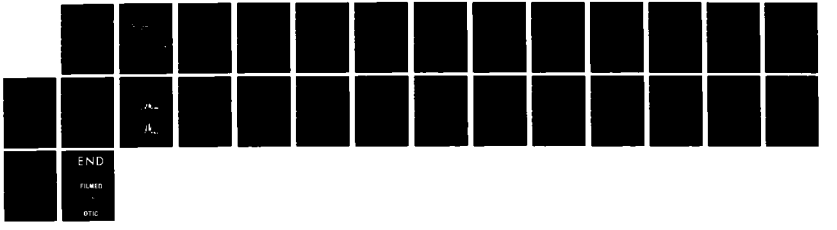
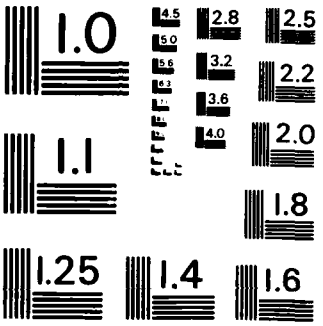
AD-A161 290 THE EFFECT OF SYSTEM WORKLOAD ON ERROR LATENCY: AN EXPERIMENTAL STUDY(U) ILLINOIS UNIV AT URBANA COMPUTER SYSTEMS GROUP R CHILLAREGE ET AL. JAN 85 CSG-40 1/1

UNCLASSIFIED N00014-84-C-0149 F/G 5/8 NL





REPORT CSG-40 JANUARY 1985

(1.2)

THE EFFECT OF SYSTEM WORKLOAD ON ERROR LATENCY: AN EXPERIMENTAL STUDY

RAM CHILLAREGE
RAVISHANKAR K. IYER

REPORT R-1027 UILU-ENG 85-2202

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

AD-A161 290

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS: None
2a. SECURITY CLASSIFICATION AUTHORITY: N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release, distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): CSG #40 R-1027	5. MONITORING ORGANIZATION REPORT NUMBER(S): N/A
6a. NAME OF PERFORMING ORGANIZATION: Coordinated Science Lab. University of Illinois; 6b. OFFICE SYMBOL (If applicable): N/A	7a. NAME OF MONITORING ORGANIZATION: Joint Services Electronics Program
6c. ADDRESS (City, State and ZIP Code): 1101 W. Springfield Avenue Urbana, Illinois 61801	7b. ADDRESS (City, State and ZIP Code): Office of Naval Research 800 Quincy Street Arlington, VA 22217
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Joint Services Electronics Program; 8b. OFFICE SYMBOL (If applicable): N/A	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: N00014-84-C-0149
8c. ADDRESS (City, State and ZIP Code): Office of Naval Research 800 Quincy Street Arlington, VA 22217	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
N/A	N/A	N/A	N/A

11. TITLE (Include Security Classification): THE EFFECT OF SYSTEM WORKLOAD ON ERROR LATENCY

12. PERSONAL AUTHOR(S): RAM CHILLAREGE, RAVISHANKAR K. IYER

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	January 1, 1985	25

16. SUPPLEMENTARY NOTATION: N/A

17. COSATI CODES		
FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Workload Measurements, ~~Error Latency Experiments,~~ Fault Tolerance, Error Detection, Reliability.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

In this paper we have established a methodology for determining and characterizing error latency. The method is based on real workload data gathered by an experiment instrumented on a VAX 11/780 during the normal workload cycle of the installation. This is the first attempt at jointly studying error latency and workload variations in a full production system. Distributions of error latency were generated by seeding errors under varying workload conditions. A family of error latency distributions so generated illustrate that error latency is not so much a function of when in time an error occured but rather a function of the workload that followed the error. The study finds that the mean error latency varies by a 1 to 10 (hours) ratio between high and low workloads. The method is general and can be applied to any system.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
		None

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

THE EFFECT OF SYSTEM WORKLOAD ON ERROR LATENCY: AN EXPERIMENTAL STUDY

Ram Chillarege and Ravishankar K. Iyer
Computer Systems Group
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign, 1985

In this report we have established a methodology for determining and characterizing error latency. The method is based on real workload data gathered by an experiment instrumented on a VAX 11/780 during the normal workload cycle of the installation. This is the first attempt at jointly studying error latency and workload variations in a full production system. Distributions of error latency were generated by seeding errors under varying workload conditions. A family of error latency distributions so generated illustrate that error latency is not so much a function of when in time an error occurred but rather a function of the workload that followed the error. The study finds that the mean error latency varies by a 1 to 10 (hours) ratio between high and low workloads. The method is general and can be applied to any system.

THE EFFECT OF SYSTEM WORKLOAD ON ERROR LATENCY:

AN EXPERIMENTAL STUDY

RAM CHILLAREGE
RAVISHANKAR K. IYER

Computer Systems Group
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign, 1985

Urbana, Illinois

TABLE OF CONTENTS

1. INTRODUCTION

There is now considerable experimental evidence to show that computer reliability is a dynamic function of system activity (as measured by the workload). Experimental studies on a number of machines, with varying configurations, [Butner 80], [Iyer 82a], [Iyer 82b] (on IBM machines) and [Castillo 80], [Castillo 81], [Castillo 82] (on DEC machines) provide evidence that CPU and memory failure rates increase exponentially as the system workload approaches saturation. i.e. there is a threshold beyond which an increase in workload results in a non-linear increase in the error rate. Measurements show that the increase in the error rate can be more than two orders in magnitude. This is an important issue since it affects both reliability and performance aspects of computer systems. It suggests that it is not useful to push a system close to its performance limits since the gain in performance is more than offset by the loss in reliability, as end points are reached.

Two possible reasons for the observed workload/failure dependency are thought to be: discovery of latent errors [Iyer 82] and stresses imposed by high currents and voltages.[1] Many failures can only be detected when a particular module or subsystem is "exercised." Thus, although the 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 "error latency" [Shedletsky 73]. The latent discovery effect can cause a noticeably high error rate during high workloads periods. A theoretical random walk model to describe this dependency is discussed in [Guther 80].

However, to-date no measure of the relative contributions of the two possible effects discussed above exist. This is an important issue because an engineering solution depends on which of the two factors is predominant. For example, if latent discovery is the predominant effect,

[1] As the activity increases within a computer the higher rates of exercise of the gates can lead to higher temperatures at the devices themselves due to switching-induced transients resulting in a higher failure rate. A model based on this is given in [Cortes 84].

new research into workload scheduling and periodic scrubbing is suggested. If however, this is not so we have to consider new designs which take workload patterns into account (much the same way as instruction mixes) in developing new designs.

