January 1987

UILU-ENG-87-2210
CSG-61

COORDINATED SCIENCE LABORATORY
College of Engineering

A MEASUREMENT-BASED STUDY OF CONCURRENCY IN A MULTIPROCESSOR

Patrick John McGuire

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) UILU-ENG-87-2210 (CSG-61)	5. MONITORING ORGANIZATION REPORT NUMBER(S) NASA NSF DOE
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Lab University of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION NASA NSF DOE
6c. ADDRESS (City, State, and ZIP Code) 1101 W. Springfield Avenue Urbana, IL 61801	7b. ADDRESS (City, State, and ZIP Code) NASA: NASA-Langley Research Center Hamptpon, VA NSF: 1800 G St. N.W., Washington, DC 20550
8a. NAME OF FUNDING/SPONSORING ORGANIZATION NASA, National Science Foundation, & DoE 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER NASA: NAG-1-613 NSF: DCR-84-10110 DOE: DOE DE FG02-85ER25001
8c. ADDRESS (City, State, and ZIP Code) NASA: Hampton, VA NSF: Washington, DC 20550 DOE: Washington, DC	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

A Measurement-Based Study of Concurrency in a Multiprocessor

12. PERSONAL AUTHOR(S)
McGuire, Patrick John

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM _____ TO _____	January 1987	56

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Concurrency, multiporcessors, Alliant FX/8 cache miss rate, bus activity

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

This thesis develops a systematic measurement-based methodology for characterizing the amount of concurrency present in a workload, and the effect of concurrency on systems performance idices such as cache miss rate and bus activity. Hardware and software instrumentation of an Alliant FX/8 at the Universtiy of Illinois Center for Supercomputing Research and Development was used to obtain data form a real workload environment. Results show that 35% of the workload is concurrent, with the concurrent periods typically using all available processores. Measurements of periods of change in concurrency show uneven usage of processors during these times. Other system measures, including cache miss rate and processor bus activity, are analyzed with respect to the concurrency measures. Probability of a cache miss is seen to increase with concurrency. The change in the cache miss rate is much more sensitive to the fraction of concurrent code in the workload than the number of processors active during concurrency. Regression models are developed to quantify the relationships

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

between cache miss rate, bus activity and the concurrency measures. The model for cache miss rate predicts an increase in the median miss rate value of as much as 300% for a 100% increase in concurrency in the workload.

ABSTRACT

This thesis develops a systematic measurement–based methodology for characterizing the amount of concurrency present in a workload, and the effect of concurrency on system performance indices such as cache miss rate and bus activity. Hardware and software instrumentation of an Alliant FX/8 at the University of Illinois Center for Supercomputing Research and Development was used to obtain data from a real workload environment. Results show that 35% of the workload is concurrent, with the concurrent periods typically using all available processors. Measurements of periods of change in concurrency show uneven usage of processors during these times. Other system measures, including cache miss rate and processor bus activity, are analyzed with respect to the concurrency measures. Probability of a cache miss is seen to increase with concurrency. The change in the cache miss rate is much more sensitive to the fraction of concurrent code in the workload than the number of processors active during concurrency. Regression models are developed to quantify the relationships between cache miss rate, bus activity and the concurrency measures. The model for cache miss rate predicts an increase in the median miss rate value of as much as 300% for a 100% increase in concurrency in the workload.

PRECEDING PAGE BLANK NOT FILMED

PRECEDING PAGE BLANK NOT FILMED

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

A MEASUREMENT-BASED STUDY OF CONCURRENCY IN A MULTIPROCESSOR

BY

PATRICK JOHN MCGUIRE

B.S.E.E., University of Santa Clara, 1981

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1987

Urbana, Illinois

ACKNOWLEDGMENT

We thank Ed Davidson at CSRD for his support and for permitting us to instrument the Alliant system. Thanks are also due to Bob McGrath, Richard Barton, Allen Maloney and Tracy Tilton, at CSRD, for their invaluable assistance.

This work was supported by NASA grant NAG-1-613 with additional support from the National Science Foundation NSF(DCR84-10110) and Dept. of Energy 3-6124(DOE DE FG02-85ER25001).

CHAPTER 1.

INTRODUCTION

The study of concurrent operations is an important part of evaluating parallel processing machines. Several analytical and simulation studies, such as those found in [1], [2], [3], and [4], have been undertaken to evaluate such machines, but few if any investigate multiprocessor performance in a real workload environment. Such measurements are important for developing realistic techniques to measure and model concurrent behavior of system workloads.

This project is concerned with the development of a measurement–based technique to study the use of loop–level concurrency in a real production workload. Measurements were performed on the Alliant FX/8 system at the Center for Supercomputing Research and Development (CSRD) at the University of Illinois at Urbana–Champaign. The FX/8 measured in the study is used primarily for development of numerical applications software. Programs developed on the machine range from high level software (FORTRAN), such as structural mechanics and circuit simulation, to assembly–level kernels for linear system solving [5], [6]. The Alliant is also networked to several other department machines.

The two main objectives of this work are 1) to find the percentage of concurrent operations in the workload and the use of processing resources in these operations and 2) to study the system overheads associated with concurrency in the workload environment, including the effect of concurrency on other system performance measures. The methodology developed is general and in principle can be applied to study other parallel systems. The thesis first proposes measures for characterizing concurrency in the system. Probability distributions of the values of these measures show the extent of concurrent operations in the workload. Particular attention is paid to

the end of concurrent loops, and the corresponding overheads in these periods. Regression techniques are used to assess the impact of increased system concurrency on other system measures, including bus utilization and cache miss rate.

Results from workload measurement show that the system workload is concurrent 35% of the time, and that concurrent periods typically use all available processors. Measurements of the end of concurrent operations indicate an uneven use of processors during these periods. Joint analysis of concurrency and system performance measures such as cache miss rate and processor bus activity shows that the probability of high values of these measures increase as concurrency levels increase. Importantly, cache miss rate is shown to depend much more strongly on the fraction of parallel code in the workload than the number of processors active during concurrent operations. In particular, a 100% increase in the fraction of concurrent operations in the workload results can result in greater than a 300% increase in cache miss rate.

The following chapter presents background information and related research in the multiprocessor performance evaluation area. Chapter 3 describes the measurement environment of the study, including the Alliant FX/8 and the instrumentation used. Chapter 4 shows results of workload measurements, with special attention to periods of changes in concurrency. Chapter 5 deals with relationships between system measures and concurrency, and Chapter 6 summarizes results.

CHAPTER 2.

BACKGROUND AND MOTIVATION

Concurrent operations in multiprocessing systems typically exist at three levels. At a high level are processes or tasks, either independent or related, which execute on separate processors. A low level of parallelism is the pipelined vector processing available on supercomputers [7]. Several simultaneous arithmetic operations are possible in these machines, on data contained in special vector registers. An intermediate level is loop concurrency, in which multiple iterations of a program loop are assigned to separate processors. A large effort has been made to apply this level of program concurrency to groups of processors, for example, see [8] and [9]. This thesis is concerned with evaluation of loop concurrency under real workload conditions.

In loop concurrent operations, processors may proceed independently, if there is no dependency between loop iterations. The efficiency of the execution of this loop (which may include several more nested loops) is dependent on how the number of total iterations matches the number of available processors; for example, assignments which require some small number of processors to execute in parallel, while others remain idle (because there are no more iterations to execute) result in lower efficiency [8]. Performance is also dependent on contention between processors for shared resources (.e.g., cache, memory). Sets of loop iterations that contain dependencies are further restricted in their execution. Dependencies may cause additional loop execution overhead, since processors may have to wait on those executing previous iterations to satisfy the dependence [10].

2.1 Related Research

There have been many studies of parallel machine performance and concurrency. Most have employed simulation and analytical–based techniques; examples are found in [1], [2], [3], and [4]. None of these studies have measured concurrency usage on a real workload of a multiprocessing system. Such measurements are important for determining the sensitivity of system performance to changes in the level of concurrency. The knowledge of these effects are essential for developing performance evaluation methodologies. Furthermore, the results can be applied to control strategies, such as processor scheduling, within the multiprocessor.

Two common measures of performance for a multiprocessing computer are *Speedup* and *Efficiency*. Speedup is defined as $S = \mathrm{T}_1/T_\mathrm{P}$, where T_1 is the execution time required for a program on a single processor, and T_P is the execution of the program on P processors. Efficiency is given by the ratio $E_\mathrm{P} = S_\mathrm{P}/\mathrm{P}$, $0 < E_\mathrm{P} < 1$ [11]. Speedup measures obtained from measurements on the Alliant FX/8 are given in [12]. Speedup and Efficiency yield information about the improvement in a program, but they are unable to provide a detailed characterization of the program or system behavior. Importantly, when performance evaluation of a *real production* workload is of interest, there is no direct applicability of the Speedup and Efficiency measures.

Other studies of multiprocessor systems, such as [13] and [14], have investigated more detailed aspects of machine performance. In [14] it is shown that shared resource contention, which typically grows with the number of processors present in a multiprocessor, can be a limiting performance factor. In [15], measurements on the FX/8 deal specifically with the effect of the machine's memory hierarchy.

Research more closely related to the work in this thesis is presented in [16] and [17]. These studies use hardware monitoring and special event marker instructions embedded in programs to acquire execution traces. Captured events on different processors are time–stamped, and the

composite trace yields information about the overlapping operations (concurrency) in the program. Since this technique requires specific code insertion in programs, it is difficult to apply to the observation of a real workload of programs generated by multiple users.

None of the other measurement studies address the issues of evaluating the amount of concurrency in a workload, or relate this measured concurrency to the behavior of other system components, such as cache and main memory. This thesis studies three aspects of concurrency. The percentage of concurrency present in the workload is measured, along with the number of processors used during parallel operations. The end of loop–concurrent operations is studied to focus on overheads associated with concurrency. These operations are subject to performance degradations due to contention for shared resources, waiting associated with dependency resolution, and less than full utilization of processing resources. The impact of concurrency on system performance measures is also analyzed to find the relationship between concurrency and other indices, including cache miss rate and processor bus activity.

To investigate the issues described above, the Alliant FX/8 at CSRD was instrumented to extract data related to concurrency and other system performance measures. The following chapter includes a brief description of the FX/8 and its loop–concurrency mechanism, the instrumentation, experiment setup, and the basic measurements made on the machine.

CHAPTER 3.

EXPERIMENTAL ENVIRONMENT

This chapter describes the Alliant FX/8, the instrumentation used for the measurements, and the experimental setup for the work. A more detailed description of the FX/8 may be found in Appendix C.

3.1 System Description

Measurements described here were performed on the Alliant FX/8 computer system. This machine is a shared–memory multiprocessor, with an advertised peak performance of 94.4 million floating point operations per second (MFLOPS) [18]. A diagram of the Alliant FX/8 configuration used in the measurements is shown in Figure 1.

Concurrency on the FX/8 is supported on the "Computational Cluster" (hereafter referred to as the Cluster) of eight Computing Elements (CEs). These processors have floating point and vector processing capabilities, and are linked through a common cache to shared memory. Interactive Processors (IPs) handle interactive traffic, operating system functions, and I/O. The machine supports an extension of 4.2 BSD UNIX, called Concentrix [18].

3.2 Alliant Concurrency

This study is concerned with the measurement and evaluation of loop–level concurrency on the FX/8. The Alliant FORTRAN compiler attempts to transform DO loops or array operations into a parallel form, where the iterations of a loop will be executed on separate CEs. Figure 2 shows how a concurrent loop is executed on the CE cluster.

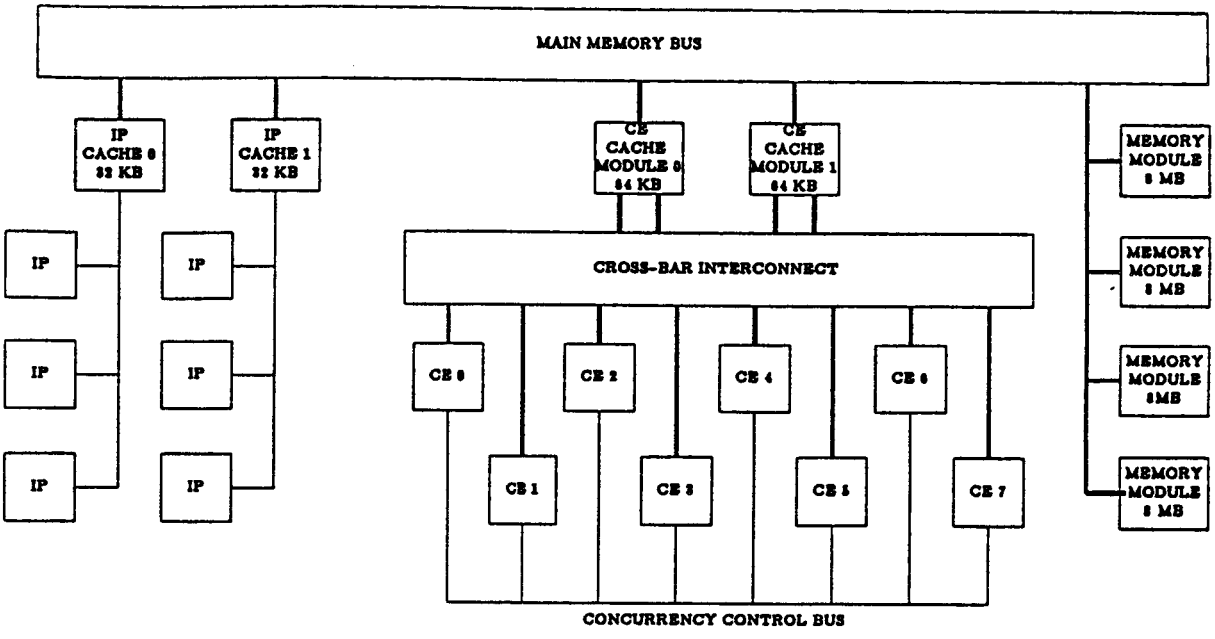

Figure 1. Configuration of the Measured Alliant FX/8.

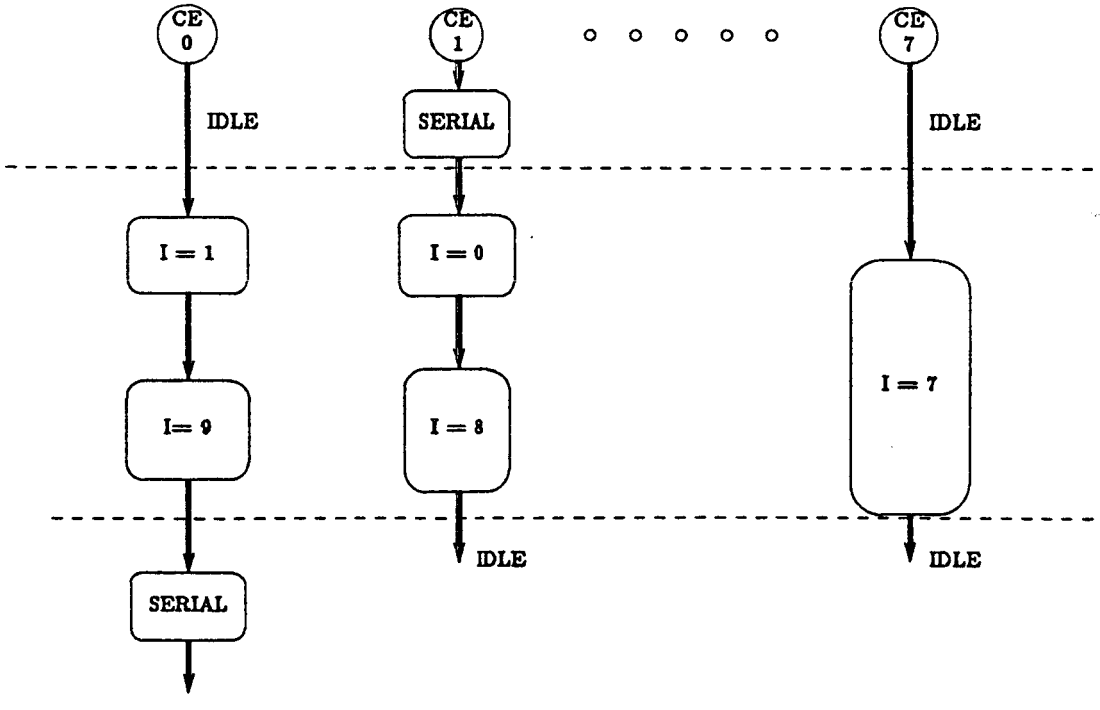

Figure 2. Execution of a Concurrent Loop.

