Understanding GPU Errors on Large-scale HPC Systems and the Implications for System Design and Operation

Devesh Tiwari[*], Saurabh Gupta[*], James Rogers[*],
Don Maxwell[*], Paolo Rech[†], Sudharshan Vazhkudai[*], Daniel Oliveira[†],
Dave Londo[§], Nathan DeBardeleben[‡], Philippe Navaux[†], Luigi Carro[†], and Arthur Bland[*]

[§]Cray Inc.
[‡]Los Alamos National Laboratory
[†]Federal University of Rio Grande do Sul
[*]Oak Ridge Leadership Computing Facility, Oak Ridge National Laboratory

Abstract

Increase in graphics hardware performance and improvements in programmability has enabled GPUs to evolve from a graphics-specific accelerator to a general-purpose computing device. Titan, the world's second fastest supercomputer for open science in 2014, consists of more than 18,000 GPUs that scientists from various domains such as astrophysics, fusion, climate, and combustion use routinely to run large-scale simulations. Unfortunately, while the performance efficiency of GPUs is well understood, their resilience characteristics in a large-scale computing system have not been fully evaluated.

We present a detailed study to provide a thorough understanding of GPU errors on a large-scale GPU-enabled system. Our data was collected from the Titan supercomputer at the Oak Ridge Leadership Computing Facility and a GPU cluster at the Los Alamos National Laboratory. We also present results from our extensive neutron-beam tests, conducted at Los Alamos Neutron Science Center (LANSCE) and at ISIS (Rutherford Appleton Laboratories, UK), to measure the resilience of different generations of GPUs. We present several findings from our field data and neutron-beam experiments, and discuss the implications of our results for future GPU architects, current and future HPC computing facilities, and researchers focusing on GPU resilience.

1. Introduction

Increase in graphics hardware performance and improvements in programmability has enabled Graphics Processing Units (GPUs) to evolve from a graphics-specific accelerator to a general-purpose computing device. Consequently, GPUs have enjoyed wide-spread adoption in various application domains, including scientific computing [1, 27]. Scientists have begun to take advantage of the unprecedented amount of parallelism available in GPUs to expedite their scientific simulations and to derive scientific insights more quickly. For example, Titan, the world's second fastest supercomputer for open science in 2014, consists of 18,688 GPUs that scientists from various domains such as astrophysics, fusion, climate, and combustion use routinely to run large-scale simulations [1, 27].

These large-scale scientific applications are often very long-running; a single simulation may take from a few hours to a couple of days. Due to the large-scale and the long duration, leadership scientific applications may encounter interruptions due to system failures. Therefore, while the performance improvement achieved via inherent parallelism available in GPUs is necessary to expedite the scientific discovery process, it is equally critical that applications are able to cope with system failures during a run, without losing all of the work. Scientific applications typically employ a checkpoint/restart mechanism to periodically take checkpoints such that it can continue to make forward progress even in the event of failures. However, the efficiency of these mechanisms depends on the resilience characteristics of the system. Unfortunately, while the performance efficiency of GPUs is well understood, their resilience characteristics in a large-scale computing system have not been well studied.

As we approach exascale, the resilience challenge will become even more critical due to increase in system-scale [23]. Additionally, GPUs are anticipated to be a part of the projected path to exascale due to their ability of offer more FLOPS compared to traditional CPUs. Therefore, understanding the nature of GPU errors is critical for operating today's large-scale HPC systems such as the Titan supercomputer as well as designing future extreme-scale systems [39]. In fact, lack of understanding about resilience characteristics of these emerging compute system components may lead to lower scientific productivity, lower operational efficiency and even, significant monetary loss [39]. This led us to conduct a large-scale field study on GPU error characterization, quantification, and impact. To the best of our knowledge, this is the first study to provide a thorough understanding of GPU errors on a large-scale GPU-enabled system using both field and experimental data.

Our study consists of three parts. First, we present findings from analyzing GPU errors on the Titan supercomputer at the Oak Ridge Leadership Computing Facility (OLCF). Our data includes single and double bit errors for all 18,688 GPUs in the Titan supercomputer. We investigate the temporal and spatial characteristics of GPU errors, their correlation with

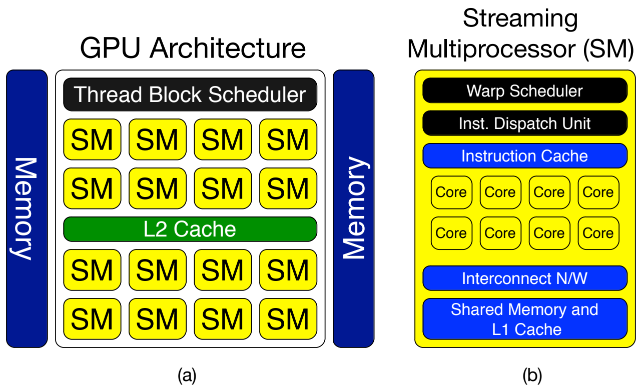

Figure 1: A representative CUDA-based GPU architecture.

temperature and per memory structure breakdown among several other factors. Our analysis reveals interesting insights about GPU error characteristics that can be used to improve operational efficiency of large-scale HPC facilities and makes recommendations to future GPU architects. The second part of the study, conducted at the Los Alamos National Laboratory, shares our experience with the first generation of GPU cards enabled with ECC support.

Finally, the third part the study presents results from our extensive neutron-beam tests, conducted at the Los Alamos Neutron Science Center (LANSCE) and ISIS (Rutherford Appleton Laboratories, UK), to measure the resilience of different generations of GPUs. Our controlled neutron beam experiments provide additional deeper insights into the resilience characteristics of GPUs, that are not otherwise possible by the field data alone. We discuss the implications of our findings for future GPU architects, the design of GPU fault-injectors, and application programmers.

We believe that insights derived from our large-scale field data analysis and extensive neutron-beam experiments carry significant implications for future generation GPU architectures, current and future HPC computing facilities, researchers focusing on GPU resilience, and end users.

2. Background

This section provides a brief background on GPU architecture, resilience support, sources of GPU errors and their impact on the application execution.

2.1. GPU Architecture and Resilience Support

Fig 1 shows a simplified, representative GPU architecture for Kepler and Fermi generations. The GPU architecture is composed of multiple streaming multiprocessors (SM). The SMs are the basic building blocks of a GPU. All SMs have access to the shared L2 cache and the device memory. The thread block scheduler dispatches one or more *blocks of threads* to an idle SM. Each SM has multiple Compute Unified Device Architecture (CUDA) cores. A group of threads (called a *warp*) to be executed next on the CUDA cores of a given SM are selected by the warp scheduler, then instructions are dispatched by the instruction dispatch unit. Note that each CUDA core executes only one thread at a time. Each CUDA core has access to the shared memory and the L1 cache region. The Shared memory and the L1 cache regions are a dedicated resource for each SM. Similarly, each SM has a dedicated register file that can be accessed only by the threads executing in the same SM.

Table 1: GPU errors and its impact

GPU Error	XID	Impact
Single Bit Error (Corrected by the ECC) Silent data corruption may occur, if no ECC support.	–	No side effect on the program.
Double Bit Error (Detected by the ECC) Silent data corruption may occur, if no ECC support.	48	Program crash.
Off the Bus	–	Program crash
Display Engine error	56	Program crash
Error programming video memory interface	57	Program crash
Unstable video memory interface detected	58	Program crash
Internal micro-controller halt	62	Program crash
ECC page retirement error	63,64	Program crash
Video processor exception	65	Program crash

All major storage structures of GPUs for HPC applications are protected with a Single Error Correction Double Error Detection (SECDED) ECC including device memory, L2 cache, instruction cache, register files, shared memory, and L1 cache region. However, not all resources benefit from ECC protection, for example, logic, queues, the thread block scheduler, warp scheduler, instruction dispatch unit, and interconnect network are not covered. It was shown experimentally that a soft error in a storage structure is likely to affect multiple threads, possibly multiple warps or thread blocks [31]. Therefore, providing costly ECC protection to storage structures is expected to be a better return on investment. Additionally, the area occupied by scheduler structures are estimated to be much smaller than the cache and memory regions, hence the likelihood of a soft error striking in that region is much lower as well. However, the details of resilience support for these structure are considered business-sensitive by vendors, and hence, unavailable. Consequently, some of the application crashes due to soft-errors in non-protected structures may not be correctly logged by the system or may even be incorrectly attributed to other causes. We point out that our radiation test results include the effect of corruptions in these areas as well when estimating the overall soft-error rate. Therefore, in addition to the field data, our neutron beam tests are critical for developing a thorough understanding of the GPU errors.

