Cosmic Rays Don't Strike Twice: Understanding the Nature of DRAM Errors and the Implications for System Design

Andy A. Hwang Ioan Stefanovici Bianca Schroeder
Department of Computer Science, University of Toronto
{hwang, ioan, bianca}@cs.toronto.edu

Abstract

Main memory is one of the leading hardware causes for machine crashes in today's datacenters. Designing, evaluating and modeling systems that are resilient against memory errors requires a good understanding of the underlying characteristics of errors in DRAM in the field. While there have recently been a few first studies on DRAM errors in production systems, these have been too limited in either the size of the data set or the granularity of the data to conclusively answer many of the open questions on DRAM errors. Such questions include, for example, the prevalence of soft errors compared to hard errors, or the analysis of typical patterns of hard errors.

In this paper, we study data on DRAM errors collected on a diverse range of production systems in total covering nearly 300 terabyte-years of main memory. As a first contribution, we provide a detailed analytical study of DRAM error characteristics, including both hard and soft errors. We find that a large fraction of DRAM errors in the field can be attributed to hard errors and we provide a detailed analytical study of their characteristics. As a second contribution, the paper uses the results from the measurement study to identify a number of promising directions for designing more resilient systems and evaluates the potential of different protection mechanisms in the light of realistic error patterns. One of our findings is that simple page retirement policies might be able to mask a large number of DRAM errors in production systems, while sacrificing only a negligible fraction of the total DRAM in the system.

Categories and Subject Descriptors B.8.0 [*Hardware*]: Performance and Reliability

General Terms Reliability, Measurement

1. Introduction

Recent studies point to main memory as one of the leading hardware causes for machine crashes and component replacements in today's datacenters [13, 18, 20]. As the amount of DRAM in servers keeps growing and chip densities increase, DRAM errors might pose an even larger threat to the reliability of future generations of systems.

As a testament to the importance of the problem, most server systems provide some form of protection against memory errors. Most commonly this is done at the hardware level through the use of DIMMs with error correcting codes (ECC). ECC DIMMs either provide single-bit error correction and double-bit error detection (SEC-DED); or use more complex codes in the chipkill family [4] that allow a system to tolerate an entire chip failure at the cost of somewhat reduced performance and increased energy usage. In addition, some systems employ protection mechanisms at the operating system level. For example, Solaris tries to identify and then retire pages with hard errors [3, 20]. Researchers have also explored other avenues, such as virtualized and flexible ECC at the software level [24].

The effectiveness of different approaches for protecting against memory errors and the most promising directions for designing future systems that are resilient in the face of increased DRAM error rates depend greatly on the nature of memory errors. For example, SEC-DED based ECC is most effective for protecting against transient random errors, such as soft errors caused by alpha particles or cosmic rays. On the other hand, mechanisms based on page retirement have potential only for protecting against hard errors, which are due to device defects and hence tend to be repeatable. In general, any realistic evaluation of system reliability relies on accurate assumptions about the underlying error process, including the relative frequency of hard versus soft errors, and the typical modes of hard errors (e.g. device defects affecting individual cells, whole rows, columns, or an entire chip).

While there exists a large body of work on protecting systems against DRAM errors, the nature of DRAM errors is not very well understood. Most existing work focuses solely on soft errors [1, 6, 9, 12, 14, 15, 21, 22, 25, 27] as soft error rates are often assumed to be orders of magnitudes greater than typical hard-error rates [5]. However, there are no large-scale field studies backing up this assumption. Existing studies on DRAM errors are quite old and rely on controlled lab experiments, rather than production machines [15, 25, 26, 28], and focus on soft errors only [10]. A notable exception is a recent study by Li et al. [11], which analyzes field data collected on 200 production machines and finds evidence that the rate of hard errors might be higher than commonly assumed. However, the limited size of their data set, which includes only a total of 12 machines with errors, makes it hard to draw statistically significant conclusions on the rate of hard versus soft errors or common modes of hard errors. Another recent field study [19] speculates that the rate of hard errors might be significant based on correlations they observe in error counts over time, but lacks more fine-grained data, in the form of information on the location of errors, to validate their hypothesis.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ASPLOS'12, March 3–7, 2012, London, England, UK.
Copyright © 2012 ACM 978-1-4503-0759-8/12/03. . . $10.00

System	Time (days)	Nodes	# DIMMs	DRAM in system (TB)	TByte years	Nodes with errors	Nodes w/ chipkill errs	Total # Errors	FIT
BG/L	214	32,768	N/A	49	28	1,742 (5.32%)	N/A	$227 \cdot 10^6$	97,614
BG/P	583	40,960	N/A	80	127	1,455 (3.55%)	1.34%	$1.96 \cdot 10^9$	167,066
SciNet	211	3,863	31,000	62	35	97 (2.51%)	N/A	$49.3 \cdot 10^6$	18,825
Google	155	20,000	$\sim$ 130,000	220	93	20,000	N/A	$27.27 \cdot 10^9$	N/A

Table 1. Summary of system configurations and high-level error statistics recorded in different systems

The goal of this paper is two-fold. First, we strive to fill the gaps in our understanding of DRAM error characteristics, in particular the rate of hard errors and their patterns. Towards this end we provide a large-scale field study based on a diverse range of production systems, covering nearly 300 terabyte-years of main memory. The data includes detailed information on the location of errors, which allows us to statistically conclusively answer several important open questions about DRAM error characteristics. In particular, we find that a large fraction of DRAM errors in the field can be attributed to hard errors and we provide a detailed analytical study of their characteristics.

As a second contribution, the paper uses the results from the measurement study to identify a number of promising directions for designing more resilient systems and evaluates the potential of different protection mechanism in the light of realistic error patterns. One of our findings is that simple page retirement policies, which are currently not widely used in practice, might be able to mask a large number of DRAM errors in production systems, while sacrificing only a negligible fraction of the total system's DRAM.

2. Background

2.1 Overview of data and systems

Our study is based on data from four different environments: the IBM Blue Gene/L (BG/L) supercomputer at Lawrence Livermore National Laboratory, the Blue Gene/P (BG/P) from the Argonne National Laboratory, a high-performance computing cluster at the SciNet High Performance Computing Consortium, and 20,000 machines randomly selected from Google's data centers. Below we provide a brief description of each of the systems and the data we obtained.

BG/L: Our first dataset is from the Lawrence Livermore National Laboratory's IBM Blue Gene/L (BG/L) supercomputer. The system consists of 64 racks, each containing 32 node cards. Each node card is made up of 16 compute cards, which are the smallest replaceable hardware component for the system; we refer to the compute cards as "nodes". Each compute card itself contains two PowerPC 440 cores, each with their own associated DRAM chips, which are soldered onto the card; there is no notion of a "DIMM" (see [7] for more details).

We obtained BG/L logs containing data generated by the system's RAS infrastructure, including count and location messages pertaining to correctable memory errors that occur during a job and are reported upon job completion.

The BG/L memory port contains a 128-bit data part that's divided divided into 32 symbols, where the ECC is able to correct any error pattern within a single symbol, assuming no errors occur in any other symbols. However, the system can still function in the event of two symbols with errors by remapping one of the symbols to a spare symbol, and correcting the other with ECC [16].

Due to limitations in the types of messages that the BG/L log contains, we are only able to report on multi-bit errors that were detected (and corrected) within a single symbol. As such, we refer to these as MBEs (multi-bit errors) for the BG/L system throughout the paper. However it's worth noting that 350 compute cards (20% of all compute cards with errors in the system) reported activating symbol steering to the spare symbol. This is indicative of more severe errors that required more advanced ECC technologies (like bit-sparing) to correct. In addition, a cap was imposed on the total count of correctable errors accumulated during a job for part of the dataset, making our results for both multi-bit errors and total correctable error counts very conservative compared to the actual state of the system.

BG/P: The second system we studied is the Blue Gene/P (BG/P) from the Argonne National Laboratory. The system has 40 racks containing a total of 40,960 compute cards (nodes). Each node in BG/P has four PowerPC 450 cores and 40 DRAM chips totalling 2GB of addressable memory. As the successor to BG/L, BG/P has stronger ECC capabilities and can correct single and double-symbol errors. The system is also capable of chipkill error correction, which tolerates failure of one whole DRAM chip [8].

We obtained RAS logs from BG/P reporting correctable error samples. Only the first error sample on an address during the execution of a job is reported, and total occurrences for each error type summarized at the end. Due to the sampling and counter size, the amount of correctable errors are once again very conservative. However, the correctable samples provide location information which allows us to study the patterns and physical distribution of errors.

