Microprocessor Reliability Enhancement Under Ionizing Radiation Using Performance Counters.

Antonio Teijeiro
Electrical and Computer Engineering
University of Texas at El Paso
El Paso, TX USA
aeteijeiro@miners.utep.edu

Eric MacDonald
Electrical and Computer Engineering
University of Texas at El Paso
El Paso, TX USA
emac@utep.edu

Rodrigo Romero
Electrical and Computer Engineering
University of Texas at El Paso
El Paso, TX USA
raromero2@utep.edu

***Abstract*—Hardware performance counters can be used to evaluate computational accuracy in the space environment without performing redundant computations or source code modification.**

Keywords—performance counters, microprocessor reliability, GPU reliability, COTS in space.

I. Introduction

Modern global security concerns have inspired a proliferation of spaceborne remote sensing research. Many of these approaches are data and processor intensive. However, high levels of ionizing radiation found in space environments precludes the introduction of terrestrial high-performance processor technology to the space environment, due to induced soft and hard upsets. These upsets are manifested as either Device Unrecoverable Errors (DUEs) or Silent Data Corruptions (SDCs). The current state-of -the-art technique is to harden critical nodes to prevent DUEs and perform redundant computations to prevent SDCs. Thus, DUEs can be handled through a system reboot, whereas SDCs can be detected and corrected through computational redundancy. This SDC detection and correction approach does not yield optimal performance, as hardware is utilized to perform redundant tasks rather than other useful work. If remote sensing and other space bound enterprises are to continue improvement, a new method of error detection and correction is necessary – one which does not require redundancy.

Hardware performance counters, special registers which track the occurrences of configurable events, are typically used for quality assurance and software tuning. However, Guererro Balaguera et al. saw that since hardware performance counters are already used to perform hardware diagnostics, then it is natural to extend their utility to upset detection. Software simulated upsets in the open source FlexGripPlus GPU confirmed hardware performance counters to be useful for detecting SEUs affecting branch and warp scheduler activity [1]. In this work, Guererro-Balaguera et al.'s research is successfully extended to include additional hardware monitoring on a modern COTS GPU [2]. Supervised and unsupervised machine learning methods are applied to hardware performance counter metrics to successfully detect software simulated GPU L0 instruction cache, Load Store Unit (LSU), Arithmetic and Logic Unit (ALU), Fused Multiply Add (FMA), and Address Divergence Unit (ADU) SEUs. Although this work focuses on GPUs for space applications, microprocessors and CPUs sometimes contain hardware performance counters of identical benefit.

II. Experimental Setup

An in-house fault injector based on the principles of NVIDIA's deprecated SASSIFI injector was developed for software fault injection. This tool intercepts GPU binary compilation following the PTX code generation step. Fault injection instructions in the form of a nonzero XOR (bit flip) operation are inserted into the appropriate place in this virtual GPU assembly code to stimulate an SDC.

The matrix_multiplication_bench benchmark within the GPU4S benchmark suite was selected for this work [3]. Per its moniker, this benchmark consists of a multiplication between two randomly generated matrices of user-selectable size. Custom Performance Application Programming Interface (PAPI) hooks were inserted to expose internal GPU hardware performance counter values to the programmer. Since PAPI can configure GPU hardware performance counters to concurrently monitor only up to about 10 of over 250,000 possible hardware events on the NVIDIA RTX 3090 GPU utilized in this experiment, only those hardware events most related to activity in the five selected hardware units are configured for monitoring.

A Support Vector Machine (SVM) is applied to performance counter metrics to analyze supervised machine learning utility for this problem. A Local Outlier Factor model is applied to determine the utility of unsupervised learning models for this problem. The success of both types of machine learning models demonstrates an ability to detect SEUs from hardware performance counter metrics without requiring access to target software source code, as discussed further in section III [4].

Our hypothesis is as follows: since L0 instruction cache holds instructions that are soon to be executed, it is reasonable to assume that any instructions related to storing information in memory will result in a cache miss if the destination address portion of the instruction is corrupted. Likewise, LSU and ALU, SEUs resulting in erroneous memory address accesses will result in cache misses. Thus, SEUs involving memory addresses

in an L0 instruction cache, LSU, or ALU should be detectable from GPU L1/Tex miss stage, frame buffer, and device DRAM activity, which are the handlers of successively further cache misses. SEUs affecting FMA kernel thread index calculations will be detectable from anomalous FMA metric activity, since the FMA pipeline is used in subsequent portions of the benchmark. ADU SEUs will also result in anomalous FMA metric activity due to abnormal branch traversal.