There is no accurate technique for determining error latency. The only available study is [McGough 81 & 83] which performed a gate-level emulation of an avionic mini processor. A set of a specific programs were used to exercise the machine. The programs do not however, represent a real workload environment. Therefore both the methodology and results are not generally applicable.

We propose a methodology to study the latency characteristics of medium to large computer systems. This is the first attempt at jointly studying error latency and workload variations in a full production environment. The technique is developed using real data from a VAX 11/780 timeshared system at the University of Illinois, Coordinated Science Laboratory. The system runs the UNIX operating system Ver 4.2, and is used mostly for scientific computing and for a variety of miscellaneous data processing activities. The measurements and analysis show a strong relationship between system workload and the Error Latency in the unpaged portion of memory. The mean Error Latency varies nearly by a 1 to 10 ratio between high and low workload. Distributions of error latency are generated by seeding errors under varying workload conditions. The resulting family of error latency distributions illustrate that latency is not so much a function of when in time an error occurred but rather a function of the workload that followed the error. The methodology is not system specific and has general applicability.

2. INSTRUMENTATION

For the purposes of this study we concentrated on memory activity. An important reason for this was that previous studies show that the memory subsystem has the largest number of errors [Rossetti 81] [Iyer 83].

The VAX 11/780 system has 4 MByte of main memory, three 300 M Byte disk drives, and has during the peak hours about 20 to 25 interactive users. The backplane of the VAX CPU was probed so as to obtain data on memory activity. This data was sampled by the instrumentation connected to the backplane, and forms the basis for the analysis from which error latency is measured.

The VAX central processor and the memory subsystem are linked through a data path called the Synchronous Backplane Interconnect (SBI). Figure 1 shows the organization of the machine. The details of the machine organization are in [DEC 80a] and [DEC 80b]. The SBI is a parallel datapath which is multiplexed for address and data and uses a 200 nsec clock to achieve a maximum information transfer rate of 13.3 million bytes per sec.

The best approach for obtaining memory activity information is to monitor the SBI through which all transactions occur. High speed devices such as Disks, connect to the SBI through an interface called the Massbus Adapter. Unibus devices [DEC 80a] similarly connect to the SBI through a Unibus Adapter. Requests to memory can arise from either the CPU or from the I/O devices and all of them are transacted through the SBI. Monitoring the SBI therefore captures all requests to the memory subsystem. The address space on the SBI is partitioned so that addresses to the main memory subsystem, Unibus subsystem or other adapters are unique, permitting them to be individually extracted.

The SBI consists of 84 signal lines that belong to five different groups, namely, arbitration, information transfer, response, interrupt and control. The information transfer group with 46 signal lines contains the memory activity information. It is used to transfer addresses, data and

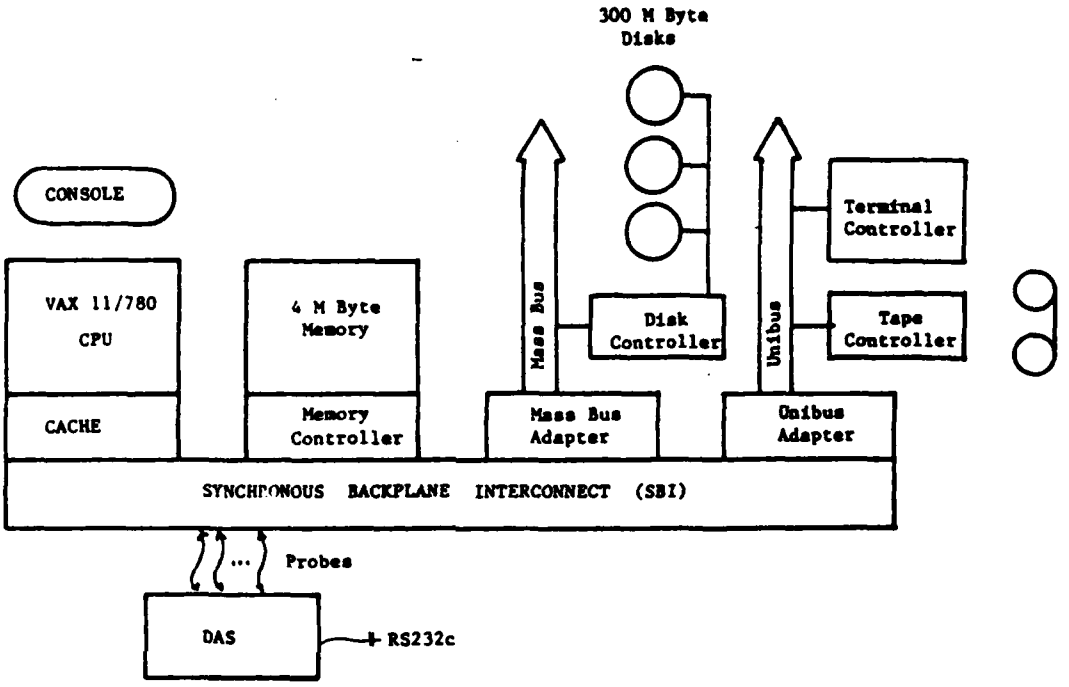


Figure 1. VAX 11/780 System Configuration

interrupt summary information. This group is subdivided into five fields that represent parity check (P), information tag (TAG), source or destination identification (ID), masks (MASKS), and 32 bits of information lines (B) as in Figure 2.

(1) P field: The parity field of 2 bits provides even parity for detecting single bit errors in the information transfer group. One of the bits provides parity over the TAG, ID and MASK fields and the other over the B field.

(2) TAG field: The TAG field is 3 bits wide and indicates the information type (being transmitted) on the information lines (B field). This field also determines the interpretation of the ID and the B fields. For example, when the tag code represents COMMAND

Figure 2. SBI Information Transfer group fields

ADDRESS, the B field contains the address.

(3) ID field: The ID field of 5 bits is used to identify the logical source of the data in a write command and the logical destination of the data in a read command. The address of the location is contained in the B field.

(4) MASK field: The mask field is 4 bits wide and is used to specify operations on any or all bytes of the data in the B field. Each bit in the mask field corresponds to a particular byte in B.

(5) B field: The B field is 32 bits wide (4 bytes) and is used to carry information/data. Depending on the TAG field the 32 bits are interpreted either as one data field of 32 bits or as containing two subfields: a FUNC field of 4 bits which identifies read or write mode and an ADDRESS field of 28 bits containing the physical address which can be either main memory or I/O.

