Unprotected Computing : A Large-Scale Study of DRAM Raw Error Rate on a Supercomputer

Leonardo Bautista-Gomez, Ferad Zyulkyarov, Osman Unsal
Barcelona Supercomputing Center, Barcelona, SPAIN

Simon McIntosh-Smith
University of Bristol, Bristol, UK

***Abstract*—Supercomputers offer new opportunities for scientific computing as they grow in size. However, their growth also poses new challenges. Resilience has been recognized as one of the most pressing issues to solve for extreme scale computing. Transistor scaling in the single-digit nanometer era and power constraints might dramatically increase the failure rate of next generation machines. DRAM errors have been analyzed in the past for different supercomputers but those studies are usually based on job scheduler logs and counters produced by hardware-level error correcting codes. Consequently, little is known about errors escaping hardware checks, which lead to silent data corruption. This work attempts to fill that gap by analyzing memory errors for over a year on a cluster with about 1000 nodes featuring low-power memory without error correction. The study gathered millions of events recording detailed information of thousands of memory errors, many of them corrupting multiple bits. Several factors are analyzed, such as temporal and spatial correlation between errors, but also the influence of temperature and even the position of the sun in the sky. The study showed that most multi-bit errors corrupted non-adjacent bits in the memory word and that most errors flipped memory bits from 1 to 0. In addition, we observed thousands of cases of multiple single-bit errors occurring simultaneously in different regions of the memory. These new observations would not be possible by simply analyzing error correction counters on classical systems. We propose several directions in which the findings of this study can help the design of more reliable systems in the future.**

I. Introduction

High-performance computing (HPC) is an important tool for scientific discovery. Next generation supercomputers will achieve unprecedented performance thanks to millions of computing cores working on the same scientific problem. However, as these machines grow, new problems arise. Reliability is considered one of the most challenging obstacles to achieve exascale computing [1]. The growing number of components is increasing the frequency at which supercomputers experience failures. If each processor of a machine has a mean time to failure (MTTF) of 25 years [2], then a supercomputer with one hundred thousand of those processors will have a mean time between failures (MTBF) of only two hours.

Besides the number of components, the computing throughput of each component is also increasing. This is achieved by embedding more transistors in each computing device through a process called *transistor scaling*. While this has certain benefits, decreasing feature size also decreases the critical charge (i.e., Q_{Crit}) required to change the logic level of a circuit. Such changes are called *upset events* and lead to data corruption, which are generally called *soft errors* in HPC. Such upset events can have several root causes, such as pollution in the packaging of the device, excessive temperatures or even energetic neutrons originating from cosmic rays [3].

The situation is worse due to power constraints [4]. To limit the power consumption of devices, industry manufacturers are decreasing voltages, which again increases the probability of data corruption [5], [6]. While most modern devices are protected against data corruption with mechanisms such as error correcting codes (ECC), it is unclear whether such mechanisms will be enough to prevent data corruption from occurring at extreme scale [7]. A recent study [8] identified DRAM as more of a challenge for exascale compared to caches composed of SRAM. In addition, such mechanisms are not flawless, meaning that certain corruptions (i.e., more than 2 corrupted bits) could occur without being detected, leading to silent data corruption (SDC). Little is known about SDC rates in supercomputers today, because of the very nature of such events (i.e., silent). SDC is often estimated as more damaging than fail-stop errors because they could lead to scientific results being produced that were unknowingly erroneous [9].

While some studies [10], [11], [12] have analyzed the error rates on standard HPC systems, they often rely on scheduler logs and ECC counters. Manufacturers *estimate* the DRAM raw error rate by disabling ECC and exposing the DIMMs to particle accelerators [13]. However, those estimates are not exact as those accelerated soft error studies fail to consider factors such as the impact of temperature or neutron flux variation due to the sun's position relative to a memory device. In this work we adopt a more radical approach: we perform a large-scale DRAM raw error rate study on an unprotected supercomputer. To reach this goal, we strip the ECC off a prototype computing node and build a cluster with almost a thousand of such *unprotected* nodes. The cluster is dedicated for debugging, benchmarking and fault tolerance research. We monitored memory errors in this system for over a year and perform a detailed analysis of the statistics that were gathered. In addition to collecting the statistical data about the frequency and nature of the observed memory errors, we attempt to uncover patterns and find correlations. The objective is to leverage that knowledge to understand failure patterns and take pro-active adjustments that could lead to higher user-perceived MTBF at large scale. Also, recording and analyzing the DRAM raw error rate (as opposed to error rate of DRAM shielded by ECC or chipkill) could help guide

SC16; Salt Lake City, Utah, USA; November 2016
978-1-4673-8815-3/16/$31.00

future design of hardware and software resiliency mechanisms for DRAM. Ultimately, this analysis could give us a glimpse of the failure rates for extreme scale systems if we do not reach the reliability level desired at that scale.

The contributions of this work are as follows.

- We monitor DRAM memory errors [1] on a system with over 900 ECC-less nodes for over a year. We gather millions of events and log the location, time, temperature, memory address and other conditions of the system.
- We analyze the gathered data and present a statistical study about soft errors in the prototype machine, including error frequency, corruption magnitude and MTBF for different nodes in the system. We present evidence that demonstrates a strong spatial and temporal correlation between DRAM memory errors.
- We study multiple factors such as temperature and the position of the sun in the sky that could affect the behavior of the machine and make it more prone to DRAM memory corruptions. We present evidence which suggests that multi-bit errors are more likely to occur during day time with a high peak when the sun is at the highest point in the sky.
- We discuss some clear correlations and propose some mechanisms to adapt and avoid failures. We classify periods of time with high fault rates and suggest a shortening in the checkpoint interval in order to adapt to the reduced MTBF. In addition we propose a quarantine strategy to cope with failure bursts and intermittent faults.
- We study intermittent faults, bursty periods of failures, and multiple potential cases of silent corruption. We provide detailed information showing which bits are flipped for multi-bit errors and we found that most multi-bit errors occur in non-adjacent bit positions in a word.
- We present evidence of multiple SDC occurring in a completely uncorrelated fashion, proving to be extremely hard to detect and/or predict.

The rest of this paper is composed as follows. Section II describes the prototype machine and explains the data collection methodology. Section III analyses the data gathered from different perspectives and tests multiple correlation hypotheses. Section IV discusses correlations and proposes several methods to adapt and avoid faulty behaviours. Section V gives an overview of the related work. Section VI concludes the paper and provides some future research directions.

II. Hunting for Memory Corruptions

Our goal is to characterize DRAM memory errors on a system without hardware ECC. To reach this goal we need a physical machine with the required hardware characteristics that will allow such a study. Given that the machine does not detect errors at the hardware level, we also need to develop a software level memory corruption detection mechanism that could run for long periods of time to gather enough data to guarantee statistical significance. Last but not least, we need a rigorous and clear methodology about how to recognize and define failures for this study.

