure Rate	EXCPS, CPU	0.93
Software Failure Rate	PAGING, EXCPS, CPU	0.91
Hardware & Software	PAGING, EXCPS	0.93

Input/output activity (measured by the variable EXCPS) and paging rate were the best utilization measures for the 1978 CIT models'. Figure 5-l displays these load profiles; there is significant similarity with those of SLAC.

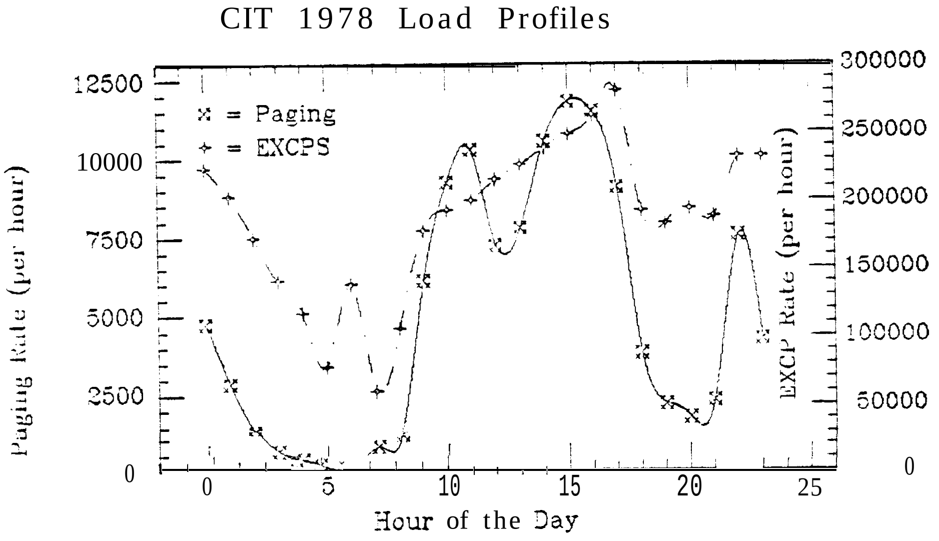


Figure 5-1: CIT 1978 Load Profiles

Our second objective was to determine if the virtual day profiles were, indeed, representative of the load at the time of failures. To do this we needed to show that the load at the time of failure was, on the average, very close to the virtual day load values for the corresponding hour. Accordingly, we computed the average load readings during a

[9] Unfortunately. these are not the same two measures that were found dominant for SLAC. For reference, all load profiles are given in Appendix II.

one-hour interval preceding each UNILOG failure. Based on the time-stamp of each failure record, we retrieved the *exact* load measures from the performance/load database. The only data conveniently accessible in this mode was system paging rate and system CPU utilization. These values are averaged, over approximate lo-minute intervals by the operating system before logging. We computed the average load over the one-hour immediately preceding each failure and attached it to the UNILOG record. In order to facilitate comparison with the virtual day, we computed profiles of this failure-time load data; Figure 5-2 depicts these for SLAC in 1978. The procedure used is further described in Appendix 1.5.

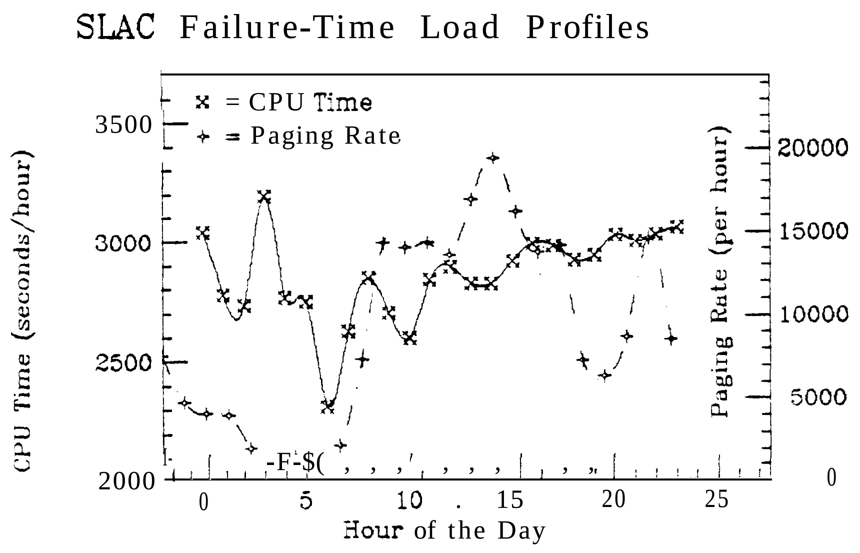


Figure 5-2: SLAC 1978 Failure-Time CPU and Paging Profiles

The curve shapes are observed to be similar to our virtual day profiles”, thus justifying the use of the virtual day in our regressions. It is important to notice also that the entire curve of CPU utilization (in seconds per hour per CPU) is above the 80% level. On the average the load was quite heavy (as measured by central processor utilization alone) at the time of component failure.

10The spike in failure-time CPU load at 3 AM is due, in large part. to periodically scheduled CPU diagnostics.

5.2 Component-Level Modelling

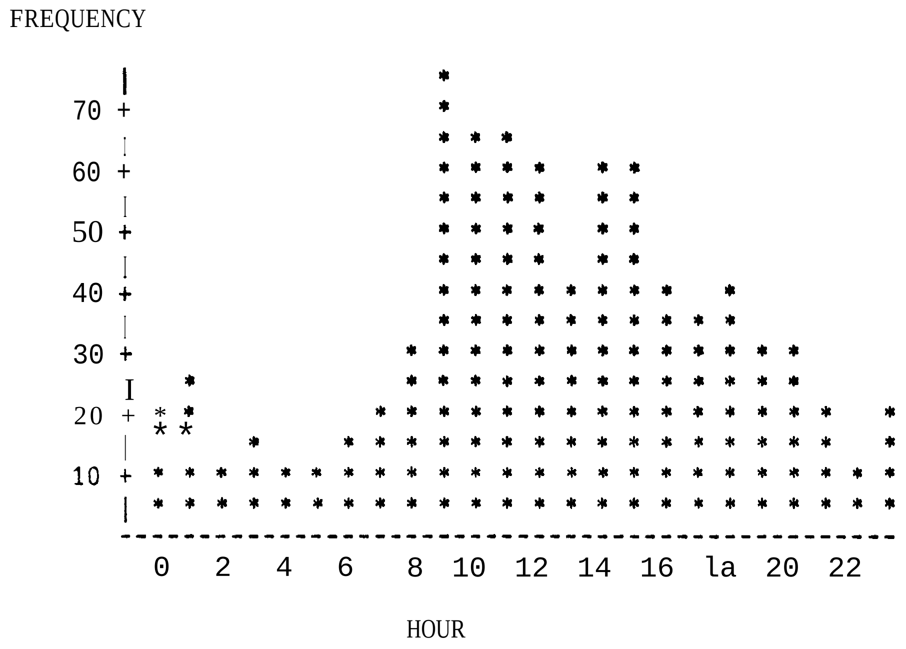
The final objective of our more detailed analysis was to investigate if strong correlations exist at the *component* level in our systems. We grouped UNILOG failures into *component types* according to equipment model numbers, e.g. IBM *2314, 3330,* and 3350 were placed in the component type "DISK". Components types with similar usage and potential influence on failures in the system were classed together as *usage groups.* For example, usage group "MEDIA" is composed of component types "DISK", "TAPE", "PRINT", "CARD", and "RLTM" (real-time experimental station equipment). Appendix III lists the complete set of component types and usage groups. For each component type and usage group, we computed failure profiles (refer to Appendix III). Figure 5-3 presents sample failure histograms for two representative usage groups at SLAC: MEDIA and CPU.

In determining load profiles we used *system-level* data in addition to our preliminary batch job-step data. This eliminated any possible bias due to a batch-only view of the system. -The new measures, system wait time (unused CPU time) and system paging rate, were obtained from the SMF system wait record. Note that in subsequent tables these new system-level measures are indicated by SYSCPU and SYSPAGING, respectively. Table 5-2 gives correlation results for linear zero-intercept regression models. Parenthesized values next to the first independent variable indicate the correlation coefficient of that factor alone with respect to the dependent variable.

The importance of the correlation results of Table 5-2 lies in the fact that the group failure rates each *individually* follow load.

We have found no single dominant device type or usage group at SLAC or CIT. The general relationship between failures and load appears to exist *across the board.* This result was a goal of our detailed analysis. The high correlation of each usage group across both our systems gives strong evidence of a general load versus failure relationship. We proceed by offering qualitative explanations for the effects.

SLAC 1979/80 Failure Histogram for Usage Group 'MEDIA'



SLAC 1979/80 Failure Histogram for Usage Group 'CPU'

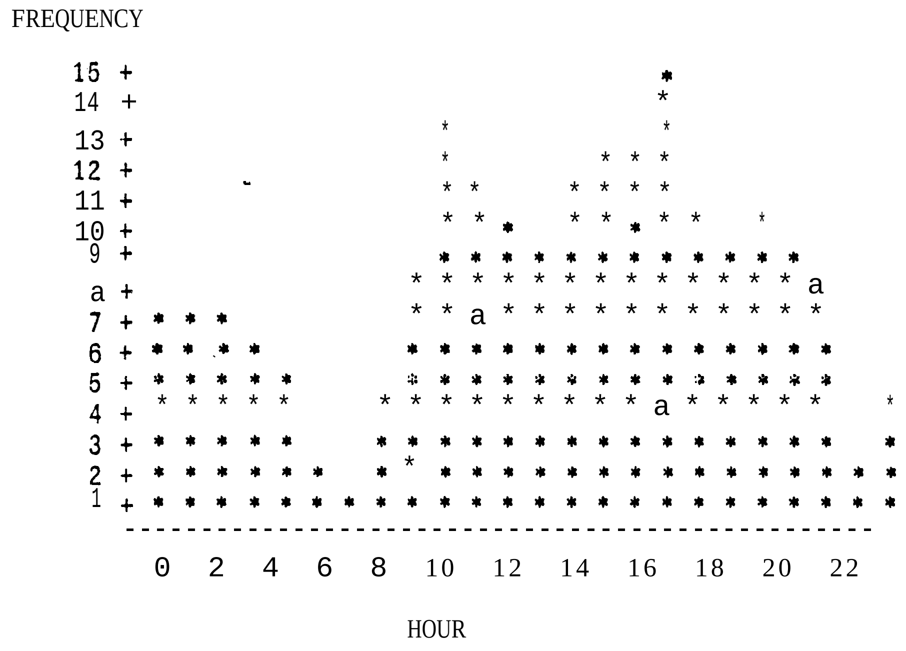


Figure 5-3: SLAC MEDIA and CPU Failure Histograms

Table 5-2: Regression Results by Usage Groups -- 1979180 SLAC & CIT

CIT 1979/80

Dependent Variable (group failure rate)	Independent Variables	Correlation Coefficient, R^2
CPU .	JOBSTEPS(.63), PAGING	0.72
CPU" §	SYSCPU(.80), SYSPAGING	0.81
MDCTL §	SYSCPU(.90), JOBSTEPS	0.91
MEDIA	SYSCPU(.86), PAGING	0.89
MEMORY §	SYSCPU(.75), EXCPS	0.76
SOFTWARE	SYSCPU(.81), PAGING	0.82
TLCHAN	SYSCPU(.79), SYSPAGTNG	0.87

§ Not a significant sample size.

SLAC 1979/80

Dependent Variable (group failure rate)	Independent Variables	Correlation Coefficient. R^2
CPU	JOBSTEPS(.92), SYSCPU	0.94
CPU* §	JOBSTEPS(.73), SYSCPU	0.75
MDCTL	SYSCPU(.88), EXCPs	0.90
MEDIA	JOBSTEPS(.91), SYSCPU	0.91
MEMORY §	SYSCPU(.78), PAGING	0.79
SOFTWARE	JOBSTEPS(.92), SYSCPU	0.94
TLCHAN	SYSCPU(.79), JOBSTEPS	0.83

§ Not a significant sample size.