Unlike BG/L, there is no bit position information for single-symbol errors. There is no way to determine the number of bits that failed within one symbol. Therefore, we report single-symbol errors as single-bit errors and double-symbol errors as multi-bit errors, and refer to the latter as MBEs for the BG/P system. A double-symbol error is guaranteed to have at least two error bits that originate from the pair of error symbols. This is once again an under-estimation of the total number of multi-bit errors.

SciNet: Our third data source comes from the General Purpose Cluster (GPC) at the SciNet High Performance Computing Consortium. The GPC at SciNet is currently the largest supercomputer in Canada. It consists of 3,863 IBM iDataPlex nodes, each with 8 Intel Xeon E5540 cores and 16GB of addressable memory that uses basic SEC-DED ECC. The logs we collected consist of hourly-dumps of the entire PCI configuration space, which expose the onboard memory controller registers containing counts (with no physical location information) of memory error events in the system.

Google: Our fourth data source comes from Google's datacenters and consists of a random sample of 20,000 machines that have experienced memory errors. Each machine comprises a motherboard with some processors and memory DIMMs. The machines in our sample come from 5 different hardware platforms, where a platform is defined by the motherboard and memory generation. The memory in these systems covers a wide variety of the most

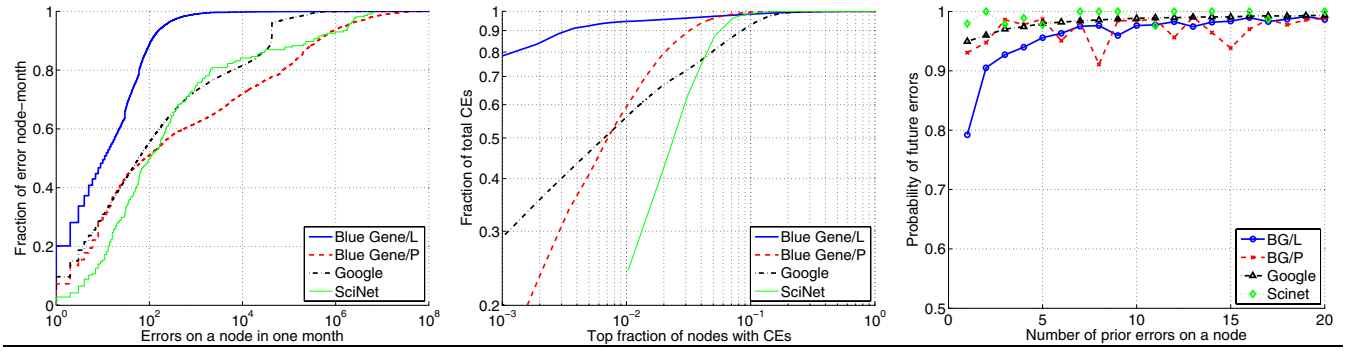

Figure 1. The left graph shows the CDF for the number of errors per month per machine. The middle graph shows the fraction y of all errors that is concentrated in the top x fraction of nodes with the most errors. The right graph shows the probability of a node developing future errors as a function of the number of prior errors.

commonly used types of DRAM. The DIMMs come from multiple manufacturers and models, with three different capacities (1GB, 2GB, 4GB), and cover the three most common DRAM technologies: Double Data Rate (DDR1), Double Data Rate 2 (DDR2) and Fully-Buffered (FBDIMM). We rely on error reports provided by the chipset. Those reports include accurate accounts of the total number of errors that occurred, but due to the limited number of registers available for storing addresses affected by errors only provides samples for the addresses of errors. For this reason, the number of repeat errors we observe and the probability of errors repeating are very conservative estimates, since there might be repeat errors that we missed because they were not sampled.

2.2 Methodology

A memory error only manifests itself upon an access to the affected location. As such, some systems employ a *memory scrubber* (a background processes that periodically scans through all of memory) to proactively detect errors before they are encountered by an application. However, except for some of the Google systems, all the systems we study rely solely on application-level memory accesses without the use of a scrubber.

Categorizing errors observed in the field as either hard or soft is difficult as it requires knowing their root cause. Obtaining a definite answer to the question of whether an observed error is hard and what type of hard error it is (e.g. a stuck bit or a bad column) would require some offline testing of the device in a lab or at least performing some active probing on the system, e.g. by running a memory test after each error occurrence to determine whether the error is repeatable. Instead we have to rely on observational data, which means we will have to make some assumptions in order to classify errors. Matters are complicated further by the fact that many hard errors start out as intermittent errors and only develop into permanent errors over time.

The key assumption that we rely on in our study is that repeat errors at the same location are likely due to hard errors since it would be statistically extremely unlikely that the same location would be hit twice within our measurement period by cosmic rays or other external sources of noise. We therefore view such repeat errors as likely being caused by hard errors. Note however that in practice, hard errors manifest themselves as intermittent rather than on every access to a particular memory location.

We consider different granularities for locations at which errors can repeat. We start by looking at repeats across nodes, but then mainly focus at locations identified by lower-levels in the hardware. Recall that a DIMM comprises multiple DRAM chips, and each DRAM chip is organized into multiple banks, typically 8 in today's systems. A bank consists of a number of two-dimensional arrays of DRAM cells. A DRAM cell is the most basic unit of storage, essentially a simple capacitor representing one bit. The two dimensions of an array are also referred to as rows and columns. We look at repeats of errors at the level of physical addresses, but also with respect to bank, rows and columns at the chip level.

3. Study of error characteristics

3.1 High-level characteristics

We begin with a summary of the high-level characteristics of memory errors at the node level. The right half of Table 1 summarizes the prevalence of memory errors in the four different systems. We observe that memory errors happen at a significant rate in all four systems with 2.5-5.5% of nodes affected per system. For each system, our data covers at least tens of millions of errors over a combined period of nearly 300 Terabyte years. In addition to correctable errors (CEs), we also observe a non-negligible rate of "non-trivial" errors, which required more than simple SEC-DED strategies for correction: 1.34% of the nodes in the BG/P system saw at least one error that required chipkill to correct it.

Figure 1 (left) and Figure 1 (middle) provide a more detailed view of how errors affect the nodes in a system. Figure 1 (left) shows the cumulative distribution function (CDF) of the number of errors per node for those nodes that experience at least one error. We see that only a minority (2-20%) of those nodes experience just a single error occurrence. The majority experiences a larger number of errors, with half of the nodes seeing more than 100 errors and the top 5% of nodes each seeing more than a million errors. Figure 1 (middle) illustrates how errors are distributed across the nodes within each system. The graph shows for each system the fraction of all errors in the system (X-axis) that is concentrated on just the y% of nodes in the system with the largest number of errors (Y-axis). In all cases we see a very skewed distribution with the top 5% of error nodes accounting for more than 95 % of all errors.

Figure 1 (left) and (middle) indicate that errors happen in a correlated fashion, rather than independently. This observation is validated in Figure 1 (right), which shows the probability of a node experiencing future errors as a function of the number of past errors. We see that even a single error on a node raises the probability of future errors to more than 80%, and after seeing just a handful of errors this probability increases to more than 95%.

The correlations we observe above provide strong evidence for hardware errors as a dominant error mechanisms, since one would not expect soft errors to be correlated in space or time. Our observations agree with similar findings reported in [11, 19]. However, the results in [11] were based on a small number of

machines (12 machines with errors) and the analysis in [19] was limited to a relatively homogeneous set of systems; all machines in the study were located in Google's datacenters. Our results show that these trends generalize to other systems as well and add statistical significance.

In addition to error counts, the BG systems also record information on the mechanisms that were used to correct errors, which we can use as additional clues regarding the nature of errors. In particular, both BG/P and BG/L provide separate log messages that allow us to distinguish multi-bit errors, and BG/P also records information on chipkill errors (i.e. errors that required chipkill to correct them). We observe that a significant fraction of BG/P and BG/L nodes experiences multi-bit errors (22.08% and 2.07%, respectively) and that these errors account for 12.96% and 2.37% of all observed errors, respectively. The fraction of nodes with chipkill errors on BG/P only is smaller, with 1.34% of nodes affected, but still significant. Interestingly, while seen only on a small number of nodes, chipkill errors make up a large fraction of all observed errors: 17% of all errors observed on BG/P were not correctable with simple SEC-DED, and required the use of chipkill ECC to be corrected.

