A Large-Scale Study of Soft-Errors on GPUs in the Field

Bin Nie*, Devesh Tiwari†, Saurabh Gupta†, Evgenia Smirni*, and James H. Rogers†
*College of William and Mary, †Oak Ridge National Laboratory

Abstract

Parallelism provided by the GPU architecture has enabled domain scientists to simulate physical phenomena at a much faster rate and finer granularity than what was previously possible by CPU-based large-scale clusters. Architecture researchers have been investigating reliability characteristics of GPUs and innovating techniques to increase the reliability of these emerging computing devices. Such efforts are often guided by technology projections and simplistic scientific kernels, and performed using architectural simulators and modeling tools. Lack of large-scale field data impedes the effectiveness of such efforts. This study attempts to bridge this gap by presenting a large-scale field data analysis of GPU reliability. We characterize and quantify different kinds of soft-errors on the Titan supercomputer's GPU nodes. Our study uncovers several interesting and previously unknown insights about the characteristics and impact of soft-errors.

1. Introduction

Parallelism provided by the GPU architecture has enabled domain scientists to simulate physical phenomena at a much faster rate and finer granularity than what was previously possible by CPU-based large-scale clusters. Scientists are already benefiting from the GPU deployments in large-scale computing systems such as the Titan supercomputer, the Blue Waters supercomputer, and the Keeneland cluster [1, 22, 25, 36]. Recognizing the performance and energy-efficiency benefits of GPUs, next generation pre-exascale supercomputers are also expected to continue taking advantage of parallelism provided by GPUs [4]. Given the challenges of power provisioning for exascale systems, GPUs will continue to be an attractive choice due to their performance per watt characteristics that are better compared to their CPU counterparts [21].

Given the technology-trends and wide-spread adoption of GPUs, many researchers have studied the performance and energy-efficiency aspects of GPU-based applications in detail. Architecture researchers have been investigating reliability characteristics of GPUs and ways to increase the reliability of these emerging computing devices. Such efforts are often guided by technology projections and simplistic scientific kernels, and are performed using architectural simulators and modeling tools. Lack of large-scale field data impedes the effectiveness of such efforts. Only recently, researchers have started to investigate the reliability characteristics of GPUs using large-scale field data [8, 33, 34]. In particular, these studies have quantified the hardware and manufacturing related failures of GPUs, firmware/application-related GPU errors, and the improvement in the resilience of GPU architecture over generations. Nevertheless, there is a limited understanding about the soft-errors on GPUs at large-scale, their impact on performance, the correlation between user jobs and GPU errors, and their relation with GPU resource utilization. The goal of this paper is to improve our understanding in these aspects and identify opportunities for future GPU system design and better resource management.

Unfortunately, developing such an understanding is challenging for multiple reasons. First, there are often multiple factors responsible for different types of GPU errors, making it hard to distill their cause and their impact on applications. Second, it is hard to study the correlation or impact of applications on GPU reliability characteristics or resource utilization since we do not have access to the end-users' application-base. Third, we often do not have control over several factors such the power/cooling conditions, user behavior or node-assignments to different jobs. This makes the development of an accurate understanding of the GPU errors more challenging. Despite these challenges, this work attempts to improve our current understanding about GPU reliability at-scale while carefully considering these challenges.

In this work, we specifically quantify and characterize the soft-errors on the Titan supercomputer's GPU nodes. Our study uncovers several interesting and previously unknown insights about the characteristics and impact of soft-errors (e.g., single bit error, dynamic page retirement error, and double bit error). We characterize the temporal characteristics of single bit errors and its association with other errors. We study the impact of workloads, resource utilization, and variance in load-level on error-affected GPU nodes. In particular, our study aims to understand the correlation between application characteristics and specific GPU errors. Our study also provides a deep understanding of possible temperature effects on soft-errors. As we describe our findings, we also point out how different methodologies may lead to different observations, and the importance of our observations to system administrators and architects. We believe that insights obtained from our large-scale field data analysis carry significant implications for current and future HPC computing facilities, system operators, and system architects.

2. Background and Methodology

In this section, we provide an overview of the Titan supercomputer and its GPU architecture. Next, we provide details about our data collection and analysis methodology. We also describe the limitations of this study.

978-1-4673-9211-2/16/$31.00

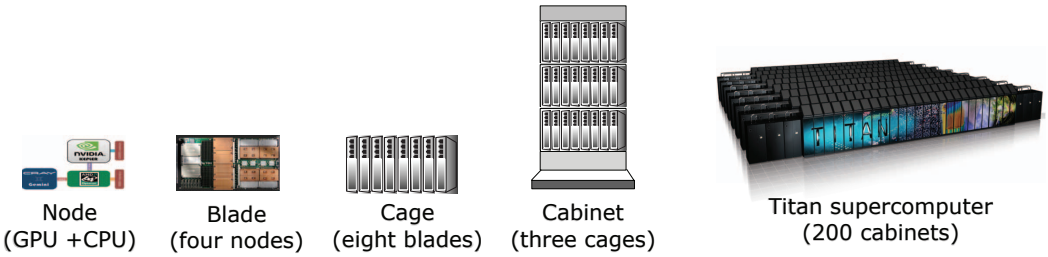


Figure 1: Overview of the Titan supercomputer's physical organization.

2.1. Titan Supercomputer Organization and GPU Architecture

Fig. 1 shows the physical organization of the Titan supercomputer. It consists of 200 cabinets organized in 25 rows and 8 columns. Each cabinet has three cages/chassis. There are eight blades in each cage. Four nodes constitute one blade. Each node has one AMD Opteron 6274 CPU (with 32 GB of DDR3 memory) and one NVIDIA K20X GPU (with 6 GB of GDDR5 memory). Each blade has two high-speed interconnect Gemini routers, each shared by two nodes. NVIDIA K20X GPU has a total of 14 streaming multiprocessors, each streaming multiprocessor has 192 CUDA cores. Each streaming multiprocessor has 64K registers, 64KB of combined shared memory and L1 cache, and 48KB of read-only data cache. The L2 cache (1536 KB) and the GDDR5 memory (6GB) is shared by all streaming multiprocessors. Overall, each GPU has peak performance of over 1.30 Tflops (double precision). On-chip and off-chip GPU memory structures including the device memory, L2 cache, instruction cache, register files, shared memory, and L1 cache are protected by a Single Error Correction Double Error Detection (SECDED) error correction code (ECC). In the K20X GPU architecture, the read-only data cache is parity protected. On the other hand, structures such as logic, thread schedulers, instruction dispatch unit, and interconnect network are not protected by the ECC.

2.2. GPU Errors: Collection and Analysis Methodology

Our study is based on the Titan supercomputer consisting of 18,688 GPUs. GPU errors can be classified into several categories. GPU hardware related errors, such as double bit errors, Off the Bus, and micro-controller halts cause the application to crash. Soft errors that can be corrected by the ECC mechanism do not result in execution loss. Single bit errors are corrected by the SECDED ECC. Two single bit errors on the same page result in a dynamic page retirement (DPR) error [3]. This particular error is also reported when a double bit error happens and the page is retired in order to improve the longevity of the card.

There is a host of GPU related errors including errors that are caused by the application, driver issues, firmware bugs, or thermal issues. Note that NVIDIA documents a list of such XID errors and their possible causes [6]. GPU applications may also terminate with a non-zero exit code, indicating that the execution was not successful. Other than hardware-related and XID errors, several other reasons may be responsible for non-zero exit codes, e.g., programming errors and expiration of time-quota. However, unlike previous works [15,33], we do not study these XID errors or system-integration errors (e.g., Off the Bus). This study primarily focuses on single bit errors, dynamic page retirement errors, and double bit errors.