6. Views on Observed Results

6.1 Discussion of Measured Results

Before discussing our results, it is important to consider the differences in environment and usage in our two systems. The CIT system is oriented more toward production (with a heavy I/O component) and general *user* services than SLAC. Accordingly, the load at CIT is more interactive, less CPU-bound than SLAC. Recall that the SLAC computer center supports a physics research community while CIT's users have much more generally distributed coniputational needs. In terms of peripheral complements, the two systems are largely equivalent. We see in excess of twenty large-capacity disk units at each site along with more than ten high-performance magnetic tape drives and several unit record devices. The fundamental architectural difference between the two systems lies in the *connection* of peripherals to central processing facilities. At CIT, a tight linkage exists between peripheral equipment and the on-going computational load of the uniprocessor system. The SLAC system, on the other hand, is relatively loosely coupled with two largely autonomous background processors *dedicated* to batch service.

As we look at the trends for the two computer systems across the three-year span of our study, one substantive effect is clear. The failure profiles at CIT have changed significantly between the 1978 and 1979/80 study periods. In mid-1979, the system was connected to a nationwide access network and accounts were made available to east coast universities and research groups. As a result, the sharp rise in failures previously seen at 8 AM became spread with a substantial effect at 5 AM *(8* AM-EST) followed by a gradual build-up until the local load start-up at 8 AM-PST. As would be predicted by our hypothesis, the relationship between load and failures continues to be strong.

We find strong correlations across the board for each component type and usage group. The measures correlating the strongest at the component level, however, are not the same at SLAC and CIT. Because of the multi-dimensional complexity of work flow within a large computer system, no one measure of utilization can describe the situation adequately. It appears that system CPU usage and job step processing rate together provide the best aggregate measure of system load (refer to Figure 5-2). We conjecture that this is so because of the *type* of exercise depicted by job step processing rate as compared with the type for CPU usage. The system can be modelled as a load-flow graph [Shooman, 1968] wherein we have increased path utilization as the system load increases. Following such a hypothesized graph model with failure discovery rate approximately linear with the fraction of the graph in exercise, we would expect the load that most completely exercises the CPU to correlate with the highest CPU failure rate. Measuring just CPU usage rate tells us little about the diversity of instruction types or the completeness of exercise of circuitry within the CPU. Job step rate, on the other hand, indirectly measures the system concurrency and, thus, is greatest when the overall system load is worst case. This tends to be late afternoon with much I/O interrupt activity and maximum CPU multiplex rate (among ready user jobs). This time is the period of most complete exercise of the CPU. The raw measure of CPU usage indicates that midnight is heaviest. This corresponds to the release and processing of large CPU and r/o-bound batch jobs.

During the three-year study, both systems were reconfigured to optimize throughput. Memory was added. region sizes were adjusted, dedicated and virtual memory allocation schemes were tuned. The effect has been a dramatic decrease in paging rate. Accordingly, that metric is no longer the best index of system activity. It does, however, continue to measure interactive load. Via tuning, its effect has been minimized in our subject systems, but it still appears as a second or third choice measure in some regression models. It is important to point out that there is some degree of overlap in all of our measures. Accordingly, one measure is sometimes selected over another for a better correlation.

6.2 Conjectures on the Model

Clearly, some type of relationship exists between system load and failures. We believe there to be several effects interacting to produce the observed phenomena. The first of these effects is the well-known constant failure rate Poisson model which is essentially load-independent. Such a model, however, cannot by itself entirely explain the effects we have observed at SLAC and CIT. We hypothesize that the other effects are *load-induced* and discuss them in the following sections.

6.2.1 Latent Discovery Effect

As failures in equipment within our systems occur, they must be *detected* in order to affect the statistics. Many failures can only be detected when the particular module or subsystem of which they are a part is exercised. Even if failures may not be caused by increased utilization, they are *revealed* by this factor. The time between the occurrence of a failure and its manifestation as a system error has been referred to as the "error latency" [Shedletsky, 1973].

This latent discovery effect allows us to explain the "8 o'clock phenomenon"; viz. the failure rates rise strongly with increased load at 8 AM and drop during lunch, then never get up as high again even though the load may rise to a level higher than the pre-lunch peak. Resident failures (latent errors) from the early morning lower-utilization hours are revealed after the system utilization starts to increase at 8 AM. The strong correlations in the pre-noon period show that latent discovery is an important factor. The continued, although somewhat weaker, correlations in the afternoon strongly suggest that latency is not the *only* effect. The constant failure rate model does not itself significantly correlate with load; it serves to explain the observed failures during low usage periods and provides the undetected failures that are revealed as latent errors after utilization increases.

6.2.2 Load-Induced Hardware Failures

There are three environmental aspects of computer systems that are effected by utilization and that, themselves, have an effect on hardware failures.

- Electronic noise . . . as more activity is taking place per unit volume, there is more electronic noise in the environment. This noise adds to the normally present variations in the analog signals which are interpreted as digital logic levels. A higher noise level contributes to a higher rate of faulty detections in memories and to timing and synchronization anomalies on busses and in cables.

- Temperature rise . . . as the activity per unit volume increases within a computer system, so does the number of average exercises per second of any given device (even within chips). Higher rates of exercise lead to higher temperatures at the devices themselves. Since temperature affects semiconductors in a variety of ways, we can only point out that statistically we expect to see higher failure rates at higher temperatures.

- Mechnical stress, vibration . . . many peripheral devices rely upon mechanical assemblies for their correct operation. Vibration in mechanical systems is largely equivalent to noise in electronic equipment. We believe that mechanical stress and vibration behave statistically as temperature and noise in semiconductor devices.

These effects are primarily significant for marginal devices. The existence of such devices can also explain transient errors, i.e. situations where under normal operating conditions all components perform correctly but, ocassionally, non-repeatable failures occur [McConnell, 1979].

6.2.3 Load-Induced Software Failures

Many of the typical software design and implementation errors [Myers, 1978] are present at SLAC and CIT. These problems divide into two groups, those triggered under high-load and those that are load-independent but appear to be load-induced because of an increased execution time effect [Musa, 1980]. Because our systems are quite mature, however, errors in the latter group have in large part been repaired. We conjecture that there are four general phenomena that affect software and explain sensitivity to system loads. These are

- Interrupt handlers or device drivers may be sensitive to fluctuations in time or sequence of events.
- Human interface effects and interactions with transient phenomena.
- Mishandled, load-induced hardware failures.
- Typical software design and implementation errors (two classes: load-dependent and load-independent)

Low-level operating system software, in particular that which handles I/O and interruptions is time and sequence critical. Out-of-sequence interrupts can cause unexpected execution of code. Due to the time-critical nature of interrupt-time software and the ever-present pressure for increased throughput, checks for the miriad of possible errant sequences are usually not made. Instead, the channel programs and interrupt-time routines are made as robust and error tolerant as is possible and economically feasible. The peaks of very high load cause a statistically higher software failure rate due to the increased stress on these programs and, thus, the increased probability of unsolicited, missed, or multiple interrupts occurring. With the use of data concentrators, local and nationwide network access, and other computer front-end equipment, the host's interrupt-time software is even more likely to see wide fluctuations in *timing* and, in some cases, *sequence* of interrupts. Such fluctuations are purely load-induced. If some software is sensitive to such fluctuations and a failure results, then the failure will appear to be due to load.

Often a transient failure or other unknown problem will result in the need for an operator to manually reset a piece of equipment. Such operator actions can sometimes result in software failures in the SLAC and CIT systems. The hardware reset action is intended to clear the error condition, but often the action confuses the software and results in a software failure. There are varying levels of reset that may be performed, including control console requests to the software. operation of a reset/clear switch on the equipment, or manual cycling of the power to the equipment. Some reset switches accomplish their mission internally by momentarily removing power from the logic so that its built-in start-up cycle performs the initialization. Such actions are rather harsh in an operating environment and can affect the correct operation of near-by equipment. The availability of software interfaces for carefully clearing such situations would reduce the failures rates in this area.

The class of software errors triggered under high-load include array bounds violations, queue overflow/underflow, accessing memory via pointers after memory had been freed, etc. Some of these only can occur when loads are high. Others occur constantly, but have no undesired effects; they go un-noticed until the system becomes heavily loaded.

Due to the organization of the SLAC and CIT software, many events which are logged as software failures are probably caused by the inability of software to adequately respond to *hardware* malfunctions. As we have mentioned earlier, there are two major interactive subsystems (WYLBUR and ORVYL) and a background batch system. Both WYLBUR and ORVYL were developed at Stanford. During the period of their development, the 2301 drums (upon which they "page" user files and programs) were extremely reliable. The equipment literally did not fail in a four year period. Consequently, correct operation of WYLBUR and ORVYL depends quite heavily upon the reliable operation of the paging devices (now largely 230%). Failures in these devices can cause the software to crash or abort. Policies such as the first-fit memory allocation technique used in ORVYL result in low utilization of certain sections of the paging devices (except during high-load periods). Hence, latent paging device failures can cause what appears to be load-induced software failures.

Many software failures are in an area of the code involved with *error condition handling.* Preliminary software testing is often functional testing and may not involve simulation of all error conditions. Thus, when a hardware failure or transient causes such an error condition to arise, the software fails. In this case, the problem lies in software, but the discovery is triggered by a hardware failure.

7. Probabilistic Models

In this section we propose a simple probabilistic model to describe the increase in failure rate with system load as measured by one of the proposed utilization factors. This allows us to estimate the underlying statistical relationship between the two variables.

We note from our regression model that the mean failure rate is linear with utilization (viz. system CPU); i.e. the slope of the line which is the ratio

$$\frac{\text{Failure Rate}}{\text{System CPU Usage Rate}} = \text{Failures per unit CPU usage}$$

is constant. We refer to this ratio as the *utilization or load-inducedfailure rate* (λ_u). As a consequence, we may employ an exponential distribution to describe the probability of a utilization-induced failure: i.e. we have the cumulative distribution function

$$F_{utilization}(U) = 1 - e^{-\lambda_u U}$$

where "U" denotes the utilization factor (in our case system CPU usage rate).

This can also be confirmed through a Kolmogorov-Smimov Test [Daniel, 1978] on UNILOG and SMF data. For CIT in 1979/80, we obtain

$$\lambda_u \text{ (hardware)} = 2.28 \times 10^{-2} \text{ failures/unit CPU}$$

$$\lambda_u \text{ (software)} = 4.62 \times 10^{-3} \text{ failures/unit CPU}$$

It is important to note that the overall failure rate still follows an exponential distribution (now with load as a parameter), a fact which allows classical models to be used in reliability computations.

Alternatively, we could view the overall failure rate as being composed of two separate quantities. The first is the system *inherent* failure rate, *Z(t)*, as determined through classical reliability models. If this is constant during the useful life of the system, the associated failure cumulative distribution is 1- $e^{-\lambda_i t}$.

The second is the *utilization-induced* failure rate depicted by the regression and probabilistic models. This failure rate is dependent upon system utilization and, hence, is cyclic on a daily basis.

Thus, the overall cumulative probability of system failure (assuming independence) is

$$F(t,U) = I - e^{-\lambda_i t} e^{-\lambda_u U}.$$

Figure 7-1 depicts these effects on the conventional "bathtub curve". Note that the observed range in failure rate is large (5:1).