We summarize our main points in the following three observations which motivate us to take a closer look at hard errors and their patterns in the remainder of this paper.

Observation 1: There are strong correlations between errors in space and time. These correlations are highly unlikely if soft errors were the dominating error mode, pointing to hard errors as the leading cause of memory errors.

Observation 2: The significant frequency of multi-bit and chipkill errors is another pointer towards hard errors as an important error mode, as these types of errors are unlikely to arise from soft errors.

Observation 3: A significant number of nodes with correctable errors activated more advanced ECC mechanisms; 20%-45% activated redundant bit-steering, and 15% activated Chipkill.

3.2 Error patterns

In this section, we attempt to categorize all banks with errors in our datasets into known error patterns related to hardware defects: single (transient) events, bad cells, bad rows, bad columns, and a whole chip error. A definite answer to the question which category a device with an error falls into would require offline testing of the device in a lab setting. Instead we have to rely on observational data, which means we will have to make a few assumptions when classifying devices. We group all banks that have at least one error into one of the following categories:

repeat address: The bank has at least one error that repeats, i.e. there is at least one address on this bank that is reported twice.
repeat row: The bank has at least one row that has experienced errors at two different locations, i.e. two different addresses on the row.
repeat column: The bank has at least one column that has experienced errors at two different locations, i.e. two different addresses on the column.
corrupt row: The bank has at least one row that has experienced errors at two different addresses on the row and one of these is a repeat address.
corrupt column: The bank has at least one column that has experienced errors at two different addresses on the column and one of these is a repeat address.

Error mode	BG/L Banks	BG/P Banks	Google Banks
repeat address	80.9%	59.4%	58.7%
repeat address w/o row/cols	72.2%	30.0%	46.1%
repeat row	4.7%	31.8%	7.4%
repeat column	8.8%	22.7%	14.5%
corrupt row	3.0%	21.6%	4.2%
corrupt column	5.9%	14.4%	8.3%
whole chip	0.53%	3.20%	2.02%
single event	17.6%	29.2%	34.9%

Table 2. Frequency of different error patterns

single event: These are banks that have only single events, i.e. they have no repeat errors on any of their addresses, rows or columns.
whole chip: These are banks that have a large number of errors ($>$ 100 unique locations) distributed over more than 50 different rows and columns.

Table 2 groups each error bank in our dataset into one of the above categories and reports for each system the fraction of banks that falls into each of these categories. Note, that the repeat-address category overlaps with the corrupt row and corrupt column categories. We therefore created an additional entry that reports banks with repeat addresses that do not exhibit corrupt rows or columns.

We make a number of interesting observations. The vast majority (65-82%, depending on the system) of all banks experiences some form of error pattern that points towards hard errors, i.e. an error pattern other than single events. This observation agrees with the findings in [11], however the conclusions on the frequency of different patterns in [11] are limited due to their small dataset (12 machines with errors). We find that among all error patterns the single most common one are repeat addresses. Consistently for all systems, more than 50% of all banks with errors are classified as repeat addresses. For all systems, corrupt rows and corrupt columns happen at a significant rate. We note that each system has a clear tendency to develop one type over the other, where the more common type is approximately twice as often observed as the other one. For example, in the case of BG/L and Google, corrupt columns are twice as likely as corrupt rows, while for BG/P it is the other way around. This is likely due to the fact that there are twice as many rows in BG/P banks than in BG/L banks.

Note that the above numbers on hard error patterns are conservative, and in practice likely higher. Since our observation period for each of the systems is limited and we depend on accesses to memory cells to detect errors, many of the non-repeat errors in our study might eventually have turned out to be repeat errors, but the repeat did not fall into our measurement period. We observe for example that for the systems with shorter observation time (BG/L and Google), the fraction of banks with only repeat addresses, but no bad rows/columns, is higher than in the BG/P system whose data spans a longer observation period. Most likely, the longer observation time increased the chances that a repeat error will manifest and move a repeat row/column to the corrupt row/column category. That indicates that a large fraction of errors we categorize conservatively as repeat rows/columns might actually be true broken rows/columns.

Observation 4: The error patterns on the majority of the banks (more than two thirds) in our study can clearly be attributed to patterns linked to hard errors. Among those patterns the most common one is a repeat address, although row and column patterns do also happen at a significant rate.

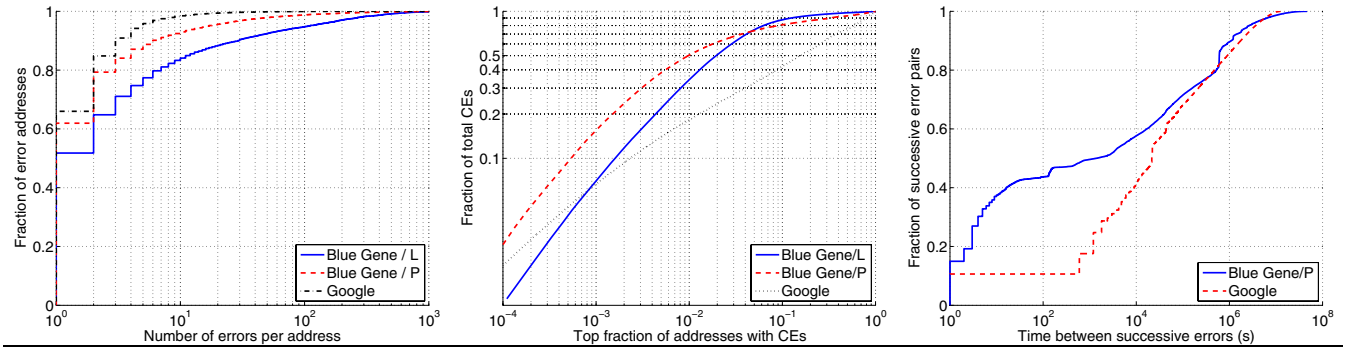

Figure 2. The left graph shows the CDF for the number of repeat errors per address (for those addresses with at least on repeat). The middle graph shows the fraction y of all errors that is concentrated in the top x fraction of addresses with the most errors. The right graph shows the CDF of the time between successive errors on an address.

3.3 Repeat errors on addresses

The previous section identified repeat addresses as the dominant error pattern, but did not provide any details on their characteristics. The only prior work that hints at repeat addresses as a common error pattern in the field is based on a data set (a dozen machines with errors) that is too small for a detailed study of repeat error characteristics [11]. We therefore study this question in more detail in this subsection.

	BG/L	BG/P	Google
# of error samples	201,206	308,170	1,091,777
# of unique addresses	9,076	44,624	556,161
% unique	4.5	14.5	50.9
% of addresses with repeats	48.2	30.6	32.6
Avg. # of errors / address	44.9	20.3	4.0

Table 3. Statistics on repeat addresses

We begin by providing some statistics on the frequency of repeat errors on addresses in Table 3 above. We observe that a high fraction of addresses with errors experience later repeat errors on the same address: a third (for Google) to nearly a half (for BG/L). The average number of repeat errors per address ranges from 4 for Google to as many as 44 for BG/L. For a more detailed view, Figure 2 (left) shows the cumulative distribution function (CDF) for the number of repeats per address. Most addresses with repeats (50-60%) see only a single repeat and another 20% see only two repeats. However, the distribution is very skewed with a long tail, where a small fraction of addresses at the end of the tail experiences a huge number of repeats. Figure 2 (middle) illustrates the skew in the distribution by plotting the fraction of errors that is made up by the top x% of addresses with the highest error count. It shows that 10% of all addresses with errors account for more than 90% of all observed errors.

When trying to protect against repeat errors it is useful to understand the temporal characteristics of errors. For example, a system using page retirement for pages with hard errors might want to wait before retiring a page that experiences an error until a repeat error occurs, providing some confidence that the problem is indeed due to a hard error. An interesting question is therefore how long it will take before a repeat error happens and an error can confidently be classified as hard. To help answer this question, Figure 2 (right) plots the CDF of the time between repeat errors on the same address. The graph shows that, if there is a repeat error it typically happens shortly after the first error occurrence. In BG/P more than half of repeats happen within less than a couple of minutes after the first occurrence. The timing information in the Google data is at a much coarser granularity (recall Section 2) and due to sampling we might not see all repeats, which leads to generally longer times until a repeat error shows up. However, we can conclude that more than half of the repeats happen within less than 6 hours. Interestingly, for larger timescales, e.g. on the order of days, where the timing granularity of the Google data should have less of an effect the trends for both systems start to look very similar. In both systems, 90% of all repeat errors are detected within less than 2 weeks.

