Concise Papers

Measurement and Analysis of Workload Effects on Fault Latency in Real-Time Systems

MICHAEL H. WOODBURY AND KANG G. SHIN

Abstract—**The effectiveness of all known recovery mechanisms is greatly reduced in the presence of multiple latent faults. The presence of multiple latent faults increases the possibility of multiple errors, which could result in *coverage failure*. In this paper, we present experimental evidence indicating workload effects on the duration of fault latency. A synthetic workload generator is used to vary the workload, and a hardware fault injector is applied to inject transient faults of varying durations. This method allows us to derive the distribution of fault latency duration. Experimental results were obtained from the fault-tolerant multiprocessor (FTMP) at the NASA Airlab.**

Index Terms—**Fault latency, fault latency experiments, isotonic regression, real-time systems.**

I. Introduction

A *fault* is the physical change of a hardware component from its intended state in a computing system, whereas an *error* is the erroneous data resulting from the manifestation of a fault. When a fault occurs, the system is not immediately aware of its presence and remains error-free. A fault will not generate an error until the faulty component is exercised in a specific manner. Thus, an error occurs when the result produced by a faulty component is different from the expected response.

The time between fault occurrence and error generation is termed *fault latency*[1] [2], as shown in Fig. 1. Fault latency depends on the type of fault, its location, and the usage of the faulty component. Depending on the system's error detection mechanism, there is also an interval between error generation and error detection called *error latency*. An efficient detection mechanism with increased coverage reduces error latency. This paper addresses the effects of fault latency only.

In critical real-time computing systems where reliable results must be generated in a timely manner, the detection of errors, isolation of faults, and subsequent system reconfiguration must be performed quickly and correctly. Recovery from a single error is usually assumed before a second error occurs. However, the effectiveness of all known recovery mechanisms is greatly reduced in the presence of multiple faults. If a second error occurs during the recovery period, *coverage failure* could result [3]. Coverage failure is a severe problem for highly reliable systems, especially real-time control computers.

Manuscript received March 21, 1989; revised September 27, 1989. Recommended by R. K. Iyer. This work was supported in part by NASA under Grants NAG 1-296, NAG 1-492, and NGT 23-005-801.

M. H. Woodbury was with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109. He is now with Bell Communications Research (Bellcore), Piscataway, NJ 08854.

K. G. Shin is with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109.

IEEE Log Number 8932233.

[1]Other researchers have referred to fault latency as *fault dormancy* [1].

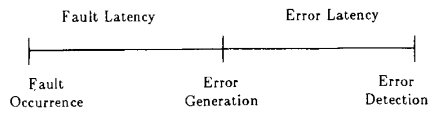

Fig. 1. Fault and error latency.

Exercising a unit with multiple latent faults could result in the near-simultaneous occurrence of errors. The presence of latent faults thus increases the possibility of multiple errors.[2] Therefore, a shorter fault latency is desirable to decrease the possibility of multiple errors. With a shorter fault latency, an error is generated quickly allowing fault recovery mechanisms more time to complete before a second error is generated.

Workload activity also has an impact on system reliability. Several researchers have reported noticeable correlations between system activity and both hardware and software failures [4]–[6]. However, these works address general-purpose time-sharing systems. Other researchers have also investigated workload effects on fault and error latency for general-purpose time-sharing systems. McGough and Swern [7] addressed fault latency through gate-level emulations, where selected programs were used as input to the emulator. A more realistic study was performed by Chillarege and Iyer [8] on a VAX-11/780 executing daily activities at a university installation.

This paper extends the analysis of workload effects on fault latency to a specific class of computers, namely real-time control systems. Exclusive analysis of real-time systems is beneficial, because of the strict reliability constraints of real-time control systems. By altering the structure of real-time workloads, fault latency can be decreased, thereby increasing overall system reliability [9].

In this paper, we present quantitative measurements of workload effects on fault latency. It is demonstrated that fault latency is affected by workload structure, and workload variations can reduce the duration of fault latency. Experiments were conducted on the fault-tolerant multiprocessor (FTMP) located at the NASA Airlab [10], [11]. FTMP is a prototype system developed to study critical real-time control applications, i.e., civilian aircraft. It provides a unique environment, although limited, to perform experimental fault injection combined with workload variations. The experiments performed use the measurement methodology outlined by Shin and Lee [12] and a workload generator developed by researchers from Carnegie-Mellon University [13].