A program is executed serially until a special instruction is encountered which enables the start of concurrent operation. Iterations of the DO loop are assigned to CEs in a self–scheduled fashion [19]. Processors are assigned iterations until all iterations are executed. As shown in the figure, the processor which executes the last iteration will continue serial execution after all iterations are complete, and need not be the same processor that entered the loop serially. The above loop execution may be complicated by dependencies which exist between iterations, where synchronization will be required between CEs to enforce correct program operation.

The Alliant FORTRAN compiler attempts to transform DO loops or array operations into a parallel form, where the iterations of a loop will be executed on separate CEs. Both synchronization and processor scheduling functions are handled in hardware, and make use of the Concurrency Control Bus shown in Figure 1 [18].

3.3 Instrumentation

The Alliant FX/8 was instrumented at both the hardware and system software levels. Hardware measurements yielded information about CE concurrency and system bus activity, while software measurements consisted of counts of events logged by the operating system kernel. The two levels of measurements occurred simultaneously but independently. Minimal system overhead was incurred for measurements; the hardware monitoring is inherently non–intrusive, and statistics–gathering software was that which was normally running in Concentrix. A feature of this instrumentation approach was the fact that no modifications were required to the system in order to perform the measurements. When monitoring a real workload, instrumentation must normally gather data without relying on any special operations of the system or programs monitored, since modification of user and system programs for measurement purposes is not possible in many cases.

The hardware monitoring was accomplished with a Tektronix DAS 9100 Series logic analyzer [20]. This instrument acquires the state of up to 80 signals (on the unit used), and stores this data in a 512–deep buffer memory. The DAS is fully controllable through an i/o port; all experiments used this feature to control the instrument, as well as to transfer acquired buffers to files resident on the Alliant system.

Probes from the DAS were connected to the FX/8 at three different logical points:

1) Bus opcode was monitored for each CE, where the bus was that between the CE and the CE Cache, on the CE's side of the crossbar switch. Bus opcode indicates what type of operation (read, write, idle, etc.) is occupying the bus.

2) The shared memory bus opcode was monitored, yielding information about interactions between memory and cache and between multiple caches.

3) The Concurrency Control Bus was also monitored, to determine whether a processor was active in concurrent operation, or not in a concurrent–active state.

As mentioned above, the software measurements were those normally collected by the Concentrix kernel, made available by a program written internally at CSRD to extract the values. The operating system logs counts continuously for a variety of memory management, scheduling, and interrupt variables. In this study, the measurement extracted was page faults generated by the CEs.

3.4 Experiment Setup

The measurements were controlled by UNIX C–Shell script programs executing on the FX/8, which controlled collection of both the hardware and software data. The programs have the ability to configure the DAS monitor, enable the monitor's triggering, transfer the data from the instrument to a host system, and reduce the acquired data to appropriate event counts (e.g., number of cache read operations). Table 1 shows the reduced set of events derived from a monitor buffer.

In these experiments, the objective was to observe the CE Cluster; therefore we chose the IP as the computing resource for executing the measurement control software. The Concentrix system allows control over what type of computing resource (IP, CE, Cluster with 1 to 8 CEs, or don't care) a program will run on, provided that resource has the correct capabilities for the program [21]. Using the IP kept measurement artifact at a minimum for our experiments.

3.5 Measurements

Two types of measurements were performed on the FX/8. The first used random sampling of the system to acquire data from the real workload on the machine. Nine sessions of this type

HARDWARE MEASUREMENT EVENT COUNTS	
Name	Event
numj	number of records with j processors active
profj	number of records with processor j active
ceopj	number of records with CE bus opcode = j
dmbopj	number of records with mem bus opcode = j

TABLE 1. Hardware Event Counts.

were performed on seven different midweek days, when the machine is used most heavily. Each session lasted between four and eight hours. Five snapshots of the system were taken and grouped together in a five–minute interval. A real–time program was written to condense the acquisition into event counts; the result was then written to disk. Software measurements were taken simultaneously with the hardware measurements. These were recorded at the time that the hardware sample was stored.

A second group of measurements was executed in order to extract system behavior when the system was executing with high concurrency. These experiments dealt with hardware measurements only. High concurrency operation was captured by triggering the hardware monitor at times when the FX/8 Cluster was operating in a concurrent mode. Two different trigger events were used. In ten of the experiment sessions, the monitor was triggered when all eight processors in the Cluster were active, while in five other experiments, the transition from eight processors active to a smaller number active was the trigger event. This latter condition was chosen particularly to try to determine the behavior of the Cluster during times when the level of concurrency in the machine was changing.

Processing of the measured data was performed with the Statistical Analysis System (SAS) package on an IBM 4381. SAS provides a large set of data analysis procedures, including graphical presentation, regression, clustering, and analysis of variance [22].

The next chapter describes the analysis performed on the above measurements. The first section presents a definition of concurrency measures, and results obtained from the random sampling of workload. A description of periods of transition of concurrency is then detailed.

CHAPTER 4.

ANALYSIS OF MEASURED DATA

The analysis of the measurements in this study are presented in this chapter. Measurements are defined which assist in characterizing concurrency in the workload, and distributions of these measures for the acquired data are given. Analysis of periods of transitions in the level of system concurrency is also performed, in order to examine the overheads associated with these transitions.

4.1 Concurrency Measures

In order to quantify concurrency in the workload, certain measures are defined. The workload is one choice of scope for the measures; they could easily be applied at other levels, such as a program or sub–program.

We first define a measure c_j as follows:

$$c_j = Prob(Number\ of\ Active\ Processors = j) \tag{4.1}$$

We call c_j *j–concurrency*. From c_j we derive the *Workload Concurrency*, which is the probability that there is any level of concurrency (2 or more processors operating in parallel) in the system:

$$C_w = \sum_{i=2}^{P} c_j \tag{4.2}$$

The above measures deal with the amount of concurrency in the workload; we now restrict our attention to times when the system is operating concurrently.

$$c_{j|c} = Prob(Number\ of\ Active\ Processors = j \mid Number\ of\ Active\ Processors > 1) \qquad (4.3)$$

$c_{j|c}$ measures the *j-concurrency* value for only those times when concurrency exists in the system. If all c_j values from 2 to P are 0, this value is undefined. From $c_{j|c}$ we calculate the *Mean Concurrency Level*, which is the average number of processors operating concurrently when at least 2 processors are active. P_c is a measures utilization of computing resources *during concurrency*, and may vary from 2 to P.

$$P_c = \sum_{j=2}^{P} j * c_{j|c} \qquad (4.4)$$

The above measures may be applied at any level of multiprocessing capability of a given machine. In our experiments, we apply the measures to the specific case of loop–level concurrency in the overall workload of the multiprocessor.

4.2 Workload Sampling Results

Figure 3 shows the distribution of the number of active processors over all the measurement sessions. This figure shows the dominant concurrency states of the system as well as the ratio of concurrency to serial activity. The high points on the distribution at eight, one, and zero processors active show that the CE Cluster spends the majority of its time in one of three states: full concurrency, serial, or idle.[1] This analysis was performed on data from individual measurement sessions and the sum of all sessions. Distributions of processor activity in individual sessions showed significant variation during different periods; examples for two sessions are shown in Appendix A.

[1] Idle in this context is with respect to *Concurrent–Mode* operation. Detached processes (exclusively serial) may constitute a portion of these states.

From the processor distribution, the concurrency measures defined in equations 4.1–4.4 may be calculated. Table 2 shows these values for the sum of random–sample sessions. The value of C_w shows that concurrency in the workload is at 35% for the full set of measurement sessions. Eight processors are active 28% of the time in the overall workload, but when the system is concurrent, the 8–active state predominates at 93% ($c_{8|c}$) resulting in a Mean Concurrency Level of 7.66.

For each sample (five minute period), a distribution of number of active processors was generated, and the corresponding concurrency measures calculated. Figures 4 and 5 show the

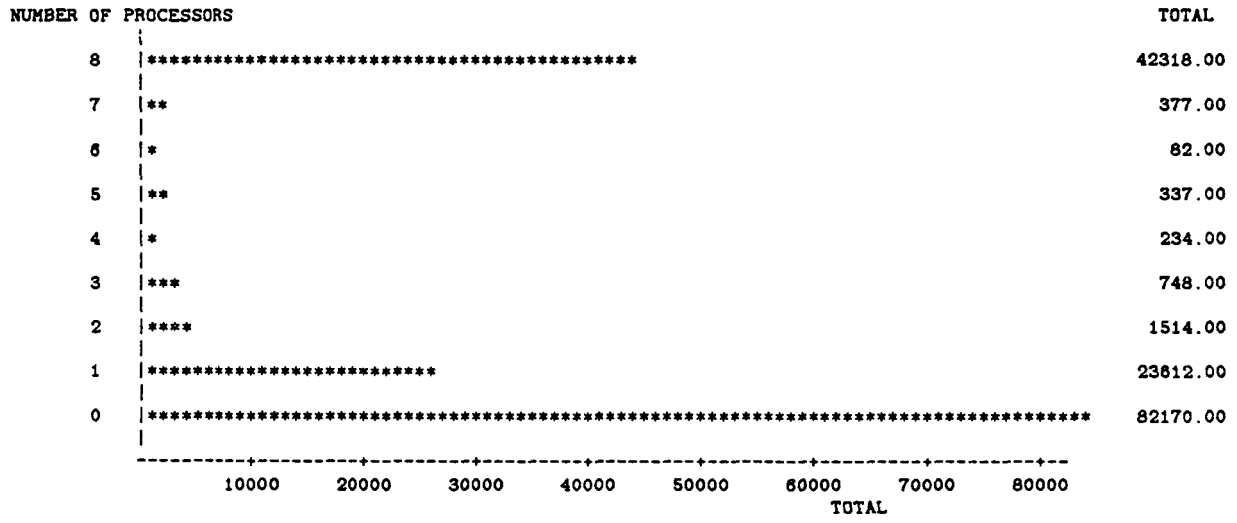

Figure 3.
Number of Records with N Processors Active / All Sessions.

CONCURRENCY MEASURES – ALL SESSIONS							
c_2	c_3	c_4	c_5	c_6	c_7	c_8	C_w
0.0100	0.0049	0.0015	0.0022	0.0005	0.0025	0.2795	0.3506
$c_{2\|c}$	$c_{3\|c}$	$c_{4\|c}$	$c_{5\|c}$	$c_{6\|c}$	$c_{7\|c}$	$c_{8\|c}$	P_c
0.0331	0.0164	0.0051	0.0074	0.0018	0.0083	0.9278	7.66

Table 2. Overall Concurrency Measures for All Sessions.

distribution of samples with particular values of Workload Concurrency and Mean Concurrency Level, respectively. Some points are of note in the distributions. The first is the large large percentage of samples with Workload Concurrency measures at or near zero, indicating serial code execution or idle states during these periods. Some concurrency in the workload exists for 55% of the samples. Note that this number is significantly different from C_w found in the overall average. The reason for this is a difference between five–minute and overall average behavior. Note from Figure 4 that there are many samples with low but non–zero Workload Concurrency, which do not contribute significantly to the value of C_w for the total group of measurement sessions. For samples with non–zero C_w, greater than 94% of samples have a Mean Concurrency Level higher than 6.5. (Recall that for any Workload Concurrency value of zero, Mean Concurrency Level is not calculated, since it is a measure of the system *when concurrent*). Hence, concurrency which does appear in the measured workload has a characteristically high utilization of the total available concurrency resource.

While concurrency level is most often high, there are some periods which are not maximum. Clearly, overheads due to multiprocessing can contribute to this problem. Study of periods of change in concurrency can yield information about these overheads; this approach is described in the following section.

4.3 Concurrency Transitions

On the FX/8, transitions in concurrency typically happen at the end of a DO loop, when there are no remaining iterations to perform, and processors begin to become idle while waiting for serial execution to continue. These transitions affect the overall efficiency of parallel operations. These idle periods correspond to a multiprocessing overhead; if the transition from P processors to one (serial) is instantaneous, processors do not incur any idle time, and Mean Con-

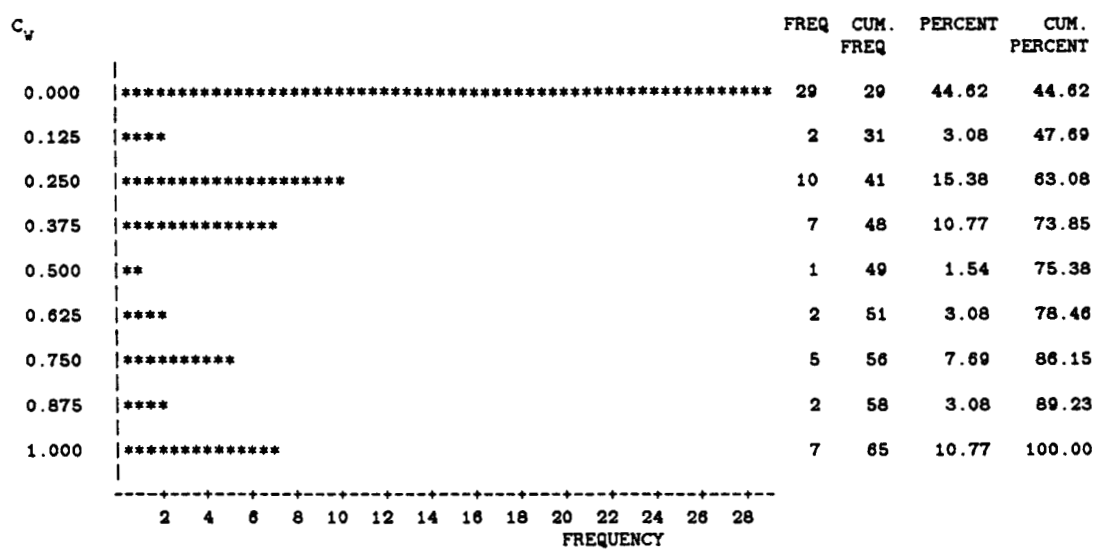

Figure 4.
Distribution of Samples by Workload Concurrency. / All Sessions.

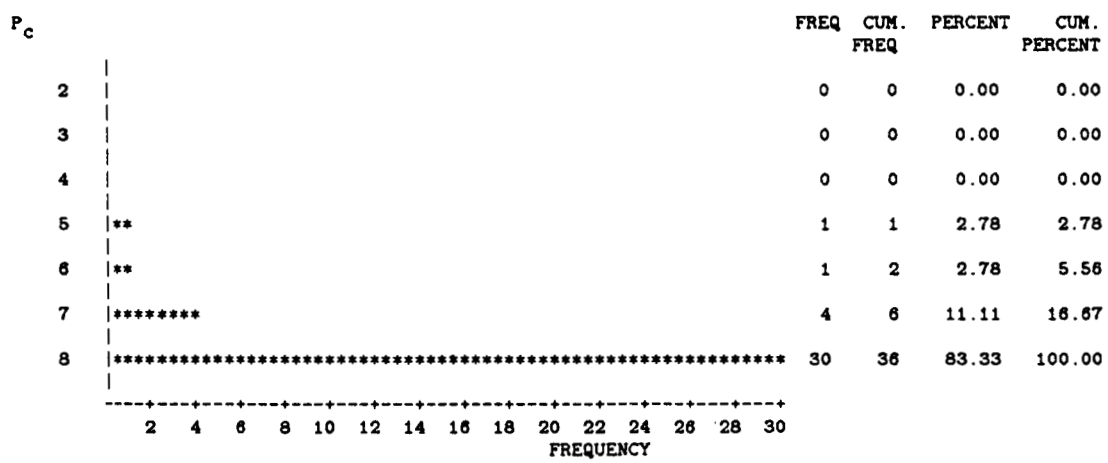

Figure 5.
Distribution of Samples by Mean Concurrency Level / All Sessions.

currency Level, P_c, is maximum. To investigate the efficiency during transition periods, a specific set of measurements was performed in which monitoring began when processor activity changed from all processors active (full–concurrency) to a lower concurrency level.

