LA-UR- 04-00352

Approved for public release; distribution is unlimited.

Title:	Neutron Beam Irradiation Study of Workload Dependence of SER in a Microprocessor
Author(s):	Ted Hong, Sarah Michalak, Todd Graves, Jerry Ackaret, Sonny Rao, Subhasish Mitra and Pia Sanda
Intended for:	SELSE5

Form 836 (7/06)

Neutron Beam Irradiation Study of Workload Dependence of SER in a Microprocessor

Ted Hong, Sarah Michalak*, Todd Graves*, Jerry Ackaret#, Sonny Rao#, Subhasish Mitra and Pia Sanda#

Stanford University,

*Los Alamos National Laboratory, and

#IBM Systems and Technology Group

Abstract— It is known that workloads are an important factor in SER, but it is proving difficult to find differentiating workloads for microprocessors. We have performed neutron beam irradiation studies of a commercial microprocessor under a wide variety of workload conditions from idle, performing no operations, to very busy workloads resembling real HPC, graphics, and business applications. There is evidence that the mean times to first indication of failure, MTFIF defined in Section II, may be different for some of the applications.

***Index Terms*—SER, RAS, Fault Tolerance**

I. Introduction

It makes sense that soft error rates (SER) in a computer system would have more impact if there is work done, and less if there is no useful work being done. Bender et. al. showed that SER is dependent upon workload in an I/O hub chip [1]. If we apply Little's Law to SER, then SER should scale with workload bandwidth and data residency. If there is little or no workload, and little or no data residency, one would think that there should be little or no SER impact. Last year at SELSE 4, Rao et. al. showed controversial results that did not differentiate between idle and busy workloads for two different machines [2]. This year, we expanded the study to include more workloads and more data points for each workload. Statistical analysis was used to see if the workloads could be distinguished. With these data, there is evidence that the mean times to first indication of failure for some of the applications may be different.

II. System and Applications

Neutron beam measurements were performed at Los Alamo National Laboratory (LANL) in New Mexico using a single test system. The instrument configuration was similar but not identical to last year's measurement [2]. Notably the beam was denser than last year and the memory configuration was comparable but slightly different.

Eighteen different applications, both single and multi-threaded, were run while the test system was exposed to the neutron beam. These were collected/developed from a variety of sources, including Linpack from Intel Corp; Parsec from Princeton University [7][8]; SpecCPU2000, SpecOMP2001-M, SpecJBB 2001, and SpecCPU2006 from SPEC; Splash-2 from Stanford University [13]; Ldpc from the University of Toronto [12] modified by Stanford [2]. In addition to these standard benchmarks, we also utilized the Idle loop of the underlying Linux Operating System and developed KpTest, an application that performs basic integer operations while self-checking the result .

Table 1: Application Description

App	Source	Type (Fp/Int) : Description
Linpack	Intel	Fp: Gaussian Elimination
SpecJBB	Spec2001	Int: Transactions Processing
Gcc	Spec2000	Int: C Compilation
Gzip	Spec2000	Int: Source Coding
Mcf	Spec2000	Int: Network Flow Scheduling
Milc	Spec2006	Fp: Quantum Chronodynamics
Blackscholes	Parsec	Fp: Financial (PDE)
Canneal	Parsec	Fp: Chip Routing Kernel
Swaptions	Parsec	Fp: Monte Carlo Simulation
Streamcluster	Parsec	Fp: Online Clustering of Data
X264	Parsec	Int: Video Encoding
Ammp	OMP2001	Fp: Molecular Dynamics (ODE)
Applu	OMP2001	Fp: Fluid Dynamics (PDE)
Ldpc	Neal	Fp: Chanel Coding for ECC
Raytrace	Splash-2	Fp: Raytracing Application
Idle	-	Linux Idle Workload
IIdle	-	Idle w/o Test Apparatus Wkld
KpTest	-	Int: Loop of Basic Operations

In Idle, the system is ready for work, but is not processing user instructions. In other words, apart from normal background tasks, we only run our test apparatus (which is mostly sleeping) and allow the operating system to maintain control of the processors in its idle loop. The difference between Idle and IIdle is that in IIdle the test environment is "sleeping".