It should be noted that memory components require a different type of fault latency analysis than that addressed here. Memory fault latency could be on the order of minutes or hours [8], whereas fault latencies for other components are on the order of milli- to microseconds. The user-defined or scrubbing access of address locations is another characteristic that differentiates memory components from other user independent components. The focus of this work is on components that exhibit short fault latencies, i.e., latencies less than one second.

The remainder of the paper is organized as follows. The next section describes the experimental system and environment. Section III presents the experimental fault latency measurements with the proper statistical analysis. This involves the use of *isotonic regression* analysis. The paper concludes with Section IV.

[2]For example, a double error is when a second error occurs before recovery from the first error is completed.

II. Experimental System Description

FTMP, located at NASA Langley's AIRLAB, is a prototype real-time multiprocessor typical of those used for real-time control applications. Its logical architecture consists of three triads, system memory, input/output links, system clock, system control registers, and a single time-shared system bus. A triad consists of three pairs of a processor and its local memory. Each system component is redundant and is either an active, standby, or shadow component. For reliability purposes, line replaceable units (LRU's) contain one component of each type.

The three processors in a triad operate in tight synchrony and should receive identical data under fault-free conditions. When there is a disagreement, an error is considered to have occurred, but masked, and task execution continues. The disagreement is detected by a hardware voter and recorded in an error latch for later identification of the faulty module or bus. The interested reader is referred to [10] for a complete architectural description of FTMP.

The operating workload of FTMP is the executive software and applications software [11]. The executive software controls the hardware and software resources. It is responsible for providing a virtual machine such that hardware redundancy, redundancy management, and the timely execution of application tasks are transparent to the user. Therefore, the executive software is present in every workload constructed.

All tasks are dispatched at periodic intervals to control repetitive applications such as flight control, configuration control, fault detection, recovery, and system displays. Three dispatch rate groups have been defined, R1, R3, and R4, with respective nominal frequencies of 3.125, 12.5, and 25 Hz. Task execution is based on priority interrupt scheduling, where a task in a rate group has priority over tasks in slower rate groups, e.g., R4 tasks have priority over R3 and R1 tasks, etc.

A hardware fault injector (FI) and a synthetic workload generator (SWG) are available for conducting experiments on FTMP. The FI allows pin level functional faults of any duration to be injected into any IC chip within FTMP [14]. The SWG, developed by researchers at Carnegie-Mellon University [13], provides an FTMP experimenter with the capability to alter the workload on FTMP.

The SWG can only vary a few features, the number of tasks in each rate group and the fixed number of repetitions of a set of five instructions (local read/write, system read/write, and assignment). The SWG was designed for other timing experiments. Despite this shortcoming, FTMP is a unique environment where fault injection experiments can be performed with workload variations on a realistic real-time system.

Once a synthetic workload is constructed, the number of tasks and their structures cannot change during execution, i.e., they are deterministic. Therefore, varying the number of tasks is equivalent to changing the number of instruction repetitions in each task. In the experiments, the number of tasks in each rate group was varied and all tasks are identical. Due to memory constraints on FTMP, the SWG can include up to three tasks in each rate group.[3]

A critical task in the executive software is the system configuration (SCC), which executes as an R1 tasks. The SCC is responsible for reading the error latches, determining which component is faulty, and initiating system reconfiguration. After a fault is injected, the FI software waits for notification of system reconfiguration or times out. If too many tasks of higher priority than the SCC are introduced, system reconfiguration cannot be completed before the FI times out. Therefore, the number of tasks is restricted. Even though this constraint is a result of the experimental environment, it parallels a real world restriction on the response time of the SCC. The FI time-out period is synonymous with a response time deadline.

We analyze four components of FTMP: the cache controller (CC) board, the system control registers (SCR) board, the system bus controller (SBC) board, and the CPU data (CPUD) board. Two synthetic workloads are constructed to represent the range of workloads within the performance constraints. The low utilization workload is the executive software only. The high utilization workload contains the greatest number of tasks executed per second without having the FI time out. For all experiments and workloads, FTMP was configured to have three fault-free triads operating. This is the intended initial state of FTMP and demonstrates the maximum contention for system components.