2.1. Experimental Setup

A Tektronix Digital Analysis System (DAS) 9100 Series was used to monitor and sample data transfer activity on the SBI. The DAS probes used can strobe the data at speed upto 40 nsec which is faster than the clock speed of the SBI (200 nsec). The SBI signals are accessible at

the card edge of the SBI control cards [DEC c]. The data was read into the DAS using the SBI clock for external synchronization.

The experiment was controlled from the VAX with the aid of Tektronix 91DVV1 software [TEK 84] and some of our own programs. The software which controlled the experiment was so that it caused negligible overhead and did not bias the experiment. The DAS was periodically triggered to acquire data from the SBI, download the acquisition memory and time-stamp the data. This data was then preprocessed to make it compatible for subsequent input into statistical analysis programs and archived on tapes.

The instrumentation was tested for correct operation and acquisition. This was performed by taking the system down into single user operation, turning the cache memory off and running a test program that accesses specific locations of memory in sequence. Data collected on the DAS was then examined for correct acquisition against a known test program.

Two types of data are collected. The first involves logging transactions of every cycle on the SBI. We refer to this as *Regular Mode*. The DAS has a memory of 511 words, and therefore each sample contains the trace of 511 cycles on the SBI. Note that, not every cycle of the SBI contains a command/address, as some cycles may contain data and some may be idle. A sample of data acquired in *Regular Mode* is shown in Figure 3a. The first line contains a time stamp for the sample. Each line of data represents one SBI cycle. The data is in one's complement form. Note there are a number of idle cycles (all 1's). Figure 3b shows the decoded version of a single observation.

In certain types of experiments, such as the latency study, it is advantageous to produce a dense trace of addresses by storing only those cycles that contain command/address transmitted

[2]The DAS is programmable via an IEEE-488 interface, or an alternative serial line RS232c link to a host machine. We have it connected to the VAX via the serial line interface.

[3]The acquisition system has been tested for data bias against itself. This is done by externally triggering the DAS and acquiring the data and comparing memory usage distributions generated by this data with the distributions generated from automatically generated data. We find that the the instrumentation is sound and does not indicate any significant influences of self-bias.

on the B lines. Toward this end a compressed mode acquisition was performed by collecting on those signals which transmitted a memory address. This is accomplished by generating a "store only" signal to the DAS by decoding the TAG field that contains a code to identify command/address transaction. In the Compressed Mode the DAS acquires 511 consecutive addresses that are generated on the SBL. Figure 3c shows a sample in the Compressed Mode

3. MEASUREMENT AND ANALYSIS

Recall that the experiment collects data on memory activity, e.g. physical memory address, access rate and read/write mode. For the purposes of this project we studied the region in memory where the OS resides. We looked at this region because errors in the operating system can be fatal. This also has the advantage of being the unpaged portion of memory hence provides an estimate of inherent latency characteristics unaffected by paging. The methodology however is equally valid for both the unpaged and paged portions of the memory.

The data acquisition was performed at intervals of 25 to 40 seconds per sample. This was found to be sufficiently frequent to provide an estimate of memory access patterns and usage. For this experiment, we acquired data for a number of different usage periods which include the varying workload environments. On this base, we implemented a methodology for accurately estimating error latency characteristics for different usage regions and workloads. In particular the effect of increasing the workload on error latency characteristics was also determined.

3.1. Workload and Memory Usage Profiles

Recall that we were interested in estimating the errors discovered during the different workload environments and also the effect of a changing workload on error latency. Therefore

```
Fri Nov 23 11:28:59 CST 1984
PROBE:   3D       3C       3B       3A       2C       2B       2A   COMMENTS
OBS
,15      11111111 11111111 11111111 11111111 11111111 11111111 11111111 idle
,16      10100101 10010101 11111010 01111111 11111111 11110011 10001111 cmd/adr
,17      11111011 00000010 11111111 11111111 11111111 11111011 11101111
,18      11111111 11111111 11111111 11111111 11111111 11111111 11111111 idle
,19      00101010 01001100 11110100 01111111 11111111 11111011 10001111 cmd/adr
,20      11111001 11001001 10001010 10000011 11111111 11111011 11101111
,21      11111111 11111111 11111111 11111111 11111111 11111111 11111111 idle
```

Figure 3a Acquired data in *Regular Mode*

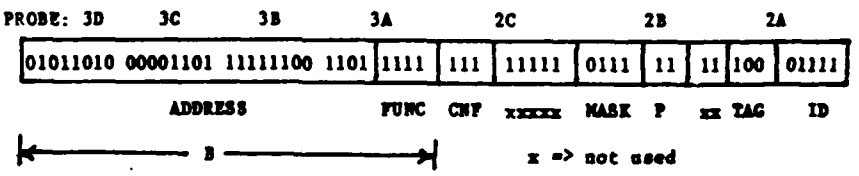


Figure 3b Decoding of the data fields

```
Fri Nov 23 12:15:11 CST 1984
PROBE:   3D       3C       3B       3A       2C       2B       2A   COMMENTS
OBS
,136     11101001 10110100 11111011 01111111 11111111 11110011 10001111 cmd/adr
,137     10001111 00110110 11110100 11011111 11111111 00000011 10001111 cmd/adr
,138     10001110 00110110 11110100 11011111 11111111 00001011 10001111 cmd/adr
,139     01011010 00001101 11111100 11011111 11111111 01111111 10001111 cmd/adr
,140     10001111 00110110 11110100 11011111 11111111 00000011 10001111 cmd/adr
```

Figure 3. Acquired data in *Compressed Mode*

it was important to have an idea of possible variations in system workload. Figure 4 shows two workload parameters CPU *user* and *system* usage as a function of the time of day. *CPU user usage* represents the percent time spent by the CPU in executing user code and *CPU system usage* corresponds to the percent time spent executing system code. The data was collected by a modified Unix system utility.

These load measures are cyclic over a 24 hours period and forms a stationary pattern for the week-days. This pattern of workload is typical of many computer installations [Iyer 82] [Castillo 81]. Between 8am and 10am there is a sharp increase in workload as noticeable in Figure 4. This period of the day is of particular interest for determining how latency characteristics change with increasing workload. During the peak periods the machine has about 20 to 25 interactive users. The workload is mostly interactive but there is a certain amount of batch processing.

An analysis of the memory access data showed two distinct patterns in the unpaged area. A 300 byte region with very high usage is shown in Figure 5a and the rest is shown in Figure 5b. These are shown separately since together one totally dominates the other. The high usage region which is most likely the kernel accounts for approximately 70% of the references made to system memory. In each of these plots it was noticed that a sampling of 20 to 30 minutes was sufficient to obtain a stable pattern. Interestingly enough, the distribution shapes did not change appreciably with workload, i.e. the relative access probabilities remained the same although the access rates changed significantly.