When interpreting data regarding repeat of errors, it is important to recall that repeat errors (or any errors) are not detected until either the application or a hardware scrubber accesses the affected cell. For the Blue Gene systems, we know that hardware scrubbers are implemented as a feature, but we were not able to determine whether this feature was actually enabled in our systems under study. On the other hand, for the Google machines we know that a subset of them does employ a hardware scrubber that periodically in the background reads through main memory to check for errors. This scrubber reads memory at a rate of 1GB per hour, which means that each memory cell should be touched at least once every day.

To determine how much earlier repeat errors could be detected if a memory scrubber is used we compared the time until a repeat error is detected for those systems with and without the hardware scrubber separately. Interestingly, we find that the use of a scrubber does not significantly reduce the time until a repeat error is detected. Even in the tail of the distribution, where it takes a relatively long time (e.g. several days or more) to identify a repeat error, there is not much difference between systems with and without a scrubber. One possible explanation is that repeat errors might not always be due to stuck bits, where a cell is permanently stuck at a particular value. Instead, they might be due to weaknesses in the hardware that get exposed only under certain access patterns.

The three points below summarize our main observations.

Observation 5: A large fraction of addresses with errors (30-50%) experience a later repeat error on the same address. This points to hard errors as a dominant error mode.

Observation 6: If an address experiences multiple errors (indicating a hard error), then 90% of all repeat errors show up within less than 2 weeks after the first error.

Observation 7: The use of a background scrubber does not significantly shorten the amount of time until a repeat error is detected. This indicates that a significant fraction of errors are intermittent and only manifest themselves under certain access patterns.

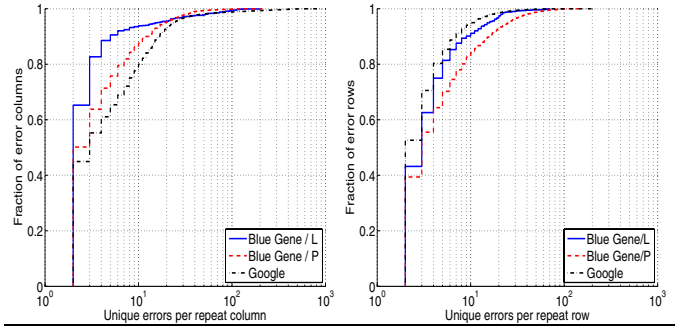

Figure 3. Number of errors per repeat row/column

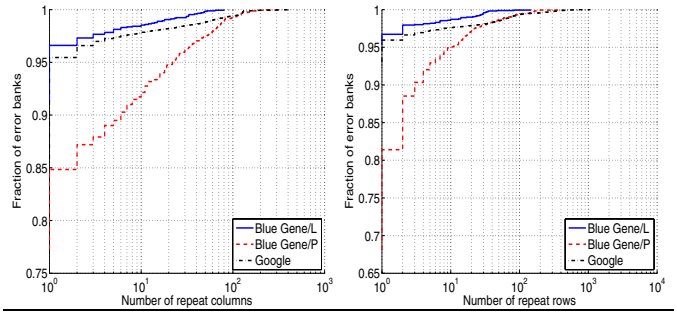

Figure 4. Number of repeat rows/columns per bank

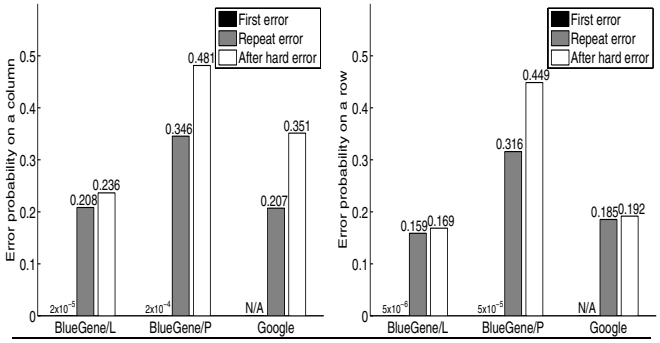

Figure 5. The two graphs show the probability that a column (left graph) and a row (right graph) will have an additional error after having one error (middle bar of each group of bars) and the probability that it will have an additional unique error (right bar of each group of bars), i.e. an additional error at an address different from the first.

3.4 Repeat errors within a row/column

Memory errors that are due to hardware problems don't only manifest themselves in the form of repeat errors on the same cell. Section 3.2 pointed to repeating errors on different locations within the same row or the same column as another common error mode (recall Table 2). This subsection takes a closer look at the characteristics of repeat rows and repeat columns.

We begin by looking at the probability that an error at an individual address will develop into a row or column error. Understanding those probabilities might help predict impending row/column errors, allowing a system to take proactive measures. The three groups of bars in Figure 5 (left) summarize our results for columns and Figure 5 (right) shows the corresponding results for rows. The gray center bar in each group of bars shows the probability that after an error occurs on an address another error will later develop at a different address along the same column (Figure 5 left) and row (Figure 5 right), respectively (turning the error from a single event error into a repeat column/row). We observe that in all three cases these probabilities are significant for all systems, with probabilities in the 15% to 30% range.

To put the above repeat probabilities in perspective, we compare them with the unconditional probability of a random bank/row/column developing an error, shown in the black (left-most) bar (the bar is barely visible due to its small magnitude). We see that the unconditional probabilities are orders of magnitude smaller.

We also study the probability that a repeat error on an address (rather than a single event error) will turn into a row/column error, i.e. the probability that after a repeat error on an address another address along the same row/column experiences an error. Those probabilities are shown in the white right-most bar in each group of bars in Figure 5. In all systems, the presence of a repeat address on a row/column further increases the probability of future errors on this bar/column. In some cases this increase is quite large. For example, for BlueGene/P after observing a repeat error on an address the probability of observing an additional error on another location on the same row/column increases to more than 40%. Again, recall that BG/P is the system with the longest observation period among our datasets, so the probabilities of repeat errors developing into row/column errors effect might be equally strong in the other systems and we might just not observe it due to the shorter timespan of the data.

While we have provided information on the probabilities of rows/columns developing multiple errors, another question is how many errors repeat rows and columns typically experience. Figure 3 (left) and (right) provide the CDF for the number of unique locations per row/column that experience errors. We see that most rows/columns with multiple errors (40-60%) don't develop more errors beyond the initial 2 errors. However, the top 10-20% of rows/columns develop errors on dozens of unique locations.

Section 3.2 showed that a significant number of error banks exhibit row/column errors, but does not quantify the number of repeat rows/columns. We find that the most common case are banks with only a single repeat row/column. Depending on the system, 80-95% of all banks with repeat rows/columns have only a single one (see Figure 4). Around 3-8% of banks develop more than 10 repeat columns and 2-5% develop more than 10 repeat rows.

Interestingly, we observe that a significant number of all banks (3.4%, 17.4%, and 4.4%, for BG/L, BG/P and Google, respectively) experience both repeat rows and repeat columns. To better understand the relationship between the number of repeat rows and columns on a bank, the scatter plot in Figure 6 shows for banks with at least one repeat row or column the number of repeat rows and columns. The marker at coordinates (x,y) reflects the fraction of banks that have x repeat columns and y repeat rows. The size of the marker in the scatter plot indicates the fraction of error banks that fall into each of the categories and is chosen to be proportional to the logarithm of the probabilities. For all three systems we observe that banks that have a larger number of repeat rows tend to also have a larger number of repeat columns, and vice versa.

Below we summarize the main observations regarding repeat rows and columns.

Observation 8: The probability of seeing an error on an address on a particular row or column increases by orders of magnitude once we have seen an error on another location on the same row or column. 15-30% of errors turn into repeat rows/columns, and up to 40% of repeat errors turn into repeat rows/columns.