[3]Recall that the executive software is included in all synthetic workloads.

III. Experimental Results and Their Analysis

The fault latency duration distribution (FLDD) is determined using the methodology developed by Shin and Lee [12]. Pin locations are randomly selected from the available locations[4] and inverted signal transient faults, varying in duration from 0 to 100 milliseconds, are injected at random times. Inverted faults provide immediate error generation when the pin is exercised.

After fault injection, FTMP waits 20 seconds for the fault to be detected. If the fault duration is shorter than the fault latency, the transient fault will not manifest an error. Since fault latency is a random variable, repeating the same transient fault duration and assuming 100 percent fault detection, the experimental probability of the FLDD at that time point can be determined by counting the number of detected faults.

Figs. 2–5 present the *isotonic regression* [15] of the experimentally derived FLDD's for the four components analyzed on FTMP. Two distributions are shown for each component showing the low and high utilization workloads. The points denoted by "Y" ("+") symbols were derived when FTMP was executing the low (high) utilization workload. The latency durations were measured in milliseconds for Figs. 2–4 and in microseconds for Fig. 5.

Let $\tilde{F}_L(t)$ be the experimental probability distribution value derived for duration t. Isotonic regression is the maximum likelihood estimate and the least-squares estimate of the sequence $\{\tilde{F}_L(t_1), \tilde{F}_L(t_2), \cdots, \tilde{F}_L(t_n)\}$, $t_1, < t_2 < \cdots < t_n$, over the set of nondecreasing sequences [15]. In determining $\tilde{F}_L(t)$, it is possible to observe $\tilde{F}_L(t_i) > \tilde{F}_L(t_j)$ with $t_i < t_j$. Since a probability distribution is a nondecreasing function, the use of isotonic regression to estimate $F_L(t)$ is justified.

Let $F_L^*(t)$ be the isotonic regression estimate of $F_L(T)$. To justify the use of $F_L^*(t)$ as and accurate approximation of $F_L(t)$, it is necessary to derive confidence bounds. Standard techniques for deriving confidence bounds do not apply for this analysis, because the values of fault latency cannot be directly measured, only the probability mass for particular durations is approximated. Specifically, the popular Kolmogorov–Smirnov cannot be used, because it requires a sample of direct observation values. Fortunately, Schoenfeld [16] introduced a method to determine the upper and lower $1 - \alpha$, $0 < \alpha < 1$, confidence bounds for a sequence of normal random variables with nondecreasing ordered means. This is exactly what is measured when the fault latency distributions are determined. The use of this method for estimating the distribution of fault latency has been outlined in [17]. The method assumes that the function is monotonically nondecreasing and infers a regression estimate over a sequence of values.

Tables I–IV list the results of the isotonic regression and calculation of the 95 percent confidence bounds for both fault latency distributions for each board. The "Note" column identifies where confidence bounds and/or regression values of the distributions for the low and high utilization workloads overlap. The minimal overlapping that occurs demonstrates that a statistically significant change in the fault latency distribution occurs when the workload of the system is varied.

For Figs. 2–4, it is observed that the regression value for the high utilization workload is always greater than or equal to the value for the low utilization workload for every time point, except

[4]Available locations are where the FI software can restore FTMP to its fault-free state after fault injection.

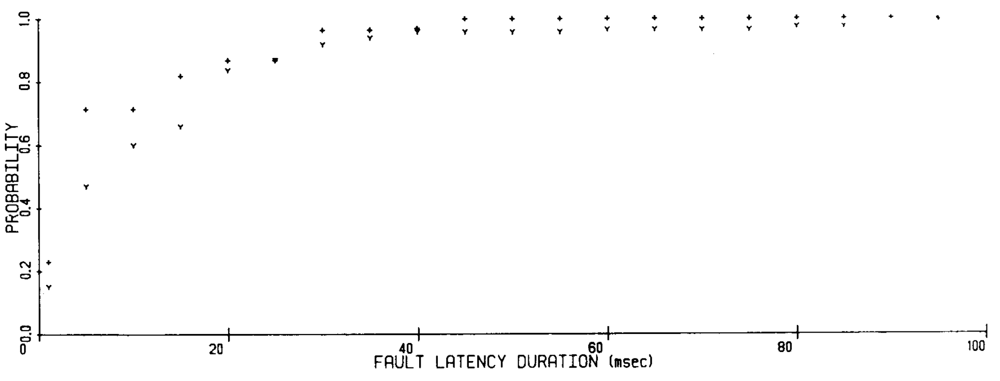

Fig. 2. Cache controller board.