3.2. Estimating Error Latency

For the purpose of this study we assume that the error rate is constant and can equally affect all memory locations. This implies that the workload does not cause any additional errors. It allows us to determine the intrinsic latency characteristics without being biased by the workload distributions. Another way to look at this is that we find the system error rate

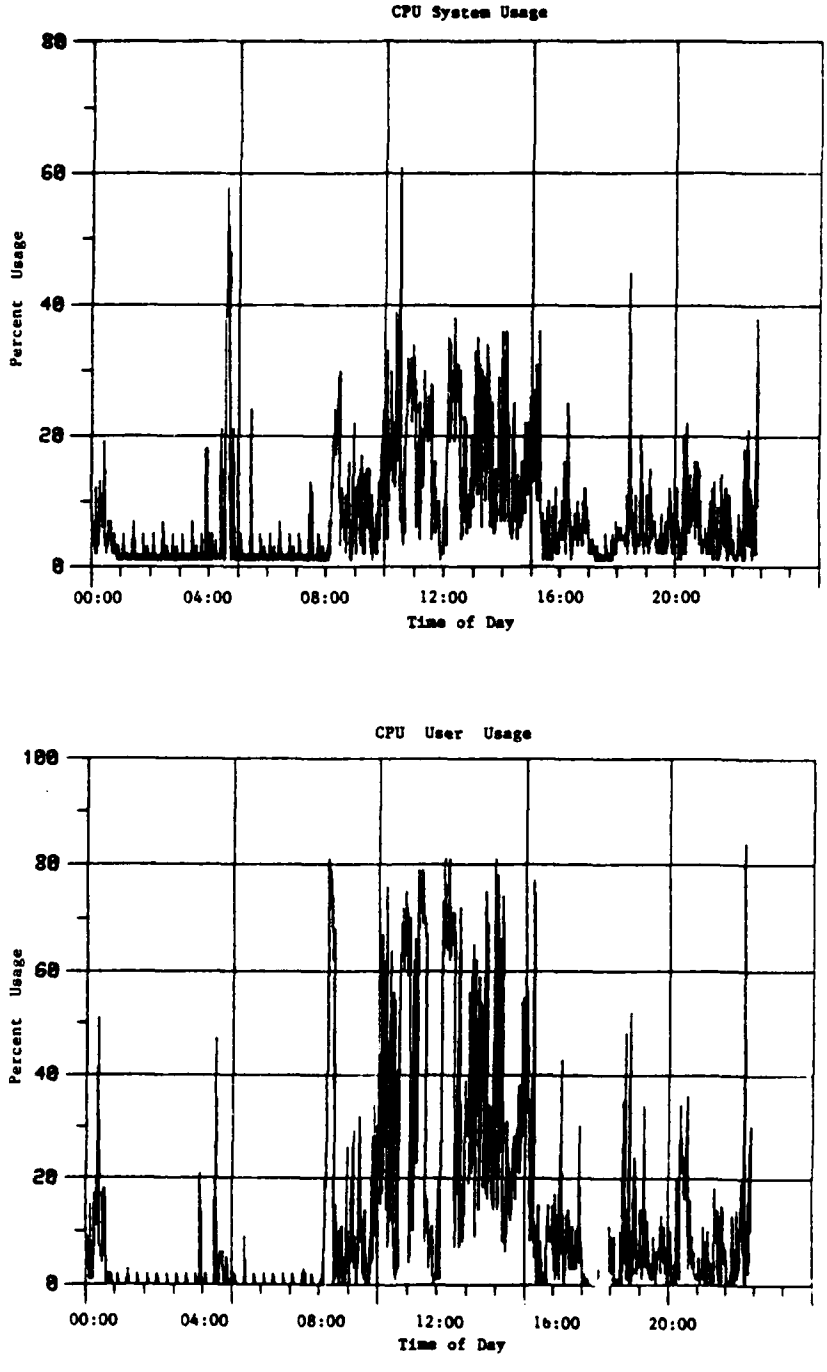


Figure 4. System workload

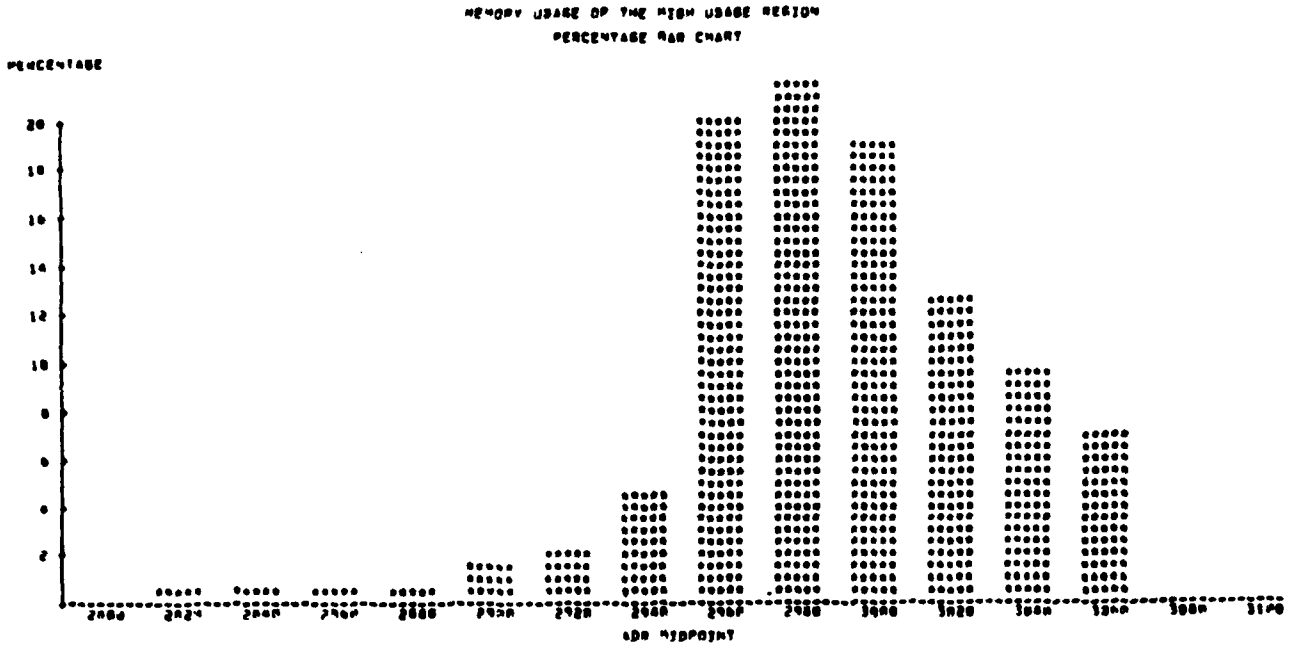


Figure 5. a

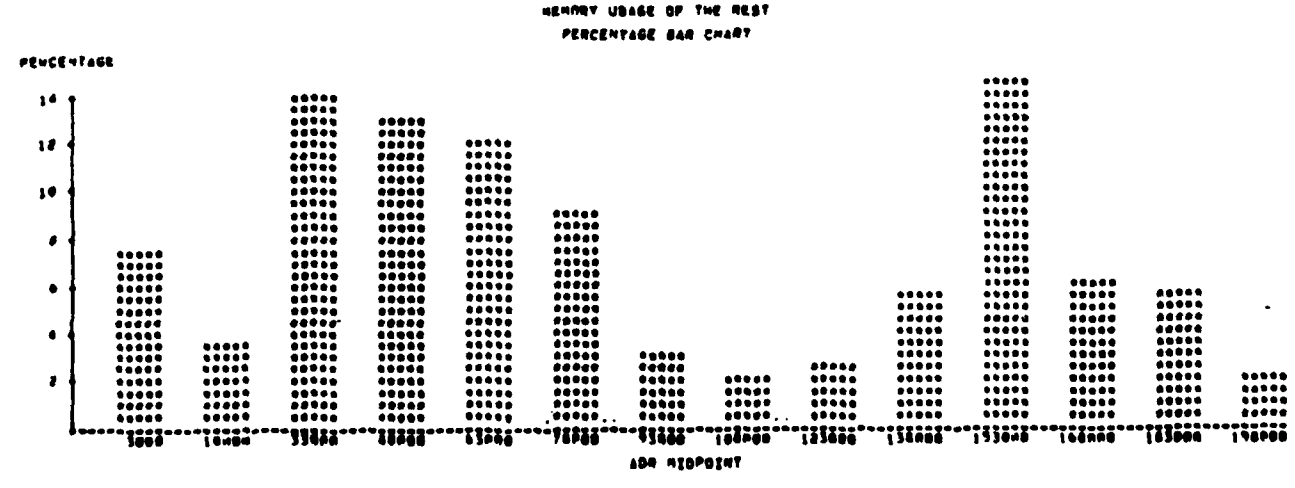


Figure 5. b

(distribution) contributed only by the latent discovery effect.

The memory addresses chosen to contain an error are picked from a uniform random distributions. However, if one intends to study the Error Latency given the assumption that increased memory usage increases the probability of error [Cortes 84] then the algorithm can be easily modified. The system memory has error detection and correction codes which corrects a single error and will detect up to two errors. An error is said to be discovered when the faulty memory location is read.

Figure 6 shows the algorithm used to generate error latency distributions. We commence by picking a random address of a memory location, say m_1 and assume that it has an error, say e_1. This is termed *seeding* an error. The error also has a time associated with it, sat t_1. The data is now scanned to find the first *memory read* to the location m_1. This is when the error would be detected by the ECC circuitry. In Figure 6 location m_1 has three memory reads to it. One before and two after the error e_1. Let t_1^* be the time of the