An Empirical Study of Memory Hardware Errors in A Server Farm*

Xin Li Michael C. Huang Kai Shen
University of Rochester
{*xinli@ece, huang@ece, kshen@cs*}*.rochester.edu*

Lingkun Chu
Ask.com
lchu@ask.com

Abstract

The integrity of system hardware is an important requirement for providing dependable services. Understanding the hardware's failure mechanisms and the error rate is therefore an important step towards devising an effective overall protection mechanism to prevent service failure. In this paper we discuss an on-going case study of memory hardware failures of production systems in a server-farm environment. We present some preliminary results collected from 212 machines. Our observations under a normal, non-accelerated condition validate the existence of all failure modes modeled in the previous literature: single-cell, row, column, and whole-chip failures. We also provide a quantitative analysis of the error rates.

1 Introduction

Modern computer systems, especially servers, demand large reliable memories. While the capacity demand is met by the relentless drive towards higher device density, technology scaling by itself does not improve memory's reliability. Quite the contrary, shrinking device size fundamentally reduces memory cell's noise tolerance and the sheer increase in the number of devices suggests higher probability for any one of them to become faulty.

At the device level, many fault mechanisms can affect different parts of the memory hardware, causing an error to a particular cell, a row, or an entire chip and so on. Some faults are largely environmental such as particle strike from radioactive decay and cosmic ray-induced neutrons [12–15] and are unpredictable and transient in nature. Others are due to manufacturing defect or device aging [2]. Once they manifest, these faults tend to cause more predictable errors as the deterioration is often irreversible. However, before transitioning into permanent faults, they may put the device into a marginal state causing apparently intermittent errors. In the rest of this paper, we use the term "permanent fault" to represent faults that are likely due to device defect or aging (regardless of whether they manifest permanently or intermittently).

*This work was supported in part by the National Science Foundation (NSF) grants CCR-0306473, ITR/IIS-0312925, CNS-0509270, CNS-0615045, and CCF-0621472. Shen was also supported by an NSF CAREER Award CCF-0448413 and an IBM Faculty Award.

Understanding memory hardware errors is important to developing an overall strategy of maintaining system dependability. For instance, the best way to handle different errors depend on their characteristics. Permanent errors seem to call for hardware replacement to prevent future catastrophic failures. However, it remains to be seen whether system-level mechanisms can help cope with certain isolated faults to reduce waste of human administrative effort and hardware. Furthermore, given the ubiquity of ECC protection in server systems, the underlying failure mechanisms are often masked, leaving the high-level system unaware of developing device fatigue. Whether (and if so, how) correctable errors should be tracked and reported to the operating system is also worth exploring.

While earlier studies have significantly improved the understanding of memory errors, they tend to be lacking in some respects: A large portion of the literature body is devoted only to transient (or soft) errors [11,14]; Many studies are limited to *accelerated* tests (tests under artificial error-prone conditions) or theoretical analysis [3]; A few studies on permanent errors (e.g., [4, 9]) focus only on error detection and mitigation mechanisms. The most closely related work that we know of [6] reported the number of non-transient errors observed in a 16-month field test of 193 machines. However, little detail on error characteristics and result analysis is provided in that study.

In this paper, we present an empirical study on memory hardware errors on a large set of production machines in a server-farm environment. In this on-going effort, we leverage the memory chipset's capability to monitor the ECC-protected DRAM main memory of 212 machines continuously and non-intrusively. Our current results can not yet provide statistically tight bounds on error rate estimations. Nonetheless, the results do suggest that permanent errors are non-trivial. Specifically, out of the 212 19-month old machines, at least 9 machines have demonstrated symptoms of permanent memory errors; all memory hardware error modes (single-cell, row, column, and whole-chip) appear to have manifested; at least one machine demonstrated the existence of uncorrectable errors.

2 Measurement Methodology

Our 212 servers were from a particular server-farm environment at ask.com [1]. All of these machines use Intel

E7520 chipsets as memory controller hub [10]. On average, each machine is equipped with 3.92 GB of DDR2 SDRAM protected by ECC capable of single-bit-error correction and double-bit-error detection (SEC-DED). Single-bit errors detected during normal accesses are automatically corrected by the system.

In addition to error detection and correction, the memory controller attempts to log some information about memory errors encountered. Due to the limited control register space available within the chipset, this logging capability is rather limited. In particular, there are two registers to track the address of two distinct errors. After a reset, these registers will capture the first two memory errors encountered. However, until the next reset, any subsequent errors will not be logged. Therefore, we periodically probe the memory controller to read out the information and reset the registers in order to capture new errors. Specifically, we probe the controller once every hour. This is set to limit both the overhead and the probability of missing errors. The required implementation for memory controller probing is straightforward, involving simple enhancements of the memory controller driver inside the OS kernel [5].