L0 i-cache, LSU, and ALU hardware upsets are simulated by XOR'ing a store address with 2^{20} – a bit position determined experimentally to belong to the set index – resulting in a cache miss. Although each hardware unit is targeted in the same manner, the architectural level at which the injection occurs differs. L0 i-cache is targeted at the Stream Multiprocessor (SM) sub-partition (warp) level, the LSU injection takes place at the quarter-warp level, and the ALU injection takes place at the Stream Processor (SP) level. FMA SEUs are simulated by adding 100,000 – an index far outside the range of thread indices used to calculate the 104x104 matrix used in this work – to the result register value of a kernel thread index calculation, resulting in an early exit for these threads. ADU SEUs are simulated by XOR'ing a predicate register with 1 to change its logical state.

A dataset consisting of 1000 golden (non-fault-injected) and 1000 fault-injected runs for each hardware unit is generated. Thus, a dataset consisting of 2000 sample points is generated for each hardware unit. The SVM is trained using 3-fold cross validation across the entirety of each dataset, whereas the LOF model is trained using all available golden data, then evaluated on the entirety of each dataset, thus treating fault detection as an outlier detection problem.

III. Results and Discussion

Table I displays the SEU detection results using those metrics experimentally determined to be the most useful for detecting each type of SEU. Displayed in each row is the targeted hardware, the name of the metric, an SVM's accuracy on its data under K-fold cross validation, and an LOF's accuracy on its data. Table II presents SEU detection results for a sampling of less sensitive metrics which were hypothesized to be informative. Table III presents results from repeating SEUs simulated in table II once per Cooperative Thread Array (CTA) [2].

The results in table II ostensibly oppose the hypothesis, as several metrics which were hypothesized to be useful yielded poor detection capability. However, the results in table III from repeating an SEU across all 49 CTAs reveal a different phenomenon to be at work. Naturally noisy metrics require more stimulus than an SEU can provide in order to be useful for upset detection. This is due to the fact that modern GPUs are incompatible with a previous version of profiling tools which allow for performance counters to be read at the individual SM level, instead of being read from across the GPU. A reversion to an older GPU model will likely yield the necessary sensitivity for detecting SEUs using noisier metrics such as these. Alternatively, one may simply utilize more informative metrics, such as those experimentally determined and listed in Table I.

Other metrics present a deterministic nature, in which the events being counted display invariant activity across benchmark trials. Thus, any small disturbance in their activity will be discernible to a machine learning model. These are not process errors, but rather a characteristic of running this benchmark in isolation. Since the benchmark contains no variability in execution between golden or injected trials, an unchanging amount of FMA instructions will be executed each time. Still other metrics are not strictly deterministic, but instead present narrow distributions that are similarly sensitive and specific to the SEUs that were injected in this work. Lts__t_requests_aperture_device_evict_normal_lookup_miss, listed in table I, is one such metric. From these observations, it is clear that the reported high accuracies in table I are due to sensitive and specific metrics, rather than overfitting [2].

Thus, the results have been determined to be reliable and generally applicable. In this work, the SVM was utilized to verify the existence of two distinct classes of data – golden and

TABLE I. Best SDC Detection Results

Targeted Hardware	Metric	3-fold CV Accuracy	LOF Accuracy
L0 i-Cache	lts__t_requests_aperture_device_evict_normal_lookup_miss.sum	[0.96701649 0.97601199 0.96696697]	91.55%
LSU	lts__t_requests_aperture_device_evict_normal_lookup_miss.sum	[0.93553223 0.93703148 0.95495495]	72.6%
ALU	lts__t_requests_aperture_device_evict_normal_lookup_miss.sum	[0.86656672 0.87556222 0.87987988]	49.95%
FMA	sm__pipe_fma_cycles_active.sum	[1. 1. 1.]	100.0
ADU	sm__pipe_fma_cycles_active.sum	[1. 1. 1.]	100.0

TABLE II. LESS SENSITIVE METRICS (SEU)