From SpecCPU2000, we chose three different integer applications: Gcc, Gzip, and Mcf. Gzip is one of the simplest benchmarks while Gcc is one of the most complex. Finally, Mcf has one of the highest CPI of all Spec integer benchmarks. Finally, from SpecCPU2006 we chose a floating-point benchmark: Milc. These four applications along with the single-threaded KpTest provide a wide range of single-threaded behaviors.

The rest of the applications are dual-threaded. For both single and dual-threaded applications, the application under consideration utilizes both cores of the CPU. We ran two instances of the single threaded applications and specified that the dual-threaded applications utilized two cores. For these multi-threaded applications we chose applications with different sharing and data-resilience characteristics. Ldpc, Blackscholes, Canneal, Swaptions, Streamcluster, Ammp, and Applu with their iterative nature are thought to be less sensitive to data errors than Linpack, SpecJBB, X264, and Raytrace. SpecJBB as a Java program relies on a Java runtime engine, which with "just in time compilation," may be more vulnerable. Canneal is a unique program in that it does not utilize any locks, and can behave correctly in the presence of data races; its data is also not easily cached, forcing sharing to be performed via main memory. X264 utilizes a pipeline parallelization model in which the entire data to be worked on needs to be transferred between processors [8]. Finally, of the two OMP benchmarks Applu is reported [1] to have one of the highest communication./computation ratios suggesting that it might be more vulnerable to errors in the caches than Ammp, which has one of the least.

III. Experiment

Each application was run repeatedly until failure, where failure is defined as having occurred when the test system no longer responded to a ping from an external controller. Several "times," measured in Monitor Units (MUs), were captured from the logs for each experiment, where each log corresponds to a particular instance in which a given application was run until failure. These experimental logs are system console logs that contain values of MU output at roughly 10-second intervals, error messages the system under test reported, a bootup and reset log, and test apparatus output and errors. The number of MUs during a given time interval is proportional to the number of neutrons/cm^2 to which the device was exposed during that time interval.

For these experiments, the following MU values were collected: 1) the MU value when the experiment was launched by the external controller, which is usually prior to the launching of the application ("MU Start"); 2) the MU value immediately prior to the launching of the application ("MU First"); 3) the MU value before the first reported error in the logs ("MU Stop Early"); and 4) the MU value following the time at which the system stopped responding to the external controller ("MU Stop Late"). Under the experimental protocol, there is a slight delay between MU Start and the application startup. Rarely, MU Start occurs after the launch of the application due to errors in the automated cleanup of a prior failed run. MU Stop Early is approximate since MU counts were output to the experimental logs every 10 seconds. MU Stop Late is also approximate since the system under test is polled every 5 seconds.

For each of the applications, interest focuses on estimating the mean number of MUs, reported in arbitrary units (AUs), until the system first issued an error message or the mean time to first indication of failure, MTFIF. (MTFIF is a good measure for the inverse of SER.) This includes any type of system failure, whether directly related to the application running during the experiment or not. Because of the manner in which the data were collected, the first error is known to have occurred within 10 seconds after the MU Stop Early value. In particular, the MU values were (ideally) output to the experimental logs every 10 seconds. With MU Stop Early, the most recent MU value before an error message appeared in the experimental log, the error message must occur within 10 seconds of MU Stop Early unless one or more MU values were not reported.

Ten KpTest experiments yielded no experimental log. There is no obvious reason why these logs should be missing. KpTest was the last application run, so it is possible that the control program was suffering difficulties at that time. All KpTest experiments for which a value of MU Stop Early could be derived are included in the analyses (an experimental log is required in order to derive MU Stop Early). For the IIdle experiments, the MU test apparatus counter wasn't used, so for these experiments, only MU Start and MU Stop Late are available.

Thirty-one experiments in which the application was allowed to run until failure were conducted for each application with the following exceptions: Ammp (n=41); KpTest (n=26); Linpack (n=40); SpecJBB (n=33); and x264 (n=41).

IV. Modelling The Experimental Data

The experimental goal was to determine whether different MTFIFs, in AUs, were obtained when different applications were running. That is, interest focuses on the values of MU Stop Early – MU First. While we could have defined MTFIF from the view of the external controller that is not in the beam (MU Stop Late - MU Start), this view suffers from two errors due to delays in the external view compared to an internal one: the delay from MU Start (command given) and the application start; and the delay from the first error and MU Stop Late (actual machine failure). MTFIF from an internal view of the machine under the beam is closer to the difference between the true application start and the first error indication, provided that the logs can be trusted. Since the acceleration

factor isn't extreme, errors are intermittent enough that this assumption is likely true. Since neither MU First nor MU Stop Early values are available for Iidle, the results of the 31 Iidle experiments are omitted from the analyses.