2.2. GPU Errors and Their Impact

An application may encounter a GPU error due to multiple reasons, for example, an application bug, driver bug, hardware or radiation-induced bit corruptions. We note that analyzing

GPU errors due to an application or a runtime system can be misleading as an excessive number of errors may be occurring due to debugging runs, program bugs, bad programming practices, and script errors. Therefore, in this paper, we primarily focus on GPU application crashes that are caused by bit errors. Different GPU errors are logged with different codes (XID numbers as per Nvidia [3]).

Radiation strike may corrupt one or more bits leading to one of the following outcomes: (1) no effect on the program output (the failure is masked or corrupted data is not used), (2) single bit errors: corruption corrected by the ECC logic (correct program output), (3) program crash, (4) silent data corruption (incorrect output, but program does not crash). Out of these outcomes, only the latter two outcomes are undesirable from a programmer's point of view as they lead to program crash/incorrect output. There may be system integration related errors that may not be specific to GPU micro-architecture; "Off the bus" error is such an error considered in this study that is related to losing the connection to the host. It may be caused due to system integration issues and is not specific to the GPU micro-architecture or radiation strikes.

Table 1 lists all the GPU errors that were considered in the this study and their possible outcomes [3]. In rest of the paper, we refer to GPU errors that lead to application crash as *GPU failures*. Section 3 describes in detail how these events are collected under different data collection methods, and what the challenges and limitations of recording these events are.

3. Methodology

In this section, we first briefly describe the systems that were used in this study and our data collection methodology. Following that we describe the neutron beam experimental setup and methodology.

3.1. Data Sources and Data Collection Methodology

Oak Ridge Leadership Computing Facility (OLCF)
We collected GPU error logs from the Titan Supercomputer, hosted at the Oak Ridge Leadership Computing Facility (OLCF). Titan has a total of 18,688 K20X GPUs (Kepler G110 processor). The K20X GPU has 2688 CUDA cores, a total of 7.1 billion transistors on each GPU at the 28nm process technology. A single GPU is capable of delivering 3.95 Tflops single precision performance and 1.31 Tflops double precision performance. There are a total of 14 SMs and 192 CUDA cores within each SM. Each SM has 64K registers, 64KB of combined shared memory and L1 cache, and 48KB of read-only data cache. SMs share 1536 KB of L2 cache and a total 6GB GDDR5 memory. The register files, shared-memory, L1 and L2 caches are SECDED ECC protected, while read-only data cache is parity protected.

We have collected the GPU error data for over 18 months (since Feb'13 to Aug'14). The GPU errors were collected in two ways. First, we analyzed the data that were collected from the console log. The console log has a record for each GPU related event. We continuously parse the console log to filter GPU related records. These records include the node location, time-stamp and description of the event. Note that single bit errors are not logged to the console log. Therefore, we rely on the second method to collect information about single bit correctable errors. We collected this data by running *nvidia-smi* utility on all the GPU nodes. In addition to reporting the single bit errors, nvidia-smi output also includes double bit and ECC page retirement related errors. There are two key differences between these two data collection methods (console log and nvidia-smi). First, nvidia-smi utility provides a snapshot of the system (cumulative count of an event at a per-node granularity) unlike the console log that records events with timestamps at which they occurred. Second, the console log does not provide memory structure in which a particular event occurred, however, this information is provided by the nvidia-smi utility. Therefore, we use both these methods of data collection to quantify and analyze the characteristics of GPU errors.

Los Alamos National Laboratory (LANL)
We collected GPU error logs and GPU counter data from the Moonlight GPGPU cluster at the Los Alamos National Laboratory (LANL). Moonlight has a total of 616 M2090 GPGPUs (Fermi architecture). The M2090 GPU has 512 CUDA cores, a total of 3.0 billion transistors on each GPU at the 40nm process technology. A single GPU is capable of delivering 1.33 Tflops single precision performance and 0.66 Tflops double precision performance. There are a total of 16 SMs and 32 CUDA cores within each SM. Each SM has 32K registers, 64KB of combined shared memory and L1 cache. SMs share 768 KB of L2 cache and a total 6GB GDDR5 memory. The register files, shared-memory, L1 and L2 caches are SECDED ECC protected. Although relatively smaller in scale, the Moonlight cluster provides us an interesting data point – one of the first Nvidia GPUs to be adopted in the HPC domain (due to first time ECC support).

3.2. Neutron Beam Test Experimental Set Up

Radiation experiments were performed at the Los Alamos National Laboratory's (LANL) Los Alamos Neutron Science Center (LANSCE) Irradiation of Chips and Electronics House II and in the VESUVIO beam line in ISIS, Rutherford Appleton Laboratories, Didcot, UK. The experiments were conducted at different times of the years in 2013 and 2014. As shown in Fig. 2, both of these facilities provide a white neutron source that emulates the energy spectrum of the atmospheric neutron flux between 10 and 750 MeV. The ISIS beam has been empirically demonstrated to be suitable to mimic the LANSCE beam and the terrestrial radiation environment [43], despite its relatively lower component of high-energy neutrons. We also point out that the ISIS beam includes a non-negligible component of thermal neutrons, which may increase the measured error rate if Boron-10 is present in the tested devices.

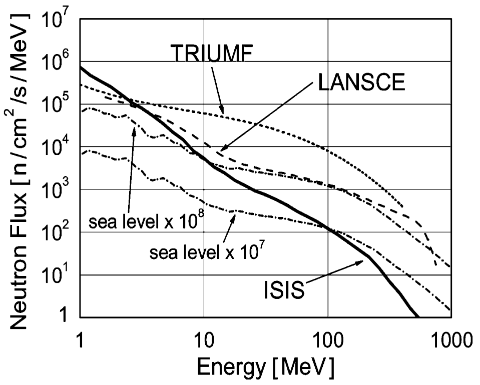

Figure 2: LANSCE, ISIS, and TRIUMF neutrons spectrum: plotted against neutrons spectrum at the sea level multiplied by 10^7 and 10^8 [43].

Therefore, in our discussions we discard thermal neutrons contribution and make a direct comparison only among experiments performed in the same facility and in the same time slot.

The neutron flux was approximately $1 \times 10^6 n/(cm^2 \times s)$ in LANSCE and $4 \times 10^4 n/(cm^2 \times s)$ in ISIS for energies above 10 MeV. The neutron fluxes used are higher than the neutron flux at sea level [20], but we have carefully designed the experiments to ensure that probability of more than one neutron generating a failure in a single code execution remains practically negligible. The observed error rates were lower than 10^{-2} errors/execution. Since a much lower neutron flux may hit a GPU in a realistic environment, it is highly likely to not have more than one corruption during one single execution. We can, therefore, scale the experimental data in the natural radioactive environment without introducing artificial behaviors.

The GPU hardware setup includes connecting the GPU to a host computer through a PCIe extender (Fig. 3). The role of the host computer is to initialize the test and gather the results from the GPU. A software and a hardware watchdog were included in the setup. The software watchdog monitors a time-stamp written by the application running on the GPU. If the time-stamp is not updated in ten seconds the GPU application is killed and launched again. Such a watchdog is required to detect and manage radiation-induced program crashes. The hardware watchdog is an Ethernet controlled switch that performs a power cycle of the host computer if the host computer itself does not acknowledge any ping requests in ten minutes. The hardware watchdog is necessary as radiation can corrupt the PCIe controller on the GPU board as well, possibly causing the host computer to hang. Irradiation was performed at room temperature with normal incidence and nominal voltages.