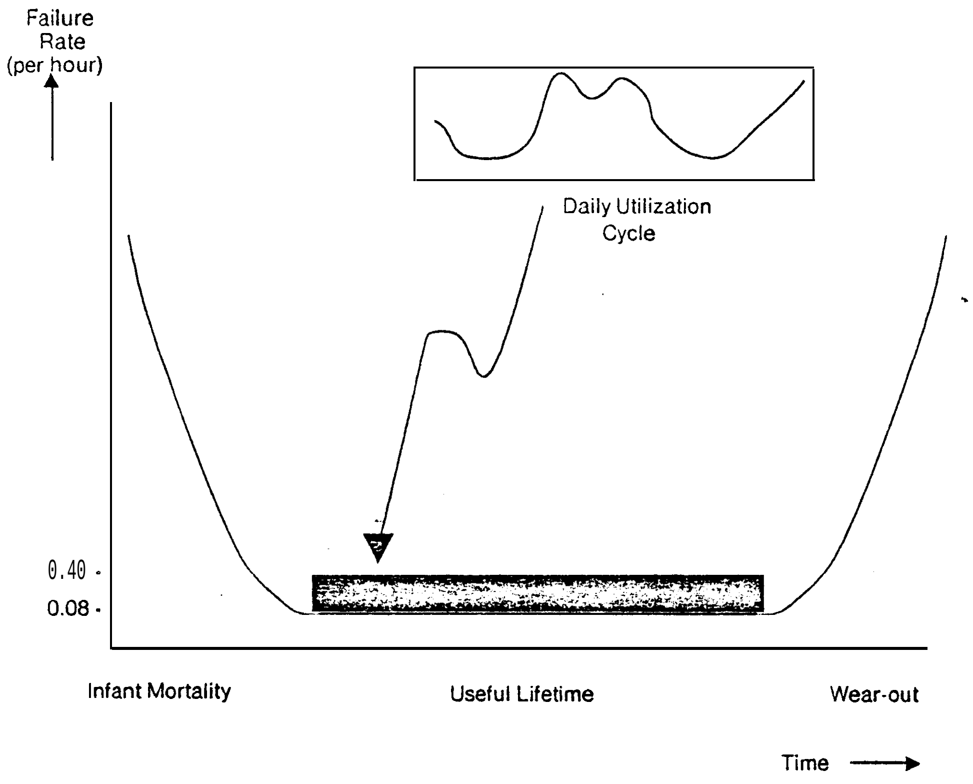


Figure 7-1: Augmented Bathtub Curve

8. Concluding Remarks

We have studied three years of real data on reliability and performance for two major computer complexes at Stanford University. We find strong correlations between observed failure rates (both for hardware and software components) and the average values of several measures of system utilization. This behavior is across all component types (e.g. media, CPU, telecommunications). During the course of our study, significant changes to the systems occurred, (e.g. major acquistion of new equipment, substantial system reconfigurations, tuning changes for optimized performance, user load pattern shifts) and in all cases, the failure profiles (both at the system and component-level) continued to correlate strongly with average measures of load. The observed changes did, however, cause different measures of load to be most significant in our correlations.

Additional substantiation of our studies can be deduced from the results presented in [Castillo, 1980]. In their studies toward developing a Poisson model for a varying load environment, the authors found that their model was only appropriate *at specific times of day*, i.e. at particular load levels. They also found a four-to-one difference between minimum and maximum failure rates which they attribute, in major part, to load.

We made some simplifying assumptions regarding the effect of averaging (viz. the virtual day profile). Our detailed analysis validated these assumptions and, in addition, indicated slightly higher than average load levels at time of failure. There was a strong agreement in shape with our virtual day profiles.

Although we are not in a position to state exactly the reasons for this observed dependency, there is sufficient grounds for speculation. Our proposed models include several combined effects: a load-independent Poisson model with latent discovery effect and load-induced effects for both hardware and software failures. It should be noted that the latent discovery effect which is predominate in new software will be minimal in a mature piece of software of the type we studied. It is our belief that other problems such as timing, synchronization, and inadequate software processing of hardware failures are the leading causes of software malfunction.

It is clear from our results that more detailed investigations are necessary and worthwhile, In particular it is important that lower-level analyses of both hardware and software failures be conducted in order to learn more about the *mechanisms* of fai lure. More challenging would be the development of simple analytical models to represent the load-failure dependency. These would be invaluable for comparisons with the statistical models.

There are many potential uses of reliability and performance models in computer design. One straightforward application is in making reliability projections under changing load situations. The Stanford Center for Information Technology is planning to implement a continuing program based on our methodology for use in future reliability and performance evaluations.

A particularly crucial and challenging component of future work involves the mathematical formulation of the multi-dimensional utilization function. $U(X_1, X_2, \ldots, X_n)$, which relates the many varied measures of load to the single concept of *system activity*. Our data reflects a number of performance optimization attempts. In most cases, the performance increased, but the failure rate followed. Clearly, we cannot continue the push for ever higher performance without considering reliability. Since increased performance involves higher utilization, there exists a strong tendency for increased failure rates. A more detailed understanding of the utilization function, $U(X_1, X_2, \ldots, X_n)$, would allow the simultaneous optimization of both performance *and reliability* in large, general-purpose time-sharing systems.

Acknowledgements

This work was supported in part by the National Science Foundation under grant MCS-7904864, the Air Force Office of Scientific Research under Contract F49620-79-C-0069 and by CSIRO (Australia). The resources and assistance given freely by SLAC and CIT personnel are also greatly appreciated. In particular, we acknowledge Ted Johnstoti, Richard Carr, and Dave Rossetti at SLAC, and Larry Rivers, Bill Bradley, and Barry Benight at CIT for their help.

Appendix I. Preprocessing Steps

1.1 Raw Data

The raw data for our analysis came from two distinct sources: the operator-maintained failure and event log (UNILOG) and the operating system-generated performance and accounting records (SMF). For a number of reasons, the raw data was not directly useable.

- The UNILOG data was not kept on-line. Two or three times per year the dataset containing the log was archived to magnetic tape. Thus, a one-time collection process was performed to retrieve the archived UNILOG data and organize it as a single on-line dataset.

- The UNILOG serves a number of purposes and, as such, contains records of significant events other than failures. Some editing of the collected log was performed to discard non-failure data. During this process. the overall UNILOG record was reshaped to make it more manageable. Fields containing data not pertaining to our study were eliminated. The result was a card image (SO byte) record with date/time stamps for down and up time, the operator-assigned failure *t,vpe code,* the component type and component address, an indication of new/recurrent and whether or not the system failed due to the component failure. There were approximately 1200 records for each year of activity.

- The SMF records of interest to our study were of four types: the system IPL record (type 0), the system wait time record (type 1), the batch user job step record (type 4), and the system end-of-day record (type 12).

- These records were interspersed with other SMF accounting and performance data and, thus, had to be selected from a large dataset. This effort was complicated due to the volume of records and the fact that they were spread across many magnetic tapes, each one holding from 24 days to one month of system SMF data. At SLAC, the data from three distinct CPUs was merged together in the same dataset.

1.2 Required Data Attributes

The requirements for the data to be used in the processing steps of our study were:

1. Dataset(s) should be on-line. This meant they must be of a size suitable for disk storage rather than the enormous multi-volume datasets we began with.

2. Records within datasets should be of a single fixed layout. This was not true of SMF records which are of fifty differing types, each with a unique record layout.

3. Data should be efficiently processable. Since many statistical runs would be made, it was important to be able to read-in the data efficiently without having to select the interesting portions while passing over unrelated records.

1.3 Processing Techniques

In our preliminary study we computed a "virtual day" by obtaining the average value of four load measures for each hour of the day. The data used was SMF batch user job step records (type 4) for SLAC in the year 1978. The processing steps performed were:

1. Month by month, select the type 4 records from the appropriate archived tape volume and place the datasets on disk.[11]

2. For each month's job step records (approximately 125,000 of them) extract the paging, CPU time, elapsed time. and I/O start-ups (the so-called "EXCPs"). Sum these data and maintain a count of the observations for eventual averaging.

3. After all months are run. combine the results by computing the arithmetic mean and standard deviation. Obtain 24 observations (one for each hour of the day). Each observation consists of HOUR, JOBSTEPS, PAGING, EXCPS, and CPU variables.

4. Save the 24 points as our SLAC 1978 virtual day load profiles and plot the four separate load measures versus time (HOUR).

Our follow-on study was to be more detailed. We wished to perform a similar virtual day-style analysis on SLAC 1979-80 and CIT 1979-80 data. During the more detailed study, we hoped to validate our preliminary findings. The added detail came from the *system* level SMF records. Recall that our preliminary work was purely based upon batch user job step data (type 4 SMF records). Much other activity was occurring in the computer system in addition to the batch service. For example, the interactive front end edit/fetch/submit processing (WYLBUR) and the timesharing component (ORVYL) presented a substantial load on the system.

[11] A general-purpose SMF record selection program written by D. X. Johnson at SLAC was used for this purpose.

To validate our preliminary findings and more accurately depict overall system activity, we chose to process the system level load records. These measure the total utilization of the CPU and provide paging counts for the entire system.

The processing of these records was more involved than that performed in our preliminary study. Most of the complexity arose from the way in which SMF system statistics are collected. The IBM operating system software collects statistics on overall performance in terms of a *time interval.* This interval is nominally ten minutes, but its exact value varies considerably and *is not stored* within the record. The interval can only be discovered by remembering the time stamp from the previous system statistics record. The time stamp contains only time-of-day (with no *date* information) and records are (obviously) not written while the CPU is halted even though the time of day clock continues to run. These facts make it important to recognize system halts (type 12 end-of-day records) and IPLs (type 0 records) and to incorporate that information into the computation of the exact interval. Since SLAC has three CPUs, each with their own clock (and, thus, their own version of time), and since the SLAC SMF data contains all three CPUs data intermixed, a program was written to extract the required information.

1.4 The UNIMERGE Program

Once the decision to write a special-purpose program was made, it became convenient to perform two relatively unrelated processing steps in the same pass through the SMF data. The program, UNIMERGE, computes hourly averages by CPU of system wait time and paging rates and merges UNILOG failure data with smoothed and lagged load data.

The averaging is straightforward and involves only summing and counting the data involved. This *is* easily done once the *interval* has been determined. It is necessary to compute the interval since the measured values of CPU wait time and paging counts must be converted to *rates* and, therefore, must be divided by the measurement interval.

The interval is computed by logging the time/date of the last event on the current CPU and correctly processing system halts and IPLS (also by CPU). This is, in principle. not difficult except for the fact that the 360/91 has a different operating system and a less-detailed time-stamping method. In fact, on the 360/91 the exact measurement interval *cannot be precisely determined* and must be “guessed” by examining the “time-written” field of the record and computing the total number of whole ten minute intervals that elapsed since the previous 360/91 record. Following this heuristic, it was necessary to do reasonability checks and log/discard suspect data. On the average, two to three suspect intervals were reported per month of triplex data

1.5 Lagged Data

In order to study the precise system load immediately preceding a component or system failure, it was necessary to merge the time-ordered datasets containing failure and load data. For each UNTLOG failure record, we wished to obtain the CPU wait time and system paging rate at the time of failure. We also wished to record average values of the two load measures during one, two, and three-hour intervals preceding the failure.

The UNIMERGE program computes the desired values by keeping a “lag buffer” of readings for each CPU as the SMF data is processed. As each SMF record is obtained, readings are recorded and stored into the (circular) lag buffer until the time-stamp of the latest record indicates a time past the time of the current UNILOG failure. Then, the lag buffer values for each CPU are used to compute the average values during the one, two, and three-hour lag intervals. Finally, the merged failure/load record is written to the output dataset and the process is repeated for the next UNILOG failure record.

When all failure records have been merged, the UNIMERGE program terminates. The lagged data can then be used to obtain load profiles indicating the measured system utilization *at the time of failures.* This is done as a validation that the virtual day style of ‘analysis is valid. Also, more intensive studies of failure as a function of load are possible.

Appendix II. SLAC and CIT Load Profiles

This appendix contains the virtual day load profiles[12] for the two systems and the two study periods (1978, 1979/80). The smooth curves are presented only as an aid to visualization; the actual data points are plotted with markers. Each graph consists of two profiles, one for CIT and one for SLAC.

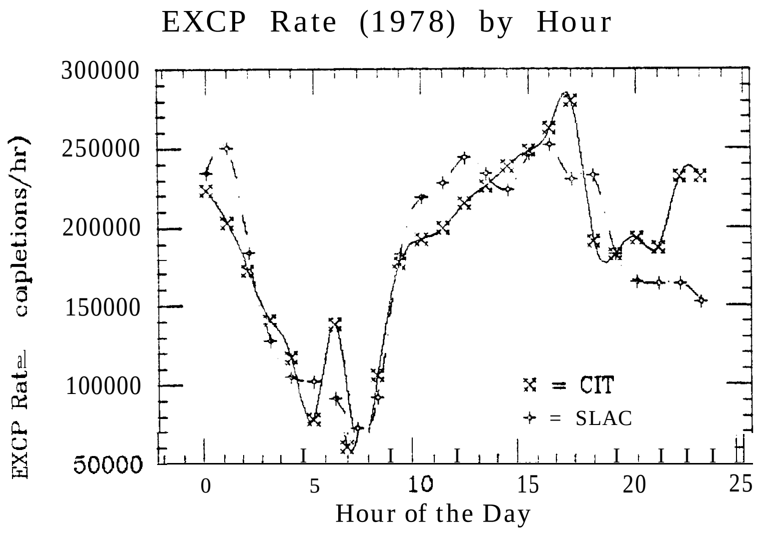


Figure II-l: 1978 EXCP Profiles

[12]The data presented within this Appendix is provided by and property of Leland Stanford, Jr. University.