58 additional experiments were not included in the analyses (in addition to the Iidle experiments). 49 were removed because the system experienced problems before the launching of the experiment. Either there were issues with the automation on the control machine or the machine in the beam experienced errors or failures before the launching of the experiment. Note that there were more than 49 instances of failures before the launching of the experiment; the automation was able to detect some and reboot the machine to recover and correctly launch an experiment. Similarly, 9 were removed because the failure happened after the launching of the experiment, but before the test application was launched. This yielded 495 experiments from the 17 applications that were included in the analyses.

The experimental data are interval censored, i.e., for a given experiment the first indication of failure is known to occur in an interval with left endpoint MU Stop Early - MU First and right endpoint equal to the MU value that is 10 seconds after MU Stop Early - MU First. Note that for some experiments, MU Stop Early equals MU First; this means that the failure occurred within 10s after MU First.

The MU value 10 seconds after MU Stop Early is unknown or missing. However, based on the MU values in the logs, the distribution of the change in the MU count over ten second intervals during the experiments may be estimated. (There are 0 values in this data that correspond mostly to when the beam was turned off; thus they are excluded in estimating the number of MUs in 10 seconds.) We used the mean number of MUs in the 10-second intervals (28) to estimate the number of MUs in a 10-second interval for each experiment. Thus, the first indication of failure for a given experiment was assumed to occur in the interval [MU Stop Early – MU First, MU Stop Early – MU First + 28] corresponding to that experiment. Methods for missing data exist (Little and Rubin [5]) so that sound inferences can be made in the presence of missing data. Using the mean as was done for the analyses can lead to faulty inferences [5]. **An analysis that incorporates the variability in the number of MUs in 10 seconds would lead to sounder inferences.**

In addition, there were 5 experiments for which MU First was actually "MU Second." In these cases, network issues meant that the MU count immediately before the application start could not be obtained. In all these cases, the network issue was transient and the second MU count value 10s later, MU Second, was correctly obtained. For these case, MU First was estimated by subtracting 28 from the recorded value.

The data from the experiments from the 17 applications were modeled jointly using a Bayesian hierarchical model [6]. With this model, the failure rate under a given application, which is the inverse of the MTFIF under that application, is approximately a weighted average of the observed failure rate when that application was running and a failure rate estimated using the data from all 495 experiments. The probability arrived at which includes information from prior data is called "posterior probability".

If the data observed when a given application was running provide a lot of information about the failure rate under that application, then those observed data will be weighted more heavily, compared to the failure rate estimated using the data from all 495 experiments, in the estimate of the failure rate under that experiment. Conversely, if the data observed when a given application was running provide relatively little information about the failure rate under that application, then those observed data will receive relatively little weight, compared to the failure rate estimate based on all 495 experiments, in the estimate of the failure rate under that application. **Because the model was fit via a simulation method (Markov Chain Monte Carlo), the draws of the rate parameters can be used to estimate the MTFIFs and related quantities discussed in this paper.**

For further information about the model and model checking results, see Appendix I.

V. Results

Results are presented in Figure 1, which includes estimated 95% posterior probability intervals for the MTFIF in AUs for the 17 applications. The estimated posterior probabilities are tabulated against each other in Table II. The vertical axis indicates the MTFIF in AUs, while horizontal axis indicates the application, with Blackscholes and Streamcluster abbreviated as "Blacksch" and "Streamcl" because of space constraints.

In the figure, the estimated 95% posterior interval for the MTFIF in AUs for each application is presented as a vertical line. The dash that intersects a given vertical line indicates the estimated posterior MTFIF for the corresponding application. The solid horizontal line provides the posterior mean of the estimate based on all 495 experiments. The dashed horizontal lines provide the estimated 95% posterior interval for this value. The results in Figure 1, the paragraph below, and in would likely be different if the missing number of MUs in 10 seconds had been addressed in a more rigorous manner.

The estimated posterior probability that the MTFIF when Ammp is running is greater than that when Raytrace is running is just over 0.95. The corresponding estimated posterior probabilities for the applications with the next 8 highest failure rates (Mcf through KPTest) and Raytrace are greater than 0.90. The estimated posterior probability that the MTFIF for Ammp is greater than that for Linpack is also greater than 0.90. Thus, there is evidence in the data that the MTFIF may differ among some of the applications. With additional data, further differences might be found.