[1]For the remainder of this paper we denote memory to mean DRAM memory (thus excluding on-chip caches)

A. The Prototype

The installed prototype has system-on-chip (SoC) nodes, each one with 2 ARM cores running at 1.7GHz, 4GB of low-power DRAM (LPDDR) without ECC and one GPU. We note that a lower supply voltage makes the chip more susceptible to soft errors or silent data corruption since the critical charge to flip a bit is lower with lower voltage. The supply voltage of LPDDR is lower than DDR at the same technology node (for example 1.2V for LPDDR3 versus 1.5V for DDR3). Since supply voltage is going to scale down for exascale technology nodes (10nm and below), the reliability qualification conducted on LPDDR of current technology is a good proxy for the future exascale era DDRs.

There are 15 SoC nodes packaged into one blade and the system is composed of 72 blades, for a total of 1080 nodes. All nodes together with the network switches are hosted in 2 racks, each rack has 4 chassis and each chassis has 9 blades. The theoretical peak performance of the system is 35 TFLOP/s double precision. The machine is located in Barcelona at an altitude of about 100 meters above sea level.

Since this prototype machine is used for multiple research projects, one chassis (i.e., 9 blades) was dedicated for another study and did not take part on this memory reliability characterization, leaving only 945 nodes available for the study. In addition, 9 nodes were dedicated as login nodes and other nodes had permanent hardware failures. In total, from the 1080 nodes, 923 were continuously scanned for memory errors from February 2015 to February 2016 inclusive.

B. Memory Error Scanning Tool

In order to characterize the memory errors in this prototype, we started a memory reliability characterization study in February 2015 using a simple memory scanning tool. For each node, the memory scanning tool creates log entries when it starts/stops scanning and when it detects a memory error. These log entries are stored in log files with each node having a separate log file. We orchestrate the execution of the memory scanner tool only when the node is idle (i.e., not executing any job) by using the job scheduler and the epilogue and prologue scripts. When a job completes, an epilogue script is run automatically. The epilogue script contains a command that triggers the start of the memory scanning tool. The memory scanner continues executing on the node until a new job is scheduled. When a new job is scheduled on that node, before the job starts, a prologue script is executed. This prologue script contains a command which terminates the execution of the memory scanner by sending a SIGTERM signal.

When the memory scanner starts on a node, it attempts to allocate 3GB of memory, which is the largest amount of memory applications can allocate on a node in this machine (the rest is dedicated for the OS and other libraries). If the allocation fails (because of memory leaked by a previously running application), the requested amount of memory for

allocation is decreased by 10MB and repeated until a successful allocation or until the amount for allocation becomes 0MB. If memory is allocated successfully, a START log entry is created. The START log entry contains a time stamp, the amount of allocated memory, host name and the temperature of the node. If memory allocation fails, that information is also logged in a separate file and contains a time stamp and the host name. After successful memory allocation, the memory scanner begins the execution of an infinite loop. Inside the loop every memory word is written with a specific value (i.e., $0x00000000$). At every iteration, the values are checked and updated with the opposite value (i.e., $0xFFFFFFFF$ if the previous one was $0x00000000$ and vice versa). If the expected and actual values do not match, an ERROR log is created. The ERROR log contains the time stamp, host name, virtual address, actual value, expected value, temperature and physical page address. We also tested another way to affect values: we start with $0x00000001$ and then keep increasing by 1 at every iteration. Most of the study was done using the former method in an attempt to stress equally all the bit positions of the memory. While we acknowledge that these write patterns might not be representative of real world applications, it is hard to capture the memory write patterns of a wide variety of HPC applications. Both value affecting strategies log exactly the same data when an error occurs. The memory scanner exits the infinite loop when it receives a SIGTERM signal. Then it logs an END command. The END command contains a time stamp, host name and the temperature of the node. In some rare cases, the node was manually rebooted and no END command was logged. This will produce a START event followed by another START event, making it impossible for us to know how many hours the memory monitor was running (i.e., when the hard reboot occurred). In such scenario, we took a conservative approach and we assumed 0 hours of memory monitoring, which leads to a slight underestimation of the total number of hours the system was monitored.

C. Error Extraction Methodology

In addition to the prototype system and the detection tool, it is critical to explain the error accounting methodology. As explained above, the scanning tool logs every error observed in the system. However, not every error log is an independent error. In many cases, a fault in a memory cell manifests as many consecutive error logs over time, but they are all related to the same original root cause: a fault in one memory cell. Even if such a fault produced many incorrect values for thousands of consecutive iterations, **we count this as one single memory error** [2].

Another special case that requires particular attention is when in one single iteration we observe multiple single-bit errors in different memory addresses. Such errors would manifest as multiple ECC corrections in a classical system with ECC but given that they occur at the same time, it would not be correct to consider them as multiple independent errors. In fact, they are highly likely to originate from the same root cause, as we will show later. Thus, we also analyze such simultaneously occurring events (See Section III-B).

[2]Given that we filter multiple error logs originating on the same root cause and count them as one single fault, we use the terms memory fault and memory error interchangeably in the reminder of this paper

III. Failure Analysis

The study lasted for over a year, accumulating over 4.2 million node-hours of error monitoring and logging over 25 million error logs from over 900 nodes with ECC-less low-power DRAM. The length and scale of this study were carefully decided in order to guarantee the statistical significance of this DRAM memory error characterization.

A. Memory scanned

The study covered a total of 12,135 Terabyte-hours of memory analysis. Figure 1 shows the total number of hours that each node was scanned during the entire period of the study. In the figure, we map the system in 63 blades with 15 SoC per blade, each SoC being an independent node. We see that the first blades do not perform any error monitoring in the first SoC; this is because they are dedicated as login nodes.

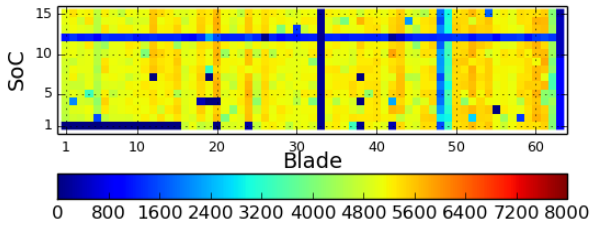

Fig. 1. Hours each node was scanned for memory errors

We also notice that the SoC 12 of most blades did not get much monitoring time. This is due to the fact that those SoC showed significant temperature issues because of their location in the rack. Given that they tend to overheat, and to produce heat for other nodes, the system administrators decided to turn them off for long periods of time. Blade 33 was also shutdown during the year due to hardware issues. We also notice that the number of monitoring hours is not necessarily the same for all the SoCs of a given blade. There are a few SoC that never got scanned as they were shutdown due to hardware issues. Most nodes got about 5000 hours of error monitoring, which is more than half of the total period of the study (i.e., about 7 months out of 13). This large number of monitoring hours gives us a fair degree of statistical confidence on the results.