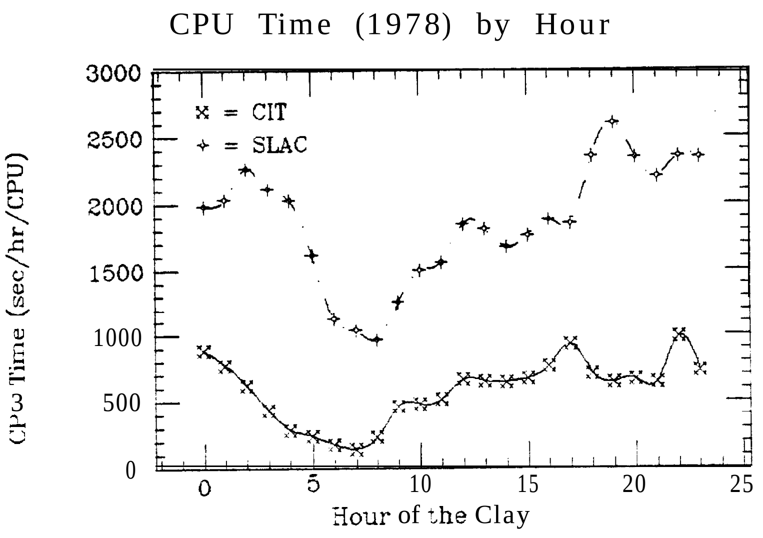


Figure II-2: 1978 Batch CPU Profiles

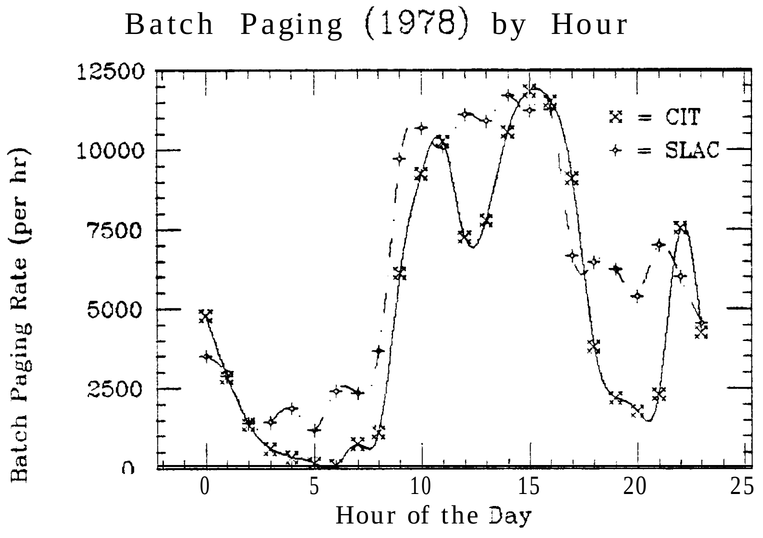


Figure II-3: 1978 Batch Paging Profiles

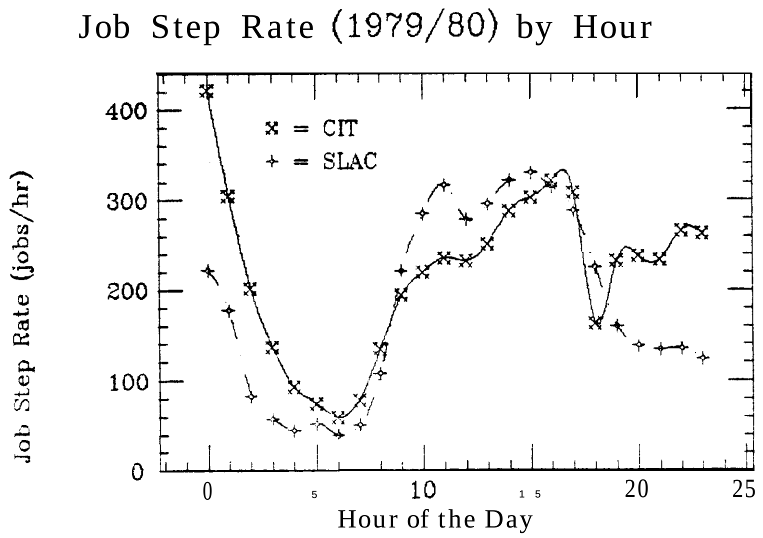


Figure II-4: 1979180 Job Step Profiles

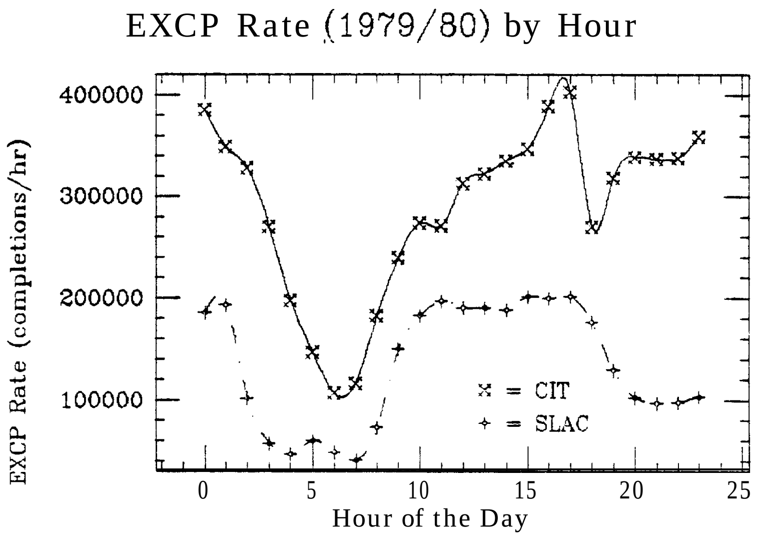


Figure II-5: 1979/80 EXCP Profiles

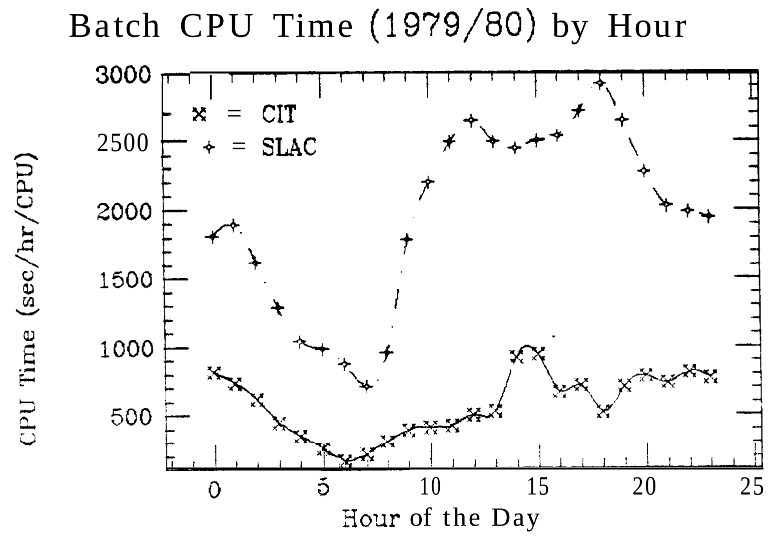


Figure II-6: 1979180 Batch CPU Profiles

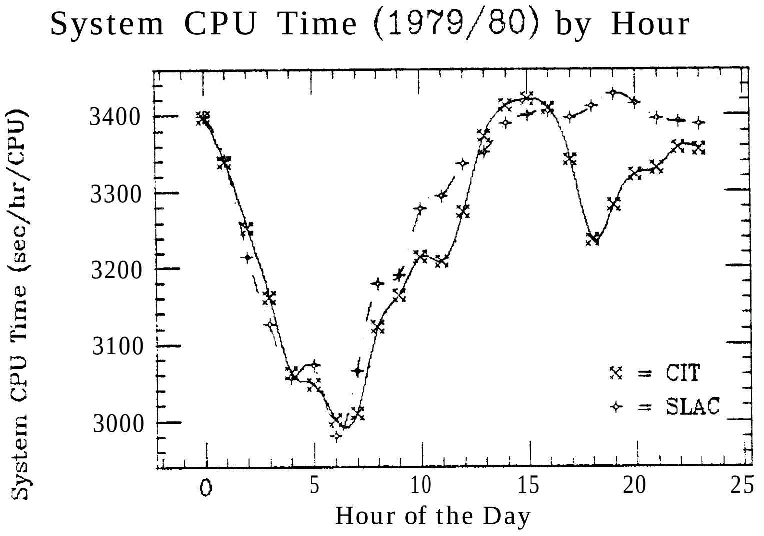


Figure II-7: 1979/80 System CPU Profiles

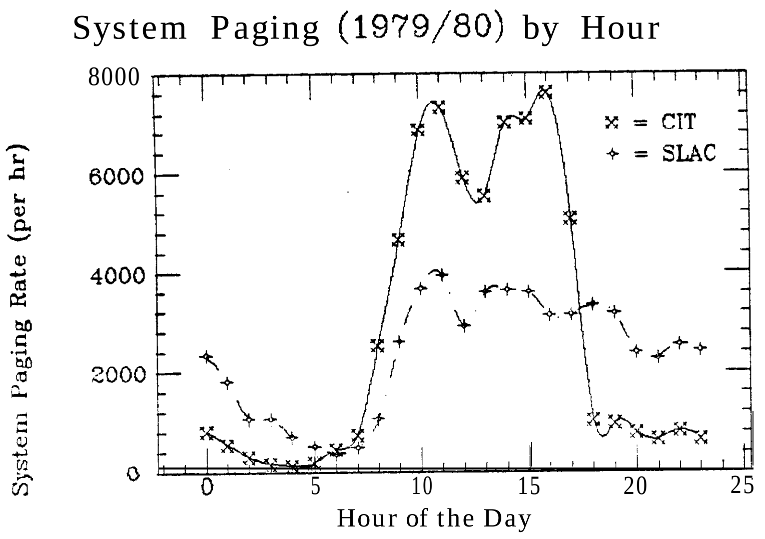


Figure II-8: 1979/80 System Paging Profiles

Appendix III. Component Type and Usage Groups Failure Charts

The data presented here[13] are failure frequency count plots for each hour and for various component type and usage groups. Each hourly reading represents the count of failures for the specified component type or usage group during the given hour of the day. Note that many component types do not contain enough failures to be considered statistically significant.

The list of all possible component types is given below:

*x*CPU	component is ‘associated with a system central processing unit,(8CPU = 3701168, 9CPU = 360/91, 3CPU = 3033);
MEMORY	component is associated with the system main memory;
DISK	component is associated with a disk subsystem;
CHANNEL	component is associated with an I/O channel;
TAPE, PRINT, CARD	component is associated with a magnetic tape, line printer, or card reader/punch subsystem;
*x*CTL	component is associated with a disk, tape, or printer controller, (x=D, T, or P);
TELE	component is associated with telecommunications (user front end or RJE);
RLTM	component is associated with SLAC’s real-time experiment stations;
SWBE, SWFE, SWOP, SW	component is back-end, front-end, or operating-system software; SW is used for all software at SLAC.

A usage group is a higher-level aggregate of component types. The significant group are:

CPU	includes groups 8CPU, 9CPU, CPU;
CPU*	equipment associated with system operator stations;

[13] The data presented within this Appendix is provided by and property of Leland Stanford, Jr. University.