We discuss several scenarios that a memory error might escape detection.

- First, a defective memory address may not be accessed by the executing software for a long time and thus error detection based solely on software access cannot detect it. To prevent such missed detections, we enable hardware memory scrubbing in our error collection tests. Memory scrubbing is a background process that scans all memory addresses to detect errors (and correct them if they are transient single-bit errors). It is typically performed at a low frequency (e.g., 1.5 hours for every 1 GB) [10] to minimize the energy consumption and contention with running applications.

- Second, permanent error on a memory bit occurs as either stuck-at-1 (the bit stays at "1" regardless how software attempts to update it) or stuck-at-0. Therefore an error may not be visible if the data written to the bit matches its sticking state. This effect of missed detections is mitigated by our continuous monitoring over a period of time (about 3 months at the time of this writing). Nevertheless, error visibility still depends on how defective memory addresses are updated and the values that are being written. Indeed, in our collected error traces, some memory addresses containing permanent errors are not detected until several days after we started monitoring.

3 Error Findings

We have collected error logs for a period of approximately 3 months. So far, the log has shown more than 8000 instances of single-bit errors. These error occurrences are from 924 distinct addresses distributed on 11 machines. There is also evidence that at least one uncorrectable multi-bit error has occurred on one machine. However, due to the artifact of the memory controller logging mechanism, we do not know the exact address of the error[1].

Machine	Single-cell	Row	Column	Whole-chip
A	1			
B		1		
C			1	
D				1
E	2	11	1	
F		1		
G		1		
H	1			
I	1			
J	1 (transient)			
K	1 (transient)			
Total	7 (2 transient)	14	2	1

Table 1. Collected errors and their modes.

We choose six of our machines with distinct error patterns and demonstrate how the errors are distributed on the physical layout of the memory arrays in Figure 1. All four memory failure modes (single-cell, row, column, and whole-chip) are observed in our log[2]. Specifically, machine A contains a single cell failure, machines B and C represent a row failure and a column failure respectively. Finally, for machine D, the errors are spread all over the chip which strongly suggests failure in the chip-wide circuitry rather than individual cells, rows, or columns. Based on the pattern of error addresses, we categorize all error instances into appropriate modes shown in Table 1. In total, there are 22 distinct sources of these errors.

Note that this categorization only serves to better understand the errors and is not definitive. Indeed, in some cases, it is hard to tell with certainty what the root causes of the manifested errors are. For instance, the two errors in machine F occur in the same row and thus may be caused by two single-cell failures or a row failure. In our categorization, we assume single cell errors are independent and therefore the chance for two independent single-cell errors to occur at the same row or column is extremely low. Therefore, when multiple errors occur in the same row (or column), we treat them as the manifestation of a row (column) failure.

While the error-correction logic can detect errors, it does not track the causes of the errors. Recall in Section 1, we defined transient errors as those that are caused solely by

[1] Basically, the tracking of multiple-bit errors and single-bit errors share the same global error information registers. When single-bit errors outnumber the multi-bit ones significantly, the registers are always filled with information of single-bit errors.

[2] The row errors were manifested as a streak of single-bit correctable errors instead of multiple-bit errors. This is due to the *chipkill* arrangement [7] that the memory controller employed.