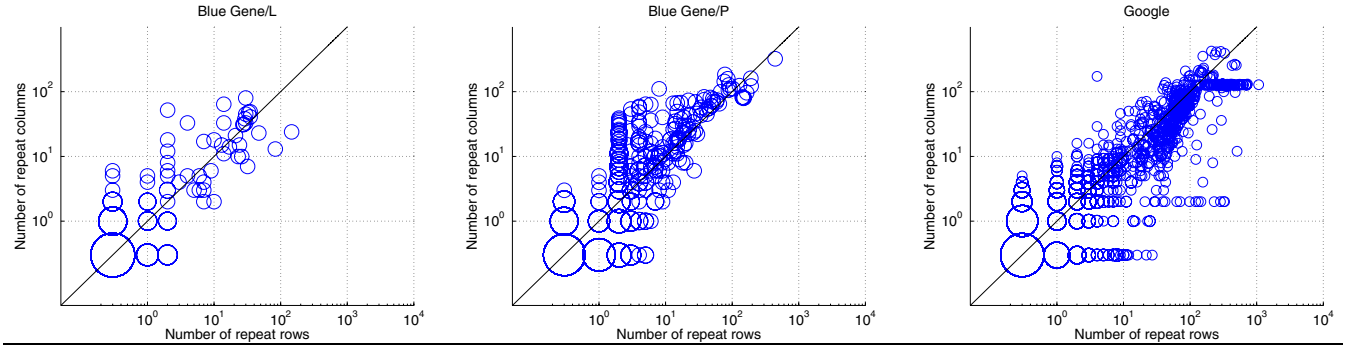

Figure 6. The distribution of the number of repeat rows and repeat columns per bank

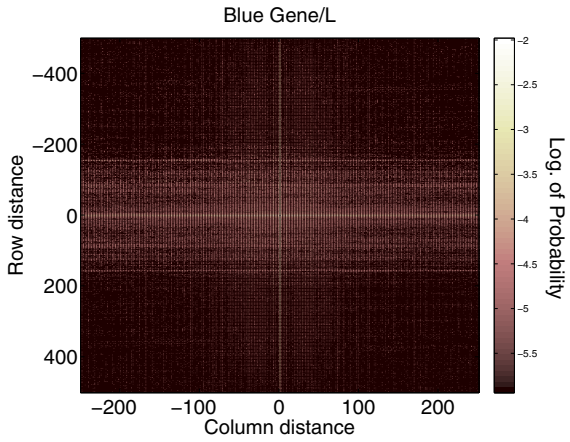

Figure 7. A visualization of correlations between errors as a function of their distances in row/column space

Observation 9: Among error banks that have both repeat rows and repeat columns, those banks that have a large number of repeat rows often also have a large number of repeat columns, and vice versa.

3.5 Correlations across rows/columns

While the previous subsection demonstrated that an error on a row or column increases the probability of follow-up errors on the same row/column, it does not tell us anything about correlations between nearby rows/columns, e.g. do multiple errors on a row make it more likely that also some nearby rows will have errors. In this subsection we answer the more general question of how the error probabilities between cells are correlated depending on the physical distance in row and column space between those cells.

The heatmap in Figure 7 is an attempt to visualize those correlations. The pixel at the center of the plot at coordinates (0,0) represents a cell with an error. The pixel at coordinates (x,y) represents the probability that the cell that is x columns and y rows away from the original error cell (i.e. at row/column coordinates (a+x, b+y) where (a,b) are the row/column coordinates of the original error) has an error as well. Lighter colors represent higher probabilities.

Not surprisingly, we observe that cells along the same row or column as the original error cell have increased error probabilities, as indicated by the bright vertical and horizontal line crossing through (0,0). This agrees with our previous observations that errors have a high probability of turning into repeat rows/columns. But we also observe that the error probabilities are increased within a wide band of neighboring rows and columns. For example, a closer study of the error probabilities as a function of row/column distance shows that rows and columns that are within a distance of 10 of the original error have an error probability of 2-5%. While this probability is clearly smaller than that of developing errors on the same row/column, it is significantly larger than that of an average row/column. We also find that the error probabilities show a roughly exponential drop-off as a function of the row/column distance, and that the probabilities are still significantly increased within a range of up to 50-100 rows/columns, compared to an average row/column.

Our study of error probabilities as a function of distance from another error also shows evidence for other patterns, beyond just proximity. In particular, we observe for some systems that cells whose column or row distance from the original error is a multiple of certain powers of two have increased likelihood of errors. Evidence for these regular patterns show up in the heatmap in the form of a grid-like background pattern. By studying the CDF of the pairwise distances between errors, we find for example that for all systems (BG/P, BG/L, Google), cells with distances in row space that are multiples of 4 have noticeably increased error probabilities. Some systems also exhibit other patterns. For example, BG/P also shows clearly increased probabilities at row distances that are multiples of 128.

Observation 10: Errors do not only tend to cluster along the same row or column. The probability of errors is also significantly increased along all cells in the band of nearby rows and columns.

3.6 Error density in different areas of a chip

In this subsection we look at the correlation between errors and their physical location on a chip, i.e. we are asking the question of whether some areas of a chip are more likely than others to experience errors. As before, we use bank, row and column information to distinguish different locations on a chip. We first divide the row/column space of each bank into equal sized square areas of 128x128 rows/columns, i.e. chunks of size 16KB. We then determine for each of these square areas the probability of observing an error in this area, i.e. the fraction of all unique banks in the system (across all nodes) that have at least one error in this area. Figure 8 shows a graphical representation of the results. For this analysis, we report results separately for BG/P and BG/L and for the four different hardware platforms that the Google data covers. Each graph represents the row/column space for one of the systems, each divided into the 128x128 sized squares as described above. Each square is colored according to the observed probability of errors in this area, where darker colors correspond to higher probabilities.

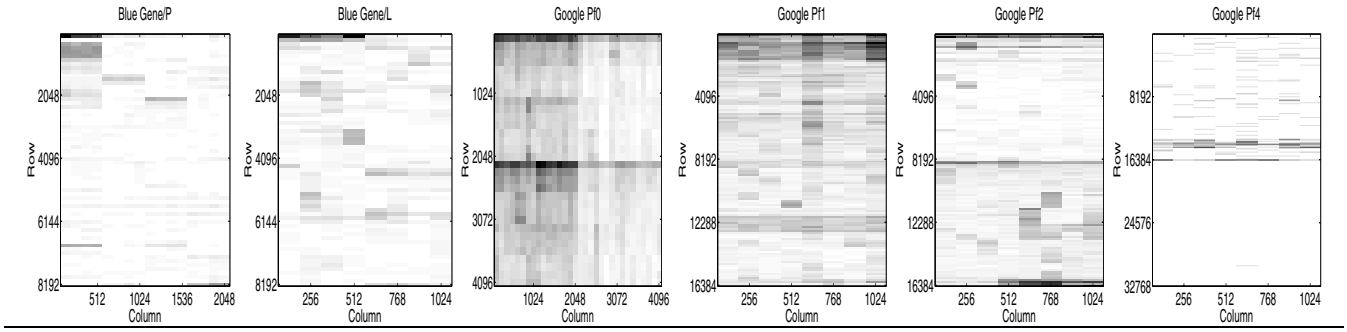

Figure 8. The error probabilities for different areas in the row/column space of a bank.

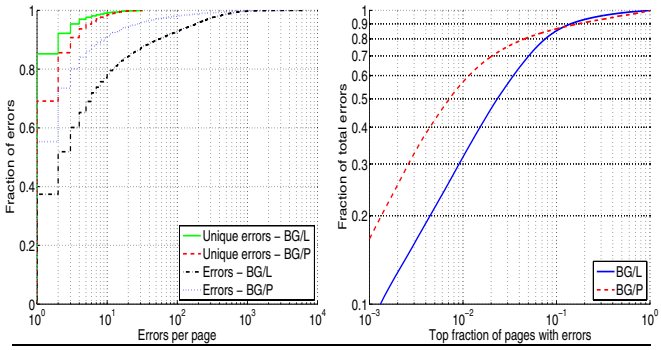

Figure 9. Error distribution over pages.

Figure 8 shows several interesting trends that are consistent across systems. Focusing on the dark areas in each graph, which present concentrations of errors, we first see that for all systems consistently the top left area shows increased error density. This area tends to span at least the first 512 columns and the first 512 rows. For several of the systems, a whole band along the very top of the row space, across all columns, shows increased error rates. For two of the six systems we observe similar concentrations of errors at the end of the row/column space, i.e. in the bottom right of the graphs at the highest numbered rows and columns. Secondly, we find that for three of the systems the entire rows in the center of the row space exhibit increased error probabilities.

Observation 11: Not all areas on a chip are equally likely to be affected by errors. In particular, the top and the bottom of the row/column space on a bank seem to be more likely to experience errors. Additionally, the upper-left corner corresponds to memory used by the OS, which may play a role in the increased error probabilities.