Targeted Hardware	Metric	3-fold CV Accuracy	LOF Accuracy
L0 i-Cache	l1tex__t_bytes_pipe_lsu_lookup_miss.sum	[0.79910045 0.77961019 0.75825826]	69.8%
L0 i-Cache	fbpa__dram_write_bytes.sum	[0.48275862 0.48125937 0.51201201]	48.8%
LSU	l1tex__t_bytes_pipe_lsu_lookup_miss.sum	[0.63568216 0.63118441 0.63963964]	53.45%
LSU	fbpa__dram_write_bytes.sum	[0.48575712 0.47826087 0.503003]	51.05%
ALU	l1tex__t_bytes_pipe_lsu_lookup_miss.sum	[0.57721139 0.5892054 0.56606607]	49.1%
ALU	fbpa__dram_write_bytes.sum	[0.55922039 0.51124438 0.51951952]	50.25%

TABLE III. LESS SENSITIVE METRICS, REPEATED SEU

Targeted Hardware	Metric	3-fold CV Accuracy	LOF Accuracy
L0 i-Cache	l1tex__t_bytes_pipe_lsu_lookup_miss.sum	[1. 1. 1.]	93.85%
L0 i-Cache	fbpa__dram_write_bytes.sum	[0.98650675 0.98350825 0.98048048]	91.6%
LSU	l1tex__t_bytes_pipe_lsu_lookup_miss.sum	[1. 1. 1.]	98.85%
LSU	fbpa__dram_write_bytes.sum	[0.95352324 0.93853073 0.95195195]	86.75%
ALU	l1tex__t_bytes_pipe_lsu_lookup_miss.sum	[0.98650675 0.988006 0.996997]]	98.35%
ALU	fbpa__dram_write_bytes.sum	[0.7946027 0.79610195 0.80930931]	70.3.%

injected – after sampling metrics from golden and injected trials. However, as a supervised machine learning algorithm, commercial application of an SVM requires the user to inject faults into source code to generate training data. Since commercial software is often closed-source, this presents a serious limitation. Therefore, LOF success is important to note. The LOF model was trained using noisy golden data, then tested on the entirety of the 2000 sample golden and injected dataset for each metric. The LOF also reported up to 100% accuracy. Therefore, it is possible to treat SEU SDC detection as an anomaly detection problem by profiling closed source software and training a novelty detection model, such as LOF, on the resulting golden data. Then the model may be deployed to its intended environment [2].

IV. CONCLUSION AND FUTURE WORK

An alternative approach to redundant computation methods currently employed for SDC detection will aid advancement in spaceborne remote sensing applications. Here, a method utilizing hardware performance counters, which are registers that maintain the number of occurrences of configurable hardware events within a processor, is expanded on. This research expands on previous work which utilized hardware that is not suitable for practical spaceborne applications, by

evaluating the utility of previously presented metrics as well as introducing new ones on a modern COTS GPU. The success of this approach indicates that further research on this approach is warranted.

Future work includes the evaluation of this work under hardware fault injection conditions. In particular, proton beam testing at low LETs to evaluate the suitability of this technique for SEU SDC detection is necessary. If successful, then MEU SDC detection may be evaluated, perhaps as a linear phenomenon. Regression tests utilizing an older GPU compatible with previous generation profiling tools (i.e. CUPTI) may prove helpful for other GPU hardware, although unnecessary for the presently targeted hardware.

REFERENCES

[1] Guerrero-Balaguera, J.-D., Condia, J. E., & Reorda, M. S. (2021). Using hardware performance counters to support infield GPU testing. *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. https://doi.org/10.1109/icecs53924.2021.9665511.

[2] Teijeiro, A. E. (2023, November 25). GitHub. https://github.com/aeteijeiro/ThesisDocument.

[3] Kosmidis, L., Rodriguez, I., Jover, Á., Alcaide, S., Lachaize, J., Abella, J., Notebaert, O., Cazorla, F. J., & Steenari, D. (2020). GPU4S: Embedded gpus in space - latest project updates. Microprocessors and Microsystems, 77, 103143. https://doi.org/10.1016/j.micpro.2020.103143.

[4] Teijeiro, A. E. (2023, November 25). GitHub. https://github.com/aeteijeiro/ThesisWork.