We collect GPU-error related data from February 2015 to June 2015 (more than 60 million node hours). The console logs from Titan are parsed to log critical system events. These critical system events alert the system operators of unexpected/undesired behavior. We point out that we apply a filter to separate a "parent" failure event from its "child" events. This methodology is similar to the one outlined in previous works [15, 24, 34, 35], but understanding the impact and effect of "parent/child" failure events is not the focus of this paper, this topic is covered in detail by other works [15, 33].

We note that single bit errors are not logged in the console log, these errors are collected via the *nvidia-smi* utility on all GPU nodes. This utility provides snapshot information, i.e., it does not timestamp individual single bit errors, but records single bit errors before and after each batch job. This allows us to do temporal analysis on single bit errors, albeit at the granularity of a "batch job". We denote a batch job as a set of submitted applications that are submitted by the same user (using a `qsub` command on Titan). Multiple "applications" (also referred as "apruns") can run within a submitted batch job (also referred to as "job" or "batch"). The single bit error count is collected at the start and end of the batch job and hence, can not be associated with an application run directly. We also note that our framework can identify the node locations on which the single bit errors occur. We collect GPU resource utilization information such as GPU core-hours, maximum memory consumption, and total memory consumption, on a per application basis.

The output from the nvidia-smi utility also includes double bit and dynamic page retirement related errors. We do not use this utility to analyze double bit or dynamic page retirement errors due to inconsistency in error logging as pointed out by previous works [34].

2.3. Limitations and Scope

While our study covers the GPU error data for a supercomputing facility over an extended period of time, we recognize that our work is subject to assumptions and limitations.

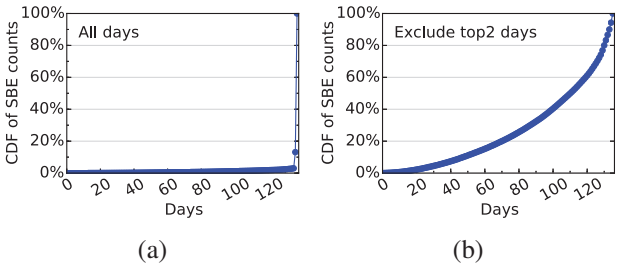


Figure 2: Cumulative single bit error (SBE) count distribution over days (a), and cumulative SBE count distribution over days excluding top 2 days (b).

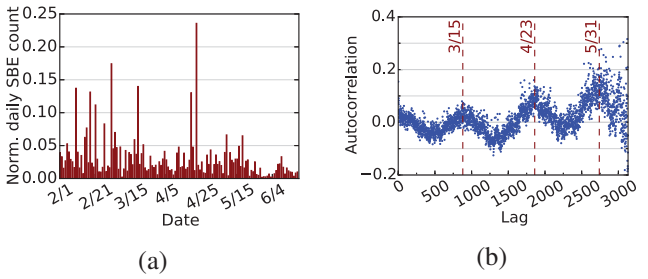


Figure 3: Daily SBE count across time excluding the top two days (a), and autocorrelation function of the SBE interarrival times (b).

First, our analysis is post-hoc in nature and hence, by definition it can not answer what-if scenarios where one may require changing the system/workload environment to observe the effect of a change.

Second, we note that such a large-scale computing facility is often very dynamic in nature with respect to software stack changes. Operational practices are continuously tweaked and unscheduled outages take place among other system updates. We have limited control over such factors. Therefore, isolating the impact of the above factors on our study is challenging. Instead, as we discuss our findings in the paper, we specifically point out the external factors that we believe may influence our findings. Previous works have also pointed out that NVIDIA's GPU error logging has improved over time [33, 34]. Our error collection framework attempts mitigate this by collecting the same error information via multiple possible methods.

Third, our study provides insights about correlation between applications/users and GPU error characteristics. Yet, it is not possible to investigate specific applications since we do not have access to application source codes. We point out that we have little to no knowledge about users' intentions. User behavior may change over time as the scientific knowledge in a particular domain improves. A new computational model or method in a particular domain may affect all applications in that domain at a given time or over a period of time. We also note that while our logs report the application name (binary name) at the end of each job, it is possible for a user to use the same binary name for two different applications, or the same application with different input types. However, for our analysis we conservatively treat them as the same application because of the lack of more detailed knowledge.

We show that performing similar analysis at the user-level mitigates some of the side-effects and provides additional understanding. However, this is a fundamental challenge that can not be rectified in a post-hoc analysis, especially for a production supercomputing facility where system resource managers can not influence user computing practices/behavior.

3. Analyzing Single Bit Errors (SBE) on the Titan Supercomputer

In this section, we aim to understand the temporal characteristics of single bit errors (SBE) on the Titan supercomputer. While previous studies have shown that most of the SBEs tend to occur only in a few GPU cards [34], the temporal characteristics of the SBEs have not been explored because of the inability to collect SBE occurrence information continuously over time. As described earlier in Section 2, our framework enables us to collect SBE counts at the batch job granularity.

Fig. 2(a) shows the CDF of the single bit error counts on a per day granularity. Recall that the time stamp of each SBE occurrence is not recorded. However, since the Titan supercomputer is highly utilized, we are able to collect the SBE data from a large number of batch jobs and aggregate them over 24-hour periods. It should be noted in Fig. 2(a) that the x-axis presents the days in the observation data in increasing order of their daily SBE count.

The steep curve of the distribution suggests that only a few days accounts for most of SBEs. In fact, only three days account for the 97.18% of the total SBEs, while the top ten days with most SBEs account for 97.84% of the total SBEs. Due to this skewness, it not clear how errors are being accumulated over the rest of the days. To better view this, we plot in Fig. 2(b) the cumulative distribution function of SBE counts but exclude the top two days. We observe that SBE occurrences are not proportionally distributed over the rest of the days either, i.e., 40% of the days with the lowest SBE daily counts account for only 10% of the total SBE counts, while the remaining 60% of the days account for 90% of SBEs.

This uneven distribution of SBEs across days led us to investigate how these errors appear across time. Fig. 3(a) shows the normalized SBE count *per day* for the whole period of the study. We normalize the daily SBE count by the average of the daily SBE count over the whole period. This figure indicates that the density of SBEs across days is fairly uneven and appears bursty.

To examine whether there is burstiness and/or periodicity in SBEs, we analyze the time series of SBE occurrences and plot the autocorrelation function of the inter-arrival times of

batches with non-zero SBE counts since SBE measurements are at the per batch granularity. Autocorrelation is a mathematical representation of the degree of similarity in a time series and a lagged version of itself. As such, it is ideal for discovering repeating patterns by quantifying the relationship between different points of a time series as a function of the time lag [18]. The autocorrelation metric is in the range of $[-1, 1]$. Higher positive values indicate that the two points between the computed lag distance are "similar", i.e., have stronger correlation. Zero values suggest no periodicity. Negative values show that the two points that are lag elements apart are diametrically different. Fig. 3(b) illustrates the autocorrelation function of the inter-arrivals of batches as a function of the distance between successive arrivals (lags). The figure illustrates a noticeable periodic pattern: the pattern repeats within every 6 weeks, and the periodic pattern for positive autocorrelation values becomes even more pronounced as the lag increases. This indicates that both burstiness and periodicity are present, it may be therefore possible to predict future SBE occurrences using this information [37]. One may argue that burstiness in SBE occurrence is an artifact of burstiness in the inter-arrival of GPU jobs. To address this, we performed the autocorrelation analysis on the number of applications executed every day. We found the autocorrelation metric to be close to zero, indicating lack of burstiness in the inter-arrival of GPU applications. This is expected since the Titan supercomputer is a highly-utilized computing platform with long job-queue waits. Therefore, we conclude that our observation about SBEs in not an artifact of the GPU job execution characteristics.