first memory read after the error. Then the latency of error e_1 is $l_i = t_1^* - t_i$. The same location may be reaccessed (as in the figure) but that amounts to rediscovery and is not a part of this latency study. If however, the data set does not contain a read to the memory location in error then it goes undetected and amounts to a miss. For example in Figure 6 error e_2 is seeded on memory location m_2. However the data does not contain a read to m_2 and therefore e_2 is undiscovered. The misses are used to estimate the percentage of undiscovered errors. This process can be repeated for a large number of errors yielding a latency time l_i for every random error e_i chosen. These different latency times now form a distribution of the error latency. The misses generated by this process are used to determine the percentage of undiscovered errors.

3.3. Algorithm Implementation

Error seeding was the means by which workload effects on latency was studied. The error seeding time was varied over the entire usage period to investigate the change in latency with

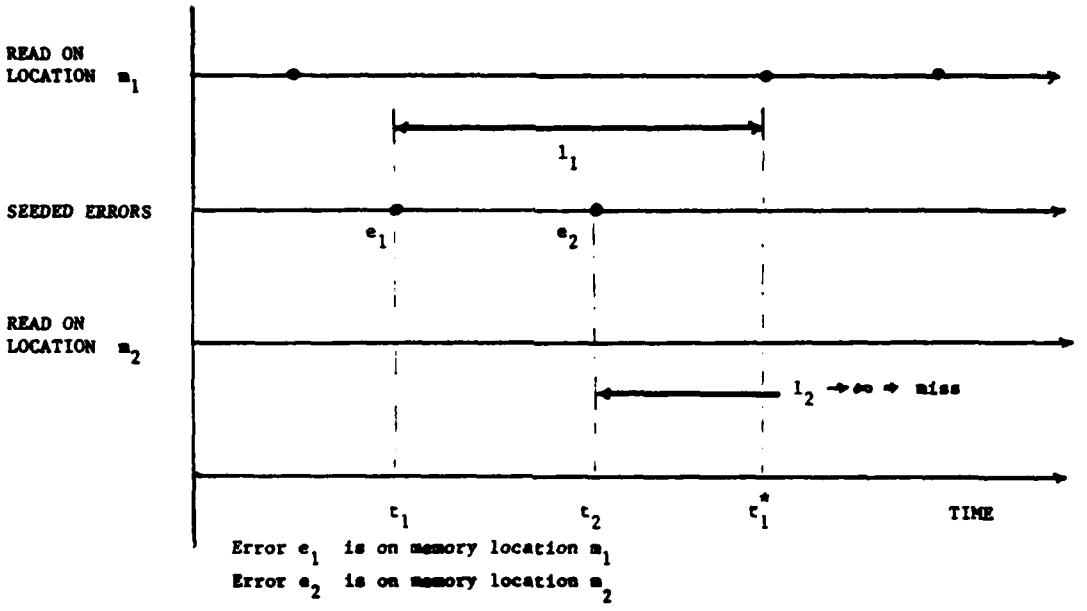


Figure 6. Latency time calculation

workload. To be able to determine the first time that the location is acessed the data should be available past the error instant for a few hours. Rather than seed all the errors at the same time, we seeded the errors in a 1 hour interval. The interval was so that the workload did not change appreciably over this period. This interval is referred to as the *error seeding window*. This is now useful in studying different workload environments by choosing the error seeding window to lie within the workload period.