3.7 Hard errors from the OS's point of view

Throughout this section we have observed different ways in which DRAM errors tend to cluster in space. We have seen that errors tend to repeat on the same address, along the addresses of a row/column and on certain areas of a chip. All these measures for spatial clustering were very hardware oriented. In order to explore protection mechanisms at the OS level, an important question is how the clustering of errors translates to the operating system level. For example, retiring pages with errors would work most efficiently and effectively if most of the errors tended to cluster on a small number of pages. Unfortunately, error clusters at the hardware level do not directly translate to clusters on pages. For example, errors along the same row or column do not necessarily lie on the same page.

To shed some light on how errors are distributed across pages, Figure 9 (left) shows the CDF for the number of errors per page and the number of unique locations with errors per page for those systems for which the information is available (BG/L and BG/P). The number of unique locations with errors per page is low (on average 1.4 and 1.8 for BG/L and BG/P, respectively) and around 90% of all pages have only a single one. However the total number of errors observed per page is still quite large, most likely due to repeat addresses. More than 60% of the pages experience more than one error, and the average number of errors per page is 31 and 12, for BG/L and BG/P respectively. More importantly, the distribution of errors across pages is very skewed, maybe not surprisingly given the frequency of repeat addresses that we observed earlier. Figure 9 (right) shows the fraction of all errors that is contributed by the fraction of the top x% of pages with the most errors. 1% of all pages with errors account for 30-60% of all errors, depending on the system, and the top 10% of all pages with errors account for more than 90% of all errors for both BG/L and BG/P. This skew in the number of errors page is good news for techniques relying on page retirement, as it means that by retiring a small fraction of pages a large number of errors can be prevented.

Observation 12: More than 60% of pages that experience an error, experience at least on follow-up error. The distribution of the number of errors per page is highly skewed, with some pages accounting for a large fraction of errors.

Observation 13: An operating system that could identify and retire those pages that are likely to develop a large number of errors, would avoid a large fraction of errors (90%) by retiring only a small fraction (10%) of pages with errors.

This observation motivates us to study the possible effectiveness of different page retirement policies in Section 4.

3.8 Hard errors and multi-bit / chipkill errors

From a systems point of view the most worrisome type of errors are multi-bit and chipkill errors, as these are the errors that in the absence of sufficiently powerful hardware ECC turn into uncorrectable errors leading to a machine crash (or if undetected to the use of corrupted data). Given the correlations we observed between errors in the previous subsections, an interesting question is whether it is possible to predict an increased likelihood of future multi-bit or chipkill errors based on the previous error behavior in the system. In particular, one might speculate that prior repeat errors, which likely indicate hard errors, will increase the probability of later multi-bit or chipkill errors. Knowledge about the increased likelihood of future multi-bit or chipkill errors could be used by an adaptive system to take proactive measures to protect against errors.

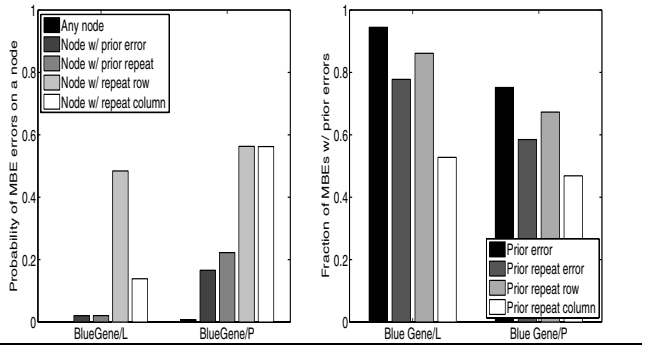

Figure 10. The relationship between multi-bit errors and prior errors.

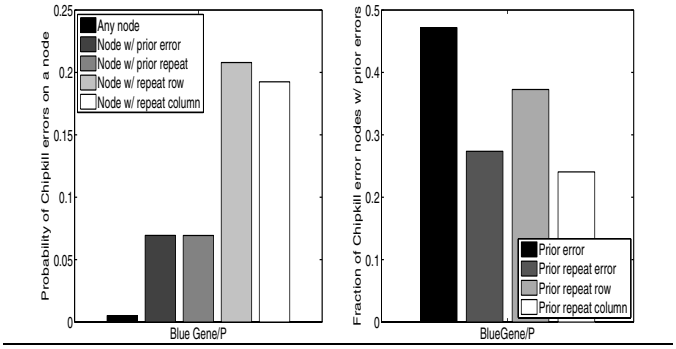

Figure 11. The relationship between chipkill errors and prior errors.

To shed some light on this question, we plot in Figure 10 (left) the probability that a node develops a multi-bit error after seeing previous errors of different types for BG/L and BG/P. More precisely, each set of 5 bars in the graph shows the following 5 probabilities: The first bar in each group of five bars represents the baseline probability of a random node seeing a multi-bit error. (Note, that this probability is so small that it is barely visible in the graph.) The second bar represents the probability that a node that has seen a prior error (of any type) will later experience a multi-bit error, The other three bars show the probability that a node will experience a later multi-bit error after experiencing a repeat address, a repeat row or a repeat column, respectively. The figure clearly indicates that for both systems the probability of a multi-bit error increases, after seeing previous errors. It also shows that the probability increases dramatically if a previous error was a repeat error.

Figure 10 (left) tells us only that the probability of multi-bit errors increases after other errors have been observed. It does not tell us whether most multi-bit errors were in fact preceded by prior errors (which a system could use as an early warning sign of impending multi-bit errors). In order to look at this side of the story, Figure 10 (right) plots the fraction of multi-bit errors that were preceded be the four types of errors we considered previously (any error, repeat address, repeat row, repeat column). The graph shows that multi-bit errors don't happen without prior warning: 60-80% of multi-bit errors were preceded by repeat addresses, 70-85% of multi-bit errors were preceded by a repeat row and 40-50% of multi-bit errors were preceded by a repeat column.

Figure 11 repeats the same analysis for chipkill errors, rather than multi-bit errors (for BG/P only, as chipkill errors do not apply to BG/L). While the overall probabilities are smaller (due to the lower rate of chipkill errors), we observe the same trends. Prior errors greatly increase the probability of a later chipkill error. Among nodes with prior error the probability increases to 7%. If there is a repeat row or repeat column present in the system, the likelihood of a later chipkill error increases to around 20%.

Observation 14: The incidence of errors in the system, in particular repeat errors, increases the likelihood of later multi-bit and chipkill errors by more than an order of magnitude. A large fraction (more than half) of multi-bit and chipkill errors are preceded by early warning signs in the form of repeat errors.

4. Implications for system design

An underlying theme throughout the previous section has been the study of hard errors as the dominating error mode among DRAM errors in the field. Compared to soft errors, hard errors have a greater potential to increase error rates, due to their repetitive nature, and to increase the chance of future uncorrectable errors. On the positive side, the repeating nature of hard errors makes them also more predictable than soft errors creating a potential for taking proactive measures against them. In this section, we discuss various implications on resilient system that follow from the insights derived from our measurement study.

4.1 Page retirement

While error protection at the hardware level in the form of ECC is effective, it is not always the most desirable option. In addition to the cost factor, another concern, in particular for the more powerful ECC codes, is the increase in energy consumption and the possible impact on performance.

As an alternative or extra level of protection in addition to the use of ECC DIMMs, one could consider the retirement of pages that have experienced previous (hard) errors. Page retirement can be accomplished by having the OS isolate pages containing errors and prevent them from being allocated in the future. While this technique is not widely used in today's data centers, some operating systems, such as Solaris [3, 20], offer build-in support for page retirement. For the standard Linux kernel there exists a patch that allows one to retire bad pages [17]. However, there is no rigorous study on the possible effectiveness of page retirement for realistic error patterns and there is no published work comparing different policies for deciding on when to retire a page.

The main trade-off in retiring pages is the amount of memory lost due to retired pages versus the number of future errors prevented. An ideal retirement policy detects as early as possible pages that are likely to develop a large number of errors in the future and retires only those pages. We have observed several indicators that lend themselves well for such predictions. Below are a few simple policies that were directly derived from the findings in Section 3.

repeat-on-address: Once an address experiences a repeat error the corresponding page is retired.

1-error-on-page: Since a large fraction of addresses with errors experiences a repeat, this policy pessimistically assumes after the first occurrence of an error on a page that it will turn into a hard error and retire the page.