Observation 1 *Our field data analysis suggests that single bit error occurrences on the Titan supercomputer are bursty in nature. These errors tend to be clustered in time. Given that most of these errors are also limited to only a few GPU cards [34], system administrators can exploit these observations together for better GPU job scheduling at a large scale (e.g., avoid scheduling critical workloads on certain nodes / days, and possibly turn off ECC on certain nodes during specific time-periods for improved performance).*

Previous work has shown that only a few selected GPU cards experience most of the SBEs in the system [34]. We note that the measurement period here does not overlap with that of a previous study on the same system but, our study reconfirms the findings presented in previous work [34]. Here, the top two SBE offenders out of all 590 SBE offenders account for 96.9% of SBE errors. Interestingly, we also found that these top two SBE offenders accumulate all the SBEs on a single day. This led us to investigate how SBE offender nodes accumulate these errors over time and look for how many distinct days each SBE offender experiences one or more SBEs. Fig. 4(a) shows the number of distinct days that a specific SBE offender node experiences an error. We make two observations. First,

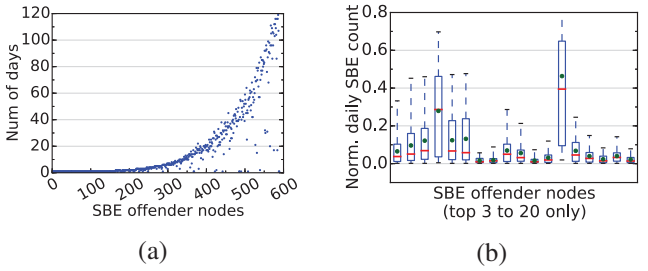


Figure 4: Number of SBE-affected days for all nodes (sorted in increasing order of total SBE count) (a), and normalized variation in the daily SBE count distribution for the top twenty SBE offender nodes excluding the top two nodes (red line in the middle represents median while green dot represents mean) (b).

as illustrated by the points in the bottom right corner of the plot, a few top SBE offender nodes experience most of their errors over a small number of days. Second, the rest of nodes do not show a linear trend in terms of the number of distinct days over which SBEs occur. For example, the bottom 65% of the SBE offenders (approximately 400 nodes) accumulate their SBEs over less than 20 days, while the top 35% of SBE offenders (approximately 200 nodes) take up to 6 times more days to accumulate their SBEs. This non-linearity in SBE accumulation can be particularly useful to HPC facility administrators for identifying high SBE offender nodes and exploiting this information for better GPU job scheduling.

Motivated by the above observation, we look deeper into the top 20 SBE offender nodes. In particular, we plot the variation in daily SBE count for top 3 to 20 nodes (we do not consider the top 2 nodes because all their SBEs occur on a single day only). Fig. 4(b) illustrates the boxplot of the daily SBE counts that shows the 25th and 75th percentiles as well as median (flat line) and mean (dot). The boxplots show that variation can be significantly high for certain nodes. This suggests that while high count SBE offenders accumulate single bit errors over a large number of distinct days, it may be challenging to predict the number of single bit errors these nodes are expected to experience on a particular day.

Observation 2 *A few top SBE offenders experience all of SBEs over a very small number of days. However, the rest of nodes do not show a linear trend in terms of the number of distinct days over which SBEs occur. High count SBE offenders experience errors over a significantly high number of distinct days compared to the low count SBE offenders nodes. Moreover, the variation of SBE occurrence among days can change significantly across SBE offender nodes.*

After investigating the temporal characteristics of SBE occurrences, we attempt to understand how GPU resource utilization affects SBE occurrences. In particular, we test if higher GPU resource utilization may lead to higher SBEs. We point out that single bit errors can occur due to multiple reasons,

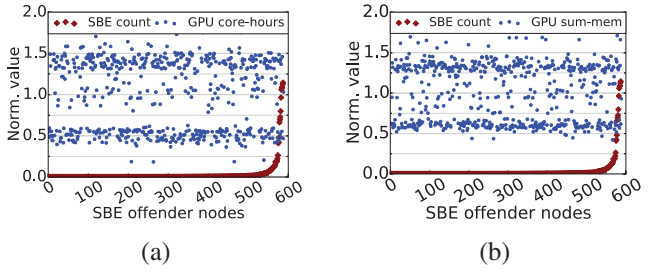


Figure 5: GPU resource distribution for the SBE offender nodes (excluding top two SBE offenders): GPU core hours (a), and GPU memory utilization (b).

therefore, higher GPU resource utilization alone may not be considered as the "cause". Fig. 5 shows the normalized GPU core hours and memory utilization for all SBE offender nodes. The normalization is performed using the average for all SBE offender nodes except the top two nodes (which are considered outliers, as their SBEs occur in a single day only). We observe that the nodes with higher SBE count do not necessarily use higher GPU core hours or run workloads with higher memory utilization.

While GPU resource utilization does not seem to be directly correlated with the SBE occurrence frequency on the GPU nodes, we suspect that the variance in GPU resource utilization may be correlated to higher SBE occurrences. More precisely, we want to test the hypothesis that days with higher variance in GPU utilization experience higher single bit errors. Fig. 6 shows the top 50 days that encountered most SBEs (in increasing order) and the corresponding variance in GPU resource utilization on that day. We note that Fig. 6(a) and (b) indicate that the couple of days with the highest SBE count may also experience the highest variance in their GPU resource utilization. However, a more closer look at top 4 to 50 days (Fig. 6(c) and (d)) shows that variance in GPU resource utilization does not imply higher daily SBEs.

Observation 3 *We found that GPU resource utilization and the variance in the GPU resource utilization do not seem to be significantly correlated with the SBE occurrences. Higher GPU resource utilization or its variance do not necessarily result in a higher SBE count. We believe that an important implication of this finding is that GPU resilience simulation and modeling frameworks do not necessarily need to vary the soft-error rate based on the compute load or variance in the load. This can potentially simplify the design of such tools without compromising the accuracy of the study.*

We learned that the GPU resource utilization is not highly correlated with the SBE frequency on SBE offender nodes. Here, we investigate the relationship between specific users/applications and SBE counts. In other words, is a certain fraction of users/applications experiencing more single bit errors than others? If so, what are the respective GPU resource utilization levels?

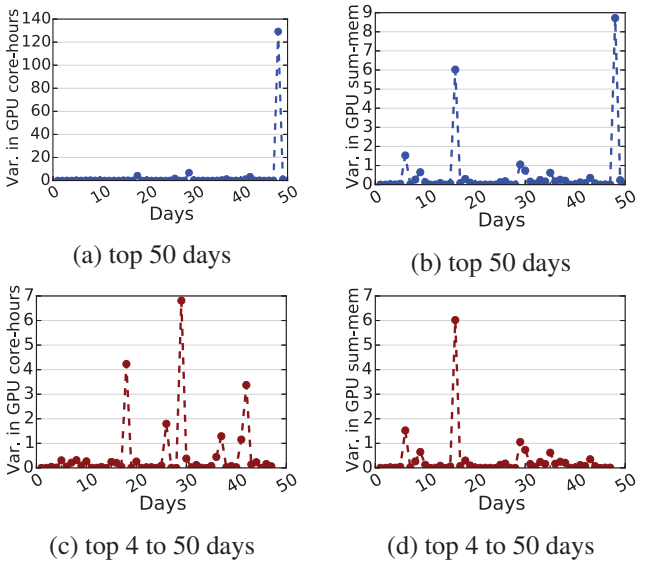


Figure 6: Variance in the GPU resource utilization and daily SBE count: GPU core hours for top 50 days (a), for top 50 days excluding the top 3 days (b), GPU memory utilization for top 50 days (c), and for top 50 days excluding the top 3 days (d). Days are sorted in increasing order of SBE count.