In addition, one instance of silent data corruption (SDC) was observed when Linpack was running.

VI. Conclusion

These results show that the MTFIFs for some of the applications may be indeed be different. The workloads with the most difference in SER were Ammp and Raytrace. These had 95% confidence of being different, as indicated by the MTFIF. Prior experiments presented at SELSE4 were not able to differentiate between workloads. The present set of experiments were broadened in scope of workloads, and number of measurements taken at each workload. Both Bayesean and Weibull analysis methods yielded the same conclusion (Weibull analysis was used for the SELSE4 study). The present studies are consistent with last year's measurements when one looks that the overall spread in MTFIF. The difference between the most divergent values for the mean MTFIF is 20%. Even taking the extremes of the error bars, the spread is within 100%. These are small differences on the scale of the SER for various design points, and for other workloads (e.g. see Dang et. al. on I/O adaptors [16]). The minor overall differences in mean MTFIF explain that it was difficult to see statistically significant difference between workloads for the SELSE4 experiments. A notable conclusion is that the idle workload did not have the highest MTFIF. That is, the processor was not less vulnerable when running idle as when running workload (e.g. Raytrace).

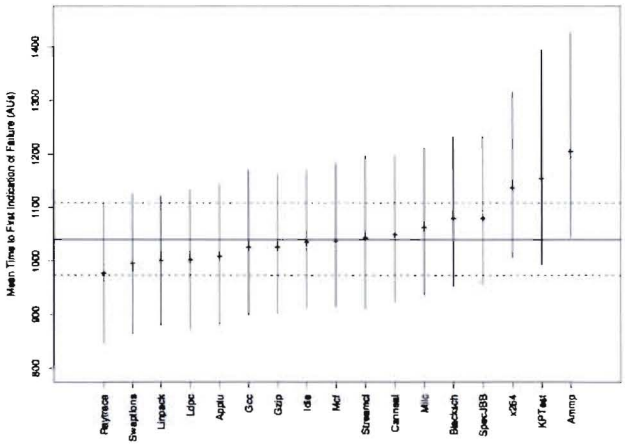

Figure 1: Estimated posterior mean times to first indication of failure, MTFIF measured in AUs and corresponding 95% posterior intervals for 17 applications

Table 2: Estimated posterior probability that the MTFIF for one application is greater than MTFIF for another. (i,j) = P(MTTIF i > MTFIF j)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	.57															
3	.58	.50														
4	.58	.51	.50													
5	.60	.53	.52	.51												
6	.66	.59	.59	.58	.56											
7	.67	.60	.59	.58	.56	.49										
8	.70	.63	.63	.62	.59	.53	.52									
9	.70	.64	.63	.62	.60	.53	.53	.50								
10	.71	.65	.64	.64	.62	.55	.55	.51	.51							
11	.74	.67	.67	.66	.64	.58	.57	.54	.53	.51						
12	.77	.72	.71	.70	.68	.62	.62	.58	.58	.56	.54					
13	.81	.76	.76	.75	.73	.67	.67	.64	.64	.61	.59	.55				
14	.81	.76	.76	.75	.73	.67	.67	.64	.64	.61	.59	.55	.49			
15	.90	.87	.88	.86	.85	.82	.82	.79	.79	.77	.76	.72	.67	.67		
16	.90	.87	.87	.86	.85	.81	.82	.79	.79	.77	.77	.73	.68	.68	.52	
17	.96	.94	.95	.94	.94	.92	.91	.90	.90	.88	.88	.86	.83	.83	.70	.65