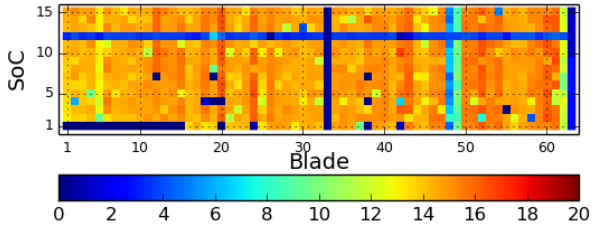

Fig. 2. Amount of memory analyzed per node (Terabyte-Hours)

The number of hours each node was analyzed gives an idea of the amount of memory scanned in the system. However, a node with more analysed time does not necessarily translate into more memory scanned. This is because for each scan, the amount of memory could be different (see Section II-B). Figure 2 shows the total terabyte-hours that each node scanned during the period of the study. We observe a strong correlation between this figure and the previous one, showing that a node with more hours of monitoring also scanned more memory, but we also see the presence of a few more marked differences between SoCs, compared to the previous figure. Overall, the vast majority of nodes scanned about 15 terabytes-hours, showing a rather homogeneous distribution.

B. Failure Rate

At the end of the study, the error monitoring tool had logged over 25 million errors. Such a large number of errors is completely out of the expected failure rate for the studied prototype in normal conditions (i.e., machine located at sea level, proper cooling system, etc.). As previously mentioned, not every error log is an independent memory error (see Section II-C). Moreover, a simple analysis showed that over 98% of the observed failures came from the same node. This node was a faulty node that was removed from the job scheduler pool and is a classic case of a node that gets replaced in production systems. Thus, this node was also removed from the error characterization study. After these filters, the system logged over $55,000$ independent memory errors, which corresponds to a node experiencing a memory error every 41 hours, or the cluster experiencing an error every 10 minutes. This raw failure rate is somewhat larger than expected but it can be explained as we will see in the rest of this study.

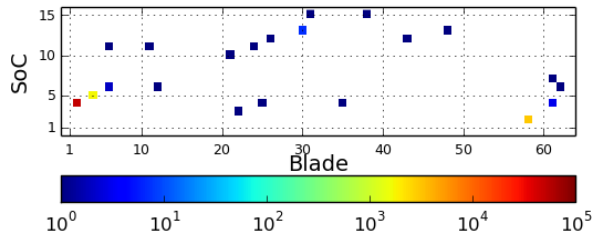

Fig. 3. Number of independent memory errors for each node

To analyze how the memory errors were distributed in the system, we plot another heat map this time showing the raw number of memory errors observed for each node during the study. Figure 3 shows the results using a logarithmic color scale because the difference in the number of failures varies drastically from some nodes to others. A first observation is that most of the nodes did not show any failure during the study (white color in the figure). Also, most nodes with failures had only one failure (dark blue spots). A few other nodes had thousands of failures (yellow, orange and red spots). Note that these represent the raw number of errors and does not take into account the amount of time each node was scanned for errors. However, the most faulty nodes show orders of magnitude higher error rates than most other nodes, much higher than the difference of time scanned between nodes.

C. Multiple Simultaneous Corruptions

In computing systems, it is well known that corruption can affect multiple bits at the same time. Memory devices protected with ECC often make use of single error correction/double error detection (SECDED) codes to *guarantee* the correction of single-bit errors and the detection of double-bit errors. Note that in some cases, SECDED could also detect more than 2 bits getting corrupted, depending on which bits are affected, but this is not guaranteed. It is usually assumed that multi-bit errors are much less frequent than single-bit errors. Most modern devices protected with ECC have counters that count the number of single-bit errors corrected and multi-bit errors detected.

During this study we observed a total of 85 failures which corrupted multiple bits of a memory word. From those 85 cases, 76 were double-bit errors, which would be detected in a SECDED-protected system, potentially producing a process or system crash depending on the location and severity of the error in system memory space. The other 9 memory errors corrupted more than 2 bits, which could pass undetected by the ECC protection, leading to silent data corruption.

TABLE I
MULTI-BIT CORRUPTIONS AFFECTING THE PROTOTYPE.

Number bits corrupted	Expected Value	Corrupted Value	Ocurrences	Consecutive
2	0x000016bb	0x000016b8	1	Yes
2	0xffffffff	0xffffeeff	2	No
2	0x000003c1	0x000003c2	2	Yes
2	0xffffffff	0xffff7dff	4	No
2	0xffffffff	0xfffff5ff	4	No
2	0xffffffff	0xfffff3ff	7	Yes
2	0xffffffff	0xfffff9ff	10	Yes
2	0xffffffff	0xffff77ff	10	No
2	0xffffffff	0xffff7bff	36	No
3	0xffffffff	0xffff75ff	1	No
3	0xffffffff	0xfffff1ff	1	Yes
4	0x00000461	0x00006e61	1	No
4	0x00002957	0x00002958	1	Yes
4	0x000071b2	0x00007100	1	No
5	0x000002e4	0x00000215	1	No
6	0x00006ab4	0x00006a5a	1	No
8	0xffffffff	0xffffff00	1	Yes
9	0x00000058	0xe6006358	1	No

Given that the memory scanning tool logged the expected and the corrupted value, it is possible to give a detailed list of the multi-bit errors observed in the system. Table I shows the detailed information of all the observed multi-bit errors. The largest number of bits that was corrupted on a memory word was 9, which is somewhat unexpectedly high. Interestingly enough, the majority of multi-bit errors did not corrupt consecutive bits. In fact, 3 bits is the average distance between corrupted bits in the same memory word and the maximum observed distance is 11 bits for this system. This could be due to DRAM layout spreading the adjacent bits of the word. Usually this *scrambling* is done to avoid resonance on the bus. This observation suggests that correcting codes optimised for adjacent bit errors are less effective.

It is also worth highlighting that about 90% of corrupted bits switched from 1 to 0 and only 10% the other way around. This is an indication that in the large majority of corruptions, the affected memory cell loses some charge. Another observation is that the majority of the multiple bit corruptions occur in the least significant bits of the word.

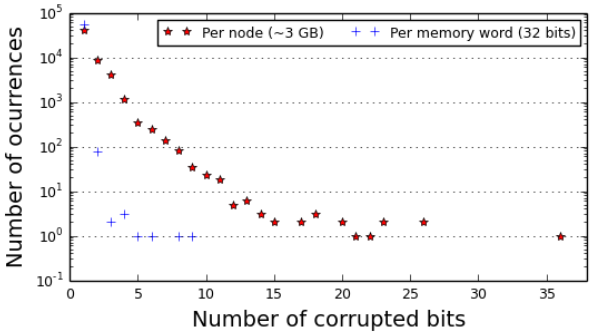

Fig. 4. Simultaneous memory errors VS multi-bit errors

One of the advantages of running this experiment on a prototype machine without ECC and with a memory scanning tool that does not crash when corruption happens, is that it is possible to observe error patterns that would not be visible otherwise. For instance, counting the ECC single-bit corrections in a classical system could lead to a wrong perception of how frequently multiple parts of the memory get corrupted simultaneously. Given that the memory scanning tool logs the exact timestamp of each detected error, it is possible to find when multiple corruptions affect different memory words simultaneously. This study revealed that over $26,000$ corruptions occurred simultaneously to other corruptions in the same node. Over 99.9% of those were multiple single-bit corruptions that occurred simultaneously in different parts of the memory of the same node. These would usually be observed as single-bit corruptions in standard systems because they affect only one bit per ECC word, but in reality they affect multiple bits on the same node. In fact, it was found that one such failure could corrupt up to 36 bits spread across different memory words. In addition, 44 double-bit corruptions happened simultaneously with a single-bit corruption in another region of the memory. Similarly, the study revealed 2 cases of triple-bit corruption happening simultaneously to a single-bit corruption. Moreover, we observed a case of two double-bit corruptions occurring simultaneously. In other words, most double-bit errors were accompanied by other errors occurring simultaneously in other regions of the memory. We suspect that the affected memory cells are in physical proximity or alignment (row, column, bank) however the memory controller maps them to different address words. These observations suggest that the root cause of such simultaneous errors could be linked to local hardware defects due to manufacturing variability [14], but they could also be related to external factors that affect multiple regions of the devices at the same time.