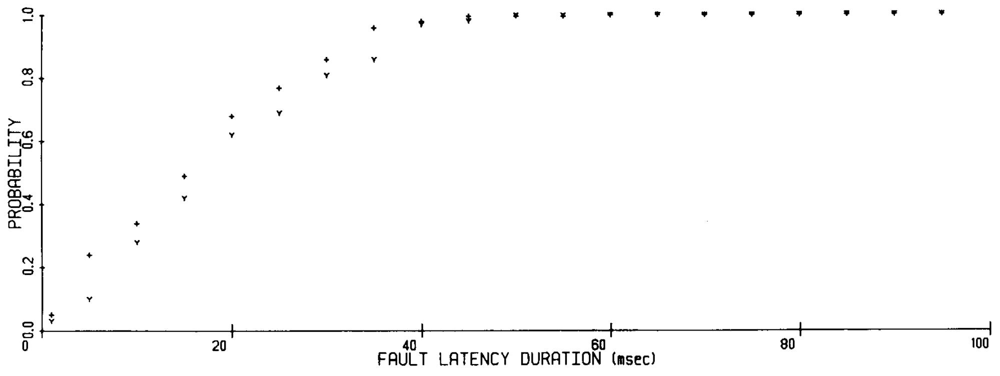

Fig. 3. System control registers board.

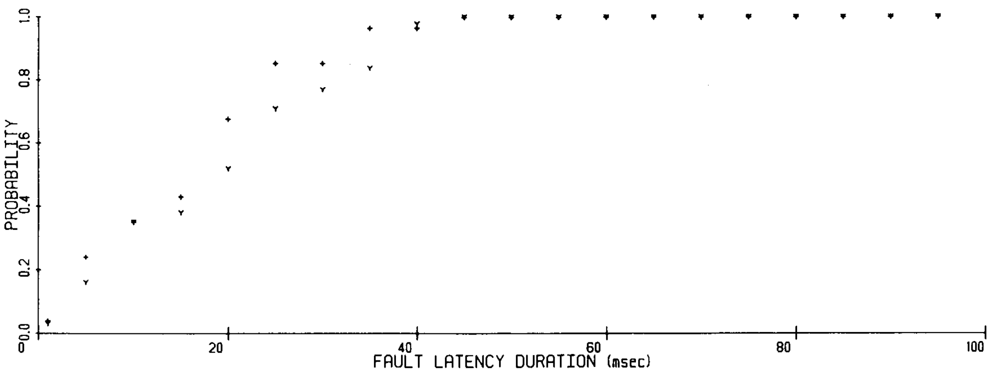

Fig. 4. System bus controller board.

for a few minor differences when the distributions approach 1. This observation supports two important ideas stressed in this paper. First, the plots quantitatively measure the workloads effects on fault latency. The type of workload executing on a system does have a marked effect on the distribution of fault latency. Second, a workload that increases system utilization will decrease the fault latency for active components.

The plots for the CPU Data Board (Fig. 5), do not support any noticeable difference between the low and high utilization workloads. This is because the CPU data path is usually at 100 percent utilization independent of which workload is being executed. Even the IDLE operation on FTMP is an infinite loop of NOP instructions. Fault latency in this area cannot be altered by variations in the workload. It should be noted, though, that the CPUD board fault latency durations are extremely short, on the order of 0-250 microseconds, which is less than 0.5 percent of the durations of all the other boards.