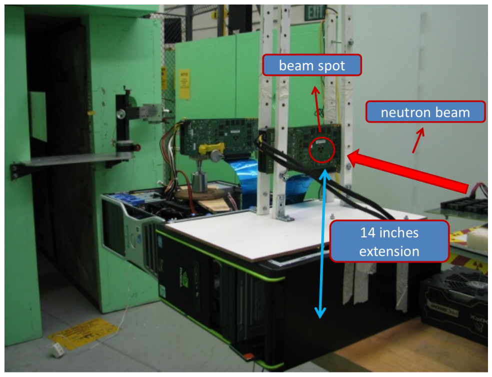

Figure 3: Radiation test setup inside the ICE House II, Los Alamos Neutron Science Center (LANSC), LANL. A similar setup was used at ISIS, Didcot, UK.

4. Understanding and Quantifying GPU Errors on the Titan Supercomputer

In this section, we present a detailed characterization of GPU failures, share interesting findings and implications for GPU architects and system operators.

Temporal characteristics of GPU failures

Fig. 4 shows the monthly-frequency of different types of GPU failures for the Titan supercomputer. First, we observe that the frequency of GPU related failure events is fairly low (typically occurring once in two days on an average). This is a significant result in the context of a such a large-scale system where more than two failures per day are likely to occur on an average, estimated using vendor-specified MTBF for the GPU card [2].

Second, we note that Off the bus, ECC page retirement errors and DBE failures are more dominant than other types of failures. We also point out that Off the bus failures were dominant only before the GPU production run (December 2013). A system integration issue with GPU cards was identified and

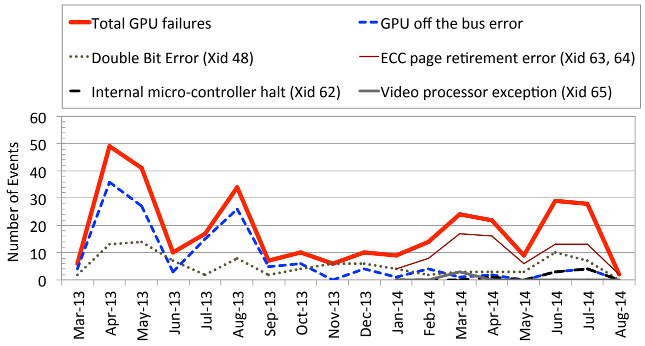

Figure 4: Monthly frequency of different types of GPU failures. Some errors (Table 1) that lead to program crash were observed to have zero error counts. ECC page retirement related errors were available only for the production run. GPU off the bus error is always followed by the micro-controller halt error after the recent driver upgrade.

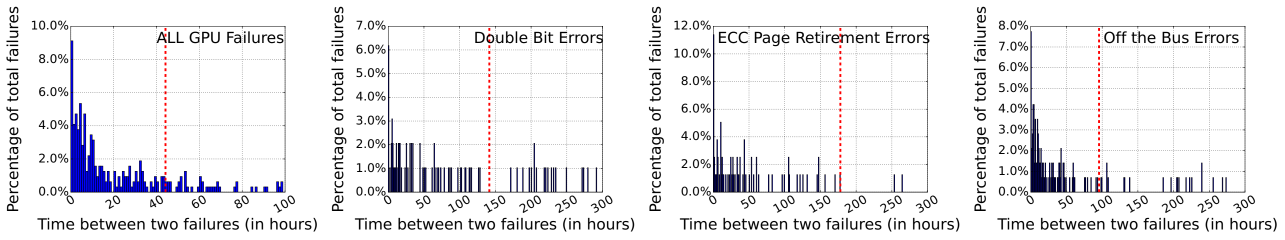

Figure 5: GPU failures exhibit temporal locality: These figures show the failure arrival time distribution. The dashed vertical line indicates the "observed" mean time between failure (MTBF). Multiple failures that occur beyond the x-axis limits are not shown here for clarity, but they contribute toward the MTBF calculation. Note that the combined MTBF is less than the MTBF for each individual types of failure.

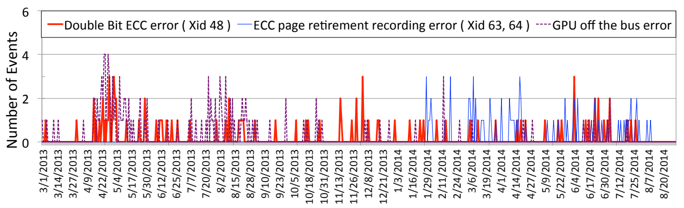

Figure 6: Frequency of daily occurrences for three dominant GPU failure types. This figure shows that the temporal locality is not an artifact of multiple errors occurring on a few number of days.

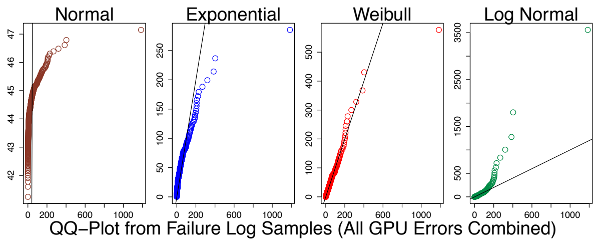

Figure 7: QQ-Plot for graphical representation of fitting different probability distribution functions (PDF). The quantiles drawn from the sample (failure log) are on the x-axis and y-axis shows theoretical quantiles. If the samples statistically come from a particular distribution function then points of QQ-plot fall on or near the straight line with slope=1. Each type of GPU failures show similar behavior (but not shown here).

GPU Failure Type	K-S test D-Statistics			Critical D-value	Weibull Shape Parameter
	Log Normal	Exp.	Weibull		
Double Bit Error	0.10	0.17	0.08	0.14	k = 0.70
Page Rtrmt Error	0.14	0.42	0.10	0.15	k = 0.49
Off the Bus Error	0.08	0.29	0.08	0.11	k = 0.55
All GPU Failures	0.06	0.23	0.04	0.08	k = 0.64

Figure 8: Result of Kolmogorov-Smirnov test (K-S test) for different types of GPU failure data. Null hypothesis that the samples for a given system come from a given probability distribution function is rejected at level 0.05 if k-s test's D-statistics is higher than the critical D-value. Comparison between D-statistics and critical D-value shows that Weibull distribution fits the best.

resolved by soldering the cards before the system went into production with GPUs. The mean time to application interruption (due to all failure events) in the production run is more than 40 hours, significantly higher than the estimated MTBF of the whole system (11.7 hours) using the vendor specified MTBF for the GPU card [2].

We perform rigorous stress tests and high-standard acceptance tests on these GPU cards before they go in production. The rigor helps the center's operations team identify bad GPU cards early enough. Even during a production run, if a GPU card exhibits a particular kind of error more than a small threshold, that particular GPU card goes under rigorous stress testing and is disqualified if the errors re-appear. As we show later as well, only a small fraction of "bad" GPU cards encounter most of the errors repeatedly, and hence, are enough to bring down the MTBF of the whole system significantly. Therefore, by doing this exercise continuously, we identify such cards early and consequently, increase the mean time to application interruption significantly.

Observation 1. *Our field data suggests that the current generation of GPUs deployed on the Titan supercomputer are fairly stable and experience failures at very low rate. We note that performing rigorous tests during the production phase and high-standard acceptance tests before the production phase helps us identify the bad cards early enough, and consequently, increases the mean time to application interruption significantly.*

While MTBF is a useful metric, it is not sufficient by itself to understand the characteristics of GPU related failures. To address this issue, we plot the inter-arrival failure distribution for GPU related failures (Fig. 5). Interestingly, a significant fraction of the failures occur much before the observed MTBF. This is true not only for all GPU failures combined, but also for dominant GPU failure types as well. These results indicate that there exists a strong temporal locality between GPU failures. This finding also implies that the average work lost due to a failure would be less, because a significant fraction of failures occur soon after a previous failure. However, this temporal locality characteristic is not artificially generated because a high number of failures occur on the same day or in the span of a couple of days. Fig. 6 shows that GPU failures do not show this kind of behavior, for example, there are only a couple of days during the observed period when more than two failures

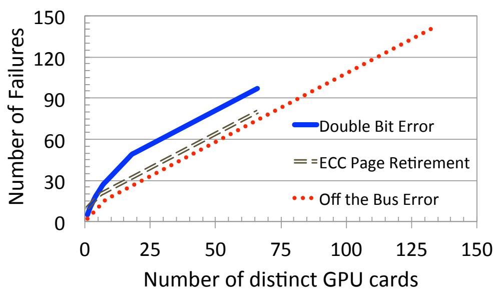

Figure 9: Number of distinct GPU cards affected by different types of GPU failures. A bend in the DBE curve suggests that a few card experience multiple DBEs.

of a type occurred on the same day.