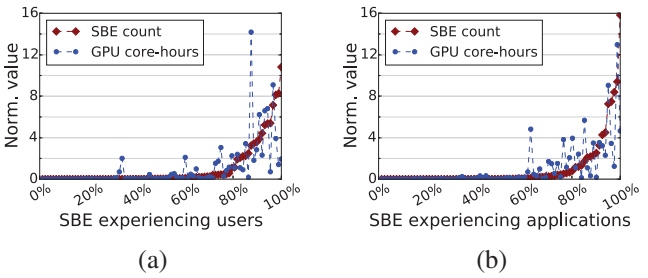


Figure 7: GPU core-hours for users (a), and applications (b) experiencing SBEs.

Fig. 7(a) shows the SBE count of different users versus their respective GPU core hours. Both SBE count and GPU core hours have been normalized by their respective average values. We also point out that only users that encountered at least one single bit error are included in the plot. We found that the correlation between GPU core hours and SBE count is significant when studied at the user-level. The Pearson coefficient is 0.59 with p-value < 0.05 while the Spearman coefficient is 0.89 with p-value < 0.05. This indicates a strong non-linear correlation. We did similar analysis between the SBE count for users versus their respective GPU memory utilization. We found similar trends in the results (not shown here due to lack of space).

Fig. 7(b) shows that SBE count for applications versus its respective GPU core hours. Only the applications affected by SBEs are included in the plot. Similar to our previous analysis for users, we found strong non-linear correlation in this case as

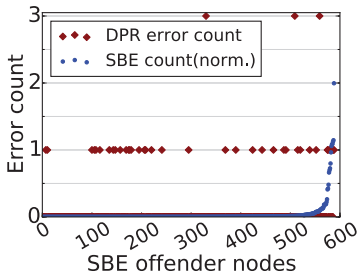


Figure 8: DPR errors for SBE offender cards (excluding top two SBE offenders which had no DPRs).

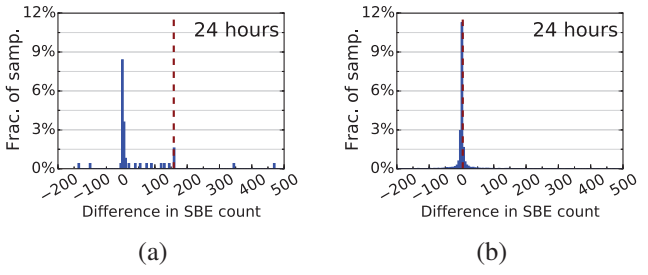


Figure 9: Histograms of difference in SBE count for a 24-hour windows after and before the DPR occurrence for DPR offender nodes (a), and non-DPR offender nodes (b). Dotted vertical lines represent the average difference in SBE count.

well. The Pearson coefficient is 0.67 with p-value < 0.05 while the Spearman coefficient is 0.89 with p-value < 0.05. Analysis between the SBE count of different applications versus their respective GPU memory utilization shows similar trends.

In summary, our data suggests that GPU resource utilization at the user-level appears highly correlated with the SBE frequency for different users and applications.

Observation 4 *SBE occurrence frequency appears to be highly correlated with users and applications. This correlation is better expressed by a non-linear relationship and is not necessarily an artifact of the bursty nature of single bit errors. This indicates the necessity and importance of application-centric GPU error resilience techniques and tools.*

4. Analyzing Dynamic Page Retirement (DPR) Errors on the Titan Supercomputer

Dynamic Page Retirement (DPR) is an important resilience feature to improve the longevity of an otherwise good GPU card. A page in the GPU device memory is blacklisted if two single bit errors or one double bit error occur on the same page. This page is not allocated to the application on the next reload of the GPU driver [3]. In this section, we study single bit errors and dynamic page retirement errors together since SBEs can cause DPRs. We also investigate the impact of GPU resource utilization and applications on DPR occurrence.

For the measurement period, we observe a total of 50 DPR errors on 43 distinct GPU cards. Recall that we observe that SBEs tend to be more concentrated in a few selected GPU cards. More generally, the distribution of SBEs is not uniform among all the 590 SBE offender cards. Therefore, we hypothesize that DPR errors are more likely to occur in the top SBE offender cards. Fig. 8 shows the DPR and SBE error frequency for all SBE offender cards (excluding the top 2 SBE offenders which do not have any DPR). The plot shows that some top SBE offender cards do observe DPR errors. For example, the top 10 SBE offender cards account for 4 DPR errors, while the top 20 SBE offender cards account for 7 DPR errors out of total 50 DPRs. Cards with low SBE counts show no DPRs during the measurement period. Most of the top SBE offenders do not experience any DPRs either. It is possible that some top SBE offender cards may potentially experience a DPR error in the future but we argue that our measurement period is long enough to account for most of such cases given the bursty occurrences of SBEs.

Observation 5 *Top SBE offender GPU cards do not necessarily experience more dynamic page retirement errors. In fact, DPR errors may occur on any SBE offender cards, even to those with relatively lower single bit error counts.*

One can also reasonably hypothesize that the SBE count is likely to be higher on DPR offender nodes before the DPR error, since two SBEs trigger a DPR. To test this hypothesis, we calculate the difference of SBE counts after and before each DPR occurrence within a certain time window (i.e., 24-hour), for both DPR offender nodes and non-DPR offender nodes. In other words, we accumulate the SBE count on the node for 24 hour window both after and before the DPR event, and then take the difference. Fig. 9 presents the histograms of the difference in SBE count for a 24-hour window for both DPR offender nodes and non-DPR offender nodes. The dotted vertical line in each graph shows the average. Average value of this difference for DPR offenders is around 160 while the value for non-DPR offenders is around 0. Similarly, the cumulative distribution in Fig. 10(a) shows that DPR offending nodes and non-DPR nodes have significantly different distribution. We also conduct the Kolmogorov-Smirnov Test (KS test) to test this hypothesis. We find that D = 0.389, p-value = 5.991×10^{-7}. For our sample size here, the critical D value is 0.19 and therefore we can reject the null hypothesis, and conclude that DPR offending nodes show significantly higher values of difference in SBE counts compared to non-DPR nodes.

Next, we test if SBEs continue to occur on the DPR offender nodes beyond the 24-hour period since the last DPR error occurrence. If so, for how long do the DPR offender nodes continue to experience single bit errors? Fig. 10 shows the cumulative distributions of difference in SBE count for two different size of time windows. As a comparison point, we present results for 24 hours time-window and 72 hours time-window (Fig. 10(a) and (b)). We observe that the cumulative

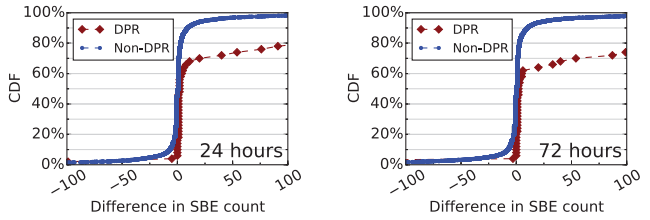


Figure 10: Cumulative distributions of difference in SBE count for a 24-hour window (a) and a 72-hour window (b) for DPR offender nodes and non-DPR offender nodes. Some outliers are omitted for clarity. Omission of outliers causes the DPR-curve not to approach 1.

distribution does not change significantly from 24 hours to 72 hours. This indicates that the majority of SBEs occurring after the DPR occurrence tend to occur within first 24 hours. We find that the likelihood of SBEs increases after the DPR occurrence, but it does not continue to remain at that level always. We also note that the time-period after which the probability of SBE occurrence returns to normal level can vary across GPU nodes. We found 24-hour to be a good threshold in our case and do not present more detailed, fine-grained results (due to space constraints). In summary, this is an interesting and counter-intuitive finding as the original hypothesis suggests higher SBE occurrences before the DPR error; on the contrary, our field data indicates that more SBEs are likely to occur after the DPR error.