To keep the computation tractable we define a *class* as a set of neighboring memory locations. Wherein, the access to any member of the class is considered equivalent to access of the whole class. The class serves two purposes. Firstly, it reduces the number of memory locations that one has to compute on. The number of memory locations is very large and the implementation of the above algorithm would necessitate a lot of computation. Secondly, it caters for the

fact that the sampled data although is representative of the memory access pattern need not contain every distinct address that is generated. The class size is chosen small enough so that the memory usage within a class is uniform (i.e. each location in a class has similar access probability) and large enough so that the computation is still tractable. We experimented with class sizes varying from ¼ page to 3 pages and found that it did not appreciably change the distribution and the median shifted by less than 5%. For a fixed class size, the percentage of misses remained almost constant.

4. ERROR LATENCY DISTRIBUTIONS

We wished to determine error latency characteristics for low, high and intermediate workload periods. The movement of the error seeding window was the vehicle by which latency characteristics were investigated. The error seeding window was shifted across the time of day axis to obtain a family of distributions. The moving of the window provides an insight into the nature of the error latency distribution and isolates the effect of workload. The change in these distributions then reflects the effect of changing workload on error latency. We keep moving the window until a cyclic behavior is seen in the distributions (which is over a 24 hour period in our data).

To determine the consequences of workload variations we consider a large span of time which includes the low and high workload periods. Recall from Figure 4 that from midnight to 7 am is the low workload, 8 to 10 am has an increasing workload (intermediate) with a peak around 11. The intermediate period where workload changes from low to high is of particular interest. Since the system workload forms a cyclic pattern every day we expect that a 24 hour observation of error seeding to be sufficient. This allows studying of both low and high workload periods. As the workload is stable for a large length of time the error seeding time

will not appreciable change the distribution so long as the measurement period is sufficiently larger than the latency period.

Figure 7 shows the latency distribution generated with a one hour error seeding window starting at midnight. The distribution is bi-modal with the second mode being the larger of the two. The initial peak corresponds to a small period of high activity which usually occurs around midnight. Within the first hour about 10% of the detected errors are found. The bulk of the errors, (70%) are found in the second mode. There is a sharp increase in the number of errors being detected from about 8 hours after the seeding. This corresponds to 8 am (real time of day) which is the start of the increasing workload period. The mean latency is 8:03 hrm. Listed in the figure are the percentages which correspond to detected errors only. Nearly 25% of the errors that are seeded were undetected. This figure of 25% miss is almost constant for all the following latency distribution.

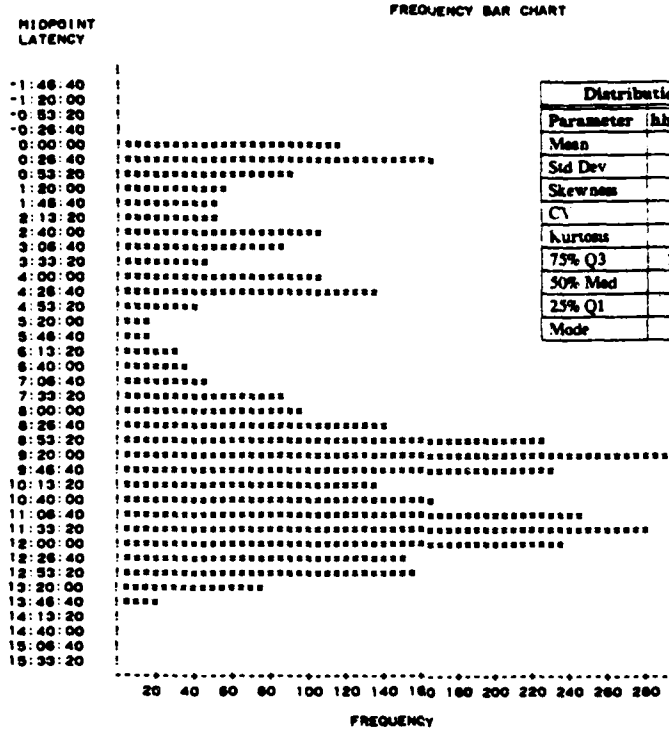
Figure 8 shows a similar latency distribution but with the error seeding window shifted in time towards the increasing workload. In Figure 8 the error seeding window it is between 4 and 5 am. Notice that the two modes have begun coming closer. It is interesting to note that the movement of the window forward in time has caused a decrease in the mean latency. This is caused by the movement of the second increase in error discovery (beginning of the second mode) closer and closer to the seeding time (shorter latency). The time of the sharp increase corresponds to 8 am, when workload starts to increase. Also note that there is dip in the distribution after the mode. This corresponds to a lull in the activity that occurs around 10:30 or so on our system.[4] This clearly shows the influence of workload in determining error latency.

Figure 9 has errors seeded at 8 am (start of the rise in workload) and Figure 10 has errors seeded at 12 pm (well into the high workload period). In contrast to Figure 7 it is seen that the mean error latency is now (Fig 10) down to 44 mins and that 70% of the detected errors are

[4]In our system which is largely used by graduate students, their day starts at around 10:30 am and continues past 12:30 pm or so. The early morning (8 am) rise in activity is due to secretarial and staff users.

ERROR LATENCY DISTRIBUTION
ERROR SEEDING WINDOW FROM 00:00:00 TO 01:00:00

FREQUENCY BAR CHART



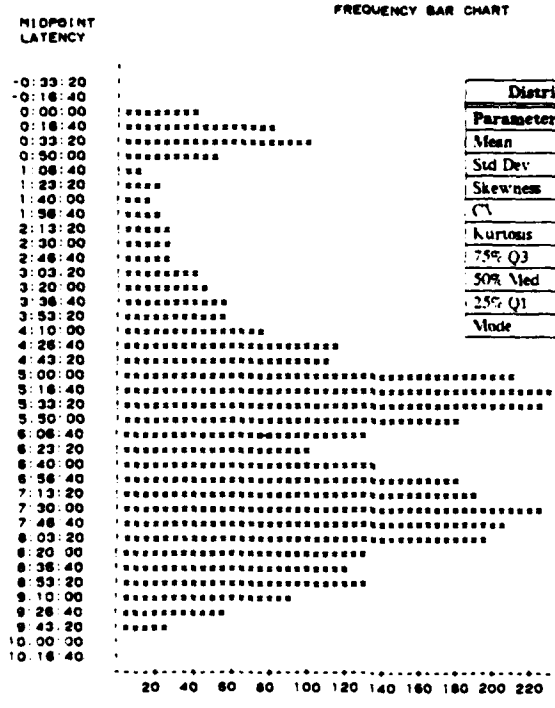
Distribution	
Parameter	hh:mm:ss
Mean	8:03:28
Std Dev	4:01:19
Skewness	-0.690
CV	49.917
Kurtosis	-0.844
75% Q3	11:18:15
50% Med	9:18:43
25% Q1	4:26:33
Mode	3:09:19