MEDIA	includesgroups DISK, TAPE, PRINT, CARD, RLTM;[14]
MDCTL	media controllers, includes DCTL, PCTL, TCTL;
MEMORY	same as group named MEMORY;
SOFTWARE	includes component types SWBE, SWFE, SWOP; no usage group was formed for SLAC, component type SW is equivalent.
TLCHAN	telecommunications and channels, includes TELE, CHAN.

```
FREQUENCY

10 +                              *
 9 +                              *                       *           *
 8 +                           *  *              *  *     *           *
 7 +                           *  *  *     *     *  *     *           *
 6 +                        *  *  *  *     *  *  *  *     *           *
 5 +                        *  *  *  *     *  *  *  *  *  *        *  *
 4 +                        *  *  *  *     *  *  *  *  *  *        *  *
 3 +                        *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
 2 + *           *  *       *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
 1 + *  *     *  *  *       *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
    ------------------------------------------------------------------------
     0     2     4     6     a     10    12    14    16    la    20    22

                                 HOUR
```

Figure III-l: CIT 1979/80 Failure Histogram for Component Type 'CARD'

```
FREQUENCY

 3 +                        *  *
 2 +             *          *  *
 1 +             *          *  *     *        *  *     *  *  *  *
    ------------------------------------------------------------------------
     0     2     4     6     a     10    12    14    16    la    20    22

                                 HOUR
```

Figure III-2: SLAC 1979180 Failure Histogram for Component Type 'CARD'

[14]This grouping differs at the two sites: at CIT we have formed an *interactive* media usage group and a *batch* usage group. The first is composed of DISK only; the second has PRINT and TAPE. At SLAC the single media usage group contains PRINT, TAPE, DISK, and RLTM component types,

```
FREQUENCY

7 + *
6 + *
5 + *               *
4 + * *             *         *               *
3 + * * * *       * *   * *   * *             *
2 + * * * *   *   * *   * *   * *       *     *   *
1 + * * * * * *   * * * * * * * * * *   *     *   *
   ------------------------------------------------
    0   2   4   6   a   lo  12  14  16  la  20  22

                     HOUR
```

Figure III-3: CIT 1979/80 Failure Histogram for Component Type ‘CHANNEL’

```
FREQUENCY

7 +                       * *
6 +                       * *
5 +                     * * *       *
4 +                 *   * * *   *   *
3 +   *             * * * * *   *   *   *     *   *
2 +   *       * *   * * * * * * * * * * *   * *   *
1 +     **    * * * * * * * * * * * * * * * * * * *
   ------------------------------------------------
    0   2   4   6   a   lo  12  14  16  la  20  22

                     HOUR
```

Figure III-4: SLAC 1979/80 Failure Histogram for Component Type ‘CHANNEL’

```
FREQUENCY

2 +                             *
1 + * *                     *   *   * *         *
   ------------------------------------------------
    0   2   4   6   a   lo  12  14  16  la  20  22

                     HOUR
```

Figure III-5: CIT 1979/80 Failure Histogram for Component Type ‘DCTL

```
FREQUENCY

15 +                 *   *
14 +                 *   * *
13 + *   *           *   * *     *         *
12 + *   *           * * * *     *         *
11 + *   *           * * * * *   *         *
10 + *   *           * * * * *   * *       *       *
 9 + * * *           * * * * *   * * *     * *     *
 8 + * * *   *       * * * * *   * * * * * * *     *
 7 + * * * * *       * * * * *   * * * * * * *     *
 6 + * * * * *       * * * * * * * * * * * * *     *
 5 + * * * * *   * * * * * * * * * * * * * * *     *
 4 + * * * * *   * * * * * * * * * * * * * * *   * *
 3 + * * * * * * * * * * * * * * * * * * * * * * * *
 2 + * * * * * * * * * * * * * * * * * * * * * * * *
 1 + * * * * * * * * * * * * * * * * * * * * * * * *
    ------------------------------------------------
     0   2   4   6   8   10  12  14  16  18  20  22

                        HOUR
```

Figure III-6: SLAC 1979/80 Failure Histogram for Component Type 'DCTL

```
FREQUENCY

                         *
11 +                     *
10 +           * *       *
 9 +           * *       *     *         *
 8 +   *     * * *       *     *         *
 7 + * *   * * * *       *   * *     *   *
 6 + * * * * * * *       *   * *   * *   *
 5 + * * * * * * *   *   *   * * * * *   * * *
 4 + * * * * * * * * *   *   * * * * *   * * * *
 3 + * * * * * * * * *   * * * * * * *   * * * * *
 2 + * * * * * * * * * * * * * * * * *   * * * * *
 1 + * * * * * * * * * * * * * * * * * * * * * * *
    ------------------------------------------------
     0   2   4   6   8   10  12  14  16  18  20  22

                        HOUR
```

Figure III-7: CIT 1979/80 Failure Histogram for Component Type 'DISK

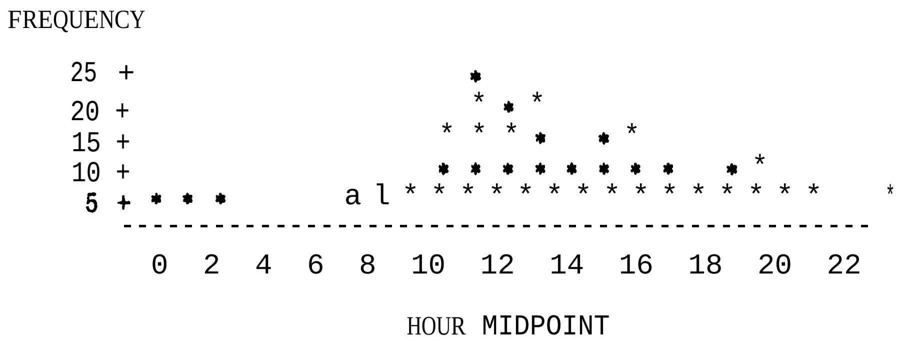


Figure III-8: SLAC 1979180 Failure Histogram for Component Type 'DISK'

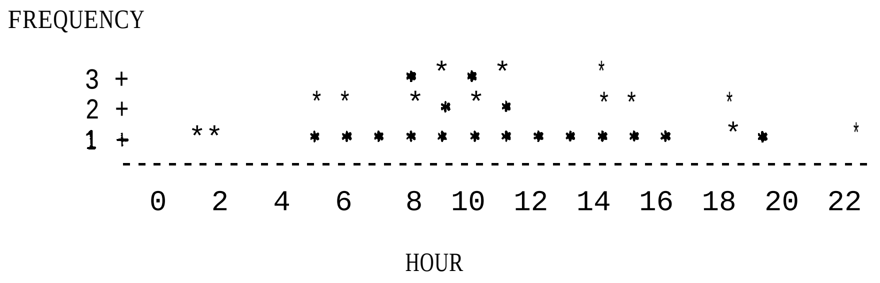


Figure III-9: CIT 1979/80 Failure Histogram for Component Type 'MCTL'

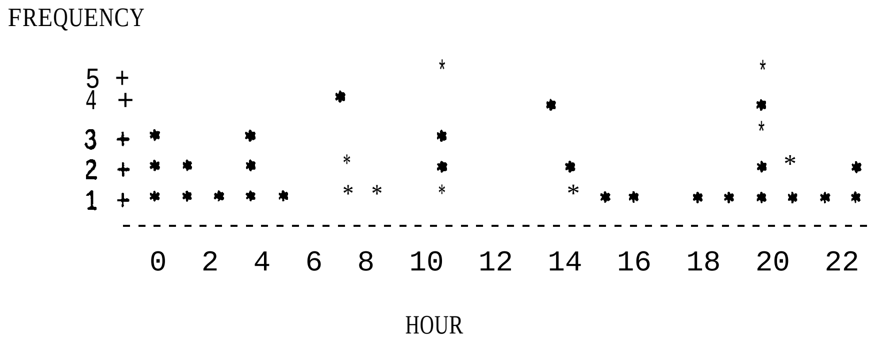


Figure III-IO: CIT 1979180 Failure Histogram for Component Type 'MEMORY'

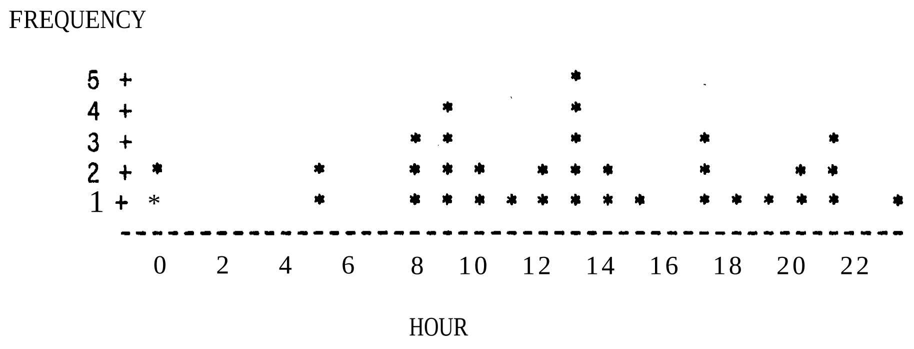


Figure III-11: SLAC 1979180 Failure Histogram for Component Type ‘MEMORY

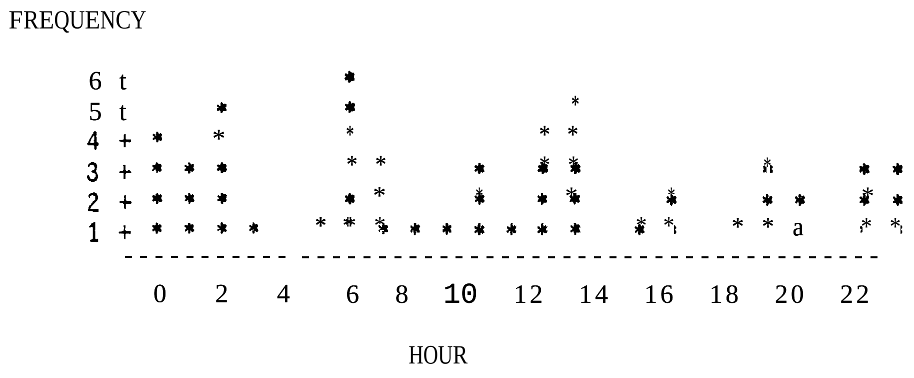


Figure III-12: CIT 1979/80 Failure Histogram for Component Type ‘OPRL’

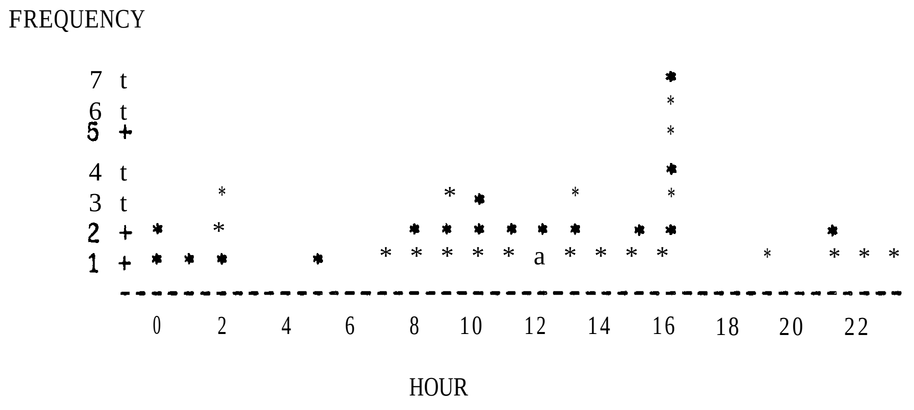


Figure III-13: SLAC 1979/80 Failure Histogram for Component Type ‘OPRL’

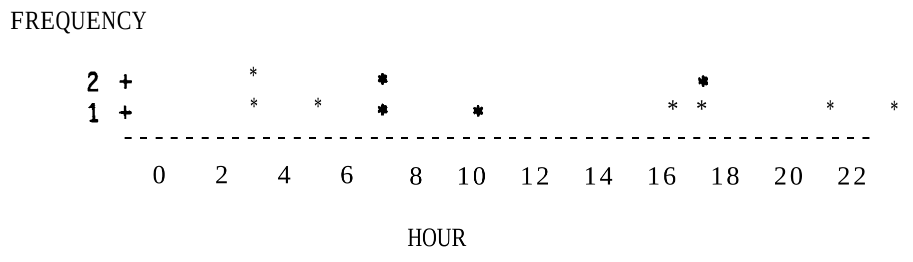


Figure III-14: CIT 1979/80 Failure Histogram for Component Type 'PCTL'