IV. Conclusion

This paper demonstrated the need to address fault latency in highly reliable real-time control computer systems. When fault arrival rates are significantly low, decreasing fault latency increases

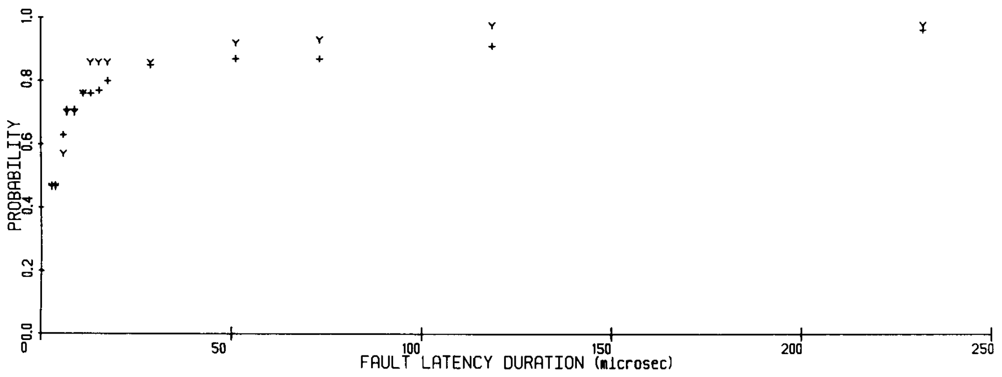

Fig. 5. CPU data board.

TABLE I
REGRESSION RESULTS FOR THE CC BOARD

t	Low Utilization Workload				Note*	High Utilization Workload			
	$\tilde{F}_L(t)$	L.Bnd.	$F_L^*(t)$	U.Bnd.		L.Bnd.	$F_L^*(t)$	U.Bnd.	$\tilde{F}_L(t)$
1.0	0.150	0.143	0.150	0.150		0.222	0.230	0.230	0.230
5.0	0.470	0.460	0.470	0.483		0.708	0.715	0.723	0.720
10.0	0.600	0.590	0.600	0.612		0.708	0.715	0.723	0.710
15.0	0.660	0.650	0.660	0.672		0.812	0.820	0.829	0.820
20.0	0.838	0.830	0.838	0.848		0.865	0.870	0.876	0.880
25.0	0.870	0.862	0.870	0.878	—	0.865	0.870	0.876	0.860
30.0	0.920	0.914	0.920	0.927		0.962	0.965	0.968	0.970
35.0	0.940	0.934	0.940	0.946		0.962	0.965	0.968	0.960
40.0	0.960	0.955	0.958	0.961		0.966	0.970	0.974	0.970
45.0	0.980	0.956	0.958	0.961		1.000	1.000	1.000	1.000
50.0	0.970	0.956	0.958	0.961		1.000	1.000	1.000	1.000
55.0	0.910	0.956	0.958	0.961		1.000	1.000	1.000	1.000
60.0	0.990	0.963	0.965	0.968		1.000	1.000	1.000	1.000
65.0	0.980	0.963	0.965	0.967		1.000	1.000	1.000	1.000
70.0	0.910	0.963	0.965	0.967		1.000	1.000	1.000	1.000
75.0	0.960	0.963	0.965	0.967		1.000	1.000	1.000	1.000
80.0	0.980	0.971	0.974	0.976		1.000	1.000	1.000	1.000
85.0	0.970	0.971	0.974	0.975		1.000	1.000	1.000	1.000
90.0	0.970	0.971	0.974	0.975		1.000	1.000	1.000	1.000
95.0	0.990	0.990	0.990	0.992		1.000	1.000	1.000	1.000

*Note**: — implies the bounds and/or regression values overlap.