Next, we mathematically capture the temporal locality characteristics of failures. We use two statistical techniques to fit our data against four distributions, normal, Weibull, log normal, and the exponential distribution. First, the QQ-plot visually shows the fitness of these distributions (Fig. 7). We notice that Weibull distribution is a better fit than exponential distribution for our failure samples. Second, Fig. 8 shows the results from the Kolmogorov-Smirnov test for different distributions [16] to reaffirm the same conclusion quantitatively. This result is particularly important for fault-tolerance studies and fault-injector tools that often assume that failure samples come from an exponential distribution – leading to incorrect estimation of the impact of failures, total execution time and checkpointing decisions.

In addition to the scale parameter (λ), a Weibull distribution is specified using a shape parameter (k). We find that the shape parameter (k) for the Weibull distribution is less than 1 for each type of failure (Fig. 8). If the shape parameter is less than one, it implies a high infant mortality rate, confirming our observation about temporal locality in failures. Due to this property, GPU programs can also take intelligent checkpointing techniques such as "Lazy checkpointing" and "Skip checkpointing" that exploit temporal locality in failures to reduce the I/O overhead significantly [42].

Observation 2. *We observed that a significant fraction of GPU failures occur much before the MTBF. This suggests that GPU failures have a strong temporal locality. This result is particularly important for reducing the I/O overhead significantly by employing techniques such as "Lazy checkpointing" [42]. This finding is also useful for fault-tolerance studies and tools that often do not take this characteristic into account – leading to incorrect estimation of impact of failures, total execution time and checkpointing decisions.*

Distinct number of cards affected by different failures

Next, we investigate how many GPU cards are affected by different GPU failure types. Fig. 9 shows that Off the bus errors have affected the most number of nodes (nearly 150 out of more than 18000 cards). Recall that almost all of these errors occurred in pre-production and were subsequently fixed by soldering. The ECC page retirement and DBE errors affect nearly half the numbers of GPU cards. Interestingly, Fig. 9 also shows that there are a few cards that experience double bit errors multiple times. In fact, six GPU cards are responsible for 25% of all DBE errors. There is only one card that exhibits multiple occurrences of ECC page retirement error. These cards were moved around the machine to see if a particular node location was more susceptible to such errors. Our results suggest that that some cards may be inherently more prone to double bit errors. We also found that one particular card was responsible for more than 10% of the ECC page retirement errors. Having learned from these trends, the Operations team takes proactive measures to identify such cards early and performs stress testing in a separate cluster before putting them back in the machine. This strategy has worked well for keeping the mean time to application interruption high. Due to this, we have seen relatively lower multiple occurrences of DBEs on the same card that would have occurred otherwise.

Observation 3. *Certain GPU cards may experience DBEs and ECC page retirement errors multiple times, motivating the Operations team to identify such cards early. Performing stress testing on these cards (on a separate cluster) before putting them back in production has proven to be an effective strategy for reducing the number of interruptions to applications.*

Temperature sensitivity of GPU failures

Next, we investigate if GPU failures have sensitivity towards temperature (Fig. 10). First, we observe that the combined GPU failures show sensitivity towards temperature, i.e., more occurrences in the hotter cages (chassis). But, not all failures individually show this trend (for example, double bit errors tend to occur least in the middle cage). However, as we had observed earlier, a few GPU cards experience multiple DBEs. This may possibly skew the sensitivity study. To address this issue, we counted only the first occurrences of DBEs on a given card. Our results show that double bit errors may be sensitive to temperature as well, while page retirement errors are not sensitive to temperature. However, we also recognize that establishing this correlation with high confidence is challenging because some cards themselves may be more prone to errors and variance in the temperature data may further complicate the problem. Additionally, the number of errors (e.g., double bit errors) are fairly small and hence, do not form a very large population size, making it more challenging to establish a correlation with high confidence. Nevertheless, we believe that our field data results may be useful and timely towards understanding this correlation better as the research community is still in the early stages of determining the effects of temperature on DBEs [4, 6].

Observation 4. *Our field data suggests that some GPU failures may be sensitive toward temperature (e.g., off the bus and double bit errors), however this is not the case for all failure types. Though our field data suggests that some GPU failures may exhibit sensitivity toward temperature, there is a need for more experimental evidence to establish the correlation between GPU errors and temperature with higher confidence.*

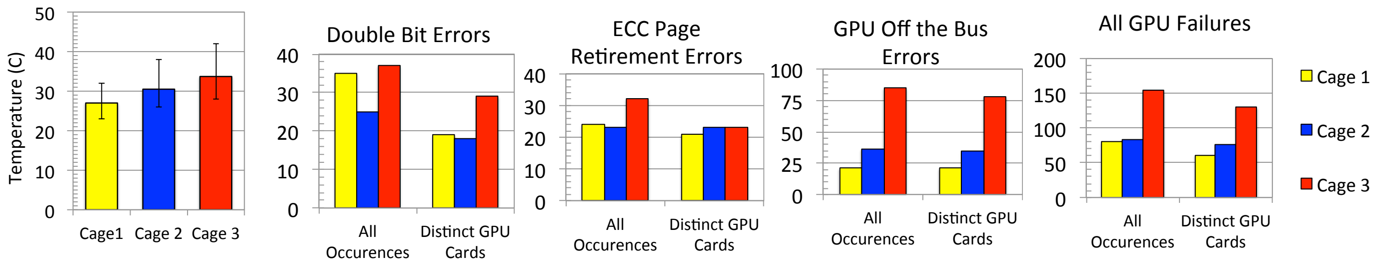

Figure 10: Temperature sensitivity of different GPU failures. Some GPU failure types tend to occur more in relatively hotter cages. "All Occurrences" counts all errors occurring in the same cage. However, in some cases, a single GPU card may see multiple errors. "Distinct GPU Cards" does not count repetitions of an error on the same card.

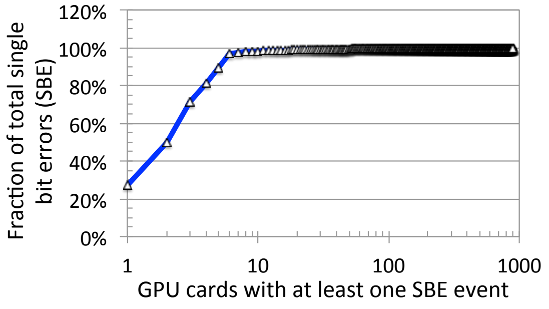

Figure 11: Single bit error affected GPU cards: Only 899 out of 18,688 cards encounter a SBE during the entire period.

Analysis of GPU Single Bit Errors (SBE)

Next, we study the characteristics of single bit errors (SBE) on the Titan supercomputer. Recall that if ECC is enabled on a system, SBE can be successfully corrected with some performance overhead. However, most of this performance overhead is due to the detection logic since the correction step only involves a few simple hardware-accelerated operations.

We found that less than 5% of the total GPU cards (899 cards) experienced one or more SBEs during Feb'13 to Aug'14. Interestingly, close to 98% of all SBEs occurred on only ten cards (out of 899 offender cards). This observation is also shown in Fig. 11. Like double bit errors, single bit errors tend to occur repeatedly on the same node, albeit this trend is much stronger in this case. We did not find any statistically sound correlation of SBEs with node or cage location, leading us to believe that some GPU cards may be more prone to single bit errors than others. Note that since SBEs do not interrupt an application and are corrected automatically. Recall that the SBE events are collected using the nvidia-smi utility which can provide only the snapshot of the system, and not continuous output to study temporal characteristics of SBEs in a statistically sound manner.