PERCENT	CUM PERCENT
0.00	0.00
0.00	0.00
0.00	0.00
0.00	0.00
3.16	3.16
4.39	7.55
2.48	10.02
1.51	11.53
1.32	12.85
1.37	14.23
2.78	17.00
2.34	19.35
1.21	20.56
2.78	23.34
3.61	26.95
1.13	28.08
0.43	28.51
0.38	28.89
0.84	29.72
0.92	30.64
1.16	31.80
2.34	34.14
2.56	36.70
3.83	40.53
5.12	45.65
7.79	54.43
6.17	60.60
3.58	64.19
4.47	68.66
6.66	75.32
7.48	82.81
5.31	88.11
4.02	93.13
4.20	97.33
2.07	99.41
0.58	100.00
0.00	100.00
0.00	100.00
0.00	100.00
0.00	100.00

Figure 7

ERROR LATENCY DISTRIBUTION
ERROR SEEDING WINDOW FROM 04:00:00 TO 05:00:00

FREQUENCY BAR CHART



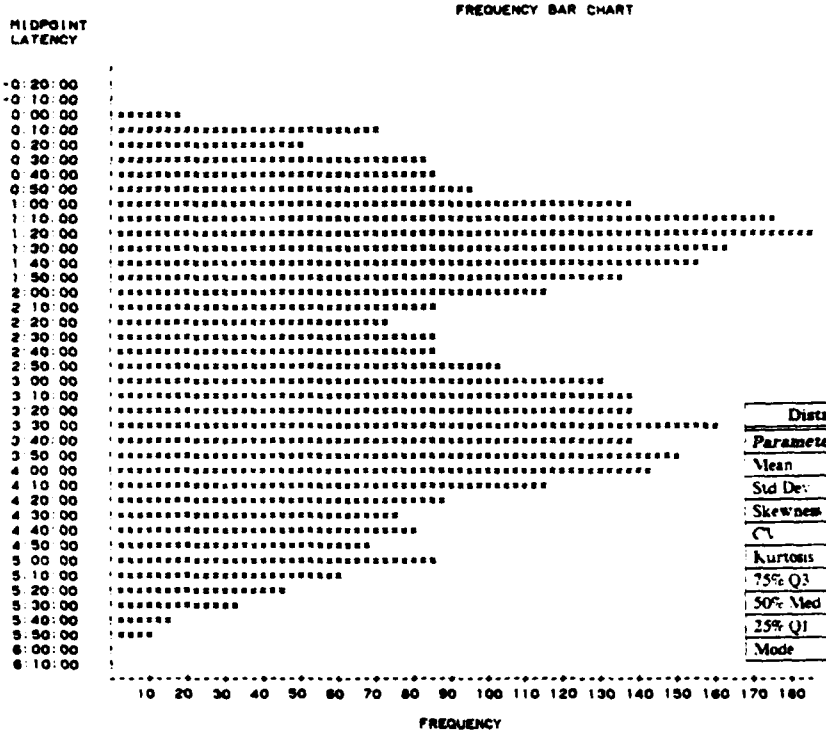
Distribution	
Parameter	hh:mm:ss
Mean	5:54:56
Std Dev	2:19:59
Skewness	-0.822
CV	39.442
Kurtosis	0.185
75% Q3	7:41:15
50% Med	6:07:25
25% Q1	4:52:34
Mode	5:09:37

PERCENT	CUM. PERCENT
0.00	0.00
0.00	0.00
1.15	1.15
2.24	3.38
2.79	6.18
1.34	7.52
0.27	7.80
0.49	8.29
0.36	8.64
0.52	9.16
0.71	9.87
0.68	10.56
0.71	11.27
1.04	12.31
1.23	13.54
1.53	15.07
1.48	16.55
2.08	18.63
3.15	21.77
3.01	24.78
5.74	30.53
8.26	38.79
6.18	42.97
4.90	47.87
3.56	51.42
2.76	54.18
3.72	57.90
4.87	62.77
5.22	68.00
6.10	74.10
5.63	79.73
5.28	85.01
3.53	88.54
3.31	91.85
3.56	95.40
2.46	97.87
1.46	99.34
0.63	99.97
0.03	100.00
0.00	100.00

Figure 8

ERROR LATENCY DISTRIBUTION
ERROR SEEDING WINDOW FROM 08:00:00 TO 09:00:00

FREQUENCY BAR CHART

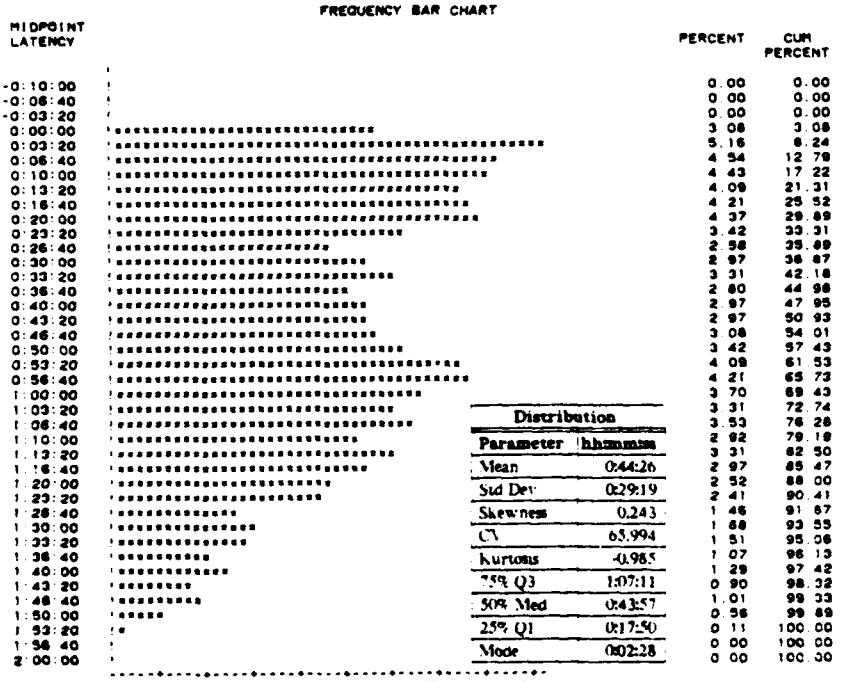


Distribution	
Parameter	hh:mm:ss
Mean	2:40:42
Std Dev	1:26:31
Skewness	0.110
CV	53.839
Kurtosis	-1.073
75% Q3	3:50:19
50% Med	2:43:16
25% Q1	1:24:36
Mode	1:30:13