Figure 6 is the distribution of active processors for the concurrency transition periods. The transition of interest is *between* 8–concurrency and 1–concurrency; hence the distribution shows the number of records for 7 through 2 processors active. The state with 2 processors active accounts for 52% of the transition states. Average transition behavior, therefore, has transitions between 7 and 2 processors active which occur significantly faster than the transition from 2 processors to serial operation.

Figure 7 shows the distribution of activity by individual processor during transitions. Processors 7 and 0 appear to be active significantly more often than the other processors, corresponding to the 2–concurrency peak in Figure 6, while processors 2, 3, and 4 are significantly less active than the others.

A simple reason for uneven distribution of processor activity is a loop count which is I = 8*j + 2, for integer j. If loop iterations throughout are equal in execution time, this would result in two iterations remaining after 8*j iterations have been executed, which would then take full

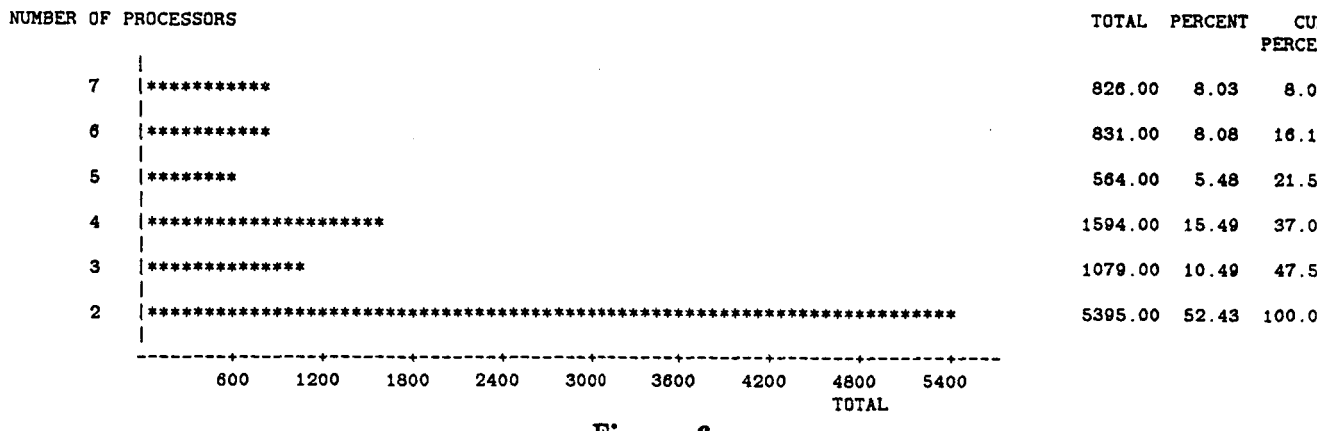

Figure 6.
Number of Records with N Processors Active / Concurrency Transition Periods.

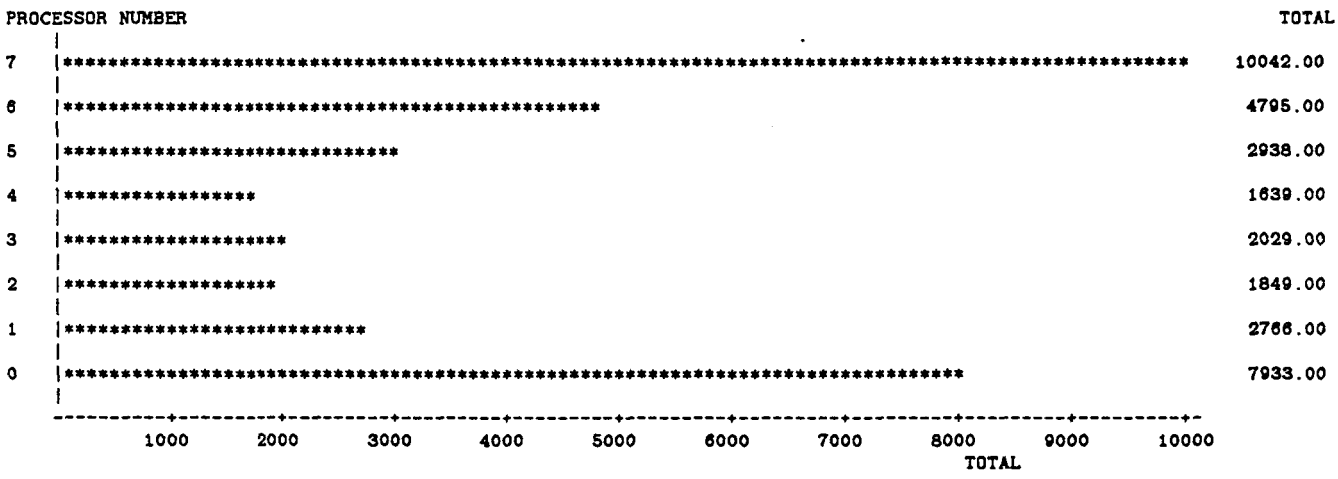

Figure 7.
Number of Records Active by Processor Number / Concurrency Transition Periods.

iteration times to complete. A particular number of "leftover" iterations occuring dominantly in the workload would seem unlikely, however. Second, processors executing a loop may not follow the same execution path, due to conditional branching which is iteration–dependent. The processor executing the shorter path will obviously finish earlier, and be available for scheduling a new iteration while the longer iteration is still executing. Data access patterns are also likely to differ between iterations, which may result in differences in memory latency, if cache misses and/or page faults are generated by the processor. Such variation in latency causes processors to lead or lag one another. Dependencies in a program loop may also result in uneven distribution of activity among processors, as some processors are waiting more than others. Finally, contention for resources such as memory can contribute to uneven processor activity. If priority schemes favor particular processors, these will suffer greater delay, increasing the probability that they will trail other processors in execution at the end of the loop.

4.4 Discussion of Results

Concurrency measures presented in this chapter describe what fraction of the workload is concurrent (C_w) and how many processors are used during concurrent operations (P_c). Random sampling of the workload during nine different sessions show that $C_w = 0.35$, indicating that about one–third of the workload is devoted to concurrent operations. An overall Mean Concurrency level of 7.66 shows that parallel operations have a high utilization of the machine's processors.

Transitions in concurrency, typically at the end of parallel loops, show uneven use of processors. The 2–concurrent state is significantly more frequent than other "transition" states. In particular, CEs 7 and 0 tend to show more activity than other processors during transition; as other processors begin to become idle, these two typically continue to execute. Several reasons for this are possible, including a high frequency of loop counts that result in two "leftover" iterations. Also, uneven distribution of waiting time among processors, due to priority assignment for shared resources and/or loop iteration dependency may result in differences in activity between processors.

The variation in system concurrency affects other system components related to processor activity; these include system busses and cache memory. Measured data is analyzed to determine this relationship in the following chapter.

CHAPTER 5.

CONCURRENCY AND SYSTEM MEASURES

In this chapter the effect of system concurrency measures on key system performances measures is described. Analysis was performed for both the combination of random sampling and high concurrency measurement periods. Three measures, CE Bus Busy, Cache Miss Rate (Missrate), and Page Fault Rate were calculated from the acquired data. CE Bus Busy is the fraction of processor–to–cache bus cycles that are not idle in the measured interval. The value shown in the following analysis is the average value of this fraction over all eight busses, and is a measure of the total information flow between CEs and Cache. Missrate is the fraction of total bus cycles corresponding to cache misses. Page Fault Rate is the sum of user–mode and system–mode page faults generated by the CEs during the measurement interval.

5.1 Cache Miss Rate

Figures 8 and 9 show the scatter–plots of the cache miss rate, against both Workload Concurrency and Mean Concurrency Level. Similar plots for CE Bus Busy and Page Fault Rate are shown in Appendix B. An inspection of Figure 8 shows that the highest Missrate values occur at maximum Workload Concurrency. In addition, an increase Workload Concurrency appears to increase the probability of a high Missrate value.

Figure 9 also shows some increasing probability of high Missrate as P_c increases, although the Missrate is relatively unchanged after $P_c > 7.0$.

The distributions for Missrate are plotted in Figures 10 and 11 for increasing values of C_w and P_c. (See Appendix B for the distributions of CE Bus Busy and Page Fault Rate.) Note that

the median Missrate value for $0.4 < C_w <= 0.8$ is .009, and increases sharply to 0.023 for $C_w > 0.8$. Median value of Missrate shows no increase between the middle and high ranges of P_c, indicating less sensitivity to this measure than C_w.

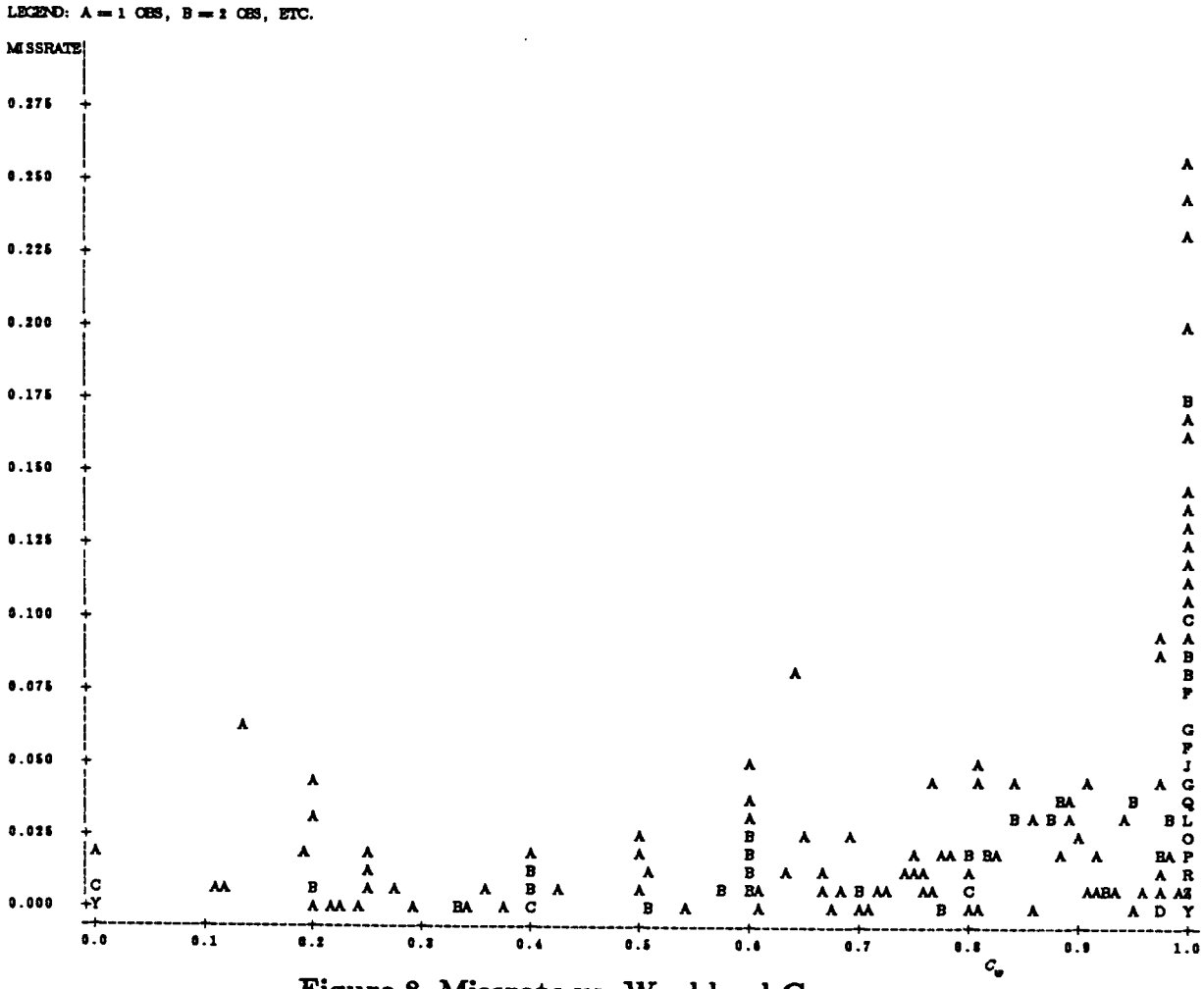

Figure 8. Missrate vs. Workload Concurrency.

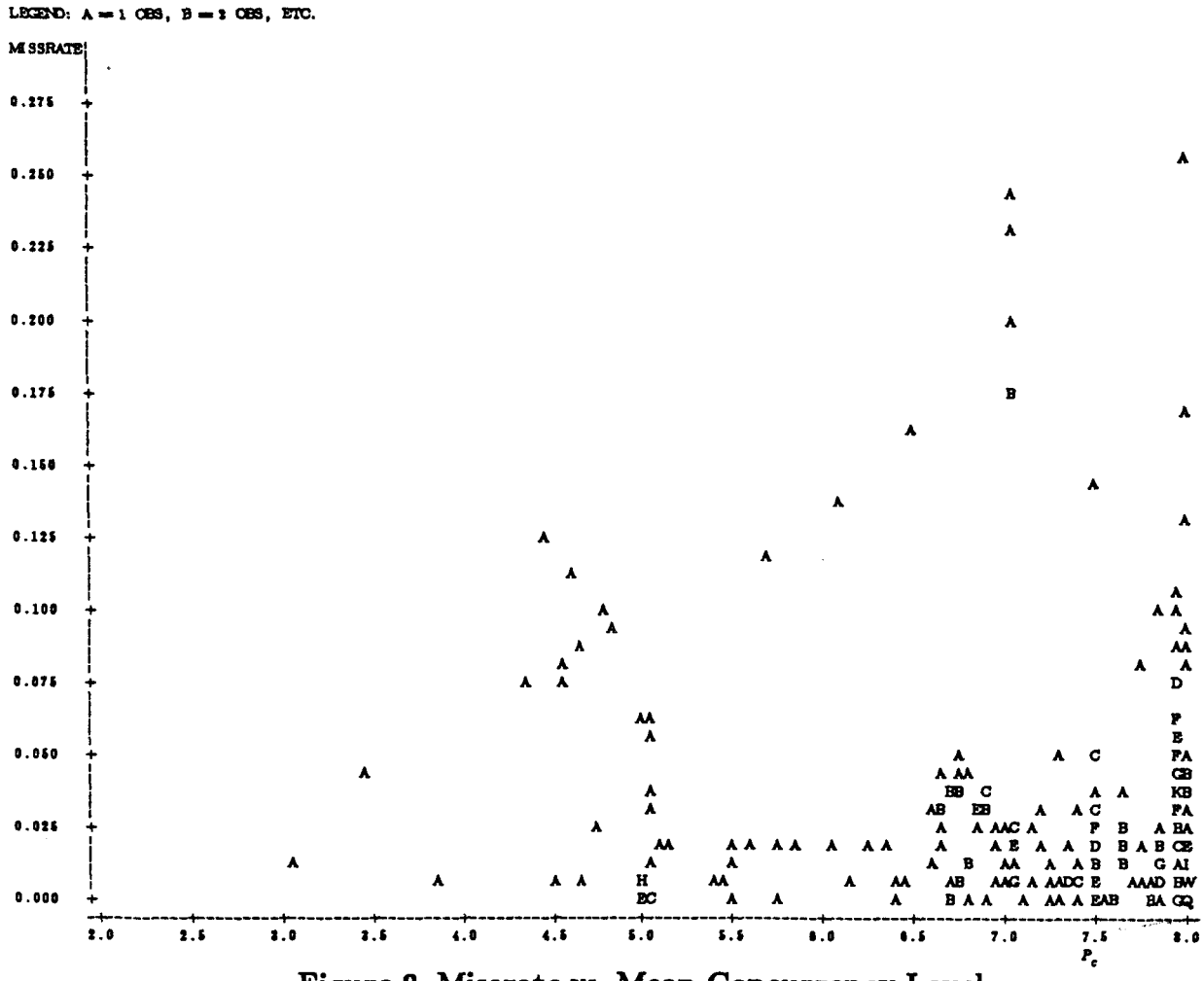

Figure 9. Missrate vs. Mean Concurrency Level.