Observation 5. *Almost 98% of all single bit errors occur in only 10 GPU cards. This suggests that a few cards may be significantly more prone to re-occurrence of SBEs. Future studies modeling, simulating and evaluating the effect of single bit errors in large-scale system should take into account this characteristic of SBEs instead of simply assuming that the probability of an SBE occurring on all nodes is equal.*

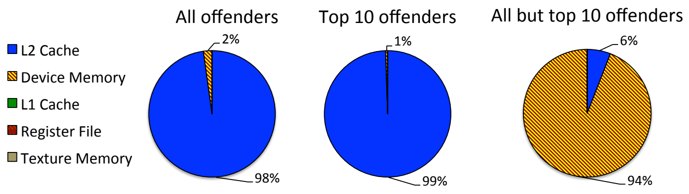

Figure 12: Breakdown of SBEs per storage structure: for GPU cards encountering at least one single bit error.

We further investigate the breakdown of SBE events in different regions on the GPU card. Fig. 12 shows the breakdown of all SBE events for all cards, for the 10 cards with the most number of SBEs, and for all cards except these top 10 offenders.

We found that overall the L2 Cache region is the major contributer in SBE events, showing that 98% of the SBE events happen in that structure alone. It is not surprising that the top 10 offenders also show the same behavior (99% of the SBE events occurred in the L2 Cache). However, it is interesting to note that once we eliminate the top 10 offenders, the device memory is the structure where most of the SBEs occur (96% of all SBEs). We also point out that the fraction of SBEs occurring in different structures is not proportional to the respective structure sizes. This finding could be used by future architects in deciding which memory structures may need better protection. Fault-injector tools could use this finding to account for the relative likelihood of soft-errors occurring in different memory structures and its corresponding impact on the application.

Observation 6. *GPU cards which experience most of the SBEs are likely to have all the SBEs occur in the device memory instead of the L2 cache. This finding can be used to identify the top offenders early on. Our results may also be useful for future architects in terms of which structures need better protection (device memory and L2 cache) and which structures may not need additional costly protection schemes (L1 cache, register file and texture memory).*

5. Experience with Moonlight GPU Cluster

In this section, we share our experience with Moonlight, a GPU-enabled HPC cluster at LANL. The GPUs deployed in Moonlight are the M2090 GPUs based on the Fermi architecture and enabled with the ECC support.

Performance Variation
Software developers and application users observed significant performance variation on these GPUs. Fig. 13(a) shows that two GPUs on the same node may exhibit significant performance variation. Each card achieves significantly different Gflops performance at different times. This is further supported by the observed variation in the performance-states of GPU cards (Fig. 13(b)). This behavior is particularly problematic for HPC workloads which heavily rely on global synchronization. Because of the global synchronization, the overall system performance is determined by the slowest GPU. Therefore, any variance in performance among GPU cards leads to an increase in application run time.

We suspect that power starvation or temperature variance is potentially causing this behavior (performance-state throttling and consequently, performance variation). Therefore, we conducted other tests on smaller testbeds at LANL using M2090 cards that were in different enclosures. However, we observed similar variance in performance and performance-states. Fortunately, this problem seems to have resolved in the recently purchased GPUs (K20, kepler architecture). Our results show less than $\pm 0.02\%$ of performance variation, indicating the maturity of GPU architecture in this particular regard.

Inconsistency in Error Logging
As discussed in Section 3, GPU error counts can be obtained via either querying the nvidia-smi tool or mining syslog. In our experiments, we observed that logged errors (as per syslog) may not always exactly match with the nvidia-smi output.

We increased the temperature of a M2090 GPU node by blocking the exhaust, and hence, increasing the probability of bit corruptions. We recall from our previous discussion related to GPU cards on the Titan supercomputer that the probability of bit errors may increase with temperature-rise. Due to increased temperature, we could observe multiple bit corruptions. However, there were several inconsistencies between syslog and nvidia-smi output. In another dedicated experiment, we ran vendor-provided HPL tests consuming 279 GPU node-hours. We observed 27 double bit errors reported in syslog but only one was captured by the nvidia-smi output. One inconsistent logging issue was observed with a GPU card on Titan too. We suspect that this is because DBE forces the node to go down, and the driver may not correctly log the error to the persistent storage on the host just-in-time, before the host loses contact or becomes unresponsive.

Observation 7. *Performance variations across GPU cards seem to be fixed on newer generation of GPU cards – making GPUs more amenable to HPC workloads. However, inconsistency in error logging may not have been completely eliminated. This is critical as system administrators often rely on error counters to monitor the health of the system.*

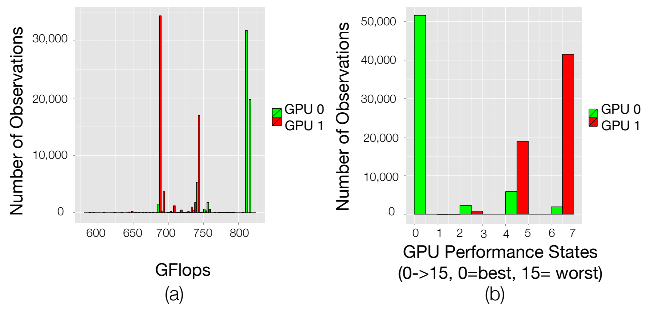

Figure 13: Performance variation on Moonlight's GPU node with two M2090 GPU cards: histogram of observed GFlops (a), histogram of observed performance-states (b).

6. Radiation Experiments

In this section, we describe our neutron-beam experiments, results, and implications. First, we measure and report the raw sensitivity of the GPU devices from different architecture generations. Second, we evaluate the silent data corruption and program crash rate for different HPC benchmarks under neutron flux (refer to Section 3 for experimental setup).

Raw sensitivity of the GPU memory structures

We tested the raw sensitivity of the GPU memory structures for two GPUs: Fermi architecture based C2050 (similar to one used in the LANL study) and Kepler architecture based K20 (similar to the Titan's GPU). Since our goal for this part of the study is to measure the raw sensitivity of the SRAM structures of the GPUs, we disabled built-in ECC mechanism.

To measure the raw sensitivity of these devices, we store a particular pattern (all $1s$ or all $0s$) in the structure, expose the device to a controlled high-energy neutron flux and, finally, check if radiation corrupted the initially stored values.

We experimentally measured the *cross section*, a widely used metric to evaluate the radiation sensitivity of a device [5], for the register file cells and the L2 cache cells of both K20 and C2050 boards. The cross section is calculated by dividing the number of observed errors by the received neutron fluence ($\frac{neutrons}{cm^2}$). If the cross section of the memory structure is divided by the number of the cells, it yields the average sensitive area of a single cell, i.e., the portion of the cell that, if hit by a neutron, will produce a failure. Higher cross section implies higher probability for an impinging neutron to corrupt a bit.

Figure 14(a) shows the *per-bit* cross section for L2 cache and register file normalized to the cross section of K20 L2 cache obtained with the $1s$ pattern. The 95% confidence intervals for our results, which is a combination of neutron counts uncertainty and statistical error, are lower than 12% for all the reported values. Note that the results reported in Figure 14(a) are normalized to the number of available bits, however, the K20 structures are significantly larger than the C2050 structures. Figure 14(b) shows the *total-area* cross section for the K20 and C2050 L2 cache and register file, obtained by multi-

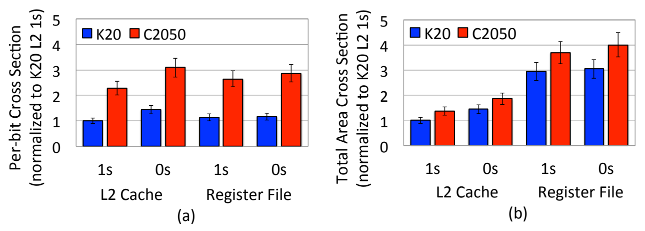

Figure 14: Normalized cross sections for the K20 and C2050 structures: per-bit (a) and total-area (b). Lower is better. The K20 is less prone to bit corruptions than C2050.

plying the *per-bit* cross section with the size of the respective memory structure.

Figure 14 shows that the Kepler generation GPU has better reliability than Fermi generation GPU per bit as well as for the whole L2 cache and register area. This is an encouraging result, because at lower feature sizes (newer process technology), the reliability can get worse [11]. The observed improvement can be attributed to various factors, including a better cell design in Kepler architecture. Unfortunately, the details of cell design are considered business-sensitive. However, this indicates that architects have improved the cell design to combat the danger of increased sensitivity at lower feature sizes.