VII. Appendix I

Under the model, the distribution of the interval-censored times to first indication of failure, in AUs, for experiment i is exponential with MTFIF $1/\lambda_i$. The λ_is are then modeled as Gamma-distributed, with mean $\exp(\beta)$ and variance $\exp(\beta)^2/\zeta$ where β and ζ are unknown. **(Since the λ_is are positive, a distribution on positive values like the Gamma is required.)** The parameters β and ζ were assigned vague priors, **with the distribution of β given fixed ζ Normal with mean -2.3 and variance 10^2** and $p(\zeta) = 10/(\zeta+10)^2$ with respect to $d\zeta$; the prior for ζ is that found in [9] when their parameter z_0=10. **The parameters of the Normal distribution and the value of z_0 were based on our prior opinion about the distribution of the λ_is and were intended to be relatively non-informative, i.e. to let the data from the experiments speak.** The model was fit using **the** YADAS **software** [10]. Model checking via plots of the observed data and posterior predictive p-values [11] suggested no gross lack of model fit. For just over half of the 17 applications, there was evidence that extremes in **the** times to first indication of failure were not consistent with the model used. **This suggests that the failure times from some of the experiments may not be exactly exponential.** For a few of the applications, further lack of model fit was suggested. Prior sensitivity analyses suggested that the results presented here are not particularly sensitive to the parameters values governing the prior distribution on β and ζ.

VIII. Appendix II

The data were also analyzed using the methods from the previous study [14]. For these analyses, ReliaSoft Weibull++ [15] was used to estimate the probability that the failure rate under application i was different from that under application j for all unique pairs i, j. The analysis is based on Equation (1):

$$P[t_2 \geq t_1] = \int_0^\infty f_1(t) * R_2(t) * dt, \quad (1)$$

where $f_1(t)$ is the probability density function under the first application and $R_2(t)$ is the reliability or survival function under the second application. (If the probability was 0.5, the distributions were considered identical. A result of 0.5 ± 0.1 was identified with approximate equality of the two distributions. There was no evidence from this analysis that conflicted with the conclusions of the analysis presented in the body of this paper.

IX. Acknowledgements

This work was performed with the aid of a research grant by the Los Alamos National Laboratory to Stanford University for neutron irradiation studies of the workload effects of SER. We thank Steve Wender of LANSCE for his help, and Ricardo Mata and Anh Dang for their assistance during the experiments.

X. References

[1] V. Aslot and R. Eigenmann, "Performance Characteristics of the SPEC OMP2001 Benchmarks" EWOMP 2001.

[2] J. Bau et al, "Error Resilient System Architecture (ERSA) for Probabilistic Applications" SELSE 3, 2006.

[3] C. Bender, P.N. Sanda, P. Kudva, R. Mata, V. Pokala, R. Haraden, M. Schallhorn, "Soft Error Resilience of the IBM POWER6 Processor Input/Output Subsystem", IBM J. Res. Dev. Vol. 52, No. 3 (2008).

[4] SELSE4 Workshop, http://www.selse.org Austin TX (2007).

[5] R. Little and D. Rubin ***Statistical Analysis with Missing Data,*** 2nd edition, John Wiley, New York (2002).

[6] A. Gelman, J. Carlin, H. Stern and D. Rubin, ***Bayesian Data Analysis,*** Chapman and Hall, London (1995).

[7] C. Bienia, S. Kumar, J.P. Singh and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," PACT 2008.

[8] C. Bienia, S. Kumar, J.P. Singh and K. Li, "The PARSEC Benchmark Suite: Characterization and Architectural Implications," Princeton University Tech Report TR-811-08 2008.

[9] C. Christiansen and C. Morris, "Hierarchical Poisson Regression Modeling", *Journal of the American Statistical Association v.92,* pp618-632 (1997).

[10] T. Graves, "Design Ideas for Markov Chain Monte Carlo Software", *Journal of Computational and Graphical Statistics v.16,* pp 24-43 (2007).

[11] A. Gelman and X. Meng, "Model Checking and Model Improvement", in ***Practical Markov Chain Monte Carlo,*** ed. W. Gilks, S. Richardson, and D. Spiegelhalter, pp 189-201, Chapman and Hall, New York (1996).

[12] Neal, "Software for Low Density Parity Check (LDPC) codes," 2006, http://www.cs.utoronto.ca/~radford/ldpc.software.html

[13] S.C Woo, et al. "The SPLASH-2 programs: characterization and methodological considerations," ISCA 1995.

[14] Rao, S., Hong, T., Sanda, P., Ackaret, J., Barrera, A., Yanez, J., Mitra, S., Kellington, JT, and McBeth, R. (2008) "Examining Workload Dependence of Soft Error Rates," *Proceedings of SELSE08.*

[15] G. G. Brown and H. C. Rutemiller, (1973) "Evaluation of Pr $\{x \geq y\}$ When Both X and Y are from Three-Parameter Weibull Distributions," IEEE Trans. Reliability R-22, No. 2, 78–82.

[16] Anh Dang et. al. submitted to this SELSE