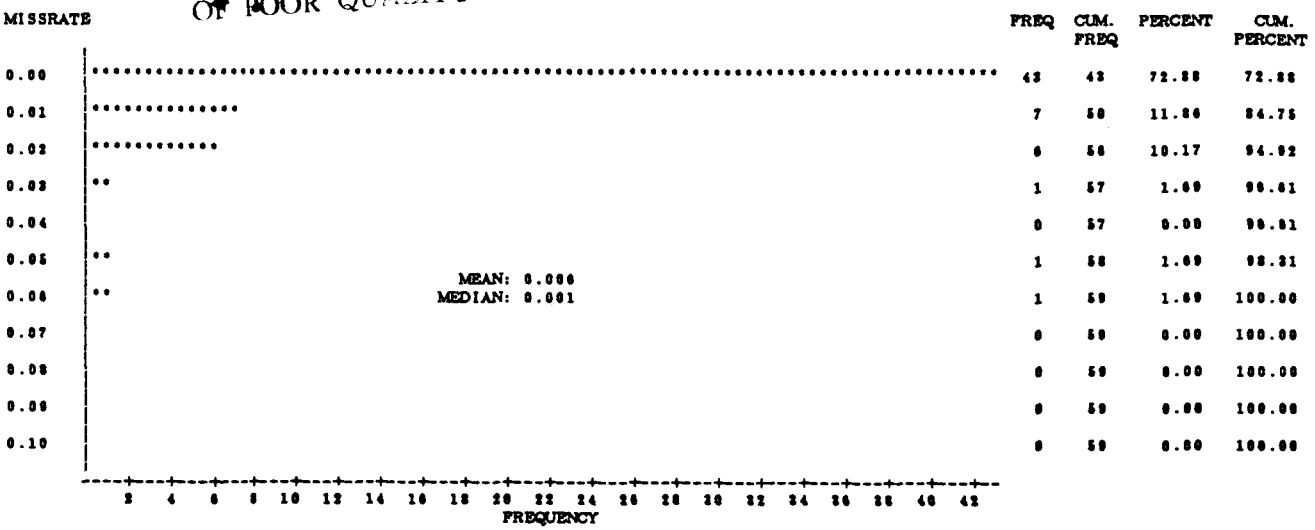

Figure 10 (a). Distribution of Miss Rate, $C_w <= 0.4$

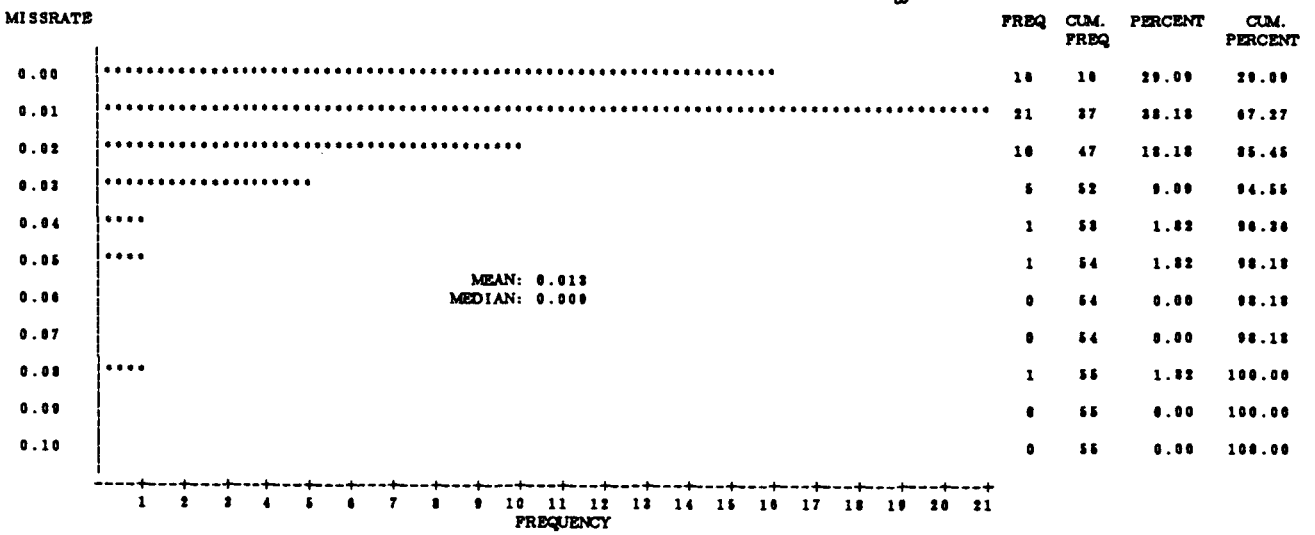

Figure 10 (b). Distribution of Miss Rate, $0.4 < C_w <= 0.8$

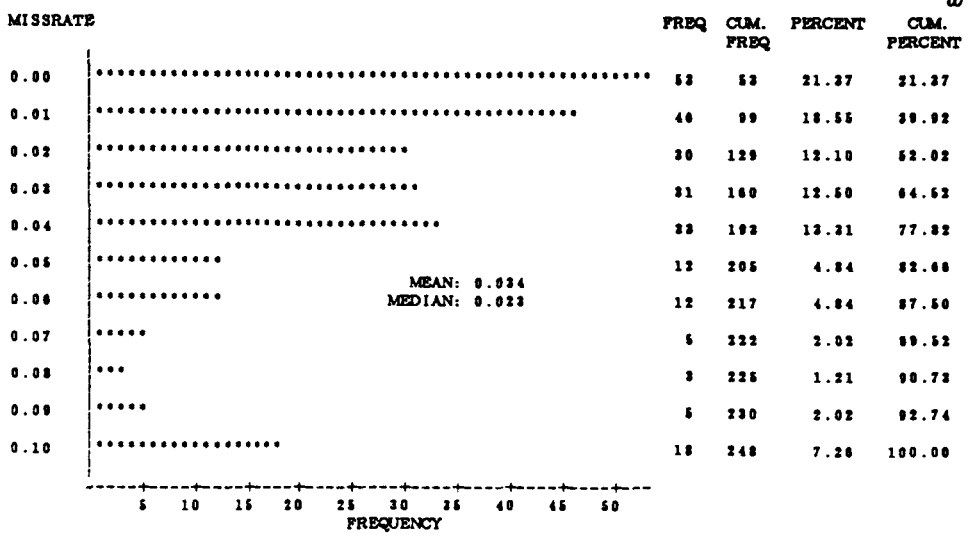

Figure 10 (c). Distribution of Miss Rate, $C_w > 0.8$

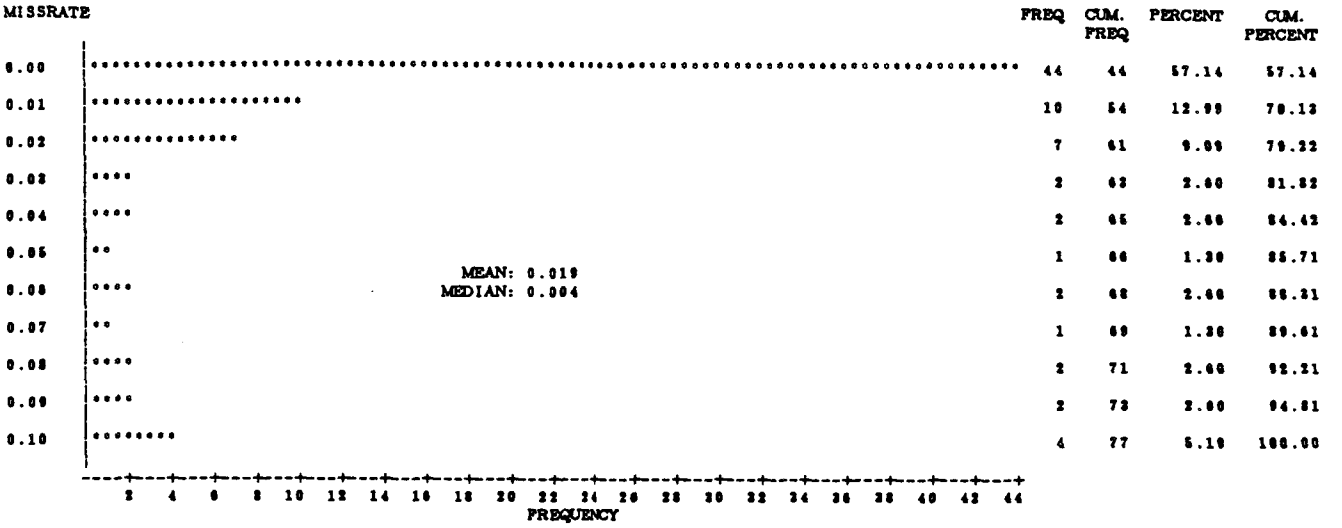

Figure 11 (a). Distribution of Miss Rate, $P_c <== 6.0$

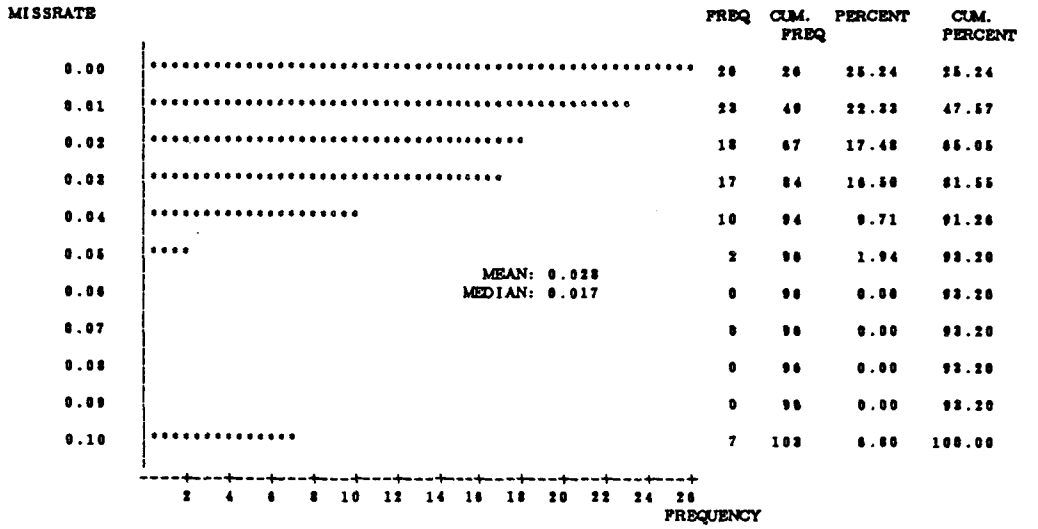

Figure 11 (b). Distribution of Miss Rate, $6.0 < P_c <== 7.5$

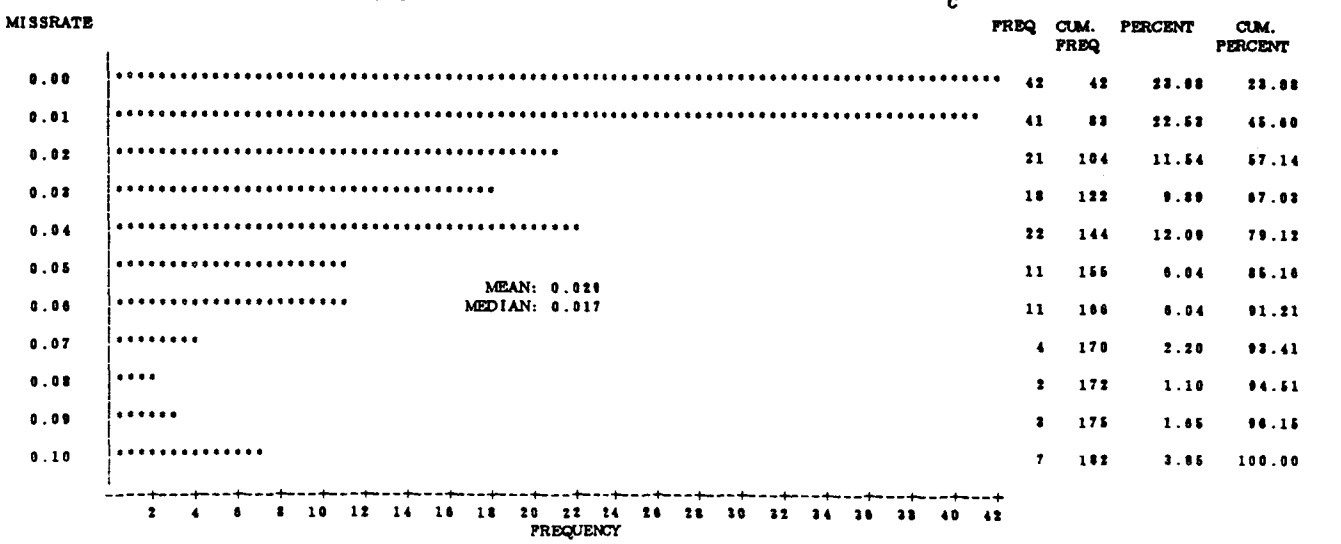

Figure 11 (c). Distribution of Miss Rate, $P_c > 7.5$

While probability of a cache miss increases with Workload Concurrency, and to a lesser extent, Mean Concurrency Level, a high value of C_w or P_c does not preclude a low Missrate value. Several observations exist at maximum C_w and low or zero Missrate. Periods of high Workload Concurrency and/or Mean Concurrency Level may generate low cache miss rates for several reasons. If the concurrent portion of the workload has well–behaved data and code locality, Missrate will obviously be low. Since each CE has an internal instruction cache (see Appendix C), loops and other program constructs which are local and "fit" in this cache will not generate successive requests to the shared cache for instruction fetch. A high degree of register–to–register operations (which may include 32–element vector operations) will reduce data traffic between CE and cache, and consequently the average number of cache misses. Data dependency within concurrent loops may also reduce cache traffic. Processors are not required to access memory while waiting for synchronization, since this mechanism uses the physically separate Concurrency Control Bus [18]. Data and instruction locality *across* processors also will lessen the overall impact on the cache of higher concurrency in the workload. Data which is fetched to the shared cache for one CE and is soon needed by one or more additional processors will not result in additional misses for these processors.

In summary, the distributions in this section show a general increase in cache miss rate with an increased amount of parallel code, and little relation between Missrate and the number of processors active within concurrent operations. In the following section, regression models are developed to quantify relationships between system and concurrency measures.

5.2 Regression Models

Regression models were developed to quantify the median behavior of system measures, with respect to concurrency measures. A median point is calculated with respect to C_w by finding the median of the system measure for the set of points clustered around their closest Workload Concurrency midpoint (0.0, 0.1, ... 1.0). The resulting set of coordinate pairs is then used to determine the model of the form *system measure verses* C_w. The same technique was used to calculate the median system measure points for Mean Concurrency Level (midpoints = 2.0, 3.0 ... 8.0), to develop models of system measure verses P_c.

Regression techniques were used to generate a fit of the median values described above and the corresponding concurrency measure midpoints. Second order linear models were determined to most accurately model the data. These models were of the form:

$$System\ Measure = \beta_1 * C_w + \beta_2 * C_w^2 + C \tag{5.1}$$

or

$$System\ Measure = \beta_1 * P_c + \beta_2 * P_c^2 + C \tag{5.2}$$

The regression model finds β_1, β_2 and C such that the equation

$$SSE = \sum [y_i - (C + \beta_1 x_i + \beta_2 x_i^2)]^2 \tag{5.3}$$

is minimized, where (x_i, y_i) is a (concurrency measure, system measure) observation. One commonly used measure of the accuracy of the model for predicting the data is given by R^2, which indicates the amount of variability in the data predicted by the model [1] [23]. The results of the regression modeling are shown in Tables 3 and 4.

[1] R^2 values are categorized in [24] as: 0 = no relationship, 0.25 = moderately weak, 0.5 = moderate, 0.75 = moderately strong, 1.0 = perfect.