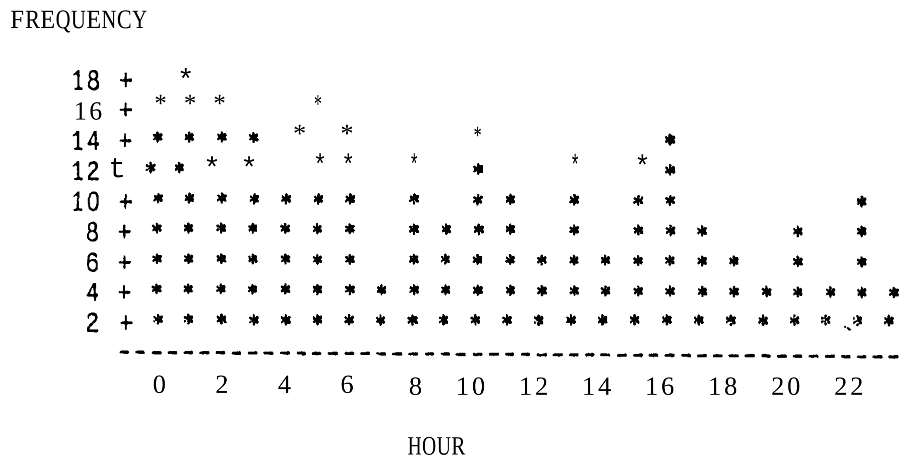


Figure III-15: CIT 1979180 Failure Histogram for Component Type 'PRINT'

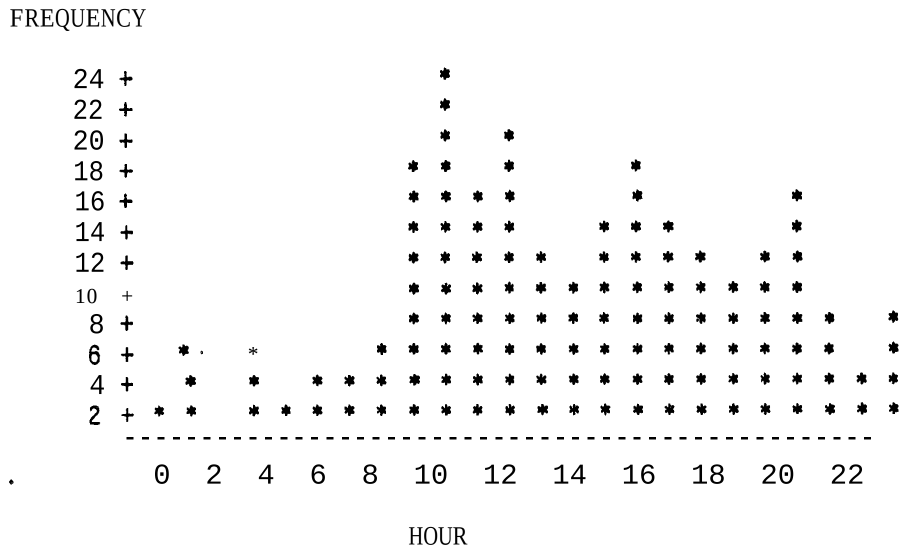


Figure III-16: SLAC 1979/80 Failure Histogram for Component Type ‘PRINT’

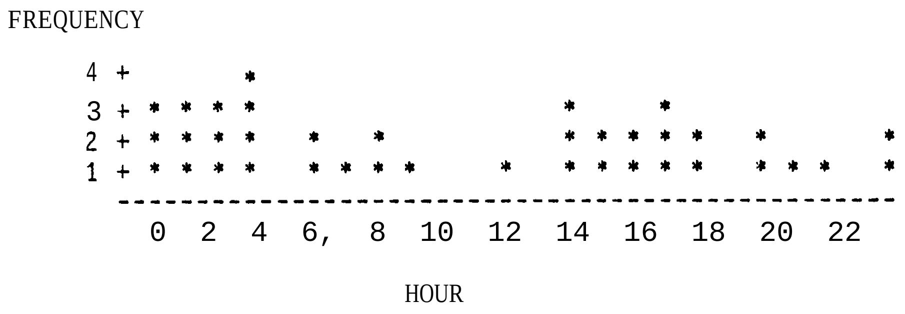


Figure III-17: CIT 1979/80 Failure Histogram for Component Type ‘SWBE’

```
FREQUENCY

30 +           *
25 +           *
20 +           *
15 +           *       *               *
10 +           * *   * * * *   * * * * *
 5 + *         * *   * * * * * * * * * * * * * * * *
     ------------------------------------------------
     0   2   4   6   8   10  12  14  16  18  20  22

                        HOUR
```

Figure III-18: CIT 1979180 Failure Histogram for Component Type ‘SWFE

```
FREQUENCY

14 + *
13 + * *
12 + * *
11 + * *
10 t * *
 g t * *   *
 8 + * * * *                       * *
 7 + * * * *                       * *
 6 + * * * *                       * *
 5 + * * * *   *       *           * * *
 4 + * * * *   *       *       *   * * *
 3 + * * * *   *       * * *   * * * * *
 2 + * * * * * *   * * * * * * * * * * *   * *     *
 1 + * * * * * * * * * * * * * * * * * *   * * * * *
     ------------------------------------------------
     0   2   4   6   8   10  12  14  16  18  20  22

                        HOUR
```

Figure III-19: CIT 1979180 Failure Histogram for Component Type ‘SWOP’

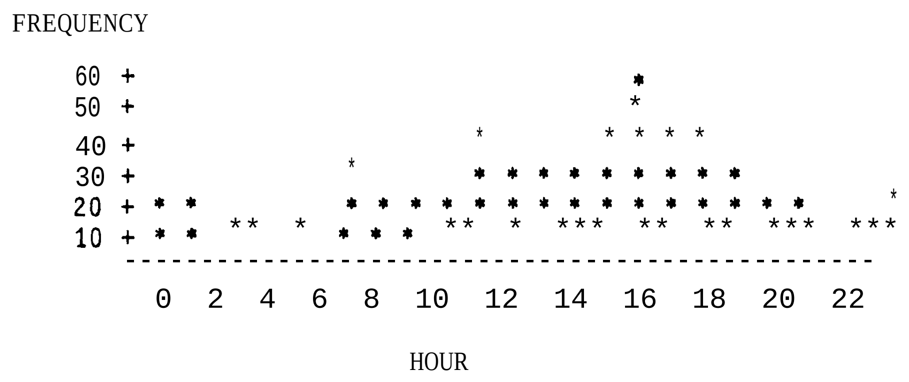


Figure III-20: SLAC 1979/80 Failure Histogram for Component Type ‘SW’

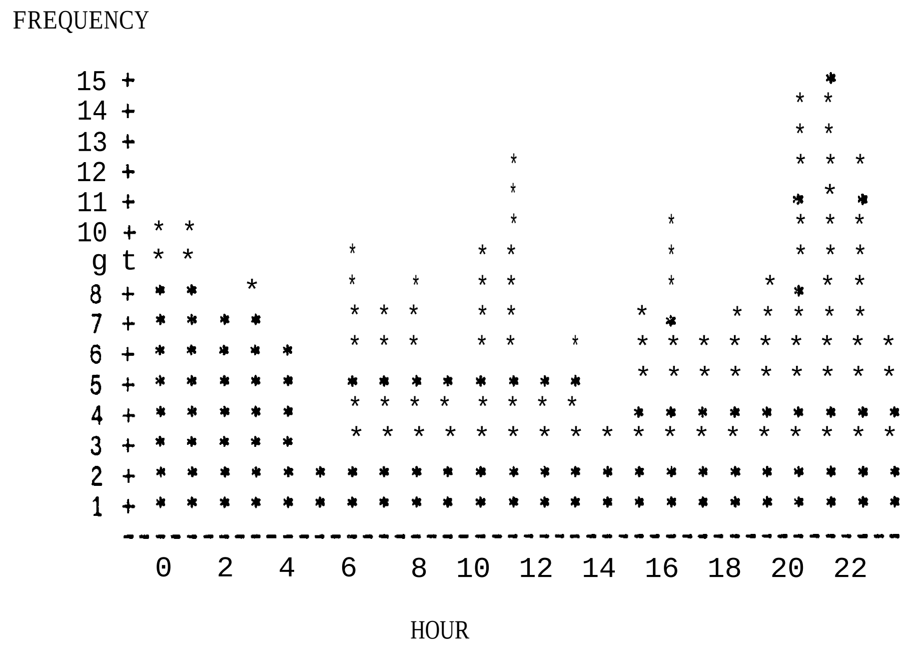


Figure 111-21: CIT 1979180 Failure Histogram for Component Type ‘TAPE

```
FREQUENCY

30 +                            *
25 +                            *  *  *  *     *
20 +                            *  *  *  *  *  *  *        *
15 + *                          *  *  *  *  *  *  *  *  *  *
10 + *  *                 *     *  *  *  *  *  *  *  *  *  *
 5 + *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
    ------------------------------------------------------------------------
     0     2     4     6     8     10    12    14    16    18    20    22

                                 HOUR
```

Figure III-22: SLAC 1979180 Failure Histogram for Component Type ‘TAPE

```
FREQUENCY

2 +    *  *        *                 *                          *  *     *
1 + *  *  *        *  *              *     *        *     *  *  *  *  *  *
   ------------------------------------------------------------------------
    0     2     4     6     8     10    12    14    16    18    20    22

                                 HOUR
```

Figure III-23: CIT 1979/80 Failure Histogram for Component Type ‘TCTL’

```
FREQUENCY

1 +                      *                 *     *           *
   ------------------------------------------------------------------------
    0     2     4     6     8     10    12    14    16    18    20    22

                                 HOUR
```

Figure 111-24: SLAC 1979/80 Failure Histogram for Component Type ‘TCTL’

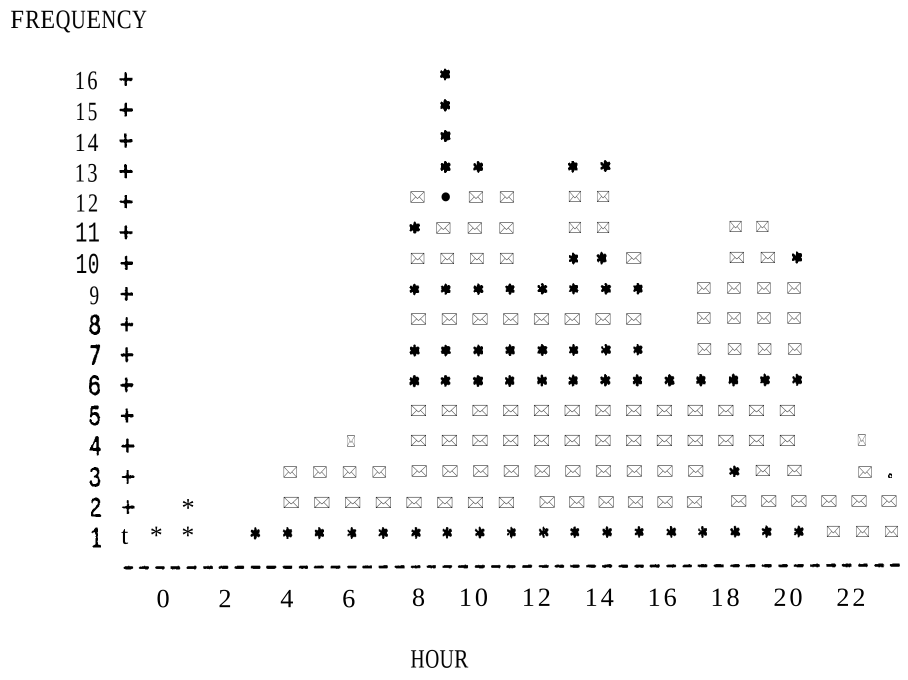


Figure III-25: CIT 1979180 Failure Histogram for Component Type ‘TELE

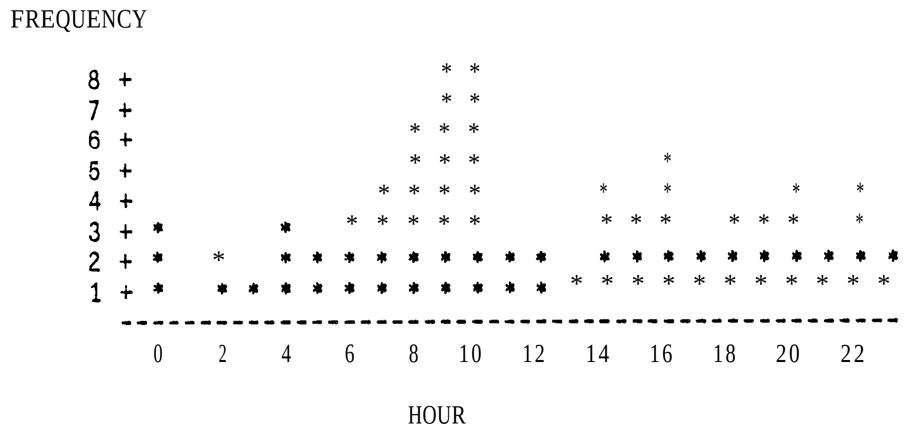


Figure III-26: SLAC 1979/80 Failure Histogram for Component Type 'TELE'

```
FREQUENCY

 1 +                 *
     ------------------------------------------------
     0   2   4   6   8   10  12  14  16  18  20  22

                        HOUR
```