Observation 6 *Our field data analysis shows that single bit errors tend to occur more frequently on the DPR offender nodes after the DPR error than before. This is counter-intuitive since single bit errors are a cause of DPR errors, and hence, one would expect the SBE error rate to be higher before the DPR error. We also observe that the majority of SBEs occurring after the DPR occurrence tend to occur within first 24 hours. This finding can be useful in cases where an application/user may turn on/off ECC support based on the probability of soft-error occurrences.*

Recall that the DBE is another cause for DPR errors. We conducted analysis to understand the relationship between DBEs and DPRs. However, due to the limited number of errors, it is not possible to draw conclusions with high statistical significance. Next, we investigate the effect of GPU resource utilization on the DPR error frequency, similar to the analysis performed for single bit errors. Fig. 11 presents the GPU resource utilization for the GPU nodes that experience DPR errors. We point out that the GPU core-hours and sum-memory metrics are normalized to the corresponding average across *all* nodes. Fig. 11 shows that GPU resource utilization points do not show any clear trend. Nodes that experience a DPR do not have higher resource utilization compared to nodes that

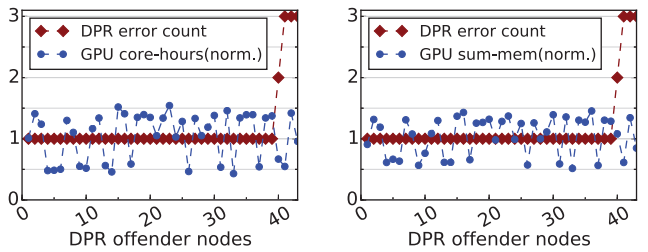


Figure 11: DPR affected GPU nodes with increasing error counts and normalized GPU core-hours (a), and normalized GPU memory utilization (b).

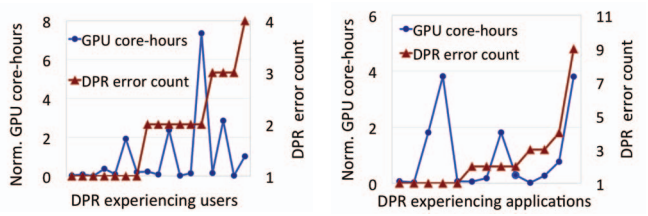


Figure 12: DPR errors and GPU core-hours for DPR affected users (a), and DPR affected applications (b).

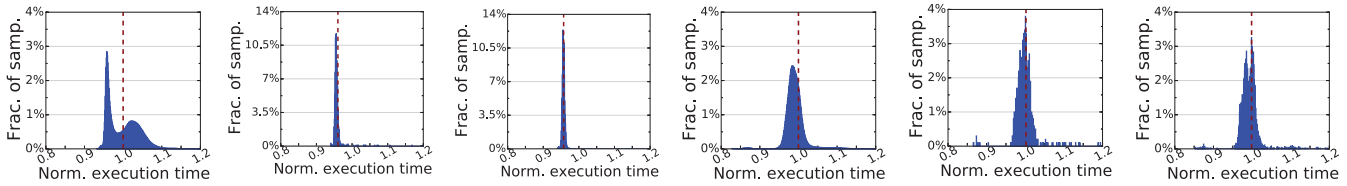
do not experience DPR errors. This finding is similar to the one expressed in Fig. 5 where SBE events do not show strong association with the GPU resource utilization.

Observation 7 *We observe that there is no significant association between DPR count and GPU resource utilization.*

As we do not find any significant relation between GPU resource usage on DPR affected GPU nodes and DPR error frequency, we now look into how GPU resource usage of certain users and applications correlates to DPR errors. Naturally, the GPU resource usage varies among different users and applications. Therefore, we investigate if applications that experience higher DPR errors also have higher GPU resource utilization. Fig. 12 shows that GPU resource utilization is not necessarily correlated to the number of DPR events experienced by different users and applications. The Spearman and Pearson coefficients show almost no correlation. In summary, we can observe from Fig. 12 that users that experience more DPR errors do not necessarily use longer GPU hours.

5. Analyzing Performance Variance in SBE and DPR Affected GPU Nodes

In this section, we investigate if nodes affected by SBE and DPR errors are more likely to show higher performance variation or significant degradation in performance compared to error-free nodes. Toward this, we perform extensive experiments on the SBE and DPR affected nodes and randomly selected nodes on Titan.



(a) MatMul. Random Nodes. (b) MatMul. Top 10 SBE Nodes. (c) MatMul. All DPR Nodes. (d) BFS. Random Nodes. (e) BFS. Top 10 SBE Nodes. (f) BFS. All DPR Nodes.

Figure 13: Distribution of execution time on random nodes, top 10 SBE nodes, and DPR offending nodes.

We run two representative GPU kernels, Matrix Multiplication (MM) and Breadth-first Search (BFS) on all DPR nodes, top 10 SBE offender nodes, and randomly selected error-free GPU nodes. These kernels have significantly different computational characteristics. MM is a regular, compute-intensive benchmark, while BFS is an irregular data-intensive one. MM and BFS kernels were obtained from the NVIDIA CUDA toolkit [2] and Rodinia Benchmark Suite [5], respectively. We collect performance data by repeatedly running these kernels on the selected GPU nodes. We conducted over 24000 experiments on Titan GPU nodes, covering more than 9000 randomly selected GPU nodes. Each kernel is run 100 times on each DPR offender node and top 10 SBE offender nodes.

Fig. 13 shows the distribution of execution times on randomly-selected nodes, top 10 SBE offender nodes, and DPR offender nodes. The execution time on the x-axis is normalized with respect to average performance across all runs. We note that some outliers in these plots are omitted for presentation clarity but their effect on mean and standard deviation is reflected on the graphs. For the MM benchmark, we notice that SBE nodes and DPR nodes have 3-4% better performance on average compared to the randomly selected nodes. This is because randomly-selected nodes exhibit a bimodal distribution of execution times, making the average execution time of these nodes slightly higher. For the BFS benchmark, there is no significant difference in average performance between the top 10 SBE and DPR offender nodes compared to randomly-selected nodes. The SBE and DPR offending nodes show slightly lower standard deviation compared to randomly selected Titan nodes. We believe that this is primarily because the number of DPR and top 10 SBE offender nodes are much smaller compared to our randomly-selected node pool (over 9000 nodes). There are other factors that can cause higher standard deviation among such a large number of nodes (e.g., variance in temperature, spatial location, device properties).