Regression Models System Measure vs. C_w *Model Parameters*				
System Measure	β_1	β_2	C	R^2
Median Miss Rate	-3.30×10^{-3}	2.57×10^{-2}	2.62×10^{-3}	0.74
Median CE Bus Busy	2.18×10^{-1}	1.01×10^{-1}	2.47×10^{-2}	0.89
Median Page Fault Rate	1.46×10^{4}	-1.02×10^{4}	1.07×10^{3}	0.65

Table 3. Regression Models verses C_w

Regression Models System Measure vs. P_c *Model Parameters*				
System Measure	β_1	β_2	C	R^2
Median Miss Rate	5.05×10^{-3}	-7.43×10^{-4}	1.86×10^{-2}	0.07
Median CE Bus Busy	1.22×10^{-1}	-7.79×10^{-3}	-1.82×10^{-1}	0.66
Median Page Fault Rate	8.12×10^{3}	-5.28×10^{2}	-2.53×10^{4}	0.61

Table 4. Regression Models verses P_c

The plot of Missrate verses Workload Concurrency model is shown in Figure 12. The model predicts that an increase in C_w from 0.5 to 1.0 will be accompanied by a greater than triple increase in Missrate, from .007 to .024. While the scatter-plots showed that Missrate values vary over a wide range for Workload Concurrency, the fact that the median value is increasing shows that probability of higher values grows with C_w.

A similar analysis was performed to estimate a relationship between CE Bus Busy and the concurrency measures. The model for this measure verses C_w and P_c are plotted in Figures 13 and 14. The figures show that the activity on the CE busses is generally increasing with both Workload Concurrency and Mean Concurrency Level. Both concurrency measures establish the fraction of time that processors may be active; median bus activity then follows this fraction. As

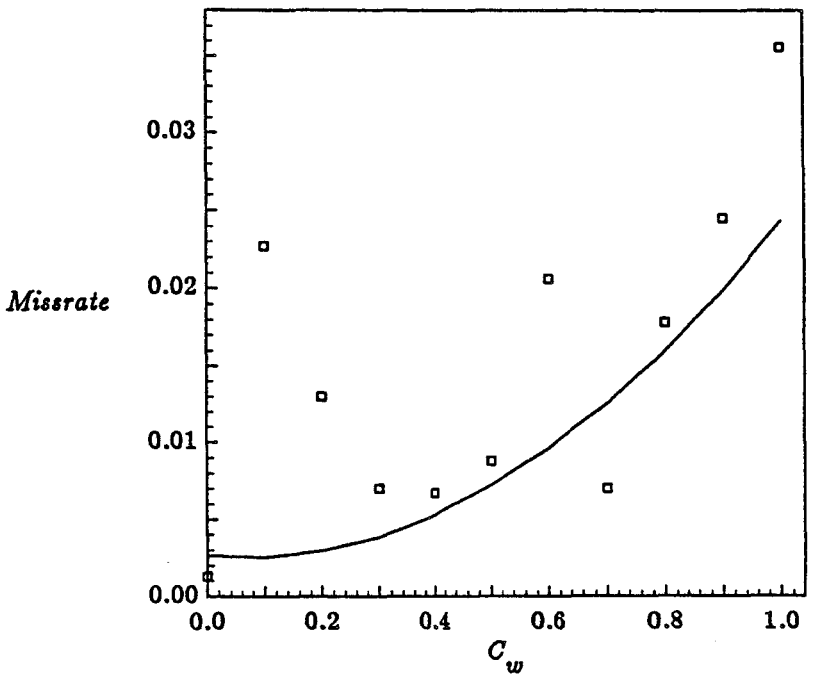

Figure 12. Plot of Regression Model, Missrate vs. C_w.

expected, the model predicts almost linear increase in bus activity with Workload Concurrency (i.e., the fraction of parallel code in the workload). With respect to Mean Concurrency Level, however, activity increases until $P_c = 6.0$, after which the Missrate levels off around 0.30. The results suggest that increased bus activity is more dependent on the percentage of parallel code in the workload (given by C_w) than the degree of concurrency within parallel operations.

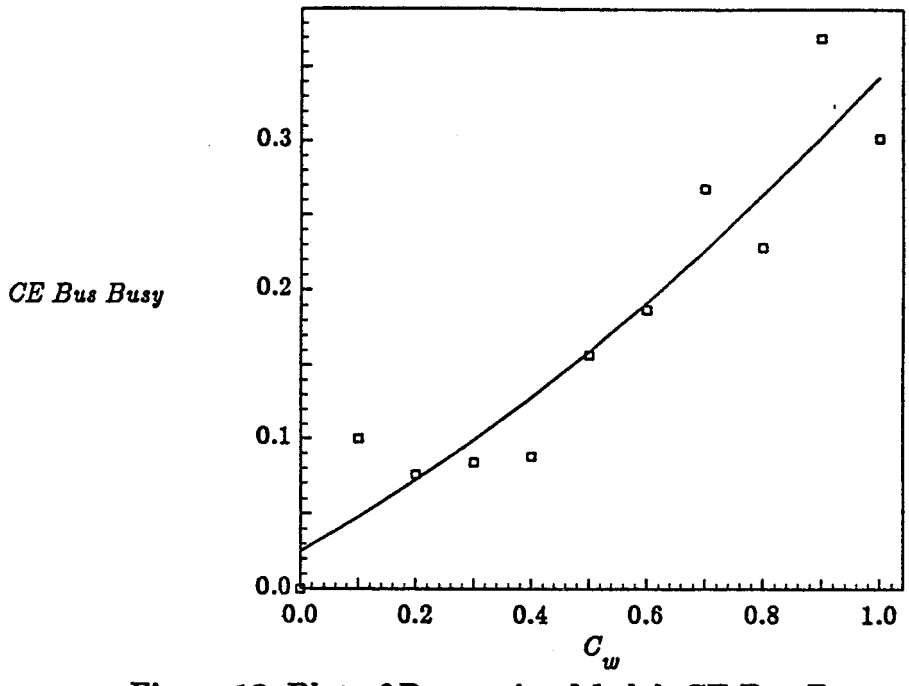

Figure 13. Plot of Regression Model, CE Bus Busy vs. C_w.

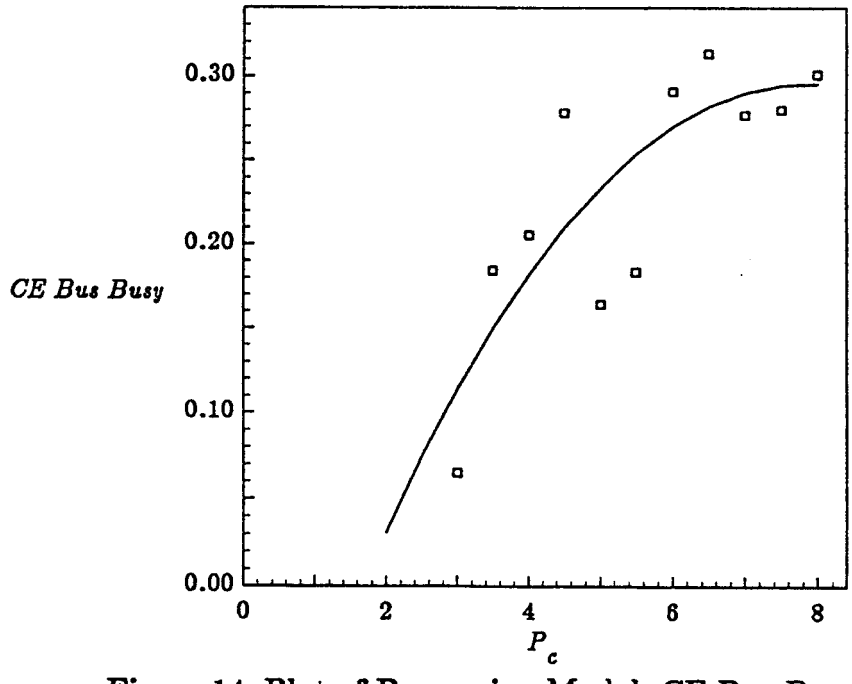

Figure 14. Plot of Regression Model, CE Bus Busy vs. P_c.

5.3 Discussion of Results

The analysis in this chapter presents a model for Missrate verses Workload Concurrency which predicts a sharp increase in Missrate, from 0.007 to 0.024, between $C_w = 0.5$ and $C_w = 1.0$. Little correlation between Missrate and P_c is seen. This means that Missrate is much more sensitive to the fraction of parallel code in the workload than the number of processors active within parallel operations. The probable cause for this result is that the kinds of functions which are suitable for parallel encoding, such as matrix and concurrent vector operations, are usually much more data intensive than general serial code. This can result in higher data traffic for parallel codes (see the regression models for CE Bus Busy in the previous section), and a greater number of cache misses.

As mentioned in section 5.1, locality of data and code across processors lessens the impact of additional processors within a parallel operation on cache misses. This is an explanation for Concurrency Level's lack of relationship with Missrate. Processors executing iterations of concurrent loops will typically follow similar instruction execution paths, ensuring good code locality. In addition, data access patterns between loop iterations will usually be related, lowering any additive effects of growth in P_c.

CE Bus activity shows a near–linear growth with increasing Workload Concurrency. With respect to Mean Concurrency Level, Bus activity increases until reaching a maximum range at $P_c = 6.0$. Increase with C_w is explained as above; the inherent difference in concurrent and serial code results in greater traffic levels as C_w grows. Relatively constant bus activity after $P_c = 6.0$ is likely a reflection of a higher degree of dependence–related waiting in periods of maximum concurrency (all processors active) in the workload; such waiting will reduce bus traffic.

The following highlights the key results arising from this work, and makes suggestions for future research.

CHAPTER 6.

CONCLUSIONS

This study used measurements on an Alliant FX/8 multiprocessor at the University of Illinois Center for Supercomputing Research and Development to evaluate concurrency found in a real workload on the machine. A systematic methodology was developed for characterizing the amount of concurrency present in the workload, and the effect of concurrency on system performance indices such as cache miss rate and bus activity.

Two measures, Workload Concurrency and Mean Concurrency Level, were defined and then measured. Random sampling of the workload showed Workload Concurrency varies from 0 (serial) to 1.0, and has an overall average of 35% for all measurement sessions. Idle, serial, and fully concurrent states dominate in the CE Cluster. Mean Concurrency Level, which measures the number of active processors during a concurrent operation, normally show a value close to maximum concurrency, or $P_c = 8$.

Analysis of transition periods between 8–concurrency and lower concurrency levels showed that processor usage was uneven during these times; for the measured data, periods of 2–concurrency dominate the transition periods. Possible reasons for this include a large percentage of concurrent loops with 2 "leftover" iterations, uneven distribution of dependency waiting times, and unbalanced sharing of resources during concurrent operation, where one or both of these processors experiences greater delays than the remaining CEs.

System measures, including cache miss rate and CE bus activity were analyzed with respect to concurrency measures to observe what relationships exist. It was shown that in general, the higher the value of C_w or P_c, the higher the *probability* of increase in the system measure. For

cache miss rate , neither concurrency measure established a lower bound on any of the system measure's value.

Second order linear regression models were developed to find the relationship between C_w, P_c, and median system measure behavior. A model for cache miss rate showed a reasonable fit verses C_w, and predicted an increase in the median value of Missrate from .007 to 0.024, while Workload Concurrency increased from 50% to 100%. Missrate showed low correlation with P_c, yielding the result that Missrate is more strongly related to the fraction of parallel code in the workload than the level of concurrency within parallel operations. CE bus activity was also modeled verses both concurrency measures, and was seen to increase with both C_w and P_c, although less strongly in the highest range of Concurrency Level.

The methodology and results presented here are useful for multiprocessor evaluation and optimization. In particular, understanding of machine characteristics in the presence of a real workload is important, since the complexity of parallel systems makes prediction of performance difficult. The techniques used here can be applied to other parallel processing systems, and be extended to other levels of concurrency and new performance indices.

Similar studies are suggested, in order to obtain a wide range of representative practical results in concurrency evaluation. Future research in the measurement of concurrency should include evaluation of individual programs, to determine their behavior within the workload environment. Also, the relationship of concurrency and software–level parameters (such as those related to job scheduling) deserves attention.

APPENDIX A.

WORKLOAD SAMPLING DATA

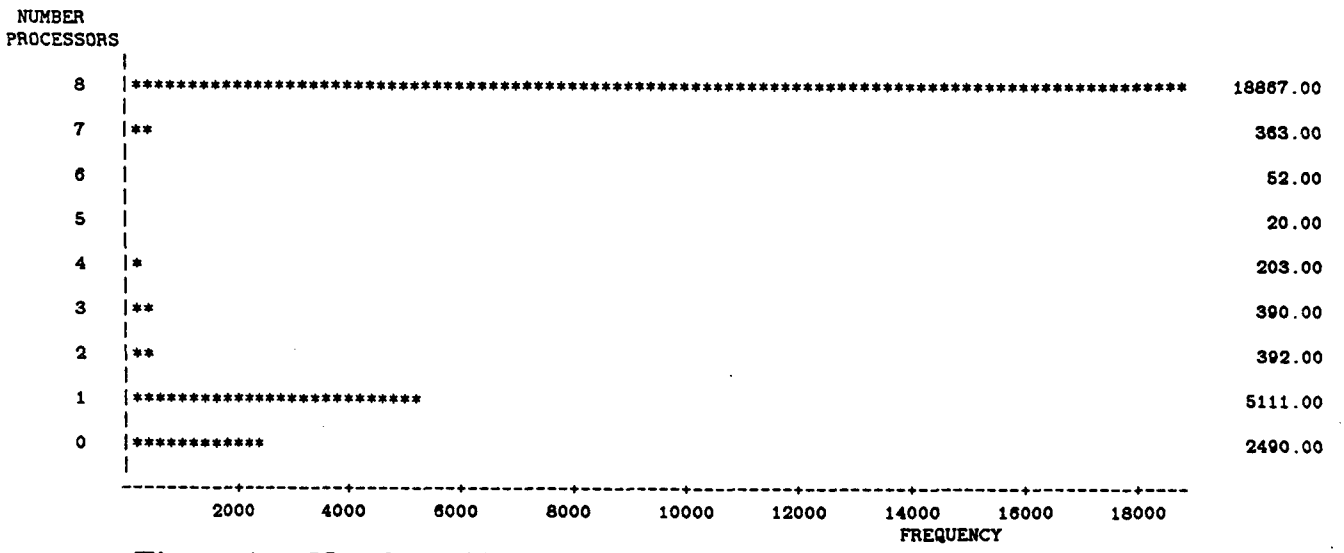

Figure A.1. Number of Records with N Processors Active / Session 1.

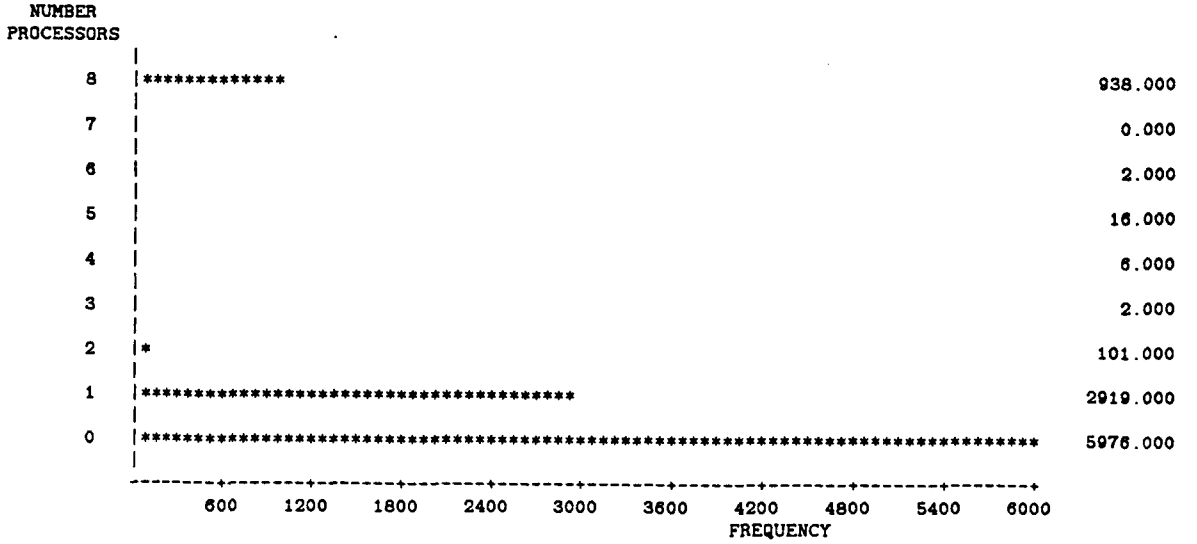

Figure A.2. Number of Records with N Processors Active / Session 9.