2-errors-on-page: This policy retires a page once two errors have been observed on this page (either on the same address or on two different addresses on the page).

repeat-on-row: Since a row with 2 unique addresses has high chances of developing additional errors, this policy retires all the pages on a row after two errors have been observed.

repeat-on-column: Equivalent to repeat-on-row, but for columns.

We simulate all of the above policies on our trace data for BG/L and BG/P and report the results in Table 4. For each policy

		All nodes with errors					Nodes w/ multi-bit/ chipkill error	
System	Policy	Pages retired	95%ile pages retired	Errors avoided (%)	MBEs avoided (%)	Chipkill avoided (%)	Pages retired	95%ile pages retired
BG/L	repeat on address	2.2	2	94.2	88.1	N/A	15.8	103.25
	any 1 error on page	3.8	4	96.8	96.7	N/A	42.4	319
	any 2 errors on page	2.4	3	94.9	94.8	N/A	24.6	123.8
	repeat on row	33.9	32	95.6	97.3	N/A	245.5	1620
	repeat on column	14,336	16,384	96.5	90.6	N/A	257,930	1,212,416
BG/P	repeat on address	4.8	18	86.3	86.4	61.8	4.8	18
	any 1 error on page	17.6	62.7	91.4	91.5	71.0	17.7	62.7
	any 2 errors on page	6.9	25.6	88.1	88.1	64.7	6.9	25.6
	repeat on row	158.0	576	92.6	92.7	77.0	158	576
	repeat on column	49,989	266,650	91.9	92	67.3	49,972	266,650

Table 4. Effectiveness of page retirement

we include the average number of pages it retires per machine with errors, the 95th percentile of the number of pages retired per machine, the percentage of all errors in the system that would have been prevented by this policy (because they fall on previously retired pages) and the percentage of all multi-bit and chipkill errors that could have been prevented.

We find that even the simple policies we are considering are quite effective at reducing the number of errors a system would observe: All policies are able to prevent nearly 90% of all errors. The most aggressive policies (retiring a page immediately after just one error, or retiring whole rows and columns) are able to avoid up to 96% of all errors. The main difference between policies lies in the cost involved in achieving this performance. The number of pages retired per machine averages at only 2.2 - 4.8 for the repeat-on-address policy, which is a small price to pay for a large gain in the number of avoided errors. For policies that retire entire rows or columns this number can grow into hundreds or thousands of pages retired per machine. Retiring entire columns is particularly expensive, due to the large number of pages that a column spans, and is prohibitive, at least in the form of the very simple policies that we have experimented with.

Another interesting finding from our simulation study is the effectiveness of page retirements in avoiding multi-bit and chipkill errors. All policies are able to avoid around two thirds of all chipkill errors and nearly 90% of all multi-bit errors. While a system with chipkill ECC would have been able to mask all of these errors, the high reduction of errors under page retirement is still interesting as it does not come with an increase in hardware cost or energy consumption. The only price to pay is the reduced amount of memory available, due to retired pages.

While the average number of pages retired per machine averaged across all machines with errors is low when the right policy is chosen, this number might be higher for machines that experience multi-bit or chipkill errors (or more precisely would have experienced multi-bit or chipkill errors in the absence of page retirement). We therefore also computed the statistics for the number of pages retired per machine for only those machines in our dataset that experienced multi-bit and chipkill errors and report the results in the right half of the above table. We find that both the average number of pages retired and the 95th percentile of the number of pages retired is still very small, compared to the total amount of memory in modern server systems. For example, under the repeat-on-address policy 5-16 pages are retired for an average machine with errors. A machines in the 95th percentile of number of retires pages, still sacrifices only 18-104 pages, i.e. less than half a MByte of total DRAM space. Even the for the more aggressive 1-error-on-a-page policy the number of retired pages is still in the same order of magnitude. More elaborate techniques based on statistical modeling or machine learning might be able to further improve on the cost-efficiency trade-off of page retirement policies.

4.2 Selective error protection

Several of our findings indicate that errors are not uniformly distributed in space. For example, we saw evidence that some parts of a chip and of the physical address space experience higher error rates than others. This implies that selective error protection mechanisms might be an interesting avenue for future work. For example, approaches along the lines of the recent work on virtualized and flexible ECC [24] might provide effective solutions that exploit the differences in error rates in different parts of the system.

4.3 Proactive error detection and monitoring

The two previous subsections provide examples for techniques that operating systems can use to exploit the characteristics of hard errors in order to reduce the negative impact of DRAM errors on system availability. However, such techniques require that the operating system has full knowledge of all errors happening at the underlying hardware level, including error counts, as well as more detailed information, such as the addresses that were affected. ECC protection used in most server systems masks the presence of errors and it is typically not trivial to obtain location information on errors. Our findings about the nature of DRAM errors provide strong encouragement to improve error tracking and reporting to the operating systems.

The second factor limiting the amount of knowledge about the underlying error process stems from the fact that DRAM errors are latent, i.e. they will not be detected until the affected cell is accessed. The chances that an error will eventually lead to an uncorrectable error, causing system downtime, increases the longer it is left latent. While hardware scrubbers provide an attempt to proactively detect errors and hence reduce this time, we observed in Section 3.3 that their effectiveness might be limited. We speculate that this is likely due to the passive monitoring approach that they are taking, rather than actively attempting to expose errors. Given the large amount of idle time that typical servers in data centers experience [2], it might be worthwhile to invest a small fraction of this idle time to periodically run a memory test, similar in functionality to memtest86, that actively probes the DRAM for memory errors. Memory tests have a much higher potential for detecting hard errors than a scrubber, since they can create access patterns that stress the memory, and because they actively write to memory cells, rather than just checking the validity of currently

written values. Such tests could either be done periodically on all machines, or they could be used selectively in cases where there is a suspected hard error, e.g. after observing an earlier error in the system (that has not yet repeated).

4.4 Effectiveness of hardware mechanisms

While this was not the original focus of our study, our analysis lets us also draw conclusions about the practical value of hardware mechanisms, such as chipkill ECC, in reducing the rate of machine crashes due to uncorrectable errors. In the presence of only soft errors, the occurrence of error patterns requiring chipkill ECC would be extremely unlikely. Our observation that a large number of errors observed in the field is likely due to hard errors provides firmer grounding for using these techniques in practice, despite their added cost and energy overheads.

Only one earlier study [19] that is based on large-scale field data comments on the effectiveness of chipkill, however they only observe that in their systems under study hardware platforms with chipkill show lower rates of uncorrectable errors than hardware platforms without chipkill. They are not able to quantify how much of this difference in error rates is due to the use of chipkill versus other hardware differences between the platforms.

Our fine-grained data allowed us to quantify exactly the number of errors that required chipkill to be corrected and that would have led to a machine crash in the absence of chipkill. We find that a significant fraction of machines experienced chipkill errors, i.e. errors whose correction was only possible with the use of chipkill techniques. In fact, among the errors in our study, a large fraction (17%) of them required the use of chipkill for correction providing some tangible benefits of the use of chipkill. We can therefore conclude that for systems with stringent availability requirements the reduction in machine crashes due to uncorrectable errors might make chipkill well worth the price.

4.5 System evaluation

Any evaluation of the impact of DRAM errors on system reliability or the effectiveness of mechanisms protecting against them relies on realistic assumptions about the characteristics of the underlying error process. In the absence of field data (or realistic models built based on field data), both analytical and experimental work typically rely on very simplistic assumptions about errors. For example, analytical models often assume that errors follow a Markov process and experimental work often relies on injecting errors at uniformly randomly generated locations. Given the high occurrence rate of hard errors, these simple approaches are likely to give misleading results (or results that represent only the less relevant scenario of a system experiencing only soft errors), as they do not capture any of the correlations and patterns present in hard errors.

While we are hoping that the findings we report here will help researchers and practitioners to base their work on more realistic assumptions on DRAM errors, we believe that more work in this direction is necessary. Towards this end, we are currently working on developing statistical models capturing the various properties of DRAM error process that can be used to generate realistic patterns in simulation or for error injection. More importantly, it seems that the current lack of publicly available field data that researchers can use to drive their experiments is a major roadblock. Li et al. have graciously made the data collected for their study on DRAM errors [11] publicly available. While we were unfortunately not able to obtain permission to share the Google data used in this study, we are currently preparing a public database of all error patterns that we have extracted from the data for the Blue Gene systems. This database and the raw logs for the Blue Gene systems will be made publicly available as part of the Usenix Failure Data Repository [23].