We also found that bits set to zero are 40% more prone to corruption than bits in the L2 caches for both generations of GPUs. However, this is not true for bits in the register file (Figure 14(a)). This is due to the intrinsic asymmetries to the cache cell design. Some memory cell designs are not symmetric, meaning that the pull-up and pull-down transistor capacitance may differ. Under radiation, this results into a different radiation sensitivity for bits set to different values. We point out that this specific result can be achieved only through radiation experiments and is fundamental to precisely evaluating the resilience of GPUs.

Observation 8. *Overall, K20 architecture is less prone to radiation-induced bit corruptions due to its improved cell design, despite its significantly bigger structure sizes. However, the probability of bit corruption highly depends on the initial value of the bit and the memory structure. Current fault-injection tools and techniques do not account for this particular characteristic. Future research studies should model and simulate this behavior correctly to capture realistic soft-error characteristics. GPU architects should focus on mitigating this behavior.*

Next, we investigate what fraction of all neutron-induced errors result into double and triple bit errors. Recall that SECDED mechanism can not correct double or detect triple bit errors. We found that, fortunately, only 4-6% errors are double bit errors in L2 cache and register file area for the K20 card – suggesting that current SECDED mechanism is a cost-effective choice for the current generation of GPU cards. Our Titan data also suggested a very low rate of DBEs in the field. Interestingly, C2050 GPU card exhibits even lower (close to 1%) DBEs. This can be explained due to difference in the process technology. Due to smaller transistor size at 28nm (K20 card), it is easier for a neutron to interact with more than one transistor in the K20 card than in the C2050 card, possibly generating more DBEs. A lower rate of DBEs is also typically achieved via radiation-aware intelligent memory interleaving [19]. No errors with more than 2 bits corruption were observed during the experiments.

Observation 9. *While K20 has a significantly higher DBE rate than the C2050 due to smaller transistor size (hence, higher likelihood of one neutron interacting with multiple bits), the absolute DBE rate remains fairly low (6%) and no triple bit corruptions were observed. This suggests that SECDED ECC mechanism should suffice for correcting most of the radiation induced single bit corruptions.*

Evaluating Radiation Sensitivity of HPC Benchmarks

To evaluate the impact of radiation-sensitivity on real-world HPC applications, we chose a representative set of benchmarks. We point out that HPC workloads running on the Titan supercomputer are considered sensitive. Since we cannot run those workloads under the neutron beam experimental set up, we have evaluated a wide variety of benchmarks that have similar characteristics as the real-workloads on Titan supercomputer. In fact, many real-world workloads often use the algorithms implemented in these benchmarks as their kernel. Our chosen benchmarks [7, 24, 31] cover a wide range of computational and data movement requirements (Table 2).

The described benchmarks were experimentally tested on NVIDIA K20s in ISIS and LANSCE neutron sources with the setup described in Section 3.2. To detect if a an execution is affected by the Silent Data Corruption (*SDC*), the desktop PC compares the program output with a pre-computed golden output and checks for mismatches. A program crash (simply, referred to as *Crash*) is detected when the program crashes or when GPU gets stuck requiring the killing of the application or a system reboot. SDC and Crash are generated by radiation-induced errors in the GPU memory, logic, or scheduler resources [44].

The benchmark cross section is measured as the observed error rate ($\frac{errors}{s}$, separately for the SDC and Crash) divided by the average particles flux ($\frac{particles}{cm^2 \times s}$), yielding an area. The execution time is normalized when measuring the cross section, so that the cross section has no dependence on the computational time, but only on the amount of resources required for computation. The experimentally observed cross section is an intrinsic characteristic of the device and benchmark, independent on the neutron source. Multiplying the cross section (cm^2) with the expected neutron flux on the GPU ($13 \frac{n}{cm^2 \times h}$ at sea level), one can estimate the GPU error rate or Failure In Time (FIT), expressed as $\frac{errors}{h}$. Table 2 shows the experimental results with a 95% confidence interval, which includes both statistical error and neutron counts uncertainty, for all the benchmarks. We also evaluate the efficiency of built-in ECC

Table 2: Benchmarks details, and FIT at NYC of the different benchmarks (ECC enabled only for the last 2 rows).

	Input	Output	Instr. Exec.	SDC Cross Section	SDC FIT	Crash Cross Section	Crash FIT
lavaMD	$1,098,500$	$878,800$	111.3 B	$(2.65 \pm 0.40) \times 10^{-7}$	$(3.44 \pm 0.52) \times 10^{3}$	$(1.17 \pm 0.18) \times 10^{-7}$	$(1.52 \pm 0.23) \times 10^{3}$
FFT	$131,072$	$131,072$	30.6 B	$(6.72 \pm 1.01) \times 10^{-8}$	$(8.74 \pm 1.31) \times 10^{2}$	$(4.69 \pm 0.73) \times 10^{-8}$	$(6.09 \pm 0.71) \times 10^{2}$
dgemm	$33,554,432$	$16,777,216$	21.8 B	$(6.59 \pm 1.06) \times 10^{-8}$	$(8.57 \pm 1.29) \times 10^{2}$	$(6.27 \pm 0.95) \times 10^{-8}$	$(8.15 \pm 1.22) \times 10^{2}$
MxM	$33,554,432$	$16,777,216$	20.4 B	$(5.83 \pm 0.87) \times 10^{-8}$	$(7.57 \pm 1.14) \times 10^{2}$	$(3.34 \pm 0.51) \times 10^{-8}$	$(4.34 \pm 0.65) \times 10^{2}$
Hotspot	$2,097,152$	$1,048,576$	266.4 M	$(1.57 \pm 0.24) \times 10^{-8}$	$(2.04 \pm 0.31) \times 10^{2}$	$(8.63 \pm 1.29) \times 10^{-9}$	$(1.12 \pm 0.17) \times 10^{2}$
MTrans	$4,194,304$	$4,194,304$	88.1 M	$(2.89 \pm 0.43) \times 10^{-9}$	$(3.75 \pm 0.56) \times 10^{1}$	$(1.55 \pm 0.23) \times 10^{-10}$	$(2.02 \pm 0.30) \times 10^{0}$
MxM ECC	$33,554,432$	$16,777,216$	20.4 B	$(5.67 \pm 1.02) \times 10^{-9}$	$(7.38 \pm 1.32) \times 10^{1}$	$(5.09 \pm 0.92) \times 10^{-8}$	$(6.62 \pm 1.19) \times 10^{2}$
Hotspot ECC	$2,097,152$	$1,048,576$	266.4 M	$(3.49 \pm 0.52) \times 10^{-9}$	$(4.53 \pm 0.68) \times 10^{1}$	$(1.90 \pm 0.29) \times 10^{-8}$	$(2.47 \pm 0.37) \times 10^{2}$

support. When the ECC is enabled, the number of output errors is significantly reduced – indicating the efficiency of ECC support. But, it also significantly increases the challenge of gather enough experimental data. Therefore, we report the results of two benchmarks with ECC support enabled for which we could gather statistically significant amount of data. First, we discuss the results for all benchmarks without ECC support, followed by our findings about implications and efficiency of ECC support.

We observed significant variance in the FIT rates across applications for both the SDC and Crash. The underlying reason is differences in the GPU resources utilization and intrinsic code characteristics like the Architectural Vulnerability Factor (AVF, i.e., the probability for a failure to propagate to the output [13]), amount of data elaborated, and the number of executions performed. We note that amount of data elaborated alone is not sufficient to indicate the program's neutron-induced error rate. For example, *dgemm* and *MxM* elaborate 3x more data than *lavaMD* but have 4x lower SDC FIT. On the other hand, the higher occurrences of program crashes in *dgemm* and *FFT* can be explained by the fact that crashes related to the number of instructions executed by a thread that, if corrupted, lead to a control flow error. The difference among *dgemm* and *MxM* is of particular interest as they solve the same problem on the same data but with two opposite approaches (computing-bounded *vs.* memory bounded). Since elaborated data are the same, their SDC FIT also remain comparable. However, being computing-bounded, *dgemm* execution requires the GPU cores to be always busy and fully loaded while during *MxM* execution the GPU cores are often waiting for data transfer. It is then more likely for *dgemm* to be affected by a corruption that leads to a control-flow error, possibly causing the *dgemm* Crash FIT rate to be twice higher than that of *MxM*.