Observation 8 *The distribution of execution time across randomly selected nodes on Titan in itself may be application-dependent. Our experimental data suggests that top 10 SBE and DPR offending nodes do not exhibit lower performance than the average performance of randomly selected nodes. The implication of this finding is that system operators do not need to replace GPU cards with high SBE / DPR error counts specifically for performance degradation or variance reasons.*

Table 1: Statistics for Temperature (°C) (DPR)

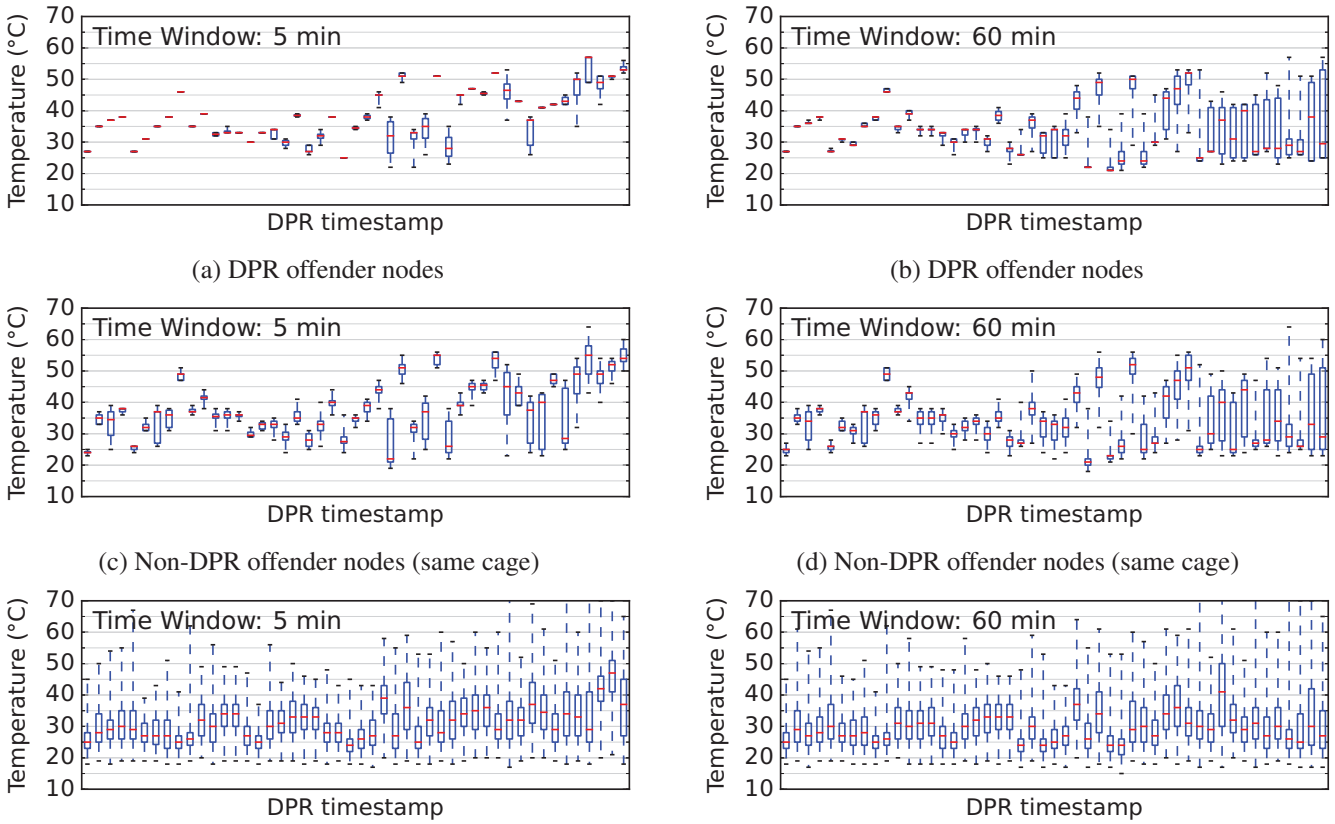
	60*min before* (avg / stddev)	15*min before* (avg / stddev)	5*min before* (avg / stddev)
DPR	34.55 / 8.53	37.00 / 8.95	39.02 / 8.79
Non-DPR (same cage)	34.68 / 8.71	36.77 / 9.24	38.56 / 9.27
Non-DPR (random)	30.54 / 7.70	31.54 / 7.79	32.47 / 8.11

6. Understanding the Effect of Temperature on Dynamic Page Retirement and Double Bit Errors

In this section, we investigate the effect of temperature on GPU soft-errors, in particular DPRs, DBEs, and SBEs. Past work points to temperature dependence of hardware errors on other systems [10,30]. Here, we perform a detailed analysis of the relationship between temperature and soft-errors on GPUs.

In the Titan supercomputer, upper cages are typically at higher temperature than lower cages. We found that the distribution of DPR errors across different levels of cages is fairly equally distributed. Therefore, this does not imply a direct impact of temperature on DPR errors as such. To investigate deeper, we collected GPU card-level temperature for different time windows of 5 minutes, 15 minutes, and 60 minutes *before* each DPR occurrence for a large number of GPU nodes. We collect temperature data every minute for each GPU card. Table 1 shows the mean and standard deviation of temperatures across the three time windows of 5 minutes, 15 minutes, and 60 minutes before each DPR occurrence. These statistics are collected for the DPR offender node, all nodes within its cage, and over 800 random nodes in the system.

First, we observe that temperature across all three types of nodes increases consistently during the hour as the DPR occurrence approaches (Table 1). This may be possibly due to power/cooling condition in the machine room or the currently running workload. Interestingly, the DPR offenders have higher average temperature than randomly selected nodes. This indicates that higher temperature may be associated with DPR errors. However, we also note that the nodes in the same cage as the DPR offenders show similar average temperature. This suggests that higher temperature may be associated with the increase in the likelihood of a DPR error. However, one



(a) DPR offender nodes

(b) DPR offender nodes

(c) Non-DPR offender nodes (same cage)

(d) Non-DPR offender nodes (same cage)

(e) Non-DPR offender nodes (random)

(f) Non-DPR offender nodes (random)

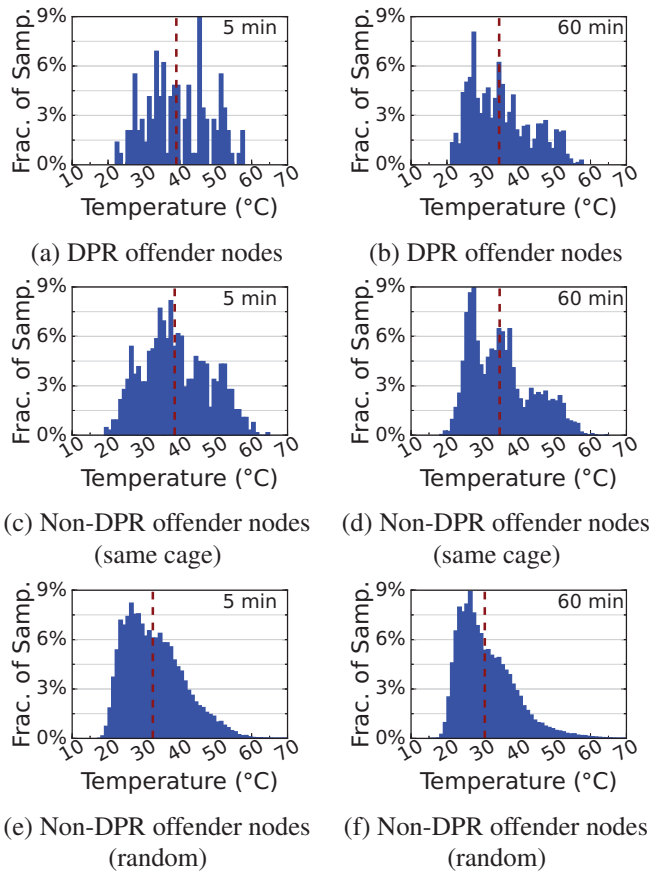
Figure 14: Temperature variation before each DPR occurrence.

can not trivially conclude that higher temperature leads to DPR errors since other GPU nodes in the same cage do not observe a DPR error despite similar average temperature.