If one would consider these simultaneously occurring errors as multi-bit corruptions, then the number of multi-bit errors is much higher. Figure 4 shows the number of multi-bit errors depending on whether we count multi-bit errors in a *per node* basis (i.e., regardless of how they affect different memory words), and in a *per memory word* basis (i.e., only those corruptions that affect multiple bits of the same memory word). We note that *per node* multi-bit errors are orders of magnitude more frequent than *per memory word* multi-bit errors, which is expected given the difference in size (i.e., 3GB vs 32 bits). However, single-bit errors are much less frequent on a *per node* basis, in comparison to *per memory word* single-bit errors. This might seem counter-intuitive: one would expect *per node* errors to always be higher or equal to *per memory word* equivalents given the difference in memory size, but the reason for this is that thousands of *per memory word* single-bit errors occur simultaneously in other memory regions of the same node, therefore they are counted as multi-bit errors in the *per node* analysis. That is to say, the tens of thousands of *per memory word* single-bit errors become *per node* multi-bit errors, keeping the total number of corruptions constant.

For the remainder of this paper, we consider as multi-bit errors, only those errors affecting multiple bits of the same memory word (i.e., the standard definition of multi-bit errors). Nevertheless, it is interesting to highlight that phenomena causing data corruption in spatially distinct locations in DRAM is much more frequent than previously assumed. Also, ECC event counters could be misleading if they do not take into account time correlations between multiple events. More importantly, single-bit errors detected by ECC are likely to be accompanied by other single-bit errors in its proximity and in some cases, linked to simultaneous uncorrectable errors in the same node. This correlation gives us opportunities to detect faulty behaviour before catastrophic failures occur.

D. Relation Between Detectable and Undetectable Errors

The previous analysis gives us some correlations between correctable and uncorrectable (detectable) errors. Now, we try to analyze the relation between detectable and undetectable errors. In particular, we focus on errors with more than 3 bit-flips (i.e., the last 7 lines of Table I). We looked at the logs of the nodes in which those undetectable errors occurred, in search for other detectable errors occurring nearly at the same time than the undetectable errors. Surprisingly, not only we did not find any errors occurring at the same time, but in addition, those nodes did not show any other error during the entire study. Those 7 undetectable errors occurred in 5 different nodes that did not show **any** other error in the whole period. In fact, 4 of those undetectable errors occurred in a node that had only that one error. Moreover, other nodes did not log any strange activity at the time of those undetectable errors. That is to say, *those cases of undetectable errors are extremely silent* in that they are both undetectable by hardware mechanisms and completely uncorrelated to detectable errors.

We noticed that 6 of these errors occurred before we turned off the overheating nodes and 4 of the concerned nodes are located near the SoC 12 (i.e., the overheating SoCs). Overheating could have partially damaged part of the memory making them more prone to SDC. Unfortunately, these errors

occurred before we started login the temperature of the system, so we do not know the temperature of the concerned nodes at the moment of the corruption. However, temperature seems to be an unlikely direct cause given that no other errors where logged. Independently of the root cause, it is surprising that no any other corruption was detected in those nodes or in other nodes at the same time.

E. Memory Errors VS Time of Day

Given that all the error occurrences were logged with their corresponding date and time, we can plot the number of memory errors observed at each hour of the day. Figure 5 shows the total number of observed memory corruptions for each hour of the day, for different numbers of corrupted bits. Note that corruption with more than 5 bit-flips are rare, thus we group them all in a single group ("6+").

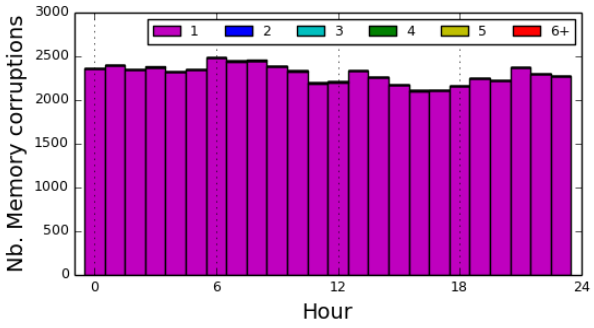

Fig. 5. Number of errors per hour for different number of corrupted bits

Clearly, single bit-flips (magenta) are the predominant type of memory errors, as shown in the analysis above. We observe a rather homogeneous distribution of memory errors through the day; that is to say, when we look at all the corruptions logged during our study, we do not find any particular time of day where memory errors are more or less frequent.

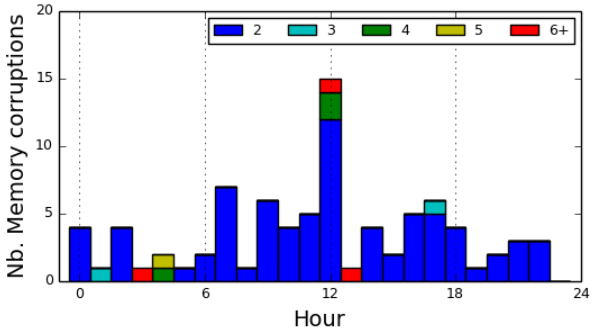

Fig. 6. Number of memory errors per hour for multi-bit corruptions

Given that single bit-flips are so predominant, we perform a second analysis in which we filter single bit-flips and we focus on multi bit-flips only. This is plotted in Figure 6. We found that the number of multi-bit corruptions between 7am and 6pm is double the number of multi-bit corruptions during the night. The distribution seems to have a bell shape with its highest point at noon. This got our attention because, in contrast with single-bit corruptions, there seems to be a high correlation between the position of the sun and the number of multi-bit corruptions. It is known that neutron showers, caused by the interaction between the solar wind and our atmosphere, can affect electronic components [15], [16]. The results of this study point in that direction, and suggest that multi-bit memory errors are mostly caused by cosmic rays. This is also supported by the previous analysis showing that multi-bit errors are often accompanied by errors in other memory regions.

F. Memory Errors VS Temperature

We study whether there is any correlation between memory errors and high temperatures. We started logging the temperature of the nodes in April 2015, so during the first months of the study we do not have information about the temperature when an error occurred. Nonetheless, we have over nine months of data with thousands of memory errors and their respective node temperature, which give as a statistically significant dataset. The room temperature was maintained between $18°\mathrm{C}$ and $26°\mathrm{C}$ during the whole period of the study.