TABLE II
REGRESSION RESULTS FOR THE SCR BOARD

t	Low Utilization Workload				Note*	High Utilization Workload			
	$\tilde{F}_L(t)$	L.Bnd.	$F_L^*(t)$	U.Bnd.		L.Bnd.	$F_L^*(t)$	U.Bnd.	$\tilde{F}_L(t)$
1.0	0.030	0.027	0.030	0.030		0.046	0.050	0.050	0.050
5.0	0.100	0.094	0.100	0.108		0.231	0.240	0.251	0.240
10.0	0.280	0.271	0.280	0.291		0.330	0.340	0.352	0.340
15.0	0.420	0.409	0.420	0.432		0.479	0.490	0.502	0.490
20.0	0.620	0.609	0.620	0.632		0.670	0.680	0.692	0.680
25.0	0.690	0.679	0.690	0.701		0.760	0.770	0.780	0.770
30.0	0.810	0.801	0.810	0.820		0.852	0.860	0.868	0.860
35.0	0.860	0.852	0.860	0.868		0.955	0.960	0.965	0.960
40.0	0.970	0.966	0.970	0.974		0.977	0.980	0.983	0.980
45.0	0.980	0.977	0.980	0.983		0.995	0.996	0.996	1.000
50.0	1.000	0.999	0.999	0.999	—	0.995	0.996	0.996	0.990
55.0	1.000	0.999	0.999	0.999	—	0.995	0.996	0.996	0.990
60.0	1.000	0.999	0.999	0.999		1.000	1.000	1.000	1.000
65.0	1.000	0.999	0.999	0.999		1.000	1.000	1.000	1.000
70.0	0.990	0.999	0.999	0.999		1.000	1.000	1.000	1.000
75.0	0.990	0.999	0.999	0.999		1.000	1.000	1.000	1.000
80.0	1.000	1.000	1.000	1.000		1.000	1.000	1.000	1.000
85.0	1.000	1.000	1.000	1.000		1.000	1.000	1.000	1.000
90.0	1.000	1.000	1.000	1.000		1.000	1.000	1.000	1.000
95.0	1.000	1.000	1.000	1.000		1.000	1.000	1.000	1.000

*Note**: — implies the bounds and/or regression values overlap.

TABLE III
REGRESSION RESULTS FOR THE SBC BOARD

t	Low Utilization Workload				Note*	High Utilization Workload			
	$\tilde{F}_L(t)$	L.Bnd.	$F_L^*(t)$	U.Bnd.		L.Bnd.	$F_L^*(t)$	U.Bnd.	$\tilde{F}_L(t)$
1.0	0.030	0.027	0.030	0.030		0.036	0.040	0.040	0.040
5.0	0.160	0.153	0.160	0.169		0.231	0.240	0.251	0.240
10.0	0.350	0.340	0.350	0.362	—	0.340	0.350	0.362	0.350
15.0	0.380	0.369	0.380	0.392		0.419	0.430	0.442	0.430
20.0	0.520	0.509	0.520	0.532		0.666	0.677	0.688	0.677
25.0	0.710	0.700	0.710	0.721		0.847	0.853	0.859	0.859
30.0	0.770	0.760	0.770	0.780		0.847	0.853	0.859	0.847
35.0	0.838	0.830	0.838	0.847		0.962	0.965	0.968	0.970
40.0	0.978	0.977	0.978	0.980	—	0.961	0.965	0.968	0.959
45.0	1.000	1.000	1.000	1.000	—	0.998	0.999	0.999	1.000
50.0	1.000	1.000	1.000	1.000	—	0.998	0.999	0.999	1.000
55.0	1.000	1.000	1.000	1.000	—	0.998	0.999	0.999	0.990
60.0	1.000	1.000	1.000	1.000		1.000	1.000	1.000	1.000
65.0	1.000	1.000	1.000	1.000		1.000	1.000	1.000	1.000
70.0	1.000	1.000	1.000	1.000		1.000	1.000	1.000	1.000
75.0	1.000	1.000	1.000	1.000		1.000	1.000	1.000	1.000
80.0	1.000	1.000	1.000	1.000		1.000	1.000	1.000	1.000
85.0	1.000	1.000	1.000	1.000		1.000	1.000	1.000	1.000
90.0	1.000	1.000	1.000	1.000		1.000	1.000	1.000	1.000
95.0	1.000	1.000	1.000	1.000		1.000	1.000	1.000	1.000

*Note**: — implies the bounds and/or regression values overlap.