These results emphasize the importance of selecting the correct methodology for comparisons: comparing the data across random nodes in the entire system and nodes within the same cage, we see the importance of selecting what to compare with. Choice of random nodes may sound as the right choice for comparison but in such-large scale systems usage behavior and node characteristics can be significantly different in randomly chosen nodes.

Table 1 also shows that the standard deviation in temperature across all three types of nodes is similar. Yet, the standard deviation for DPR offenders is generally higher than the standard deviation for randomly selected nodes. Since the standard deviation is a single number and may not capture the entire picture, we investigate deeper to understand the effect of temperature variation of DPR errors. Fig. 14(a)-(f) illustrate how temperature variations occur for different type of nodes with a DPR occurrence for the extreme time windows: 5 minutes and 60 minutes. Each element on the x-axis corresponds to a DPR occurrence (i.e., its timestamp), each box-plot shows the 25 and 75 percentiles, the median (as a flat line), as well as the ending points of the temperature distribution (whiskers). We observe that there is more variation in the temperatures if the time window is longer. This observation is true across all types of nodes. However, closer to the DPR occurrence, the temperature variations decrease significantly for DPR offender nodes as compared to randomly selected nodes and nodes in the same cage as the DPR offender. Unlike previous research for hard-disk related errors [10], our analysis suggests that higher temperature variation does not necessarily increase the probability of DPR errors. In fact, the majority of DPR offender nodes remain comparatively hotter and with non-fluctuating temperatures. We also point out that the temperature variation for randomly selected nodes is also affected by the large number of samples, partially contributing toward higher variance.

Next, Fig. 15 presents histograms of the frequencies of temperatures for the three categories of nodes: DPR offenders, DPR cages, and random. The average values are denoted by the dashed lines in each histogram. The figure indicates that for the randomly selected nodes the right tails (corresponding to higher temperatures) are thinner than those of the DPR and DPR-cage ones. Focusing on the histograms that correspond to the 5 min observations (i.e., the left column of Fig. 15), one can notice the difference in shapes across the three histograms. The random nodes, shown in Fig. 15(e) have significant prob-



(a) DPR offender nodes (b) DPR offender nodes

(c) Non-DPR offender nodes (same cage) (d) Non-DPR offender nodes (same cage)

(e) Non-DPR offender nodes (random) (f) Non-DPR offender nodes (random)

Figure 15: Temperature variation before each DPR error.

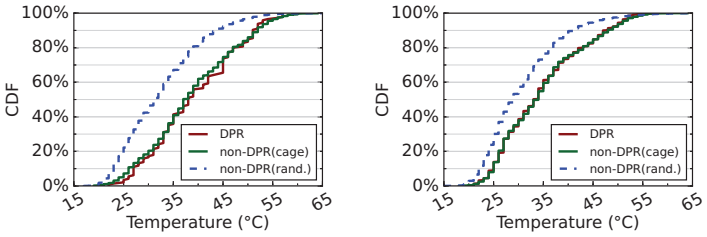
ability mass that is below 40 degrees comparing to the DPR offenders and non-DPR offenders within the same cage. This mass may not be as pronounced in the 60 min observations, but it is still present across all histograms in the right column of Fig. 15. Overall, the six histograms shown in this figure allow the reader to appreciate how the mere differences in standard deviation that are shown in Table 1 indeed correspond to significantly different temperature frequencies.

To better compare these histograms quantitatively, we compare them as CDFs in Fig. 16. Fig. 16(a) shows all CDFs for the 5 minutes case and Fig. 16(b) shows all CDFs for the 60 minutes case. Across both graphs, we see that the random nodes (non-DPR) have significantly lower temperature than those of DPR offenders. For example, in the 5-minute window, we see that 50% of random nodes have temperature less than 35°C, but only 25% of those within the DPR case reach this mark. This trend is consistent across most temperatures, nearly 20% of nodes that are randomly selected are consistently cooler than those in the DPR categories (individual and cage). Further we see that even within the same temperature percentile level, there is a difference in temperatures ranging between three to ten degrees. For the longer time window of 60 minutes, these differences still exist but not as large.

In summary, we have seen that while the temperatures of

Table 2: Statistics for Temperature (°C) (DBE)

	60*min before* (avg / stddev)	15*min before* (avg / stddev)	5*min before* (avg / stddev)
DBE	32.64 / 5.97	32.02 / 5.54	33.30 / 6.18
Non-DBE (same cage)	32.14 / 6.24	32.23 / 6.07	33.14 / 6.82
Non-DBE (random)	32.89 / 8.54	32.79 / 7.96	33.39 / 7.89



(a) 5 min before (b) 60 min before

Figure 16: CDF of temperature variation before DPR errors.

DPR offenders may be similar to nodes within the same cage, but they are consistently hotter than randomly selected nodes in the machine. This further supports the observation that high temperature may precipitate the occurrence of a DPR, especially if it remains consistently high (i.e., temperature variations are rather limited).

We conduct similar analysis for DBE occurrences, results are shown in Table 2. We observe that there is no significant difference in temperature of DBE offender node, other nodes in the same cage as the DBE errors, and randomly selected nodes. Therefore, we can not conclude the effect of temperature on DBEs as per this analysis. However, we found that DBEs occur more frequently in the upper cages than the lower cages (similar to previous work [34]). This indicates some association with temperature, since the upper cages are typically hotter than the lower cages. It should be noted that, this in itself can not lead to well-formed conclusion due the varying temperature of nodes over time. Recall that single bit errors are collected at start and end of each batch job and hence, we do not have the exact timestamp of occurrence. This limits our capability to perform fine-grained analysis on the effect of temperature on single bit errors.

Observation 9 *Temperature may have an impact on GPU soft errors (DPR and DBE), but this conclusion is highly dependent on the choice of nodes to compare against. Our analysis clearly shows that a comprehensive methodology should be followed and described when making such assessments. We found that the higher temperature may be correlated with DPR and DBE errors, and the higher variability in temperature does not necessarily lead to increased probability for DPR errors.*

7. Related Work

Quantifying and characterizing the system failures is key to improving the reliability of any computing system. This is even more important for large-scale computing systems since the impact of system failures is larger on these system and may lead to significant scientific productivity loss and monetary loss. Consequently, researchers have investigated failures on large-scale systems in detail [8, 9, 19, 20, 24, 26–28]. Several studies have exploited the insights from such efforts to predict failures and adapt fault-tolerance mechanisms to minimize the impact of system failures. Some of these studies propose to predict failure by identifying the correlation among failure events [11–13, 19]. Such proposals often rely on machine learning and other prediction techniques on the RAS logs and the system logs. This may result in high-overhead and low lead time for prediction, but nevertheless they demonstrate that failure prediction is possible and effective in certain cases.

Several studies have focused on studying the reliability aspect of large-scale computing systems. For example, Liang et al. investigated different component failures including network, disk, memory and CPU for the Blue Gene/L system, and proposed failure prediction models [19]. Oliner et al. investigated system failure logs for multiple HPC systems at the Los Alamos National Laboratory and the Sandia National Laboratory, including RedStorm and Thunderbird system [24]. They studied both software and hardware errors and developed the methodology for applying filtering to failure logs. Schroeder et al. have studied the system failures and its impact on multiple HPC systems at LANL [28].