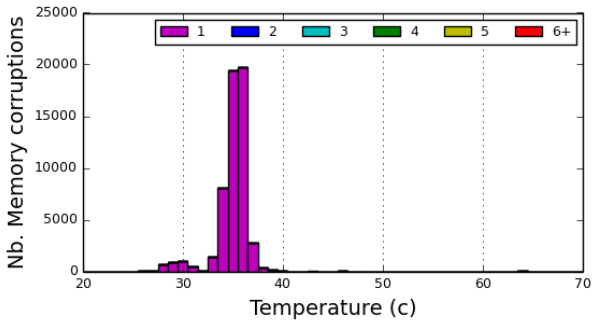

Fig. 7. Memory errors vs temperature for different numbers of corrupted bits

The temperature of the node for the different types of memory errors (i.e., single bit-flips, etc.) is shown in Figure 7.

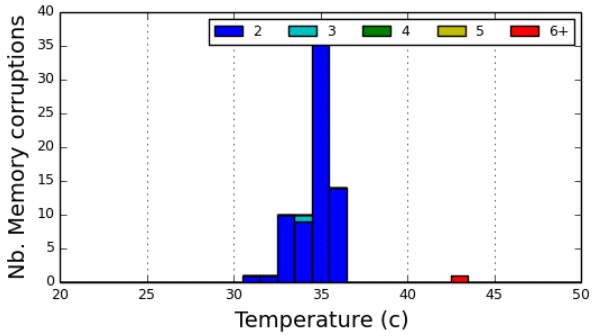

Fig. 8. Number of errors vs temperature for multi-bit corruptions

There is a small set of memory errors that show a node temperature over $60°C$ which is higher than the normal working temperature for a compute node and those errors could be temperature induced. Nevertheless, most errors happen when the node has a temperature between $30°C$ and $40°C$, which

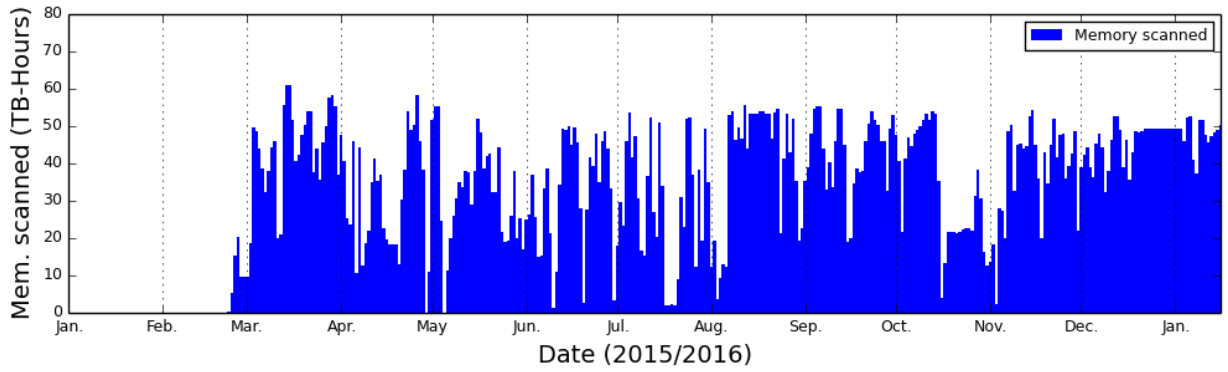

Fig. 9. Total amount of memory scanned per day (in Terabyte-Hours)

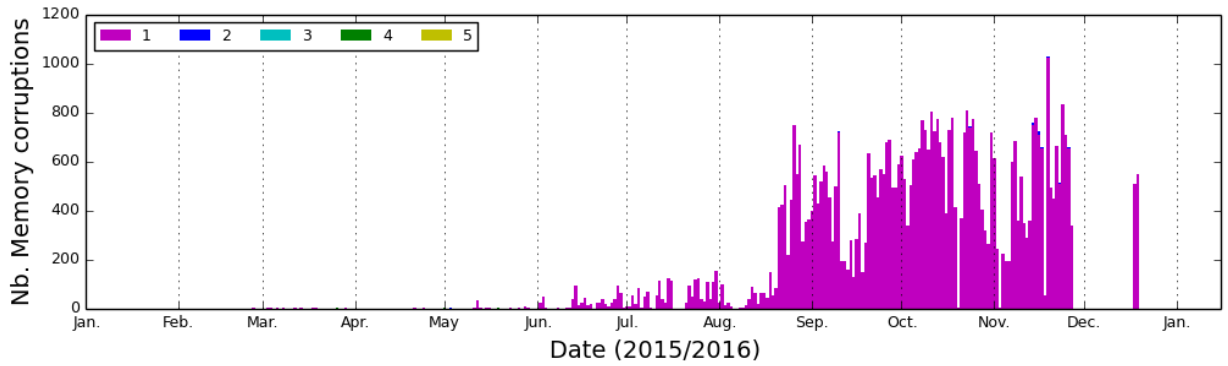

Fig. 10. Number of errors per day for different number of corrupted bits

is a nominal temperature for a computing node. Note that the memory scanning tool does not stress the CPU, hence the low temperatures. Thus, we observe no high correlation between temperature and high memory error rates in our test system.

Again, we do a second study in which we filter single-bit corruptions and focus on multi-bit corruptions and their respective node temperature. As we can see on Figure 8, all multi-bit corruptions occur at nominal temperatures and we do not see any correlation between high temperatures and multi-bit errors (for those errors with temperature information). It is important to note that the SoC 12 of the blades were turned off due to temperature issues. Also, our memory error monitoring program does not stress the CPU, hence the low temperatures observed. This analysis, does not challenge previous findings that show correlations between high temperature and errors, and we do not claim that such a correlation does not exist. We simply state that given this error gathering methodology, the hardware is not stressed enough to expose that phenomena and this is reflected in the data gathered.

G. Memory Errors VS Memory Scanned

The next analysis aims to study if there is any correlation between the amount of memory scanned per day and the number of memory corruptions observed each day. Figure 9 shows the number of terabyte-hours scanned each day. We observe large periods of intense memory scanning in August, September and December which seem to coincide with the low activity periods of academic vacations. We also observe lower levels of memory scanned from April to July which seems to coincide with the last part of the academic year.

If we look at how many memory errors were observed each day, we get a rather different picture. As shown in Figure 10, we can see that there are more memory errors from September to December and rather fewer ones in the first half of the year. To quantify this contrast we performed a correlation analysis between both datasets using Pearson correlation. We found that these two datasets have a Pearson correlation of -0.17966 with a p-value of 0.0002; this is a rather low level of anti-correlation between the two datasets. This demonstrates that the memory scanning methodology does not influence in any way the number of memory errors observed.

A second analysis filtering the single-bit corruptions and plotting only the multi-bit memory errors observed each day is shown in Figure 11. Multi-bit corruptions are rather rare, and they only occur a few times during the year, except for several days of unusually high rates of multi-bit errors in November 2015. However, during those dates the system also experienced high rates of single-bit errors, and those might be correlated (see Section III-I). We also observe two undetectable errors occurring the same day in March and May. Although they occur on the same day, those events are separated by hours.