TABLE IV
REGRESSION RESULTS FOR THE CPUD BOARD

t	Low Utilization Workload				Note*	High Utilization Workload			
	$\tilde{F}_L(t)$	L.Bnd.	$F_L^*(t)$	U.Bnd.		L.Bnd.	$F_L^*(t)$	U.Bnd.	$\tilde{F}_L(t)$
2.8	0.480	0.458	0.465	0.474	—	0.463	0.470	0.479	0.480
3.8	0.450	0.458	0.465	0.474	—	0.463	0.470	0.479	0.460
5.9	0.570	0.560	0.570	0.582		0.620	0.630	0.642	0.630
6.8	0.700	0.690	0.700	0.708	—	0.703	0.710	0.718	0.720
8.8	0.700	0.693	0.700	0.711	—	0.703	0.710	0.718	0.700
11.1	0.760	0.750	0.760	0.770	—	0.754	0.761	0.768	0.790
13.1	0.860	0.852	0.858	0.862	—	0.754	0.761	0.768	0.730
15.3	0.880	0.854	0.858	0.862	—	0.760	0.770	0.780	0.770
17.6	0.860	0.854	0.858	0.862	—	0.790	0.800	0.809	0.800
28.8	0.830	0.854	0.858	0.862	—	0.841	0.850	0.858	0.850
51.2	0.920	0.913	0.920	0.926	—	0.862	0.870	0.875	0.870
73.2	0.930	0.924	0.930	0.935	—	0.864	0.870	0.877	0.870
118.5	0.980	0.972	0.975	0.977	—	0.903	0.910	0.916	0.910
232.0	0.970	0.972	0.975	0.977	—	0.955	0.960	0.964	0.960

*Note**: — implies the bounds and/or regression values overlap.

the reliability of the entire system. Through experiments on FTMP, we provided evidence that the duration of fault latency is dependent on the system workload. These results indicate that a methodology could be developed to enable the optimal design of the workload structure to decrease system fault latency.

ACKNOWLEDGMENT

The authors would like to thank P. Padilla and R. Angallatta for their help in performing the FTMP experiments and E. Ellis and R. Butler for assistance in the isotonic regression analysis.

References

[1] A. Avizienis and J. C. Laprie, "Dependable computing: From concept to design diversity," *Proc. IEEE*, vol. 74, pp. 629-638, May 1986.

[2] K. G. Shin and Y. H. Lee, "Error detection process—Model, design, and impact on computer performance," *IEEE Trans. Comput.*, vol. C-33, pp. 529-540, June 1984.

[3] A. L. Hopkins, T. B. Smith, and J. H. Lala, "FTMP—A highly reliable fault-tolerant multiprocessor for aircraft," *Proc. IEEE*, vol. 66, pp. 1221-1240, Oct. 1978.

[4] R. K. Iyer, S. E. Butner, and E. J. McCluskey, "A statistical failure/load relationship: Results of a multicomputer study," *IEEE Trans. Comput.*, vol. C-31, pp. 697-706, July 1982.

[5] R. K. Iyer and D. J. Rosetti, "Effect of system workload on operating system reliability: A study on IBM 3081," *IEEE Trans. Software Eng.*, vol. SE-11, pp. 1438-1448, Dec. 1985.

[6] X. Castillo and D. P. Siewiorek, "Workload, performance, and reliability of digital computing systems," in *Proc. 11th Annu. Int. Symp. Fault-Tolerant Computing*, 1981, pp. 84-89.

[7] J. G. McGough and F. L. Swern, "Measurement of fault latency in a digital avionic mini processor," Tech. Rep. 3651, NASA Contractor Rep., Jan. 1983.

[8] R. Chillarege and R. K. Iyer, "Fault latency in the memory—An experimental study on VAX 11/780," in *Proc. 16th Annu. Int. Symp. Fault-Tolerant Computing*, 1986, pp. 258-263.

[9] M. H. Woodbury and K. G. Shin, "Workload effects on fault latency for real-time computing systems," in *Proc. Real-Time Systems Symp.*, Dec. 1987, pp. 188-197.

[10] T. B. Smith and J. H. Lala, "Development and evaluation of a fault-tolerant multiprocessor (FTMP) computer: Volume I FTMP principles of operation," NASA Contractor Rep., Tech. Rep. 166071, May 1983.

[11] J. H. Lala and T. B. Smith, "Development and evaluation of a fault-tolerant multiprocessor (FTMP) computer: Volume II FTMP software," NASA Contractor Rep., Tech. Rep. 166072, May 1983.

[12] K. G. Shin and Y. H. Lee, "Measurement and application of fault latency," *IEEE Trans. Comput.*, vol. C-35, pp. 370-375, Apr. 1986.

[13] F. Feather, "Validation of a fault-tolerant multiprocessor: Baseline experiments and workload implementation," Master's thesis, Dep. ECE, Carnegie-Mellon Univ., Pittsburgh, PA, 1984.

[14] J. H. Lala and T. B. Smith, "Development and evaluation of a fault-tolerant multiprocessor (FTMP) computer: Volume III FTMP test and evaluation," NASA Contractor Rep., Tech. Rep. 166073, May 1983.