Figure III-27: CIT 1979180 Failure Histogram for Component Type ‘TLCT

```
FREQUENCY

18 + *
17 + *
16 + *
15 + *
14 + *
13 + *
12 + *
11 + * *
10 + * *
 g t * *     *
 8 + * * *   *
 7 + * * * * *
 6 + * * * * *                 *
 5 + * * * * * *             * *                   *
 4 + * * * * * * *       *   * * *     *           *
 3 + * * * * * * * * *   * * * * *     *           *
 2 + * * * * * * * * * * * * * * * *   *   *     * *
 1 + * * * * * * * * * * * * * * * * * * * * *   * *
     ------------------------------------------------
     0   2   4   6   8   10  12  14  16  18  20  22

                        HOUR
```

Figure III-28: CIT 1979/80 Failure Histogram for Component Type ‘3CPU’

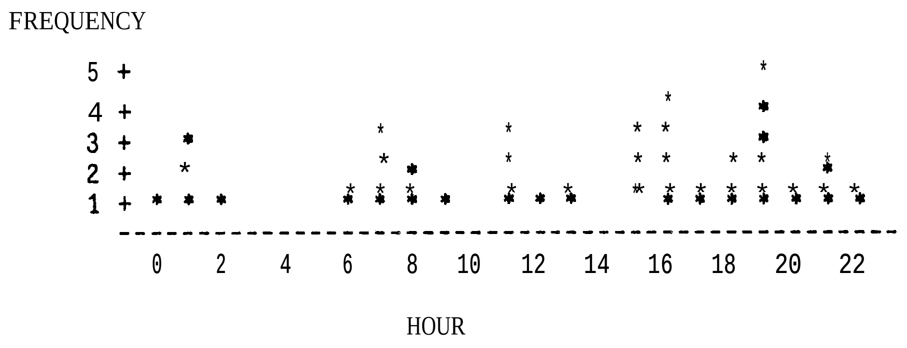


Figure 111-29: SLAC 1979/80 Failure Histogram for Component Type ‘8CPU’

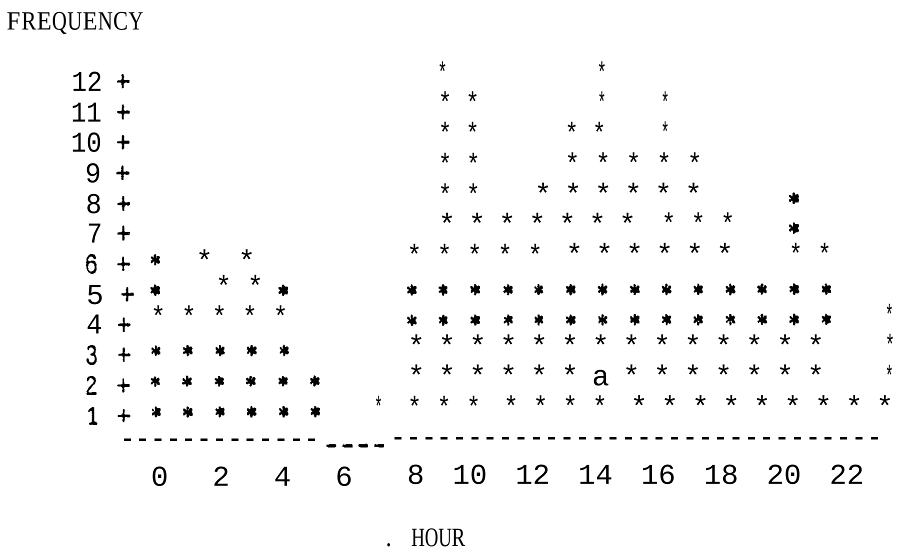


Figure III-30: SLAC 1979/80 Failure Histogram for Component Type ‘9CPU’

```
FREQUENCY

18 t *
17 + *
16 + *
15 + *
14 + *
13 + *
12 + *
11 + * *
10 + * *
 g t * *     *
 8 + * * *   *
 7 + * * * * *
 6 + * * * * *                 *
 5 + * * * * * *             * *                   *
 4 + * * * * * * *       *   * * *     *           *
 3 + * * * * * * * * *   * * * * *     *           *
 2 + * * * * * * * * * * * * * * * *   *   *     * *
 1 + * * * * * * * * * * * * * * * * * * * * *   * *
     ------------------------------------------------
     0   2   4   6   8   10  12  14  16  18  20  22

                          HOUR
```

Figure 111-31: CIT 1979/80 Failure Histogram for Usage Group ‘CPU’

```
FREQUENCY

5 +     *
4 + *   *                   *
3 + * * *       * *         *                   * *
2 + * * *       * *     *   * *     *     *     * *
1 + * * * *   * * * * * *   * *   * *     * *   * *
    ------------------------------------------------
    0   2   4   6   8   10  12  14  16  18  20  22

                         HOUR
```

Figure 111-32: CIT 1979/80 Failure Histogram for Usage Group ‘CPU*’

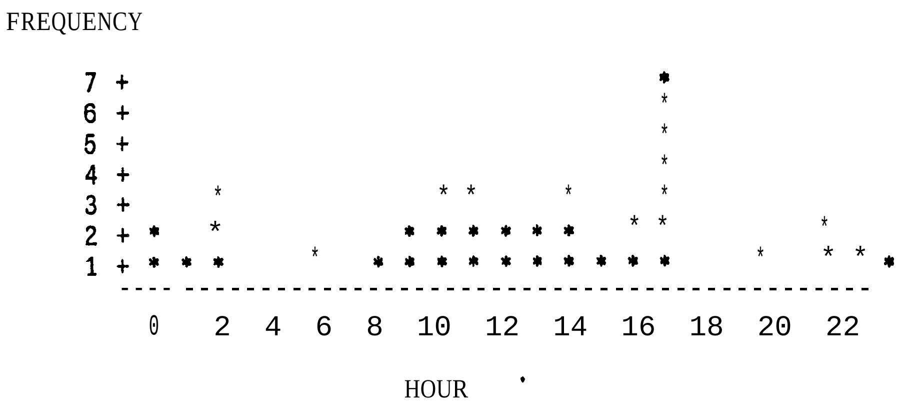


Figure 111-33: SLAC 1979/80 Failure Histogram for Usage Group ‘CPU*’

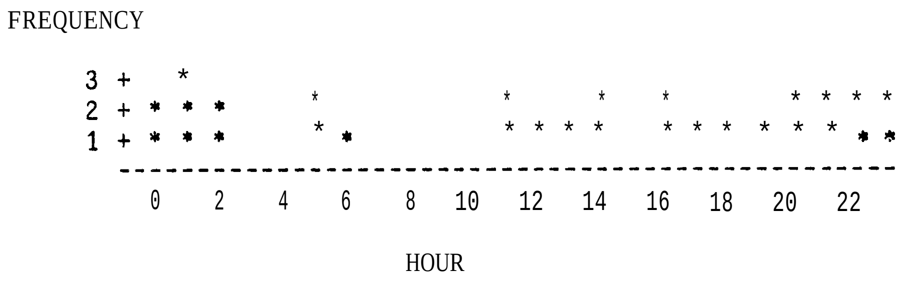


Figure III-34 CIT 1979/80 Failure Histogram for Usage Group ‘MDCTL’

```
FREQUENCY

15 +                 *   *
14 +                 *   * *               *
13 + *   *           *   * *     *         *
12 + *   *           * * * *     *         *
11 + *   *           * * * * *   * *       *
10 + *   *           * * * * *   * * *     *       *
 9 + * * *           * * * * *   * * * *   * *     *
 8 + * * *   *       * * * * *   * * * * * * *     *
 7 + * * * * *       * * * * * * * * * * * * *     *
 6 + * * * * *     * * * * * * * * * * * * * *     *
 5 + * * * * *   * * * * * * * * * * * * * * *     *
 4 + * * * * *   * * * * * * * * * * * * * * *   * *
 3 + * * * * * * * * * * * * * * * * * * * * * * * *
 2 + * * * * * * * * * * * * * * * * * * * * * * * *
 1 + * * * * * * * * * * * * * * * * * * * * * * * *
    -------------------------------------------------
     0   2   4   6   8  10  12  14  16  18  20  22

                         HOUR
```

Figure III-35: SLAC 1979180 Failure Histogram for Usage Group 'MDCTL'

```
FREQUENCY

25 + * *   *             *           *
20 + * * * *   * *   *   * *         *       * * *
15 + * * * * * * *   *   * *   *   * * *     * * *
10 + * * * * * * * * * * * * * * * * * * * * * * * *
 5 + * * * * * * * * * * * * * * * * * * * * * * * *
    -------------------------------------------------
     0   2   4   6   8  10  12  14  16  18  20  22

                         HOUR
```

Figure 111-36: CIT 1979/80 Failure Histogram for Usage Group 'MED*'

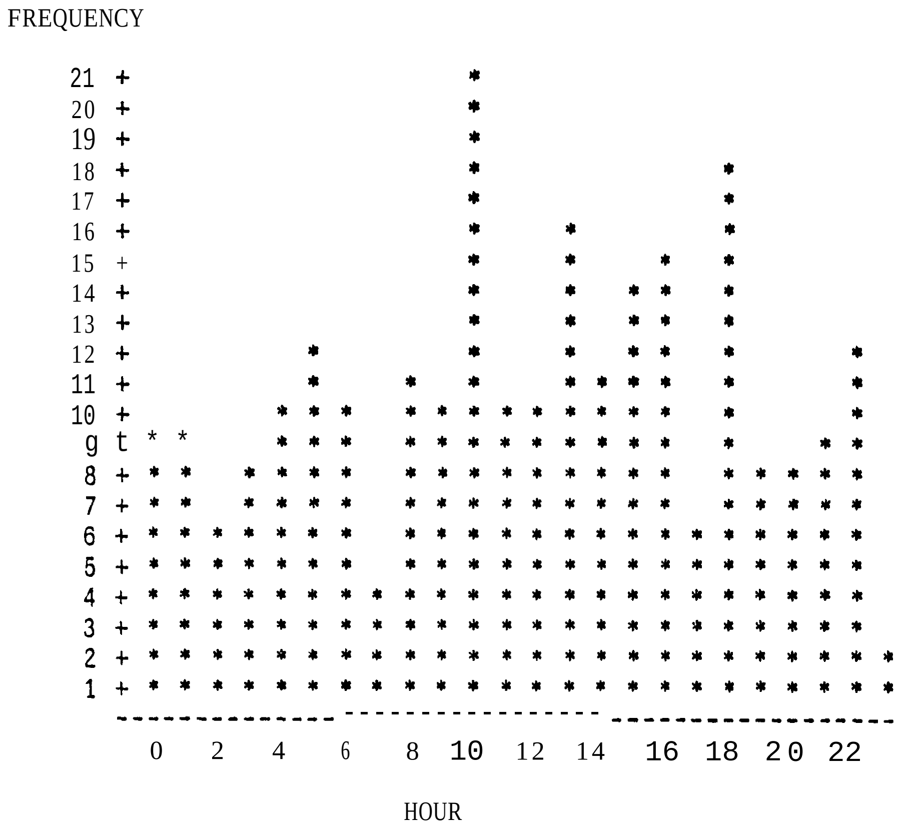


Figure III-37: CIT 1979/80 Failure Histogram for Usage Group 'MEDIA'