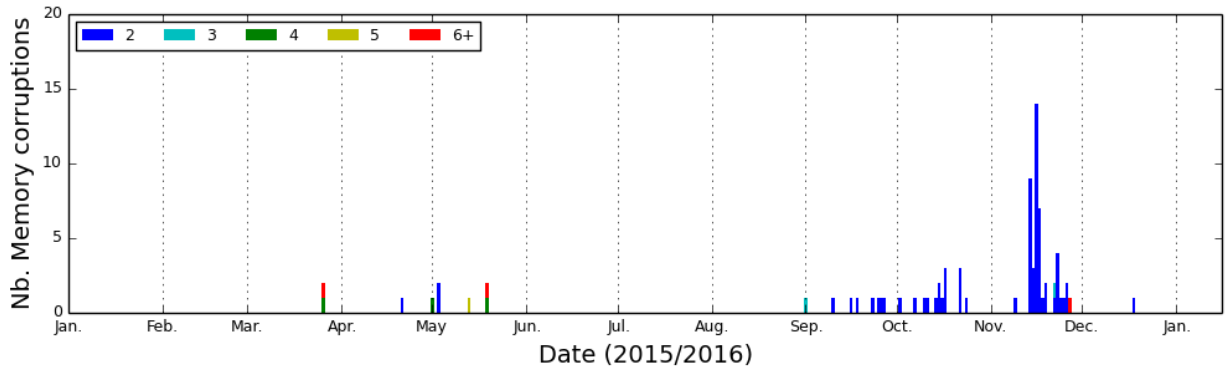

Fig. 11. Number of errors per day for multi-bit errors

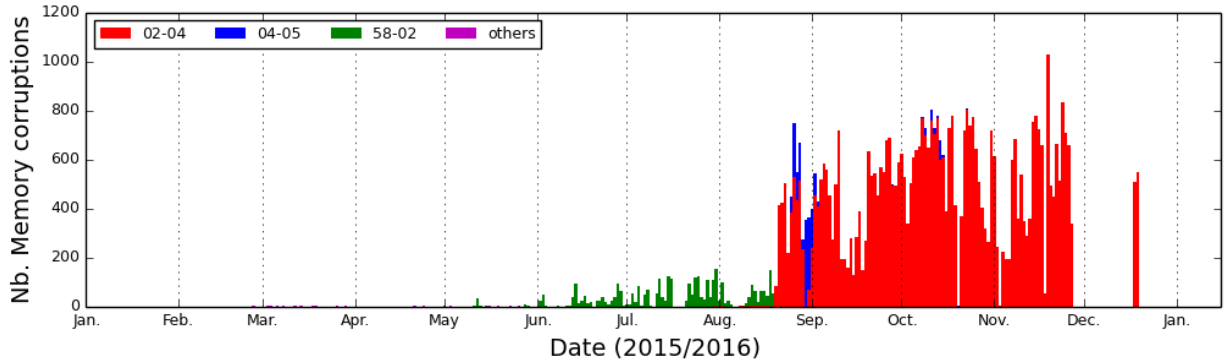

Fig. 12. Number of errors per day for different nodes

H. Spatial Correlation

The space correlation between memory errors can show evidence for different root causes and provide ideas for failure prediction and ultimately failure avoidance. Figure 12 shows memory errors for the three nodes with the highest error rate in the system and all other nodes in another group (purple). Clearly, node $02-04$ (red) is responsible for the majority of memory errors (over $50,000$ of them). The errors seem to stop abruptly toward the end of November, re-appear shortly for a couple of days in December and then disappear again until the end of the study. However, a detailed analysis showed that the reason for those silent periods is due to the fact that no memory monitoring was done on that node during those dates. This means that node $02-04$ started having errors from August and its behaviour degraded to over a 1000 memory errors per day in November without any sign of improvement.

Thanks to the detailed information recorded for the errors, it was possible to count the number of different memory addresses that were affected by errors. Over $11,000$ addresses showed memory corruption with almost 30 different corruption patterns, with the vast majority of them corresponding to single bit-flips switching from 1 to 0. Such large numbers of locations affected in such a random way suggests that corruption might have been happening in another component of the node and not in the memory itself, but it could also be due to a loose DIMM connection or even some capacitative noise in one of the chips. Such permanent failure would force system administrators to replace the failed component in production systems.

The other two nodes (i.e., $04-05$ & $58-02$) showed a completely different behaviour. Although, they were responsible for over five thousand memory errors, absolutely all the memory errors were identical. In other words, the corrupted bit was the same in 100% of the cases (although in a different bit position for each node). This suggests that in these two cases, the intermittent memory errors were caused by a faulty memory cell that would occasionally leak charge. This could be due to what is termed as a *weak bit*. Those bits are faulty due to manufacturing variability, and normally they are caught during *burn-in* [17] which accelerates device aging by subjecting the chips to maximum voltage at very high temperature - typically at $120°C$ - in test ovens before product is shipped. However, error-detection coverage of burn-in to isolate weak bits is not 100% effective, and so sometimes DRAM devices are shipped with *weak bits*. The manifestation of this error only affects a single bit, and are therefore corrected by ECC. We note that if we did the experiment with ECCs we might not be able to see this phenomenon because of the lack of information about which bit was corrupted.

All other nodes combined (i.e., purple) had less than 30 memory errors. This shows a very strong spatial correlation between memory errors, with over 99.9% of errors occurring in less than 1% of the nodes. Such an error pattern is highly

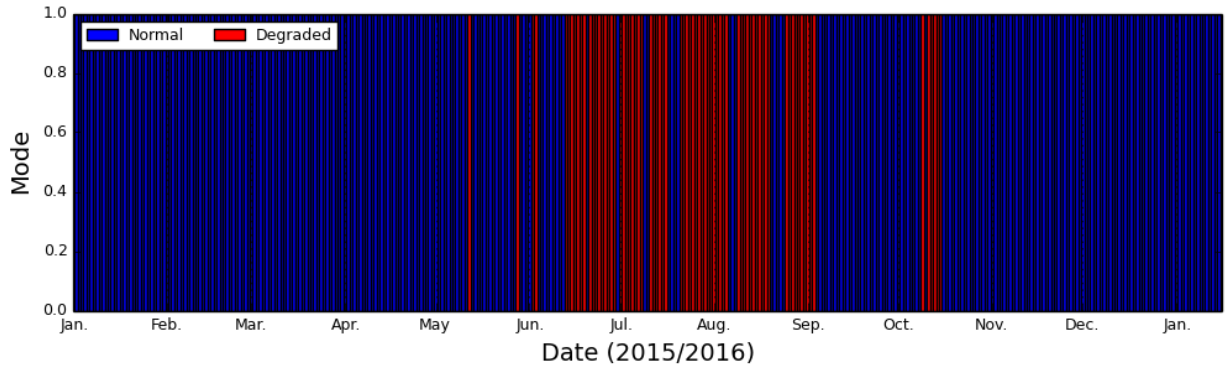

Fig. 13. Regime of the system for each day of the study

beneficial for failure prediction and avoidance. For instance, spatial correlation information can be added into the scheduler algorithm to avoid large high priority jobs running in nodes with a long history of failures. A more aggressive approach would be to run only short *debugging* jobs on those nodes.