Figure 9

ERROR LATENCY DISTRIBUTION
ERROR SEEDING WINDOW FROM 12:00:00 TO 13:00:00

FREQUENCY BAR CHART



Distribution	
Parameter	hh:mm:ss
Mean	0:44:26
Std Dev	0:29:19
Skewness	0.243
CV	65.994
Kurtosis	-0.985
75% Q3	1:07:11
50% Med	0:43:57
25% Q1	0:17:50
Mode	0:02:28

Figure 10

discovered in the 1st hour. Thus errors occurring at low workload are discovered withing 10 hours (on the average) versus only 1 hour for errors occurring at high workload.

4.1. Discussion of Results

The family of distributions generated by moving the error seeding window reveal a number of interesting issues.

It is seen that error latency is not so much a function of when in time the error occurred but rather a function of the workload that followed the occurrence of the error. A mode in the error latency distribution always occurs when there is a rise in system workload. Thus any rise in the workload results in a mode appearing in the latency distribution. A steady rise in workloads sweeps the errors out (higher error discovery) after which few if error remain to be discovered (low error discovery). The results suggest that an increase in workload causes a temporary increase the observed error rate. The error rate drops again after the errors have been discovered.

These results, apart from providing insight into the latency behavior, also suggest that the safest period for computing (assuming only latency effects) is at the tail end of an increasing workload. Since errors that have occurred in the past would have been discovered and the error rate will be nearly constant. The SLAC [Iyer 82b] and CMU [Castillo 81] results however show a sustained increase in the error rate as workload increases. Thus we have fresh evidence to suggest that there is more to a workload/error relationship than simply a latent discovery effect. This work however serves to isolate the impact of error latency which was not possible so far.

5. CONCLUDING REMARKS

In this report we established a methodology for determining and characterizing error latency. The method is based on real workload data gathered by an experiment instrumented on a VAX 11/780 during the normal workload cycle of the installation. Distributions of error latency were generated by seeding errors under varying workload conditions. The family of error latency distributions obtained illustrate that error latency is not so much a function of when in time an error occurred but rather a function of the workload that followed the error. The study finds that the mean error latency varies by a 10 to 1 between low and high workloads. The methodology is not restricted to systems studied and has general applicability.

ACKNOWLEDGMENT

We thank Professor E. S. Davidson for valuable discussions during the course of this work; G. D. McNiven, R. Berry and S. Laha for their assistance in instrumenting the acquisition system on the VAX.

This work was supported in part by the Joint Services Electronics Program (U.S. Army, U.S. Navy and the U.S. Air Force) under contract N00014-84-C-0149 and in part by the Graduate Research Board, University of Illinois, Urbana-Champaign.

REFERENCES

[Butner 80]
S. E. Butner and R. K. Iyer, "A Statistical Study of Reliability and System Load at SLAC," *Digest, Tenth International Symposium on Fault Tolerant Computing*, Kyoto, Japan, Oct 1980.

[Castillo 80]
X. Castillo and D. P. Siewiorek, "A Performance Reliability Model for Computing Systems," *Digest, Tenth International Symposium on Fault Tolerant Computing*, Kyoto, Japan, Oct 1980.

[Castillo 81]
X. Castillo and D. P. Siewiorek, "Workload, Performance and Reliability of Digital Computing Systems," *Digest, Eleventh International Symposium on Fault-Tolerant Computing*, Portland, Maine, June 1981, pp. 84-89.

[Cortes 84]
Mario L. Cortes and R. K. Iyer, "Device Failures and System Activity: A Thermal Effects Model," "*Digest, Fourteenth Inter. Symposium on Fault-Tolerant Computing*," Orlando, Florida June 1984.

[DEC 80a]
Digital Equipment Corporation, *VAX Architecture Handbook*, Digital Equipment Corporation, Maynard, MA, 1980.

[DEC 80b]
Digital Equipment Corporation, *VAX Architecture Handbook*, Digital Equipment Corporation, Maynard, MA, 1980.

[DEC 80c]
Digital Equipment Corporation, *KA780 Field Maintenance Print Set*, Digital Equipment Corporation, Maynard, MA, 1980.

[Gunther 80]
N. L. Gunther and W. C. Carter, "Remarks on the Prob. of detecting faults," *Digest 10th International Symposium on Fault-Tolerance Computing*, Kyoto, Japan, Oct 1980.

[Iyer 82a]
R. K. Iyer, S. E. Butner and E. J. McCluskey, "A Statistical Failure/Load Relationship: Results of a Multi-Computer Study," *IEEE Transactions on Computers*, July 1982.

[Iyer 82b]
R. K. Iyer and D. J. Rossetti, "A Statistical Load Dependency of CPU Errors at SLAC," *Digest, 12th International Symposium on Fault Tolerant Computing*, Santa Monica, California, June 1982.

[Iyer 83]
R. K. Iyer and David J. Rossetti, "Permanent CPU Errors and System Activity: Measurement and Modeling", *Digest, Real-Time Systems Symposium*, Arlington, Virginia, Dec 1983.

[McGough 81]
John G. McGough and Fred L. Swern, "Measurement of Fault Latency in a Digital Avionic Mini Processor," *NASA Contractor Report 3462, Oct 1981.*

[McGough 83]
John G. McGough and Fred L. Swern, "Measurement of Fault Latency in a Digital Avionic Mini Processor Part II," *NASA Contractor Report 3651, Jan 1983.*

[Rossetti 81]
D. J. Rossetti and R. K. Iyer, "A Software System for Reliability and Workload Analysis," *CRC Tech Rpt 81-18*, Center for Reliable Computing, Computer Systems Laboratory, Stanford Univ, Stanford, C.A., Dec 1981.

[Shedletsky 73]
J. J. Schedletsky and E. J. McCluskey, "The Error Latency of a Fault in a Combinational Circuit," *Digest FTCS-3, June 1973.*

[TEK 84]
Tektronix, *User's Manual 91DVV1 For VAX/UNIX 4.1bsd Release1, 1984.* Tektronix, Oregon, USA.

END

FILMED

1-86

DTIC