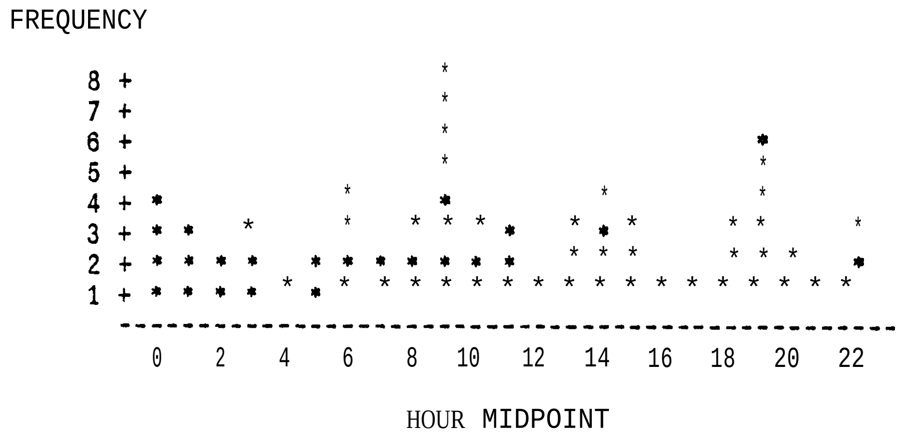


Figure 1X1-38: CIT 1979180 Failure Histogram for Usage Group 'MEMORY'

```
FREQUENCY

  5 +                                        *
  4 +
  3 +                            *  *           *           *           *
  2 + *              *        *  *  *     *  *  *                 *  *
  1 + *              *        *  *  *  *  *  *  *  *     *  *  *  *  *     *
    we--.-------------------------------------------------------------------
      0     2     4     6     8     10    12    14    16    18    20    22

                              HOUR
```

Figure 111-39: SLAC 1979/80 Failure Histogram for Usage Group 'MEMORY'

```
FREQUENCY

  35 +                *
  30 +                *
  25 + *              *
  20 + *              *           *           *     *  *  *
  15 + *  *  *  *     *  *     *  *     *     *  *  *  *  *
  10 + *  *  *  *     *  *     *  *  *  *     *  *  *  *  *        *  *     *
   5 + *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
       ------------------------------------------------------------------------
       0     2     4     6     8     10    12    14    16    18    20    22

                              HOUR MIDPOINT
```

Figure III-40: CIT 1979/80 Failure Histogram for Usage Group 'SOFTWARE

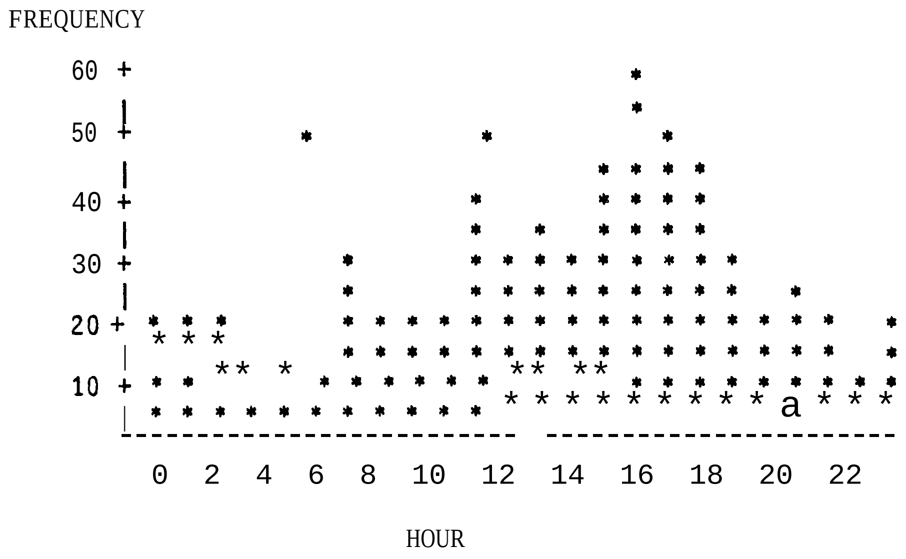


Figure 1X1-41: SLAC 1979180 Failure Histogram for Usage Group ‘SOFTWARE

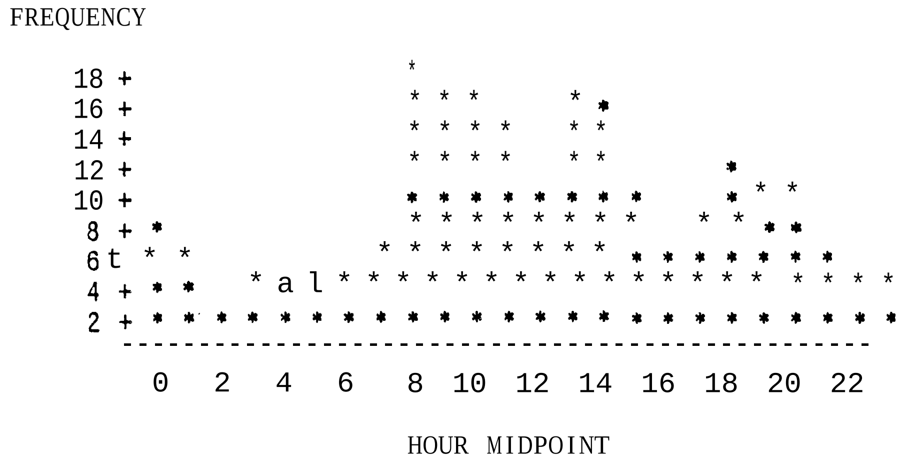


Figure III-42: CIT 1979/80 Failure Histogram for Usage Group ‘TLCHN’

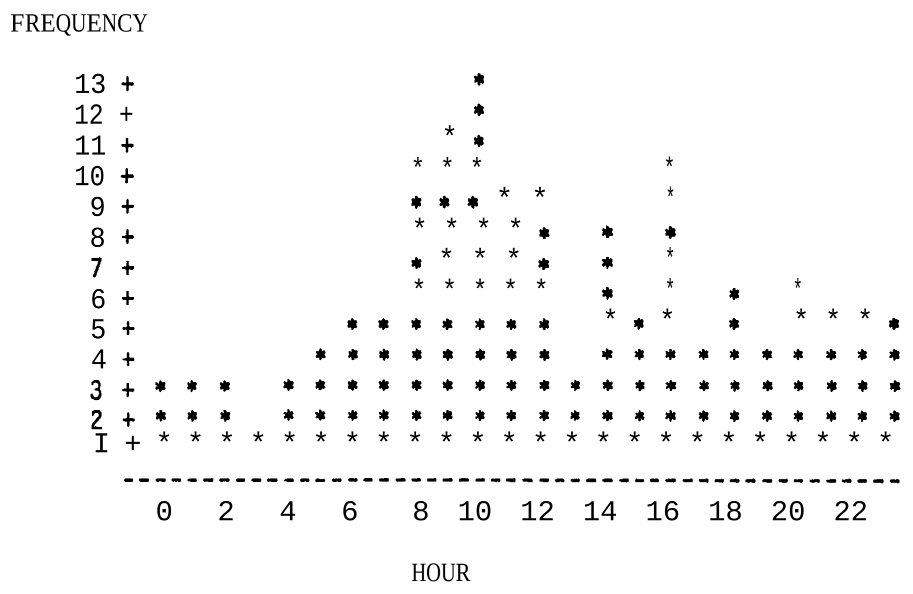


Figure III-43: SLAC 1979180 Failure Histogram for Usage Group ‘TLCHN’

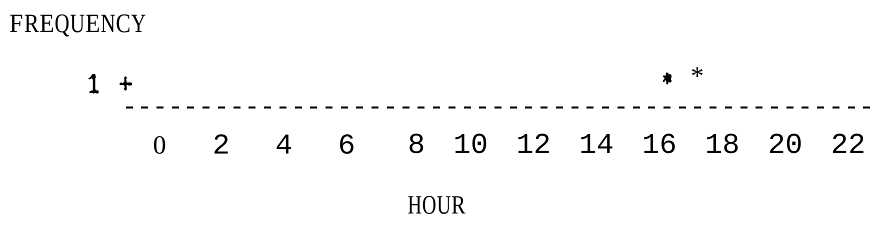


Figure III-44: SLAC 1979/80 Failure Histogram for Component Type ‘OCTL

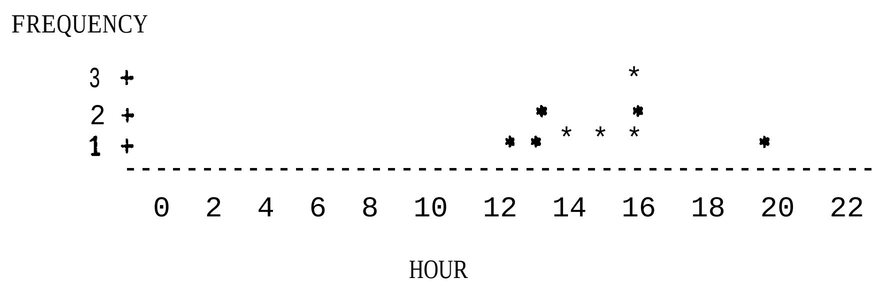


Figure III-45: SLAC 1979180 Failure Histogram for Component Type ‘RLTM’

References

[Barr, 1976] Barr, Goodnight, Sall & Hellwig. A User's *Guide to SAS 76* SAS Institute, Raleigh, North Carolina, 1976.

[Beaudry, 1978a] Beaudry, M. D. *Performance Considerations for the Reliability Analysis of Computing Systems.* PhD thesis, Stanford University, 1978.

[Beaudry, 1978b] Beaudry, M. D. *Performance Considerations **for** Reliability Analysis: A Statistical Case Study.* Technical Report 126, Center for Reliable Computing, Computer Systems Lab, Stanford University, May, 1978.

[Butner, 1980] Butner, *S.* E. and Iyer, R. K. *A Statistical Study **of** Reliability **and** System Load at SLAC.* Technical Report 188, Center for Reliable Computing, Computer Systems Lab, Stanford University, Jan, 1980.

[Castillo, 1980] Castillo, X. & Siewiorek, D. P. A Performance-Reliability Model for Computing Systems. In *Proceedings **of** the Tenth Annual Fault-Tolerant Computing Symposium,* pages 187-192. October, 1980.

[Castiilo, 1981] Castillo, X. & Siewiorek, D. P. Workload. Performance, and Reliability of Digital Computing Systems. In *Proceedings **of** the Eleventh International Symposium on Fault-Tolerant Computers,* pages 84-89. June, 1981.

[Daniel, 1978] Daniel, W. W. *Applied Nonparametric Statistics.* Houghton Mufflin Co., 1978.

[Draper, 1966] Draper, *N.* & Smith, H. *Applied Regression Analysis.* Wiley, 1966.

[IBM, 1972] *Introduction to Virtual Storage in System/370* IBM Corp., Poughkeepsie, N.Y., 1972.

[McConnell, 1979]
McConnell, S. R., Siewiorek, D. P. & Tsao, M. M. The Measurement and Analysis of Transient Errors in Digital Computing Systems. In *Proceedings **of** the Ninth Annual International Symposium on Fault-Tolerant Computing,* pages 67-70. June, 1979.

[Meyer, 1978] Meyer, J. F. On Evaluating the Performability of Degradable Computing Systems. In *Digest of Papers: The Eighth Annual International Symposium on Fault-Tolerant Computing,* pages 44-49. June, 1978.

[Meyer, 1980] Meyer, J. F. On Evaluating the Performability of Degradable Computing Systems. *IEEE Transactions on Computers C-22:720-731,* August, 1980.

[Musa, 1980] Musa, J. D. The Measurement and Management of Software Reliability. *Proceedings* ***of*** *the IEEE* 68 No. 9:1131-1143, September, *1980.*

[Myers, 1978] Myers, G. J. *Advances in Computer Architecture.* Wiley & Sons, 1978.

[SCIP, 1975a] *WYLBUR/370 Reference Manual* SCIP, Stanford University, 1975.

[SCIP, 1975 b] *ORVYL/370 Reference Manual* SCIP, Stanford University, 1975.

[Shedletsky, 1973]
Shedletsky, J. J. & McCluskey, E. J. The Error Latency of a Fault in a Combinational Digital Circuit. In *Digest* ***of*** *Papers, The Fifth Annual International Conference on Fault-Tolerant Computing.* June, 1973.

[Shooman, 1968] Shooman, M. L. *Probabilistic Reliability - An Engineering Approach.* McGraw Hill, 1968.

[Younger, 1979] Younger, M. *S. A Handbook* ***for*** *Linear Regression.* Wadsworth, Inc., 1979.