I. Temporal correlation

Memory errors are not only clustered in a few nodes, but also clustered in time, as it can be seen in Figure 12. When a node starts having errors, many subsequent errors are observed in the following hours. Those periods of abnormal behaviour could be taken into account while setting resilience parameters, such as the checkpoint interval of a long job. To quantify the benefits of detecting periods of higher error rate, we classify days in two categories. In *normal* conditions, the system observes between one and two memory errors per day. Days with more errors than that show when the system is working in a *degraded* mode. To add a safety margin, we consider any day with three or less errors as *normal*.

Given that node $02-04$ kept having errors until the end of the study (i.e., permanent failure) we assume that such a node would be taken offline on production systems, hence we do not take it into account for computing the MTBF and other similar analyses in the remainder of this paper. During the entire period of this study, the prototype spent 77 days (18.1% of the time) in degraded mode as opposed to 348 days with normal behaviour. The error monitoring tool detected about 50 memory errors occurring during normal days, so the MTBF is 167 hours (i.e., more than six days). In contrast, almost $5,000$ errors were observed during the days on which the system was working in degraded mode; giving as a result a MTBF of only 0.39 hours (i.e., less than 30 minutes). Figure 13 shows in red the days the system suffered from a high error rate.

When the system starts to experience several failures in a short period of time, it is relatively simple to foresee future failures using the spatio-temporal analysis above. This high spatio-temporal correlation has been observed in several recent studies [2], [18], [19]. Thus, the system can adapt to the new MTBF by increasing the checkpoint frequency or using more robust algorithms with extra safety checks.

IV. Failure Avoidance and Resilience Directions

Given the high spatio-temporal correlation between memory errors shown in the previous section, we propose putting compute nodes in *quarantine* as soon as they show an abnormally high error rate. Putting nodes in quarantine has been proposed in the past for nodes that have a long history of failures [20]. What this study suggests is that given the high spatio-temporal correlation of this type of error, it is preferable to put the node in quarantine as soon as it shows abnormal behaviour, instead of waiting for it to create a long failure history.

TABLE II
System MTBF for different quarantine periods

Quaratine period (Days)	Number of errors	Day-Nodes in Quarantine	System MTBF (Hours)
00	04779	000	002.1
05	00131	090	077.9
10	00095	100	107.4
15	00077	135	132.5
20	00067	140	152.2
25	00073	150	139.7
30	00065	180	156.9

We implemented this quarantine algorithm in a simulator and fed it with the error logs gathered during this study. The time a node stays in quarantine is a parameter that can be set by the system administrator. Long quarantine periods can prevent many failures but it can also decrease the productivity of the system. Table II shows the number of failures, the MTBF and the node-days spent in quarantine for different quarantine periods. The results shows that we can obtain a MTBF of almost 157 hours for 30 days of quarantine and the system spends only 180 node-days in quarantine. Please note that this does not mean that the whole system was in quarantine for 180 days, but rather that during 180 days one node was in quarantine from the 945 nodes in the system. This can be translated as a loss in node availability of lower than 0.1% for the whole system. As we see, the number of memory errors occurring per day is reduced dramatically, as any node showing abnormal behavior is immediately put in quarantine. By leveraging the spatio-temporal correlation of memory errors, it is possible to increase the system MTBF by almost three orders of magnitude while imposing a negligible

decrease in productivity. This, however, does not completely remove the possibility of SDC, which has been shown to be a real concern given the results presented in Section III-D.

Another simple strategy that could partially solve some cases of intermittent memory errors is page retirement [21]. This mechanism could be useful in particular for nodes showing evidence of a *weak bit*. Nonetheless, the evidence of multiple single-bit corruptions happening simultaneously in different regions of the memory, leads us to conclude that such a technique would not be effective in all cases.

We acknowledge that the ultimate strategy to efficiently cope with all type of memory errors at extreme scale (including SDC) is not clear yet, but large-scale detailed research studies of unprotected systems, like this one, are important for the design, implementation and testing of future resilience mechanisms at extreme scale.

V. Related Work

The HPC community has arrived to the consensus that reliability is one of the most important challenges to achieve exascale computing [22]. Several reports [23], [1] have highlighted the importance of large-scale failure characterization studies in order to drive technology design and development. In particular, SDC is known to be particularly harmful for these systems and their applications [24], [25], [26].

Several research works [11] have studied the failure rates of HPC systems. Some studies focus on disk failures [27] while others try to analyze all type of failures [10]. Also, a large-scale study [28] of DRAM errors on Google datacenters was done some years ago, showing that error rates are higher than previously reported. Of particular relevance are these large-scale studies of DRAM errors in production systems [21], [29] with a full characterization of hard and soft errors in production systems. These studies are important milestones given the scale of the analysis (i.e., millions of DIMM days, hundreds of Terabyte-years). Unfortunately, since the analysis was based on pre-processed failure logs of detected failures, little information is revealed about multi-bit and silent errors.

Temporal and spatial correlation between failures have been analyzed quite extensively in recent years [19], [2], [30]. The findings of those research works are of major importance for the efficient utilization of extreme scale production systems. They have demonstrated the huge gains that can be obtained by taking into account the spatio-temporal correlation of failures. However, those studies do not target memory errors specifically, hence a detailed analysis of corruption errors (including silent errors) is out of the scope in those papers.

A study [31] of single-bit and multi-bit errors on DRAM memory showed that chipkill ECC offers 42x higher reliability than SECDED ECC. The study also found that DRAM are susceptible to multi-bit errors affecting an entire column or row. In addition, DRAM and SRAM faults in HPC systems have been analyzed and compared [32], in particular to understand the impact of aging on those devices. Interestingly, the authors found a marked shift from permanent to transient failures in the first years of usage. They also found a correlation between SRAM faults and rack positioning. Studies in production systems are critical for the understanding of failure rates in HPC, but they are also limited by the error data gathering methodology, as they cannot get information about which bits are frequently corrupted or when silent errors occur. The analysis of unprotected electronic devices has been restricted to first-level caches [33] showing that reliability is highly application-dependent at that level, which is consistent to different application having different cache miss and cache eviction rates. However, we are not aware of any other large-scale study of unprotected low-power DRAM memory.

VI. Conclusion

This work shows a large-scale study of DRAM memory errors in a low-power unprotected system. The study logged millions of events from over 900 compute nodes for over a year, covering more than $12,000$ terabyte-hours of error monitoring. More than $55,000$ DRAM faults were observed, analyzed and characterized. The study revealed the presence of multiple single-bit errors occurring simultaneously in different regions of the memory, affecting sometimes tens of different memory words. In addition, multi-bit errors were shown to be twice as likely to occur during the day time, with an important increase at noon, which suggests that they could be correlated to the position of the sun in the sky. Moreover, a large fraction of double-bit errors happened simultaneously with single-bit errors in other parts of the memory showing a correlation between correctable and uncorrectable errors. Interestingly, the vast majority of errors happened to switch from 1 to 0 and not the other way around. Another finding worth noting is that most multi-bit errors affected non-adjacent bits in the memory word, sometimes having as much as 11 non-corrupted bits between corrupted bits. Moreover, we present disturbing evidence of SDC occurring in an isolated and independent fashion, making it extremely hard to detect and/or predict. All this detailed information can help with the design of more reliable and efficient devices in the future by targeting specific types of corruption, such as the ones found in this study. We discuss some strategies that could be implemented in HPC systems to limit the impact of DRAM errors on production.