5. Conclusions

While a large body of work has been dedicated to studying the characteristics of DRAM errors and how to best protect against them, the large majority of this work has focused on soft errors in DRAM. Our work presents the first study based on data from a large number of production systems that shows that a large fraction of errors observed in the field can be traced back to hard errors. For all systems we studied, more than a third of all memory banks that experienced errors show signs of hard errors, most commonly in the form of repeating errors on the same physical address within less than 2 weeks. Repeating errors on the same row/column are also common error modes. For some systems, as many as 95% of all observed errors can be attributed to hard errors. We also provide a detailed study of the statistical characteristics of hard errors. Some of these provide direct insights useful for protecting against errors. For example, we observe that not all areas in memory are equally likely to be affected by errors; specific regions such as low rows/columns have higher error probabilities. We speculate that this might be due to different usage patterns in different memory areas, as we observe for example that those areas used by the OS tend to see larger error counts. Furthermore, from the perspective of the OS, a large fraction of the errors observed in a system is usually concentrated on a small set of pages providing some motivation for proactively retiring pages after they experience errors. We also observed that errors that have the highest potential to be uncorrectable, such as multi-bit errors and errors that require chipkill for correction, are usually preceded by more benign early warning signs, such as repeating errors on individual addresses, rows or columns. Finally, we observe that a significant number of errors is complex enough to require more than simple SEC-DED error correction to be corrected. A significant number of nodes with correctable errors in our study activated more advanced ECC mechanisms (20%-45% activated redundant bit-steering, and 15% activated Chipkill) and a large fraction (17%) of all errors required the use of chipkill for error correction.

As a second contribution, we identify various implications on resilient system design that follow from the insights derived from our measurement study. One of our findings is that simple page retirement policies can potentially mask a large number of errors with only a small sacrifice in the amount of available DRAM. For example, a simple policy that retires a page after the first repeat error on an address on this page can mask up to 95% of all errors and up to 60% of errors that would require chipkill for correction, while giving up only a few dozen pages of main memory. This is an interesting finding, since based on discussions with administrators of large datacenters, the use of page retirement is not widely spread in practice, although it has been implemented in some systems in the past [3]. On the other hand, we find that a commonly used technique for proactively detecting memory errors, the use of background memory scrubbers, might not be as effective as one might think. We hypothesize that this is because a large fraction of errors are intermittent, i.e. they manifest only under certain access patterns. This observation, together with the observed high frequency of hard (and hence repeatable) errors, might make it worthwhile to use the idle time that most servers in datacenters experience to periodically run a memory test to actively probe for errors, in particular after observing prior errors on a node. Finally, the fact that different areas of memory experience different error rates and that usage likely plays a role in error frequencies suggests an interesting avenue for future work might be selective error protection mechanisms, where different protection mechanisms are used for different areas of memory.

Acknowledgments

We would like to thank Thomas Gooding, Mark Megerian, and Rob Wisniewski from IBM for helping us acquire very detailed information about the BlueGene systems. We would also like to thank Pete Beckman, Rinku Gupta, Rob Ross and everybody else at Argonne National Laboratory who was involved in collecting and making available the BG/P data and helped us interpret the data. The third author thanks Google for hosting her as a visiting faculty during the summer of 2009, where part of this work started. In particular, she would like to thank John Hawley, Xin Li, Eduardo Pinheiro, Nick Sanders, and Wolf-Dietrich Weber for their help in accessing the data and answering questions about the data and systems at Google. We thank Adam Oliner, Jon Stearley and Sandia National Laboratories for making the BG/L data available. Finally, we would like to thank the members of SciNet, in partiular Chris Loken and Ching-Hsing Yu, for providing us with the data from their GPC system. This work has been funded by an NSERC discovery grant.

References

[1] Soft errors in electronic memory – a white paper. *Tezzaron Semiconductor*. URL `http://tezzaron.com/about/papes/soft_errors_1_1_secture.pdf`.

[2] L. A. Barroso and U. Hölzle. The case for energy-proportional computing. *IEEE Computer*, 40(12), 2007.

[3] T. M. Chalfant. Solaris operating system availability features. In *SunBluePrints Online*, 2004.

[4] T. J. Dell. A white paper on the benefits of chipkill-correct ECC for PC server main memory. *IBM Microelectronics*, 1997.

[5] T. J. Dell. System RAS implications of DRAM soft errors. *IBM J. Res. Dev.*, 52(3), 2008.

[6] P. E. Dodd. Device simulation of charge collection and single-event upset. *IEEE Nuclear Science*, 43:561–575, 1996.

[7] A. Gara. Overview of the Blue Gene/L system architecture. *IBM J. Res. Dev.*, 49:195–212, March 2005.

[8] IBM journal of Research and Development staff. Overview of the IBM Blue Gene/P project. *IBM J. Res. Dev.*, 52(1/2):199–220, January 2008.

[9] H. Kobayashi, K. Shiraishi, H. Tsuchiya, H. Usuki, Y. Nagai, and K. Takahisa. Evaluation of lsi soft errors induced by terrestrial cosmic rays and alpha particles. Technical report, Sony corporation and RCNP Osaka University, 2001.

[10] X. Li, K. Shen, M. Huang, and L. Chu. A memory soft error measurement on production systems. In *Proc. USENIX Annual Technical Conference (ATC '07)*, pages 21:1–21:6, 2007.

[11] X. Li, M. C. Huang, K. Shen, and L. Chu. A realistic evaluation of memory hardware errors and software system susceptibility. In *Proc. USENIX Annual Technical Conference (ATC '10)*, pages 75–88, 2010.

[12] T. C. May and M. H. Woods. Alpha-particle-induced soft errors in dynamic memories. *IEEE Transactions on Electron Devices*, 26(1), 1979.

[13] B. Murphy. Automating software failure reporting. *ACM Queue*, 2, 2004.

[14] E. Normand. Single event upset at ground level. *IEEE Transaction on Nuclear Sciences*, 6(43):2742–2750, 1996.

[15] T. J. O'Gorman, J. M. Ross, A. H. Taber, J. F. Ziegler, H. P. Muhlfeld, C. J. Montrose, H. W. Curtis, and J. L. Walsh. Field testing for cosmic ray soft errors in semiconductor memories. *IBM J. Res. Dev.*, 40(1), 1996.

[16] M. Ohmacht. Blue Gene/L compute chip: memory and Ethernet subsystem. *IBM J. Res. Dev.*, 49:255–264, March 2005.

[17] R. V. Rein. BadRAM: Linux kernel support for broken RAM modules. URL `http://rick.vanrein.org/linux/badram/`.

[18] B. Schroeder and G. A. Gibson. A large scale study of failures in high-performance-computing systems. In *Proc. Int'l Conf. Dependable Systems and Networks (DSN 2006)*, pages 249–258, 2006.

[19] B. Schroeder, E. Pinheiro, and W.-D. Weber. DRAM errors in the wild: a large-scale field study. In *Proc. 11th Int'l Joint Conf. Measurement and Modeling of Computer Systems (SIGMETRICS '09)*, pages 193–204, 2009.

[20] D. Tang, P. Carruthers, Z. Totari, and M. W. Shapiro. Assessment of the effect of memory page retirement on system RAS against hardware faults. In *Proc. Int'l Conf. Dependable Systems and Networks (DSN 2006)*, pages 365–370, 2006.

[21] H. H. Tang. Semm-2: a new generation of single-event-effect modeling tools. *IBM J. Res. Dev.*, 52:233–244, May 2008.

[22] H. H. K. Tang, C. E. Murray, G. Fiorenza, K. P. Rodbell, M. S. Gordon, and D. F. Heidel. New simulation methodology for effects of radiation in semiconductor chip structures. *IBM J. Res. Dev.*, 52:245–253, May 2008.

[23] USENIX. The computer failure data repository (CFDR). URL `http://cfdr.usenix.org/`.

[24] D. H. Yoon and M. Erez. Virtualized and flexible ECC for main memory. In *Proc. 15th Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS '10)*, pages 397–408, 2010.

[25] J. Ziegler. IBM experiments in soft fails in computer electronics. *Political Analysis*, 40(1):3–18, 1996.

[26] J. F. Ziegler. Terrestrial cosmic rays. *IBM J. Res. Dev.*, 40:19–39, January 1996.

[27] J. F. Ziegler and W. A. Lanford. Effect of Cosmic Rays on Computer Memories. *Science*, 206:776–788, 1979.

[28] 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. *IEEE J. Solid-state Circuits*, 33:246–252, 1998.