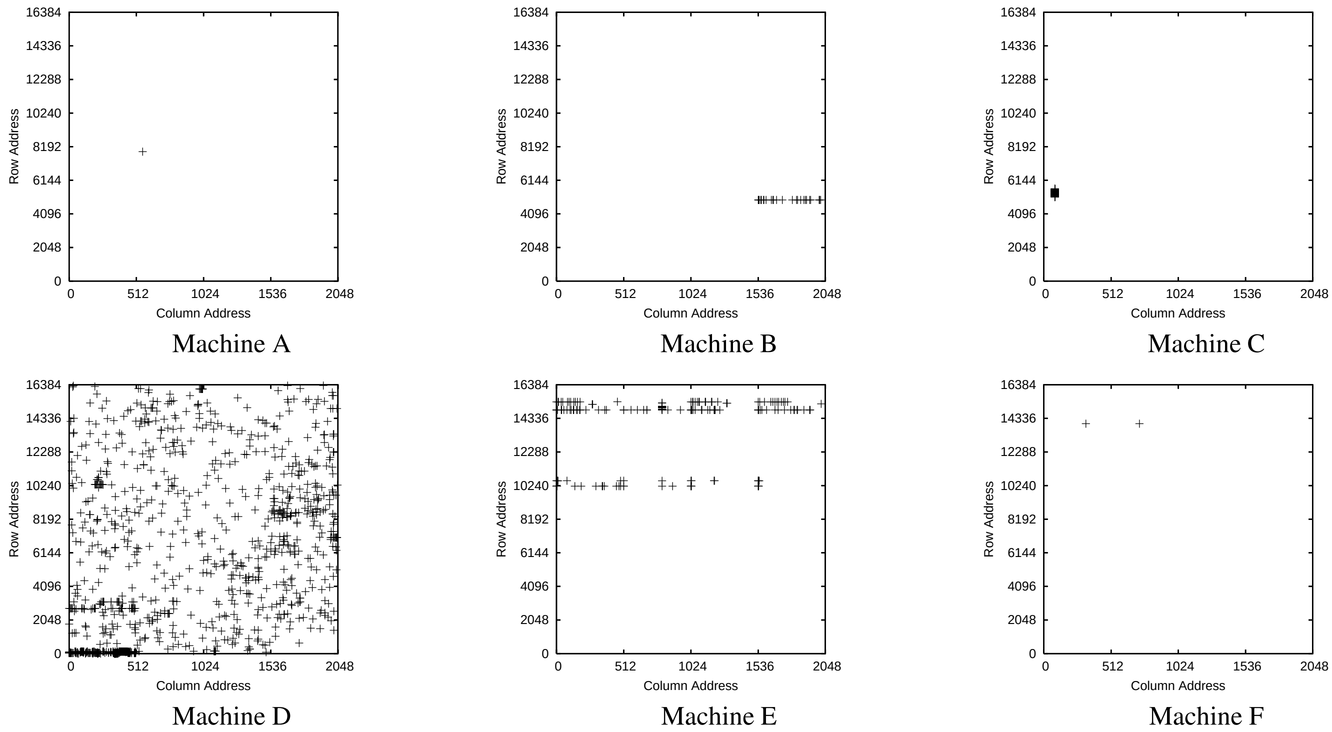

Figure 1. The visualization of errors on physical memory arrays. The y-axis shows the row address, and x-axis shows the column address. The lower two bits of the column address are masked in the memory controller registers and we assume they are "00" in the visualization. The system address to row/column address translation is obtained from the official Intel document [10]. Each cross represents an erroneous cell at its row/column address. For machine D, the errors were obtained from all four memory banks of the machine, whereas in the other figures, errors only occur on one of the banks.

environmental factors. Conversely, permanent errors are the errors whose causes are related to hardware defects. In this paper we do not target at making a clear distinction between the error types. However, by examining the manifested characteristics, we can still tell some signs of the nature of errors. First, transient errors are caused by external noises and should affect all elements with largely the same probability. With the assumption that they manifest independently on different memory addresses, we can be virtually certain that repeated errors on the same address are caused (at least partially) by permanent defects. Second, there are four failure modes in memory: single-cell, row, column, and whole-chip [3]. Of these, certain memory failure modes will necessarily generate many errors with correlated addresses. For instance, memory row failures will manifest as a series of errors with addresses on the same memory row. Some addresses on this row may be caught on the log only once. Yet, the cause of the error is clearly permanent.

To summarize, when an error address meets either one of the following conditions, we treat it as a permanent error:

1. The same address is observed more than once in the machine's error log.
2. The address is observed only once, but there are other addresses in the same machine with multiple errors instances in the log.

If we apply this rule, there are 9 machines with permanent memory errors. The other 2 machines each logged only one single-bit error during the entire period. In all likelihood, these are transient errors. Given the extensive treatment of transient (or soft) errors in the literature (including our own [11]) and the relative abundance of permanent errors discovered in our measurement, the rest of this paper focuses on the error rate analysis of permanent errors that are likely due to device defect or aging.

4 Error Rate Analysis

In total, we discovered 22 permanent hardware errors out of the memory modules on 212 machines. Below we calculate a probabilistic interval on the failure-in-time rate for each one of these failure modes.