Mean of Concurrency Measures – Random Samples							
c_2	c_3	c_4	c_5	c_6	c_7	c_8	C_w
0.0158	0.0082	0.0084	0.0068	0.0062	0.1326	0.3045	0.4828
$c_{2\|c}$	$c_{3\|c}$	$c_{4\|c}$	$c_{5\|c}$	$c_{6\|c}$	$c_{7\|c}$	$c_{8\|c}$	P_c
0.0347	0.0138	0.0135	0.0143	0.0109	0.2598	0.6529	7.3438
Missrate			CE Bus Busy			Page Fault Rate	
0.0084			0.1361			7539	

Table A.1. Mean Concurrency Measures for Random Samples.

MIDPOINT CE BUS BUSY		FREQ	CUM. FREQ	PERCENT	CUM. PERCENT
0.00	**	25	25	21.74	21.74
0.05	**	22	47	19.13	40.87
0.10	*************************************	19	66	16.52	57.39
0.15	*****************************	14	80	12.17	69.57
0.20	**************	7	87	6.09	75.65
0.25	**********************	11	98	9.57	85.22
0.30	*****************	8	106	6.96	92.17
0.35	******	3	109	2.61	94.78
0.40		0	109	0.00	94.78
0.45	**	1	110	0.87	95.65
0.50	**********	5	115	4.35	100.00

2 4 6 8 10 12 14 16 18 20 22 24
FREQUENCY

Figure A.3. Distribution of Samples by CE Bus Busy.

MIDPOINT MISSRATE		FREQ	CUM. FREQ	PERCENT	CUM. PERCENT
0.00	**********	41	41	63.08	63.08
0.01	**********	13	54	20.00	83.08
0.02	**********	6	60	9.23	92.31
0.03	****	2	62	3.08	95.38
0.04	**	1	63	1.54	96.92
0.05	****	2	65	3.08	100.00
0.06		0	65	0.00	100.00
0.07		0	65	0.00	100.00
0.08		0	65	0.00	100.00
0.09		0	65	0.00	100.00
0.10		0	65	0.00	100.00

2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
FREQUENCY

Figure A.4. Distribution of Samples by Miss Rate.

MIDPOINT PAGE FAULT RATE		FREQ	CUM. FREQ	PERCENT	CUM. PERCENT
0	******	3	3	2.54	2.54
1000	**********	17	20	14.41	16.95
2000	**********	24	44	20.34	37.29
3000	**********	8	52	6.78	44.07
4000	**********	15	67	12.71	56.78
5000	**********	9	76	7.63	64.41
6000	********	4	80	3.39	67.80
7000	**********	6	86	5.08	72.88
8000	**********	7	93	5.93	78.81
9000	****	2	95	1.69	80.51
10000	****	2	97	1.69	82.20
11000	******	3	100	2.54	84.75
12000		0	100	0.00	84.75
13000		0	100	0.00	84.75
14000		0	100	0.00	84.75
15000		0	100	0.00	84.75
16000		0	100	0.00	84.75
17000	**	1	101	0.85	85.59
18000		0	101	0.00	85.59
19000		0	101	0.00	85.59
20000	******	3	104	2.54	88.14
21000	**	1	105	0.85	88.98
22000	****	2	107	1.69	90.68
23000	******	3	110	2.54	93.22
24000	**********	8	118	6.78	100.00

2 4 6 8 10 12 14 16 18 20 22 24
FREQUENCY

Figure A.5. Distribution of Samples by Page Fault Rate.

APPENDIX B.

CONCURRENCY VS. SYSTEM MEASURE DATA

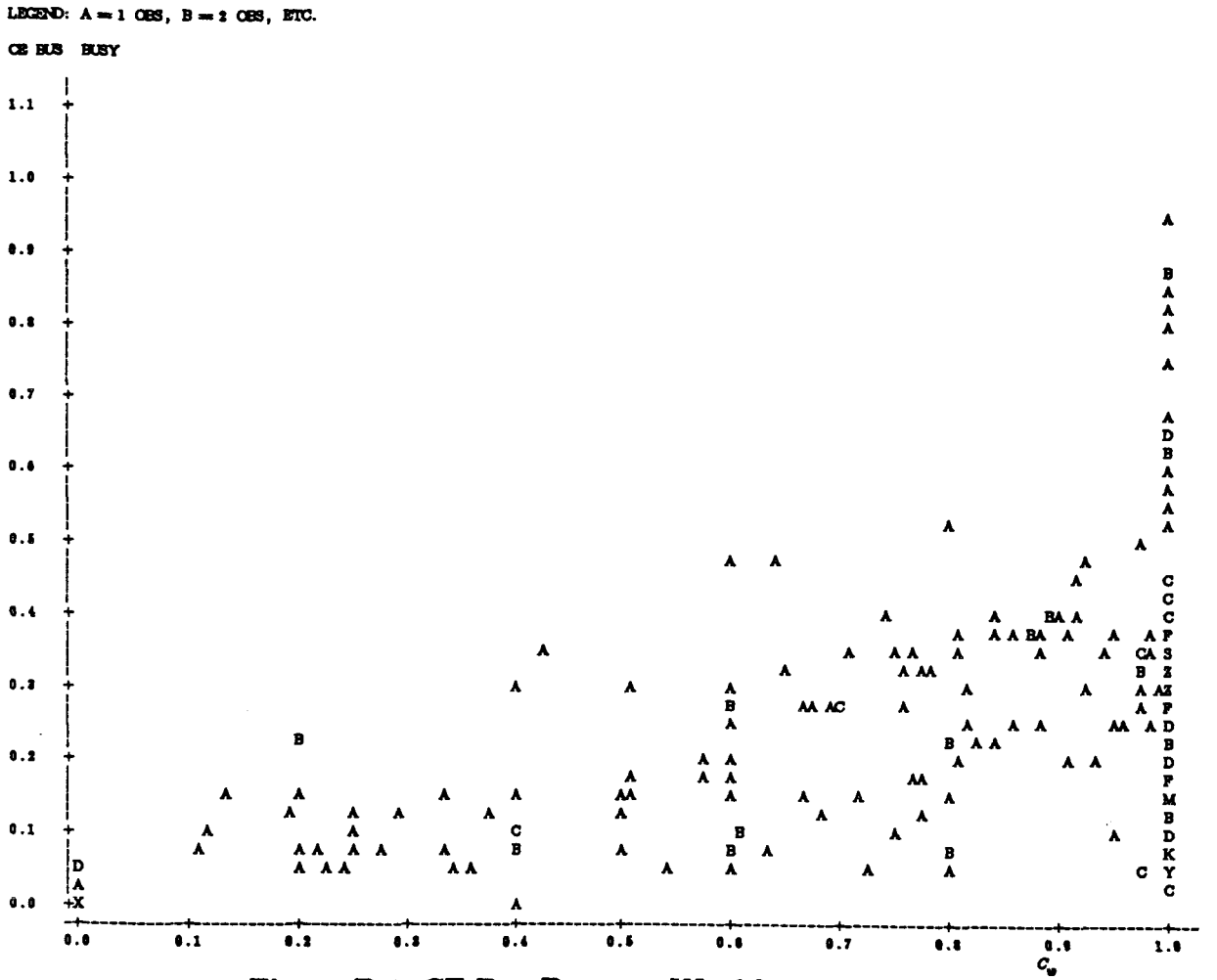

Figure B.1. CE Bus Busy vs. Workload Concurrency.

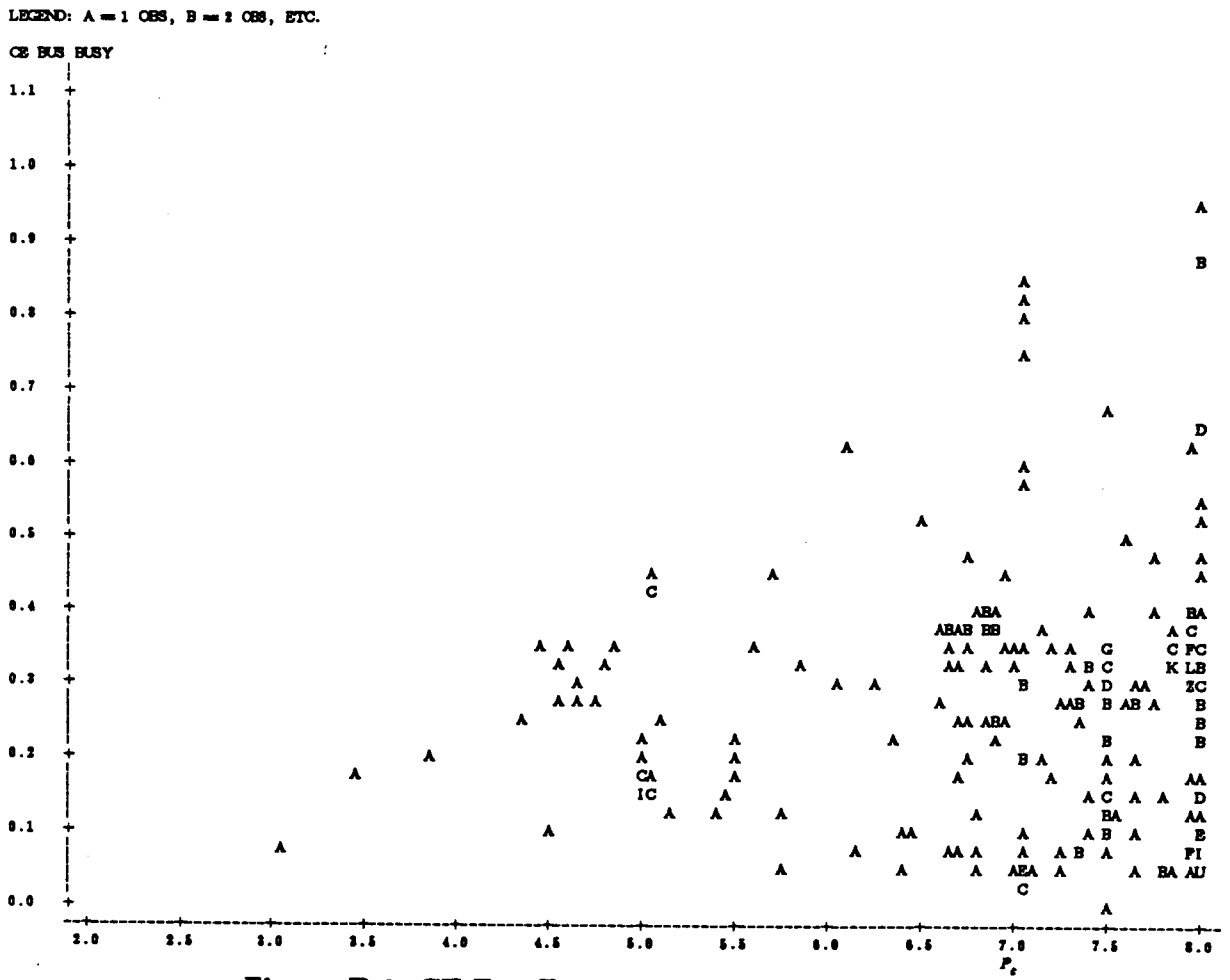

Figure B.2. CE Bus Busy vs. Mean Concurrency Level.

CE BUS BUSY		FREQ	CUM. FREQ	PERCENT	CUM. PERCENT
0.0	**********	31	31	52.54	52.54
0.1	**********	24	55	40.68	93.22
0.2	**	3	58	5.08	98.31
0.3	*	1	59	1.69	100.00
0.4		0	59	0.00	100.00
0.5		0	59	0.00	100.00
0.6	MEAN: 0.056 MEDIAN: 0.0045	0	59	0.00	100.00
0.7		0	59	0.00	100.00
0.8		0	59	0.00	100.00
0.9		0	59	0.00	100.00
1.0		0	59	0.00	100.00

2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
FREQUENCY

Figure B.3 (a). Distribution of CE Bus Busy, $C_w <= 0.4$

CE BUS BUSY		FREQ	CUM. FREQ	PERCENT	CUM. PERCENT
0.0	*****	1	1	1.82	1.82
0.1	**********	18	19	32.73	34.55
0.2	**********	13	32	23.64	58.18
0.3	**********	17	49	30.91	89.09
0.4	**********	3	52	5.45	94.55
0.5	**********	3	55	5.45	100.00
0.6	MEAN: 0.216 MEDIAN: 0.195	0	55	0.00	100.00
0.7		0	55	0.00	100.00
0.8		0	55	0.00	100.00
0.9		0	55	0.00	100.00
1.0		0	55	0.00	100.00

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
FREQUENCY

Figure B.3 (b). Distribution of CE Bus Busy, $0.4 < C_w <= 0.8$

CE BUS BUSY		FREQ	CUM. FREQ	PERCENT	CUM. PERCENT
0.0	***	6	6	2.42	2.42
0.1	**********	44	50	17.74	20.16
0.2	**********	33	83	13.31	33.47
0.3	**********	101	184	40.73	74.19
0.4	**********	43	227	17.34	91.53
0.5	***	5	232	2.02	93.55
0.6	**** MEAN: 0.286 MEDIAN: 0.305	7	239	2.82	96.37
0.7	*	2	241	0.81	97.18
0.8	**	4	245	1.61	98.79
0.9	**	3	248	1.21	100.00
1.0		0	248	0.00	100.00