Our results for both *Hotspot* and *MxM* benchmarks show that ECC is efficient in reducing the FIT rate significantly, in accordance with our previous findings where we observed that DBEs rate fairly low in the field as well as under the neutron beam. Interestingly, SDC FIT rate are reduced significantly when ECC is enabled; *MxM* SDC rate is reduced by one order of magnitude, while *Hotspot* SDC rate is four times lower. We point out that remaining SDC errors are due to corruptions in logic resources or scheduler as they are not protected by the ECC.

Observation 10. *ECC reduces the Silent Data Corruption rate up to one order of magnitude. However, a non-negligible amount of SDC may still occur due to errors in unprotected areas (including queues, flip-flops, logics, and schedulers). GPU architects should focus on improving the reliability of these unprotected structures to further reduce the SDC rate.*

Interestingly, we observe that while ECC can decrease the overall FIT rate (SDC and Crash combined), it may actually increase the Crash FIT. For example, Crash FIT for *Hotspot* is almost doubled. The underlying reason is relatively more complex than reasoning about the observed decrease in SDC FIT rate. When ECC mechanism detects a double bit corruption, it launches an exception that eventually leads to program crash (see Table1). Recall our Observation 9 that the number of double bit corruptions is about an order of magnitude smaller than single bit corruptions. However, that does not directly imply that the Crash rate will be necessarily decreased by an order of magnitude when ECC is enabled. We observed that a large portion of single-bit and double-bit errors are benign and do not affect the program output. Hence, they do not contribute towards SDC and Crash rate when ECC is disabled. But, when ECC is enabled, all the double bit errors (including the benign ones) are converted into program crashes. Even *benign* double bit errors cause program crash when ECC is enabled, causing the crash FIT rate to go up. However, increase in program crash rate is not necessarily undesirable. In fact, that shows ECC is effective in decreasing the more undesirable outcome (SDC) – critical for long running HPC workloads that often do not have mechanism in place to detect SDC, but can recover from program crashes by reading in the last saved checkpoint.

Observation 11. *ECC may increases the program crash rate due to the fact that many benign double bit corruptions, that did not result in SDC or crash, are being detected by ECC and result in program crash. This finding is useful for others to identify double bit errors that may actually be benign. If such corruptions could be predicted successfully, the program does not need to crash proactively and incur high I/O overhead due to reading previously saved checkpoint from the parallel file system [39].*

Next, we scale the FIT rate of these benchmarks to Titan's scale. Notice that we can only compare the Crash FIT, since SDC for the Titan system can not be measured. Also, recall that SDC rates are reduced by an order of magnitude when ECC is enabled. Crash FIT rates, when scaled, results in the mean time to application interruption of approx. 85 hours (ECC enabled) and 35 hours (ECC disabled) for lavaMD application. Recall that lavaMD benchmark experiences an order of magnitude higher FIT rate than other applications. Other applications are estimated to be interrupted over 100 hours on average. These results align well with our field observed MTBF for different GPU errors, for example MTBF for DBE was observed 144 hours on Titan and combined MTBF of all types of GPU failures was approximately 44 hours (Fig. 4 and 5). Our radiation experiments indicate that Kepler generations of GPU cards perform well under the controlled radiation tests and our results matches with what we observe in the field.

Observation 12. *Radiation experiments and field data from two Supercomputing facilities align well and indicate that Kepler generation of GPU are significantly more resilient than Fermi generation of GPUs.*

7. Related Work

System failures and their characteristics has been studied extensively in the past to dissect the reliability of large-scale systems in general [9, 12, 21, 22, 25, 28, 35, 36]. There are more focused studies such as [18, 38, 40] on DRAM failures and [37] on disk failures. Our work is the first study focused specifically on GPU failures on large-scale systems. A recent work on the Blue waters system [9], primarily focusing on the characterizing all system failures, reports only the number of uncorrectable errors on GPU memory. This study provides a more detailed and rich characterization both in terms of system size and time, accompanied with neutron beam tests. We also present several interesting findings and implications of various different kinds of GPU errors for architects and system operators.

Recently, there have been extensive scientific efforts studying and improving GPU reliability, motivated by the spread of GPUs in both HPC and safety-critical applications [15]. Some initial work in the HPC domain have started to look at reliability evaluation of GPUs exposed to terrestrial radiation [8, 30, 33]. At the same time, fault injection experiments have been performed to track error propagation towards the GPU outputs and evaluate the Architectural Vulnerability Factor (AVF) of several parallel codes [13, 14, 17, 41]. Some recent works have focused on evaluation of the parallel code robustness to soft errors [10, 32, 34]. Experimentally-tuned and optimized hardening strategies for a limited set of algorithms have been investigated as well [26, 29, 31].

In contrast to these efforts, this is first study that provides radiation experiments for a variety of workloads and evaluates the efficiency of ECC on these codes. No prior work has combined the field study and radiation tests to understand the GPU failures in a unified way. Our study provides insights about differences in the reliability of memory. If these insights are taken into account, it can make fault injection tools/models more realistic and accurate. The radiation experiments presented here ensure that all resources are exposed and were carefully conducted to ensure that all resources can be corrupted. This is also the first study to provide operational insights and recommendations for current and future large-scale GPU-enabled HPC centers.

8. Conclusion

We present an in-depth study of GPU failures on large-scales system and derive insights about GPU error characteristics that can be used to improve operational efficiency of large-scale HPC facilities. Our insights related to the failure characteristics based on raw sensitivity of the GPU memory structures tested using neutron-beam experiments can be incorporated into the future failure/soft-error modeling, simulation and tool frameworks. Overall, we believe that insights derived from our large-scale field data analysis and neutron-beam experiments carry significant implications for future generation GPU architectures and exascale systems.

9. Acknowledgment

This manuscript used the resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at the Oak Ridge National Laboratory, which is managed by UT Battelle, LLC for the U.S. Department of Energy, under the contract No. DEAC05-00OR22725.

This manuscript has been authored by UT-Battelle, LLC, under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non- exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

Part of this work was performed at the Ultrascale Systems Research Center (USRC) at Los Alamos National Laboratory, supported by the U.S. Department of Energy contract DE-FC02-06ER25750. The publication has been assigned the LANL identifier LA-UR-15-20004.

The authors would like to thank Stephen Keckler, Timothy Tsai, and Siva Hari for providing K20 boards for the radiation tests. The authors would also like to thank Heather Quinn, Thomas Fairbanks, and Christopher Frost for their help with the radiation tests.

References

[1] "Computational science requirements for leadership computing, 2007, http://www.olcf.ornl.gov/wp-content/uploads/2010/03/ORNL_TM-2007_44.pdf."

[2] "Tesla k20 gpu accelerator, board specification," http://www.nvidia.com/content/PDF/kepler/Tesla-K20-Passive-BD-06455-001-v07.pdf.

[3] "Understanding xid errors," http://docs.nvidia.com/deploy/xid-errors/index.html.

[4] M. Bagatin, S. Gerardin, A. Paccagnella, C. Andreani, G. Gorini, and C. Frost, "Temperature dependence of neutron-induced soft errors in srams," *Microelectronics Reliability*, vol. 52, no. 1, pp. 289–293, 2012.

[5] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *Device and Materials Reliability, IEEE Transactions on*, vol. 5, no. 3, pp. 305–316, Sept 2005.

[6] G. Bruni, "Temperature effects on soft error rate due to atmospheric neutrons on 28 nm fpgas," 2014.

[7] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The scalable heterogeneous computing (shoc) benchmark suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, ser. GPGPU '10. New York, NY, USA: ACM, 2010, pp. 63–74. [Online]. Available: http://doi.acm.org/10.1145/1735688.1735702

[8] N. DeBardeleben, S. Blanchard, L. Monroe, P. Romero, D. Grunau, C. Idler, and C. Wright, "GPU Behavior on a Large HPC Cluster," *6th Workshop on Resiliency in High Performance Computing (Resilience) in Clusters, Clouds, and Grids in conjunction with the 19th International European Conference on Parallel and Distributed Computing (Euro-Par 2013), Aachen, Germany,*, August 26-30 2013.