[15] R. E. Barlow *et al.*, *Statistical Inference Under Order Restrictions*. New York: Wiley, 1972.

[16] D. A. Schoenfeld, "Confidence bounds for normal means under order restrictions, with application to dose-response curves, toxicology experiments, and low-dose extrapolation," *J. Amer. Stat. Assoc.*, vol. 81, pp. 186-195, Mar. 1986.

[17] E. L. Ellis and R. W. Butler, "Estimating the distribution of fault latency in a digital processor," NASA Tech. Memo., Tech. Rep. 100521, Nov. 1987.

Experimentally Characterizing the Behavior of Multiprocessor Memory Systems: A Case Study

K. GALLIVAN, D. GANNON, W. JALBY, A. MALONY, AND H. WIJSHOFF

***Abstract*—Although architectural improvements in memory organization of multiprocessor systems can increase effective data bandwidth, the actual performance achieved is highly dependent upon the characteristics of the memory address streams; e.g., the data access rate, and the temporal and spatial distributions. Accurately quantifying the performance behavior of a multiprocessor memory system across a broad range of algorithmic parameters is crucial if users (and restructuring compilers) are to achieve high-performance codes. In this paper, we demonstrate how the behavior of a cache-based multivector processor memory system can be systematically characterized and its performance experimentally correlated with key features of the address stream. The approach is based on the definition of a family of parameterized kernels used to explore specific aspects of the memory system's performance. The empirical results from this kernel suite provide the data from which architectural or algorithmic characteristics can be studied. The results of applying the approach to an Alliant FX/8 are presented.**

***Index Terms*—Characterization, memory systems, multiprocessor, performance.**

Manuscript received April 3, 1989; revised October 2, 1989. Recommended by R. K. Iyer. This work was supported by the National Science Foundation under Grant US NSF MIP-8410110, the Department of Energy under Grant US DOE-DE-FGO2-85ER25001, the Air Force Office of Scientific Research under Grants AFOSR-85-0211 and AFOSR 86-0147, and an IBM Donation.

The authors are with the Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

IEEE Log Number 8932234.

I. Introduction

For shared memory multiprocessors, access to the common memory is one of the key limiting factors in performance. One of the most attractive solutions to this problem is the use of a hierarchical memory system. This approach reduces the apparent memory latency as well as the memory contention. However, the performance is far from uniform and depends not only upon the characteristics of the memory hierarchy itself, but also on the characteristics of the address streams and the interaction between the two. This implies that the relationship of code characteristics to machine characteristics must be taken into account. For example, knowing the precise penalty in terms of number of cycles for a cache miss is not enough to understand the effectiveness of a given cache organization. We need to determine precisely, as a function of the temporal and spatial distribution of the requests, the data access rate and to try to correlate observed behavior with code characteristics. This requires a systematic investigation of the parameter space (code characteristics).

Classically, two main approaches are used for performance analysis: analytical or experimental (simulation or measurement). The first solution is extremely powerful in the sense that it allows the analytical correlation of the performance with organizational parameters. The drawback is that, in order to be tractable, they typically require a drastic simplification of the hardware model and of the memory request stream. For example, queueing theory-based models assume a randomly distributed (both in time and space) memory request stream. This is particularly disturbing when modeling scientific codes on vector machines because these codes tend to exhibit very regular data access patterns and the vector instructions used to implement the codes must exploit, and thereby emphasize, this regularity in the spatial and temporal distribution of the requests. Experimental performance analysis (simulation or measurement) provides more accurate information in the sense that it is possible to take into account more details of the hardware and code characteristics. The drawback of such a solution is its experimental nature which limits the number of codes analyzed and generally does not provide any methodology for extrapolating the performance of an arbitrary code from the performance of the benchmark codes. Furthermore, even when using very simple benchmarks, there is no general method for correlating code characteristics with the performance observed.

Our primary goal in this paper is to present a systematic methodology for investigating and correlating the performance of a cache-based memory system (in our case, the Alliant FX/8) in terms of architectural parameters and code characteristics typical of scientific numerical computations. The resulting characterization can be used for performance prediction of scientific codes. Furthermore, the design of the empirical kernels upon which the meth-