10 20 30 40 50 60 70 80 90 100
FREQUENCY

Figure B.3 (c). Distribution of CE Bus Busy, $C_w > 0.8$

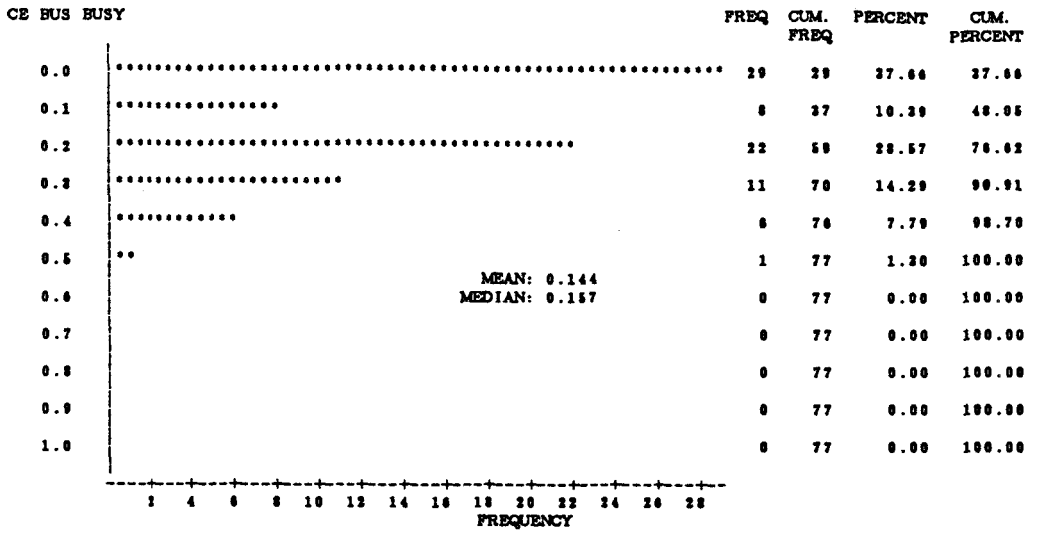

Figure B.4 (a). Distribution of CE Bus Busy, $P_c <= 6.0$

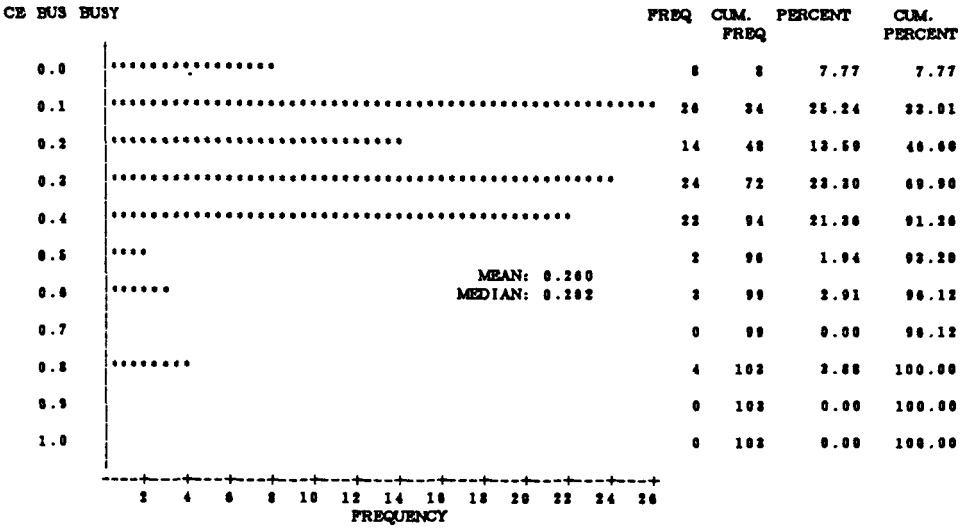

Figure B.4 (b). Distribution of CE Bus Busy, $6.0 < P_c <= 7.5$

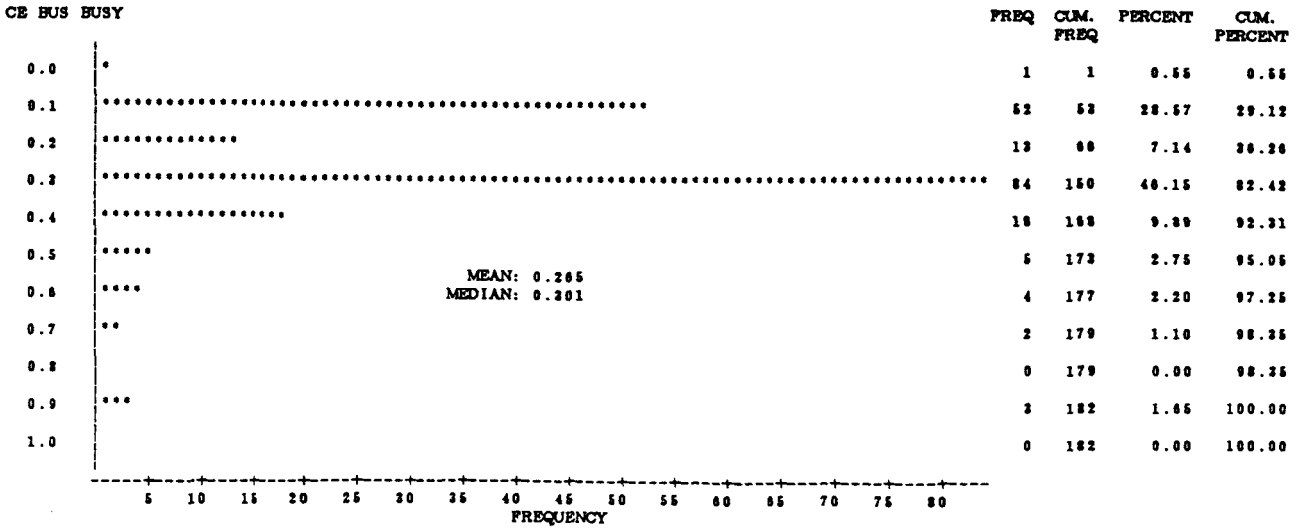

Figure B.4 (c). Distribution of CE Bus Busy, $P_c > 7.5$

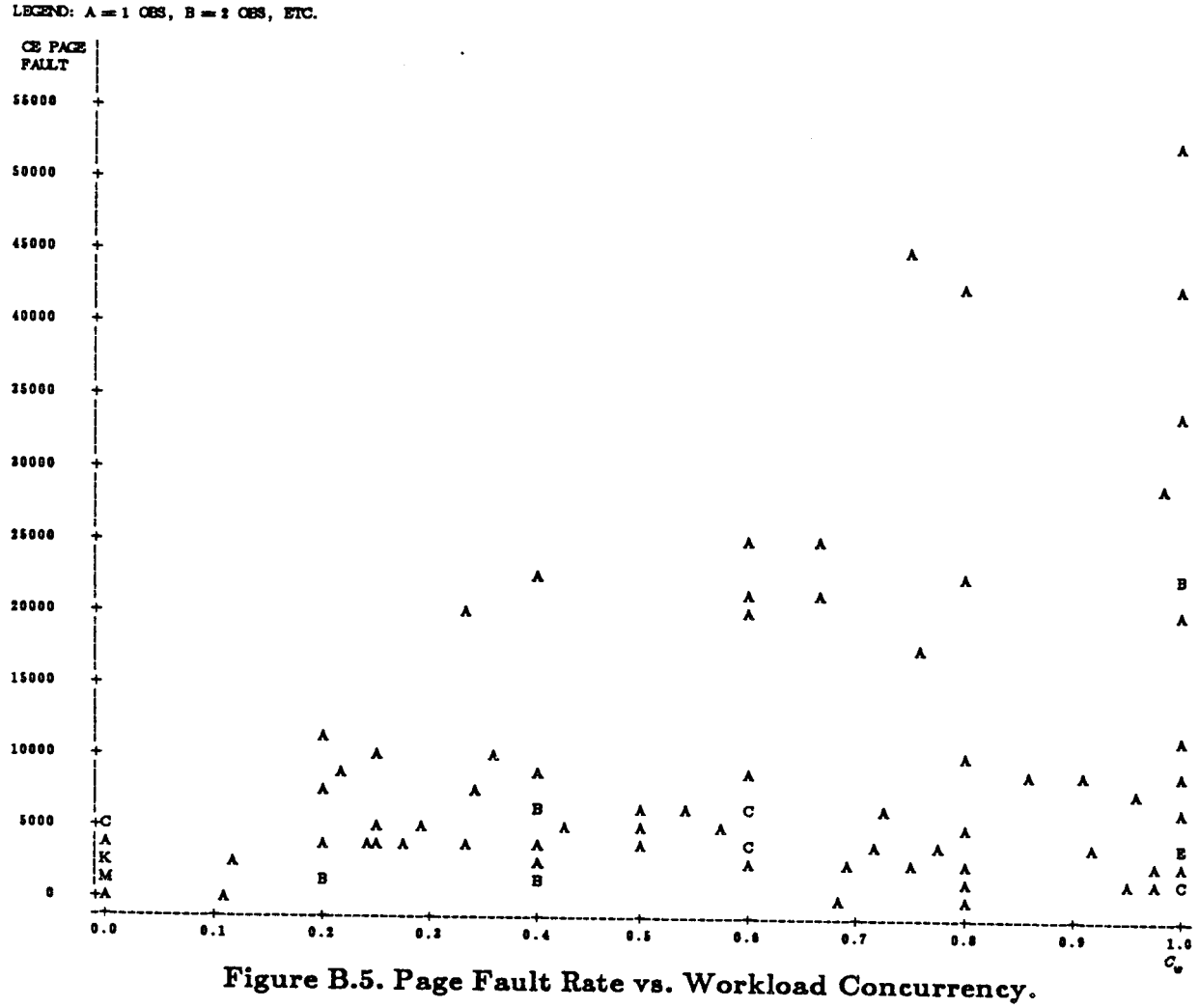

Figure B.5. Page Fault Rate vs. Workload Concurrency.

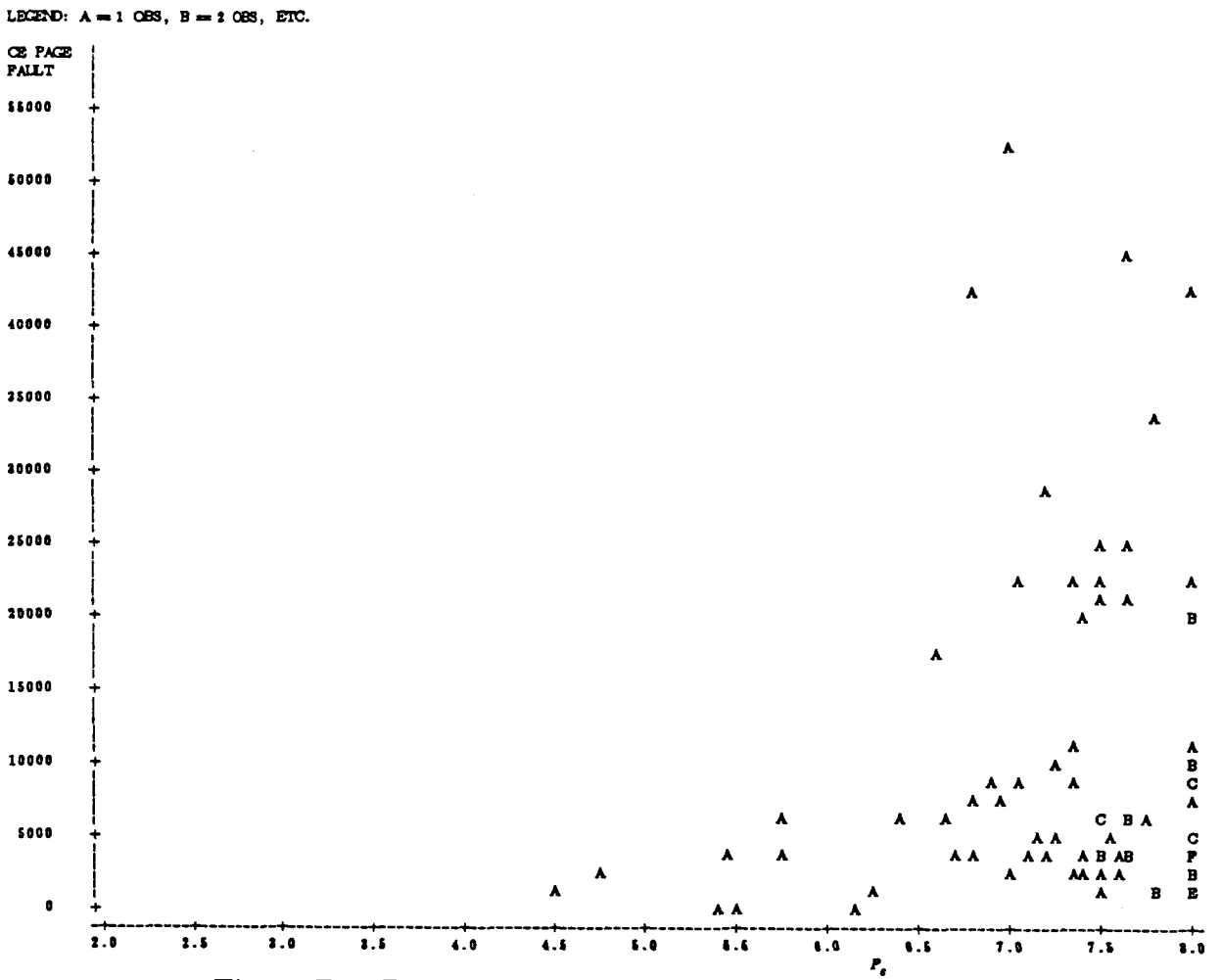

Figure B.6. Page Fault Rate vs. Mean Concurrency Level.