As future work, we are planning to stress test our system by turning on the nodes with heating issues and monitoring them as well as their neighbors. In addition, we want to swap some components from the most faulty nodes with some healthy nodes to further improve the memory error characterization.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under the Mont-Blanc 2 Project (www.montblanc-project.eu), grant agreement n 610402 and it has been supported in part by the European Union (FEDER funds) under contract TTIN2015-65316-P.

REFERENCES

[1] 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*, 2014.

[2] 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.

[3] J. F. Ziegler, M. E. Nelson, J. D. Shell, R. J. Peterson, C. J. Gelderloos, H. P. Muhlfeld, and C. J. Montrose, "Cosmic ray soft error rates of 16-mb dram memory chips," *Solid-State Circuits, IEEE Journal of*, vol. 33, no. 2, pp. 246–252, 1998.

[4] B. Giridhar, M. Cieslak, D. Duggal, R. Dreslinski, H. M. Chen, R. Patti, B. Hold, C. Chakrabarti, T. Mudge, and D. Blaauw, "Exploring DRAM Organizations for Energy-efficient and Resilient Exascale Memories," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '13. New York, NY, USA: ACM, 2013, pp. 23:1–23:12. [Online]. Available: http://doi.acm.org/10.1145/2503210.2503215

[5] S. Borkar, "Designing reliable systems from unreliable components: The challenges of transistor variability and degradation," *IEEE Micro*, 2005.

[6] C. D. Martino, Z. Kalbarczyk, R. K. Iyer, F. Baccanico, J. Fullop, and W. Kramer, "Lessons learned from the analysis of system failures at petascale: The case of blue waters," in *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*, 2014, pp. 610–621.

[7] B. Nie, D. Tiwari, S. Gupta, E. Smirni, and J. H. Rogers, "A Large-Scale Study of Soft-Errors on GPUs in the Field," in *2016 IEEE 22nd International Symposium on High Performance Computer Architecture (HPCA)*, 2016.

[8] V. Sridharan, N. DeBardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi, "Memory errors in modern systems: The good, the bad, and the ugly," *SIGARCH Comput. Archit. News*, vol. 43, no. 1, pp. 297–310, Mar. 2015. [Online]. Available: http://doi.acm.org/10.1145/2786763.2694348

[9] A. Geist, "How to kill a supercomputer: Dirty power, cosmic rays, and bad solder," 2016. [Online]. Available: http://spectrum.ieee.org/computing/hardware/how-to-kill-a-supercomputer-dirty-power-cosmic-rays-and-bad-solder

[10] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance computing systems," *TDSC*, 2010.

[11] B. Schroeder and G. A. Gibson, "Understanding failures in petascale computers," *Journal of Physics: Conference Series 78:012022*, 2007.

[12] C.-D. Lu, "Failure data analysis of hpc systems," *Technical Report CoRR abs/1302.4779*, 2013.

[13] L. Borucki, G. Schindlbeck, and C. Slayman, "Comparison of accelerated DRAM soft error rates measured at component and system level," in *Reliability Physics Symposium, 2008. IRPS 2008. IEEE International*. IEEE, 2008, pp. 482–487.

[14] O. S. Unsal, J. W. Tschanz, K. Bowman, V. De, X. Vera, A. Gonzalez, and O. Ergin, "Impact of parameter variations on circuits and microarchitecture," *IEEE Micro*, vol. 26, no. 6, pp. 30–39, Nov 2006.

[15] P. N. Sanda, J. W. Kellington, P. Kudva, R. Kalla, R. B. McBeth, J. Ackaret, R. Lockwood, J. Schumann, and C. R. Jones, "Soft-error resilience of the IBM POWER6 processor," *IBM Journal of Research and Development*, vol. 52, no. 3, pp. 275–284, 2008.

[16] P. E. Dodd and L. W. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," *Nuclear Science, IEEE Transactions on*, vol. 50, no. 3, pp. 583–602, 2003.

[17] I.-G. Kim, S.-K. Choi, J.-H. Choi, and J.-S. Park, "Real impact of dynamic operation stress during burn-in on DRAM retention time," *Electron Devices, IEEE Transactions on*, vol. 51, no. 4, pp. 603–608, 2004.

[18] D. Tiwari, S. Gupta, J. Rogers, D. Maxwell, P. Rech, S. Vazhkudai, D. Oliveira, D. Londo, N. DeBardeleben, P. Navaux, L. Carro, and A. Bland, "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*, Feb 2015, pp. 331–342.

[19] 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," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '15. New York, NY, USA: ACM, 2015, pp. 38:1–38:12. [Online]. Available: http://doi.acm.org/10.1145/2807591.2807666

[20] S. Gupta, D. Tiwari, C. Jantzi, J. Rogers, and D. Maxwell, "Understanding and exploiting spatial properties of system failures on extreme-scale hpc systems," in *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*. IEEE, 2015, pp. 37–44.

[21] A. Hwang, I. Stefanovici, and B. Schroeder, "Cosmic rays don't strike twice: understanding the nature of DRAM errors and the implications for system design," *SIGARCH Comput. Archit. News*, 2012.

[22] J. Dongarra *et al.*, "The international exascale software project roadmap," *International Journal of High Performance Computing Applications*, 2011.

[23] F. Cappello, A. Geist, B. Gropp, L. Kale, B. Kramer, and M. Snir, "Toward exascale resilience," *International Journal of High Performance Computing Applications*, 2009.

[24] C. Constantinescu, I. Parulkar, R. Harper, and S. Michalak, "Silent data corruption: Myth or reality?" in *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*, June 2008, pp. 108–109.

[25] M. Casas, B. R. de Supinski, G. Bronevetsky, and M. Schulz, "Fault resilience of the algebraic multi-grid solver," in *Proceedings of the 26th ACM international conference on Supercomputing*. ACM, 2012, pp. 91–100.

[26] R. A. Ashraf, R. Gioiosa, G. Kestor, R. F. DeMara, C.-Y. Cher, and P. Bose, "Understanding the propagation of transient errors in hpc applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015, p. 72.

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

[28] 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.

[29] C.-Y. Cher, M. S. Gupta, P. Bose, and K. P. Muller, "Understanding soft error resiliency of bluegene/q compute chip through hardware proton irradiation and software fault injection," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2014, pp. 587–596.

[30] T. J. Hacker, F. Romero, and C. D. Carothers, "An analysis of clustered failures on large supercomputing systems." in *JPDC*, 2009.

[31] V. Sridharan and D. Liberty, "A study of DRAM failures in the field," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE Computer Society Press, 2012, p. 76.

[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 *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*. IEEE, 2013, pp. 1–11.

[33] G.-H. Asadi, V. Sridharan, M. B. Tahoori, and D. Kaeli, "Balancing performance and reliability in the memory hierarchy," in *Performance Analysis of Systems and Software, 2005. ISPASS 2005. IEEE International Symposium on*. IEEE, 2005, pp. 269–279.