[9] C. Di Martino, F. Baccanico, W. Kramer, J. Fullop, Z. Kalbarczyk, and R. Iyer, "Lessons learned from the analysis of system failures at petascale: The case of blue waters," *44th international*.

[10] C. Ding, C. Karlsson, H. Liu, T. Davies, and Z. Chen, "Matrix multiplication on gpus with on-line fault tolerance," in *Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium on*, May 2011, pp. 311–317.

[11] A. Dixit and A. Wood, "The impact of new technology on soft error rates," in *Reliability Physics Symposium (IRPS), 2011 IEEE International*. IEEE, 2011, pp. 5B–4.

[12] N. El-Sayed and B. Schroeder, "Reading between the lines of failure logs: Understanding how hpc systems fail, DSN," 2013.

[13] B. Fang, K. Pattabiraman, M. Ripeanu, and S. Gurumurthi, "Gpu-qin: A methodology for evaluating the error resilience of gpgpu applications," 2014.

[14] B. Fang, J. Wei, K. Pattabiraman, and M. Ripeanu, "Poster: Evaluating error resiliency of gpgpu applications," in *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, Nov 2012, pp. 1504–1504.

[15] L. A. B. Gomez, F. Cappello, L. Carro, N. DeBardeleben, B. Fang, S. Gurumurthi, S. Keckler, K. Pattabiraman, R. Rech, and M. S. Reorda, "Gpgpus: How to combine high computational power with high reliability," in *2014 Design Automation and Test in Europe Conference and Exhibition*, Dresden, Germany, 2014.

[16] D. M. H. and M. J. Schervish, "Probability and statistics. 3rd ed. boston, ma: Addison-wesley, 2002."

[17] I. Haque and V. Pande, "Hard Data on Soft Errors: A Large-Scale Assessment of Real-World Error Rates in GPGPU," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, 2010, pp. 691–696.

[18] A. A. Hwang, I. A. Stefanovici, and B. Schroeder, "Cosmic rays don't strike twice: understanding the nature of dram errors and the implications for system design," *ACM SIGPLAN Notices*, vol. 47, no. 4, pp. 111–122, 2012.

[19] E. Ibe, S. Chung, S. Wen, H. Yamaguchi, Y. Yahagi, H. Kameyama, S. Yamamoto, and T. Akioka, "Spreading diversity in multi-cell neutron-induced upsets with device scaling," in *Custom Integrated Circuits Conference, 2006. CICC'06. IEEE*. IEEE, 2006, pp. 437–444.

[20] JEDEC, "Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices," JEDEC Standard, Tech. Rep. JESD89A, 2006.

[21] Y. Liang, Y. Zhang, M. Jette, A. Sivasubramaniam, and R. Sahoo, "Bluegene/l failure analysis and prediction models," in *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*. IEEE, 2006, pp. 425–434.

[22] Y. Liang, Y. Zhang, A. Sivasubramaniam, R. K. Sahoo, J. Moreira, and M. Gupta, "Filtering failure logs for a bluegene/l prototype," in *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*. IEEE, 2005, pp. 476–485.

[23] R. Lucas, "Top ten exascale research challenges," in *DOE ASCAC Subcommittee Report*, 2014.

[24] NVIDIA, "uBLAS Library User Guide," http://docs.nvidia.com/cuda/pdf/CUBLAS_Library.pdf, 2014.

[25] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on*. IEEE, 2007, pp. 575–584.

[26] D. A. G. Oliveira, P. Rech, L. L. Pilla, P. O. A. Navaux, and L. Carro, "Gpgpus ecc efficiency and efficacy," in *International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT 2014)*, 2014.

[27] "Preparing for Exascale: ORNL Leadership Computing Facility Application Requirements and Strategy, 2009, http://www.olcf.ornl.gov/wp-content/uploads/2010/03/olcf-requirements.pdf."

[28] A. Pecchia, D. Cotroneo, Z. Kalbarczyk, and R. K. Iyer, "Improving log-based field failure data analysis of multi-node computing systems," in *Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on*. IEEE, 2011, pp. 97–108.

[29] L. Pilla, P. Rech, F. Silvestri, C. Frost, P. Navaux, M. Reorda, and L. Carro, "Software-based hardening strategies for neutron sensitive fft algorithms on gpus," *Nuclear Science, IEEE Transactions on*, vol. PP, no. 99, pp. 1–7, 2014.

[30] P. Rech, C. Aguiar, R. Ferreira, C. Frost, and L. Carro, "Neutron radiation test of graphic processing units," in *On-Line Testing Symposium (IOLTS), 2012 IEEE 18th International*, June 2012, pp. 55–60.

[31] P. Rech, C. Aguiar, C. Frost, and L. Carro, "An Efficient and Experimentally Tuned Software-Based Hardening Strategy for Matrix Multiplication on GPUs," *Nuclear Science, IEEE Transactions on*, vol. 60, no. 4, pp. 2797–2804, 2013.

[32] ——, "Experimental evaluation of thread distribution effects on multiple output errors in gpus," in *Test Symposium (ETS), 2013 18th IEEE European*. IEEE, 2013, pp. 1–6.

[33] P. Rech, L. Carro, N. Wang, T. Tsai, S. K. S. Hari, and S. W. Keckler, "Measuring the Radiation Reliability of SRAM Structures in GPUS Designed for HPC," in *IEEE 10th Workshop on Silicon Errors in Logic - System Effects (SELSE)*, 2014.

[34] D. Sabena, M. Sonza Reorda, L. Sterpone, P. Rech, and L. Carro, "On the evaluation of soft-errors detection techniques for gpgpus," in *Design and Test Symposium (IDT), 2013 8th International*, Dec 2013, pp. 1–6.

[35] R. K. Sahoo, M. S. Squillante, A. Sivasubramaniam, and Y. Zhang, "Failure data analysis of a large-scale heterogeneous server environment," in *Dependable Systems and Networks, 2004 International Conference on*. IEEE, 2004, pp. 772–781.

[36] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance computing systems," *Dependable and Secure Computing, IEEE Transactions on*, vol. 7, no. 4, pp. 337–350, 2010.

[37] B. Schroeder and G. A. Gibson, "Disk failures in the real world: What does an mttf of 1, 000, 000 hours mean to you?" in *FAST*, vol. 7, 2007, pp. 1–16.

[38] B. Schroeder, E. Pinheiro, and W.-D. Weber, "Dram errors in the wild: a large-scale field study," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1. ACM, 2009, pp. 193–204.

[39] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson *et al.*, "Addressing failures in exascale computing," *International Journal of High Performance Computing Applications*, p. 1094342014522573, 2014.

[40] V. Sridharan, J. Stearley, N. DeBardeleben, S. Blanchard, and S. Gurumurthi, "Feng shui of supercomputer memory: positional effects in dram and sram faults," in *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2013, p. 22.

[41] J. Tan, N. Goswami, T. Li, and X. Fu, "Analyzing soft-error vulnerability on gpgpu microarchitecture," in *Workload Characterization (IISWC), 2011 IEEE International Symposium on*, Nov 2011, pp. 226–235.

[42] D. Tiwari, S. Gupta, and S. S. Vazhkudai, "Lazy checkpointing: Exploiting temporal locality in failures to mitigate checkpointing overheads on extreme-scale systems," *International Conference on Dependable Systems and Networks (DSN)*, 2014.

[43] M. Violante, L. Sterpone, A. Manuzzato, S. Gerardin, P. Rech, M. Bagatin, A. Paccagnella, C. Andreani, G. Gorini, A. Pietropaolo, G. Cardarilli, S. Pontarelli, and C. Frost, "A New Hardware/Software Platform and a New 1/E Neutron Source for Soft Error Studies: Testing FPGAs at the ISIS Facility," *Nuclear Science, IEEE Transactions on*, vol. 54, no. 4, pp. 1184–1189, 2007.

[44] J. F. Ziegler and H. Puchner, *SER–history, Trends and Challenges: A Guide for Designing with Memory ICs*. Cypress, 2010.