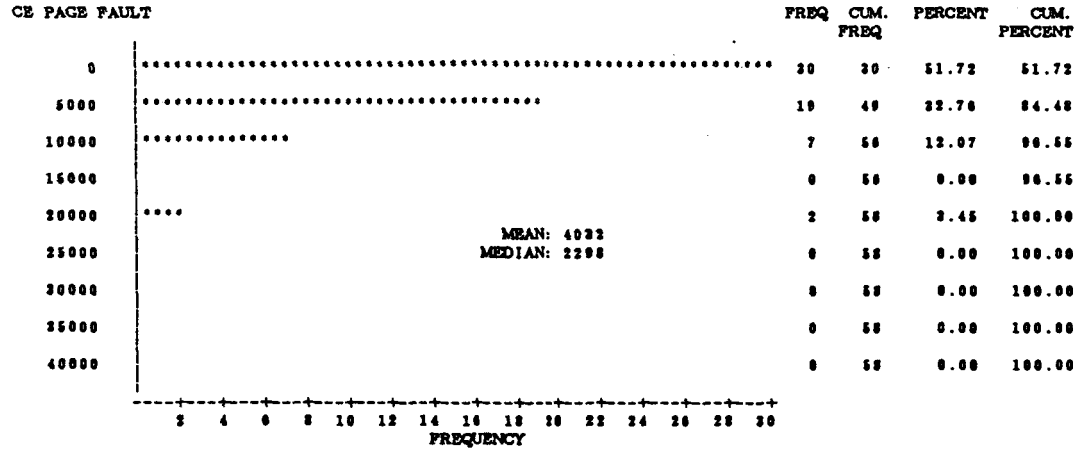

Figure B.7 (a). Distribution of Page Fault Rate, $C_w <= 0.4$

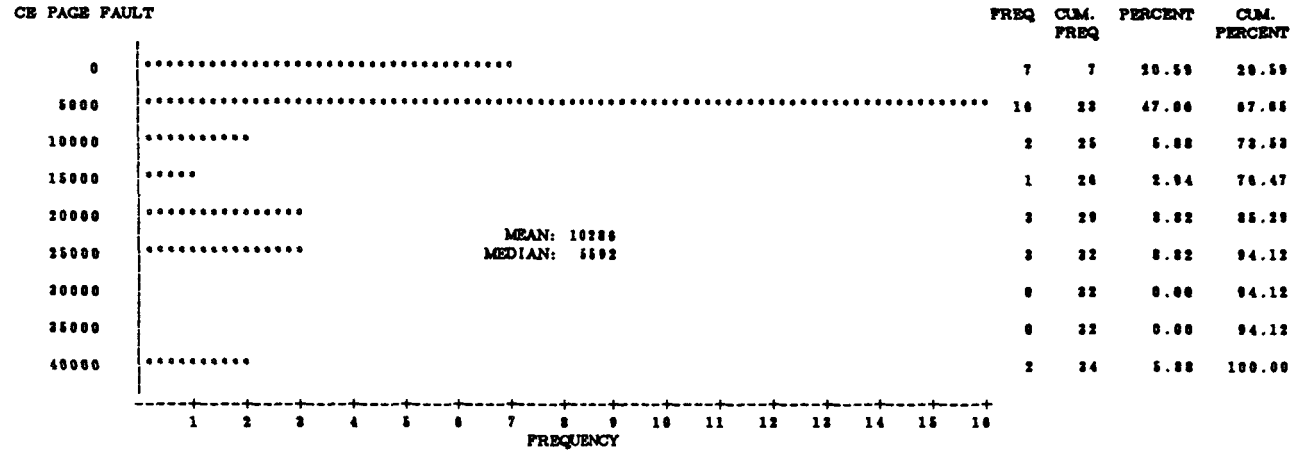

Figure B.7 (b). Distribution of Page Fault Rate, $0.4 < C_w <= 0.8$

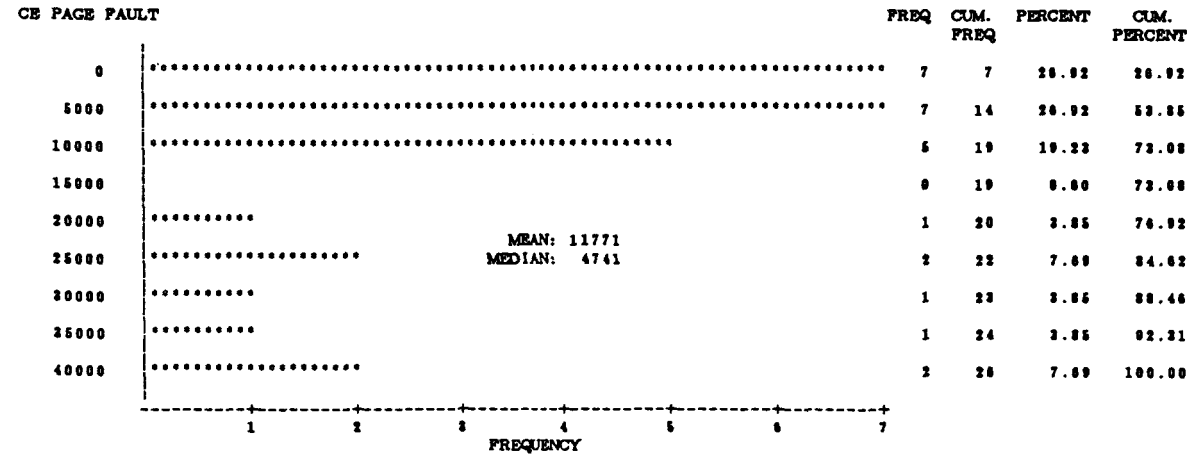

Figure B.7 (c). Distribution of Page Fault Rate, $C_w > 0.8$

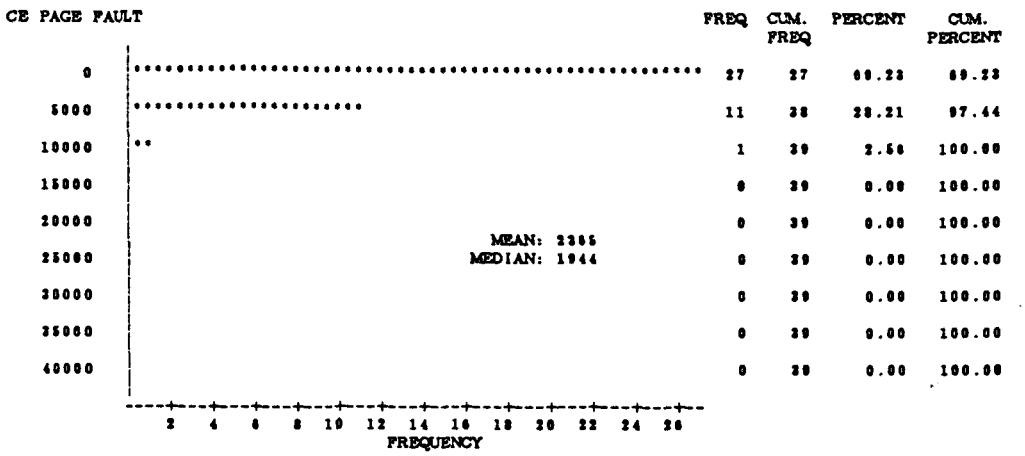

Figure B.8 (a). Distribution of Page Fault Rate, $P_c <= 6.0$

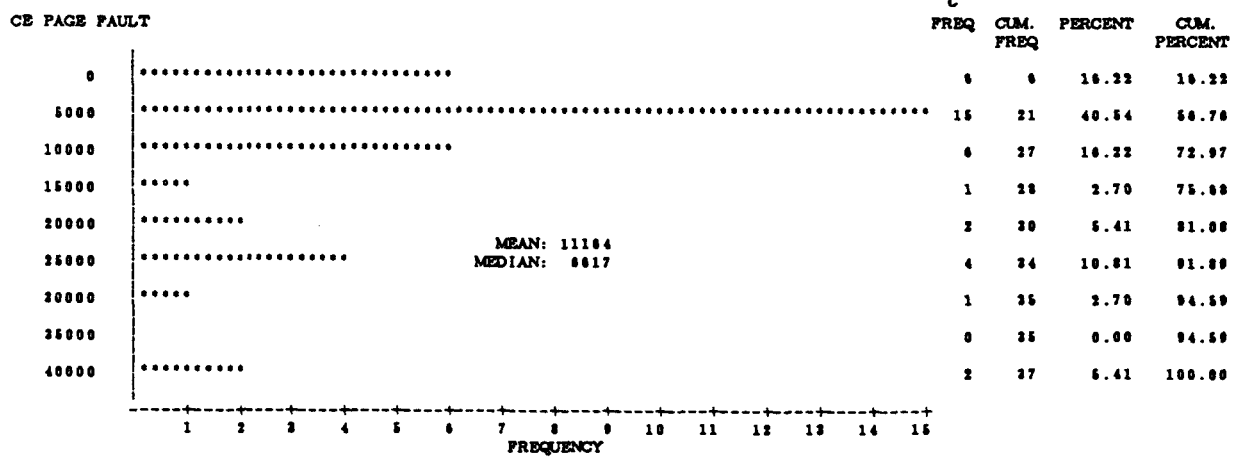

Figure B.8 (b). Distribution of Page Fault Rate, $6.0 < P_c <= 7.5$

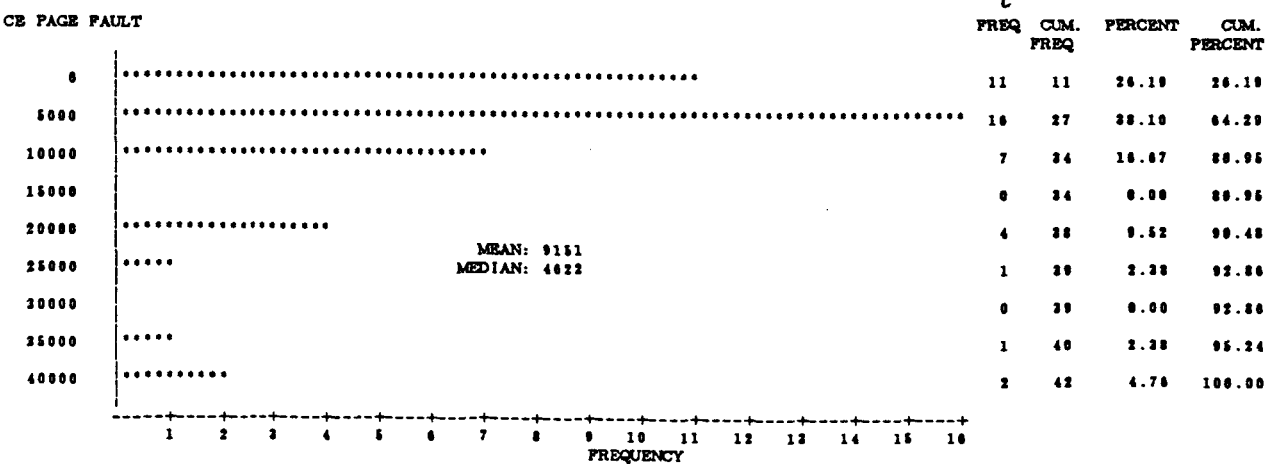

Figure B.8 (c). Distribution of Page Fault Rate, $P_c > 7.5$

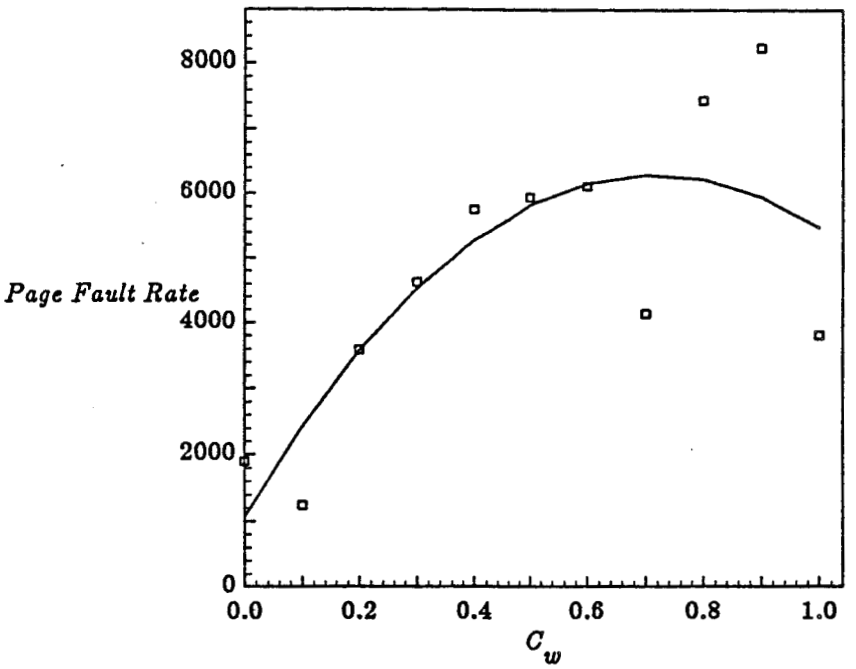

Figure B.9. Plot of Regression Model, Page Fault Rate vs. C_w.

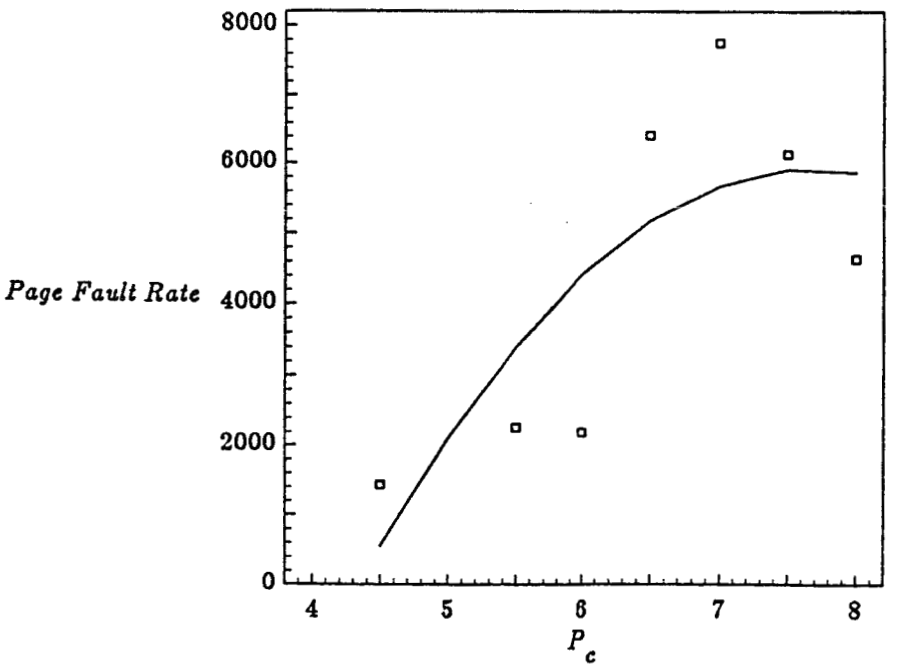

Figure B.10. Plot of Regression Model, Page Fault Rate vs. P_c.

APPENDIX C.

ALLIANT FX/8 DESCRIPTION

The Alliant FX/8 supports two types of processors, an Interactive Processor (IP), and a Computational Element (CE). The machine may be configured with as little as 1 IP and 1 CE (*FX/1*), or up to 12 IPs and 8 CEs (*F/X 8*). The IP is based on a Motorola 68012 microprocessor. 512 Kbytes of local memory is available to the IP; it is also directly connected to a 32 Kbyte IP cache (IPC), which is in turn connected to the system memory bus. IPs handle all system I/O through a Multibus™ system. The IPs' function is support of interactive load, support of the operating system, and control of I/O.

The CE has a base instruction set similar to the Motorola 68020 microprocessor. Additionally, the CE supports vector processing, floating point operation, and concurrent execution in the Computational Cluster, where loop–level multiprocessing takes place. Vector operations may occur simultaneously with multiprocessing on the Cluster. Each CE contains a 16 Kbyte instruction cache for efficient handling of loops and other localized portions of code. The CEs share a four–way interleaved cache memory, with a total size of 128 Kbytes, divided into two Computational Element Caches (CPCs). Connection to these cache modules is accomplished through a crossbar switch which routes both address and data between cache and CE.

All data traffic between processors (CE or IP) and shared memory takes place through the processors' respective caches. The caches maintain data coherency by requiring that a cache possess a "unique" copy of data before modifying it. Traffic between caches and main memory is over two 64–bit wide data busses, with a total maximum bandwidth of 188 Mbytes per second. The main memory has an interleaving factor of four, and has a maximum size of 64 Mbytes. The

system's virtual address spaces are organized as 1024 segments of 1024 pages per segment; pages are 4 Kbytes in length [18] [25].

The operating system on the Alliant FX/8 is called Concentrix, and is an implementation of 4.2 BSD UNIX, with extensions including support of multiple processors. Languages supported include C, F/X FORTRAN, and assembler. FORTRAN is the only high-level language which generates code using the Cluster concurrency feature; this function can also be accessed by the assembly language programmer. Programs may be specified to run on either the CE or the IP, (the latter only if floating point or vector processing is not required), or on the Cluster with a particular number of processors [21].

REFERENCES

[1] P. Heidelberger and K. Trivedi, *Queueing Network Models for Parallel Processing with Asynchronous Tasks.* **IEEE Trans. on Comp.**, v. C–31, 1982, pp. 1099–1108.

[2] ————————, *Analytic Queueing Models for Programs With Internal Concurrency.* **IEEE Trans. on Comp.**, v. C–32, January, 1983, pp. 73–82.

[3] U. Herzog, W. Hoffman and W. Kleinöder, *Performance Modeling and Evaluation for Hierarchically Organized Multiprocessor Computer Systems.* **Proc. of the 1979 Int. Conference on Parallel Processing,** pp. 103–114.

[4] D. Kuck et. al., *The Effects of Program Restructuring, Algorithm Change, and Architecture Choice on Program Performance.* **Proc. of the 1984 Int. Conf. on Parallel Processing**, pp. 129–135.

[5] K. Gallivan, W. Jalby, and U.Meier, *The use of BLAS3 in Linear Algebra on a Parallel Processor with a Hierarchical Memory.* **CSRD Report No. 610, University of Illinois at Urbana–Champaign**, October 14, 1986.

[6] M. Berry and R. Plemmons, *Algorithms and Experiments for Structural Mechanics on High Performance Architectures.* **CSRD Report No. 602, University of Illinois at Urbana–Champaign,** September 1986.

[7] K. Hwang and F. Briggs, *Computer Architecture and Parallel Processing.* McGraw–Hill, New York 1984.

[8] C. Polychronopoulos, D. Kuck, and D. Padua, *Execution of Parallel Loops on Parallel Processor Systems.* **Proc. of the 1986 Int. Conf. on Parallel Processing**, pp. 519–527.

[9] R. Cytron, *Compile Time Scheduling and Optimization for Asynchronous Machines.* **Ph.D. Thesis, University of Illinois at Urbana–Champaign, UIUC Report No. UIUCDCS–R–84–1177,** October, 1984.

[10] S. Midkiff and D. Padua, *Compiler Generated Synchronizations for DO Loops.* **Proc. of the 1986 Int. Conf. on Parallel Processing**, pp. 544–551.

[11] D. Kuck, *The Structure of Computers and Computations.* John Wiley and Sons, New York, 1978.

[12] W. Abu-Sufah and A. Malony, *Vector Processing on the Alliant FX/8 Multiprocessor.* **Proc. of the 1986 Int. Conf. on Parallel Processing**, pp. 559–566.

[13] A. Jones and P. Schwarz, *Experience Using Multiprocessor Systems – A Status Report.*, **Computing Surveys**, v. 12, no. 2, June 1980, pp. 121–165.

[14] R. Vaughan and M. Anastas, *Limiting Multiprocessor Performance Analysis.* **Proc. of the 1979 Int. Conf. on Parallel Processing**, pp. 55–64.

[15] W. Jalby and Ulrike Meier, *Optimizing Matrix Operations on a Parallel Multiprocessor with a Hierarchical Memory System.*, **Proc. of the 1986 Int. Conf. on Parallel Processing**, pp. 429–432.

[16] U. Hercksen, R. Klar, W. Kleinöder, and F. Kneissl, *Measuring Simultaneous Events in a Multiprocessor System.* **Proc. of the 1982 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems**, pp. 77–82.

[17] H. Fromm et al., *Experiments with Performance Measurement and Modeling of a Processor Array.* **IEEE Trans. on Comp.**, v. C–32, no. 1, January, 1983, pp. 15–31.

[18] Alliant Computer Systems Corp., *FX/Series Product Summary.* June, 1985.

[19] P. Tang, P. Yew and C. Zhu, *Processor Self-Scheduling in Large Multiprocessor Systems.* **CSRD Report No. 536, University of Illinois at Urbana–Champaign**, October, 1985.

[20] Tektronix, *DAS 9100 Series Operator's Manual.* February 1985.

[21] Alliant Computer Systems Corp., *Concentrix Commands and Applications Manual.* May 1985.

[22] SAS Institute, *SAS User's Guide: Basics 1982 Edition.*

[23] W. Mendenhall and Terry Sincich, *Statistics for the Engineering and Computer Sciences.* Dellen Publishing, San Francisco, 1984.

[24] M. Younger, *A Handbook for Linear Regression.* Wadsorth, Inc., 1979.

[25] Alliant Computer Systems Corp., *FX/Series Architecture Manual.* May 1985.