There have also been more focused effort on studying failures for a given system components such as DRAM, disks, and SSDs. For example, DRAM-focused efforts have shown the effect of height and vendors on soft-errors [17, 31, 32]. These studies also showed the pitfalls in studying the DRAM errors and its impact on the reliability assessment of the system. Disk-focused efforts demonstrate that disk failure in the field can be significantly higher than what one would estimate from the vendor's sheet [7, 29]. Such studies also show that peripheral components fail more often than one may expect in large scale storage systems. Recent study [23] on SSD failure in the field provide insights about differences in the early detection life cycle between SSDs and Disks, lack of read disturbance error in the wild, and implication of these findings for future SSDs. However, large-scale GPU reliability characterization studies have been relatively limited [8, 16, 34], primarily because GPU architecture is relatively newer technology to be deployed at such a large-scale.

Recently, there have been efforts focusing on studying and improving GPU reliability at scale [14]. Several recent studies [8, 33, 34] present error characterization for the GPU-enabled Cray supercomputers such as the NCSA Blue Water and Titan supercomputer. They study the spatial and temporal characteristics of GPU errors, how these errors propagate spatially in a short time-window, frequency of GPU errors in different memory structures of a GPU, correlation between batch jobs and correctable GPU errors, etc. These efforts have primarily focused on understanding XID errors, manufacturing errors (e.g., Off the Bus error), and its effect on application-execution. These studies have also shown via neutron beam testing that more recent generation of GPUs are more error resilient than previous generation of GPU architecture. These studies have also focused on issues and challenges with current GPU error logging methods. Previous efforts by Haque et al. [16] have deployed a software-based GPU soft-error detector on Folding@home distributed platform for two different architectures, the G80 and GT200 architectures. They showed that newer generation of GPUs observed significantly lower soft error rate. Additionally, they found that the GPUs were sensitivity to memory faults in a pattern-dependent manner.

However, none of these studies study presents detailed analysis and characterization of soft-errors on GPUs at large-scale. Our study discovers several previously unknown insights about the characteristics of single bit errors, dynamic page retirement errors, and double bit errors. For the first time, we characterize the temporal characteristics of single bit errors and its association with other errors. In contrast to previous works, we investigate the impact of workloads, resource utilization, and variance in load-level on error-affected GPU nodes in detail. Our study also provides a deep understanding of possible temperature effects on soft-errors in GPU architecture, and quantification of performance variation on soft-error affected GPU nodes. Given that GPUs are likely to be an important part of an exaflop HPC system, we believe that our study with the world's largest GPU-enabled system will help the whole community in improving the understanding the impact of GPU errors on scientific applications and its implications for large-scale GPU resource management.

8. Conclusion

In this study we focus on single bit errors, dynamic page retirement errors, and double bit errors on Titan's GPUs and analyze their characteristics and relationships with resource usage, applications, users, and temperature. Our study discovers several previously unknown insights about the characteristics of SBE, DBE, and DPR errors. For example, we show that SBEs happen in bursts and tend to be clustered in time. Average GPU resource utilization and its variance do not seem to be significantly correlated with the SBE occurrences, but shows strong dependence with respect to users and applications. Interestingly, our analysis also shows that top SBE offending GPUs do not necessarily experience more dynamic page retirement errors or DBEs. Another counter-intuitive finding is that SBEs are more likely to occur on the DPR offending GPUs after the DPR error rather than leading to the DPR error. We also provide interesting and deep analysis about possible performance-variation effects of soft-errors and its association with temperature.

9. Acknowledgment

We thank the reviewers for their feedback that has significantly improved the paper. This research is partially supported by NSF grant CCF-1218758. This work also used the resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is managed by UT Battelle, LLC for the U.S. DOE (under the contract No. DE-AC05-00OR22725).

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] "Cuda c programming guide," https://docs.nvidia.com/cuda/cuda-c-programming-guide.

[3] "Dynamic page retirement," http://docs.nvidia.com/deploy/dynamic-page-retirement/index.html.

[4] "Ibm and nvidia launch supercomputer centers of excellence with the u.s. department of energy's oak ridge and lawrence livermore national labs," http://www-03.ibm.com/press/us/en/pressrelease/47318.wss.

[5] "Rodinia: Accelerating compute-intensive applications with accelerators," http://www.cs.virginia.edu/~skadron/wiki/rodinia/index.php/Rodinia:Accelerating_Compute-Intensive_Applications_with_Accelerators.

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

[7] L. N. Bairavasundaram, *Characteristics, impact, and tolerance of partial disk failures*. ProQuest, 2008.

[8] 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*.

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

[10] N. El-Sayed, I. A. Stefanovici, G. Amvrosiadis, A. A. Hwang, and B. Schroeder, "Temperature management in data centers: why some (might) like it hot," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1, pp. 163–174, 2012.

[11] S. Fu and C. Xu, "Quantifying temporal and spatial correlation of failure events for proactive management," in *Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium on*. IEEE, 2007, pp. 175–184.

[12] A. Gainaru, F. Cappello, J. Fullop, S. Trausan-Matu, and W. Kramer, "Adaptive event prediction strategy with dynamic time window for large-scale hpc systems," in *Managing Large-scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques*. ACM, 2011, p. 4.

[13] A. Gainaru, F. Cappello, M. Snir, and W. Kramer, "Fault prediction under the microscope: A closer look into hpc systems," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 77.

[14] 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.

[15] S. Gupta, D. Tiwari, C. Jantzi, J. Rogers, and D. Maxwell, "Understanding and exploiting spatial properties of system failures on extreme-scale hpc systems," *International Conference on Dependable Systems and Networks (DSN)*, 2015.

[16] I. S. Haque and V. S. 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*. IEEE, 2010, pp. 691–696.

[17] 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.

[18] L. M. Leemis and S. K. Park, *Discrete-Event Simulation, A First Course*. Prentice Hall, 2006.

[19] 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.

[20] 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.

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

[22] C. L. Mendes, B. Bode, G. H. Bauer, J. Enos, C. Beldica, and W. T. Kramer, "Deploying a large petascale system: The blue waters experience," *Procedia Computer Science*, vol. 29, pp. 198–209, 2014.

[23] J. Meza et al., "A large-scale study of flash memory errors in the field," *ACM SIGMETRICS Performance Evaluation Review*, 2015.

[24] 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.

[25] "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."

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] ——, "Understanding failures in petascale computers," in *Journal of Physics: Conference Series*, vol. 78, no. 1. IOP Publishing, 2007, p. 012022.

[31] 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.

[32] 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.

[33] D. Tiwari, S. Gupta, G. Gallarno, J. Rogers, and D. Maxwell, "Reliability lessons learned from gpu experience with the titan supercomputer at oak ridge leadership computing facility," *Proceedings of SC15: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2015.

[34] D. Tiwari, S. Gupta, J. Rogers, D. Maxwell, P. Rech, S. Vazhkudai, D. Oliveira, D. Londo, N. DeBardeleben, P. Navaux *et al.*, "Understanding gpu errors on large-scale hpc systems and the implications for system design and operation," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on*. IEEE, 2015, pp. 331–342.

[35] D. Tiwari, S. Gupta, and S. S. Vazhkudai, "Lazy checkpointing: Exploiting temporal locality in failures to mitigate checkpointing overheads on extreme-scale systems," in *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*. IEEE, 2014, pp. 25–36.

[36] J. S. Vetter, R. Glassbrook, J. Dongarra, K. Schwan, B. Loftis, S. McNally, J. Meredith, J. Rogers, P. Roth, K. Spafford *et al.*, "Keeneland: Bringing heterogeneous gpu computing to the computational science community," *Computing in Science and Engineering*, vol. 13, no. 5, pp. 90–95, 2011.

[37] J. Xue, F. Yan, R. Birke, L. Y. Chen, T. Scherer, and E. Smirni, "Practise: Robust prediction of data center time series," in *Proceedings of the 11th International Conference on Network and Service Mangement (CNSM 15)*, 2015.