The actual random process that the error occurrences follow is debatable. We follow a previous study [8] and assume that the error occurrences follow the Poisson process. Therefore according to the probability mass function

Failure Mode	Average Rate	Lower Bound	Upper Bound
Single-cell	1759.2 FITs	478.5 FITs	4366.4 FITs
Row	4925.8 FITs	2589.6 FITs	8363.4 FITs
Column	703.7 FITs	52.8 FITs	2846.4 FITs
Whole-chip	351.8 FITs	3.5 FITs	2118.1 FITs

Table 2. Average error rate and the 99%-probability error rate interval for four failure modes. The rates are in FITs (failure in time, errors in 10^9 hours) per 4 GB (approximately the amount of memory on each machine).

of Poisson distribution, within a time-memory extent T, the probability that k errors happen is

$$Pr_{\lambda,T}[N = k] = \frac{e^{-\lambda T}(\lambda T)^k}{k!} \tag{1}$$

where λ is the average error rate (i.e., the error occurs λ times on average for every unit of time-memory extent).

For a given T, and error count k, let us call $[\Lambda_1, \Lambda_2]$ a p-probability interval of the average error occurrence rate, which is defined as:

$$\forall \lambda \notin [\Lambda_1, \Lambda_2]: \quad Pr_{\lambda,T,k}[N = k] < 1 - p$$

In other words, if a computing environment has an average error occurrence rate that is outside the p-probability interval, then the chance for k occurrence during a measurement of time-memory extent T is always less than 1-p.

According to Equation (1), we can calculate a 99%-probability interval of the average error occurrence rate, by plugging in the error numbers k, and the time-memory extent T. There are about 831.04 GB on these machines and they have been operational for about 570 days[3]. Since $T = 831.04\text{GB} \times 570\text{day}$ for our measurement, the per 4 GB 99%-probability interval of rates for the four modes are given in Table 2, along with the observed average error rates.

5 Conclusions and Future Work

In this paper, we have presented an empirical study on permanent memory hardware errors. In contrast to earlier studies, our error collection is performed under normal, non-accelerated conditions on real production systems. After monitoring 212 19-month-old machines in a server farm environment for 3 months, we were able to collect information that identifies 22 errors in total. They contain patterns that validate all the failure modes in the previous literature: single-cell, row, column, and whole-chip failure. The result suggests that there are non-trivial amount of permanent memory hardware errors.

[3]We should use the machine operational time (570 days) instead of our measurement duration (around 3 months) when calculating the rate of permanent errors.

This is on-going research work. In the future, as we obtain more information from the monitoring, we will be able to make more detailed analysis on the properties of memory errors. In particular, we will study whether error rate accelerates. We will also explore the underlying mechanism of the errors, and make a prediction of the error trends. Furthermore, we shall study how these errors manifest at higher system levels (e.g., operating systems and software applications) given existing hardware counter-measures such as ECC.

References

[1] IAC search & media (formerly Ask Jeeves Search). `http://www.ask.com`.

[2] J. Black. Mass Transport of Aluminum By Momentum Exachange with Conducting Electrons. In *International Reliability Physics Symposium*, Los Angeles, California, Nov. 1967.

[3] M. Blaum, R. Goodman, and R. Mceliece. The reliability of single-error protected computer memories. *IEEE Trans. on Computers*, 37(1):114–118, 1988.

[4] D. M. Blough. On the reconfiguration of memory arrays containing clustered faults. In *Int'l Symp. on Fault-Tolerant Computing*, pages 444–451, 1991.

[5] EDAC (Error Detection and Correction) project. `http://bluesmoke.sourceforge.net`.

[6] C. Constantinescu. Impact of deep submicron technology on dependability of VLSI circuits. In *Int'l Conf. on Dependable Systems and Networks*, pages 205–209, 2002.

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

[8] S. A. Elkind and D. P. Siewiorek. Reliability and performance of error-correcting memory and register arrays. *IEEE Trans. on Computers*, (10):920–927, 1980.

[9] M. Franklin and K. K. Saluja. Pattern sensitive fault testing of RAMs with bullt-in ECC. In *Int'l Symp. on Fault-Tolerant Computing*, pages 385–392, 1991.

[10] Intel e7520 chipset datasheet: Memory controller hub (mch). `http://www.intel.com/design/chipsets/E7520_E7320/documentation.htm`.

[11] X. Li, K. Shen, M. Huang, and L. Chu. A memory soft error measurement on production systems. In *USENIX Annual Technical Conf.*, 2007.

[12] T. C. May and M. H. Woods. Alpha-particle-included soft errors in dynamic memories. *IEEE Trans. on Electron Devices*, 26(1):2–9, 1979.

[13] 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. of Research and Development*, 40(1):41–50, 1996.

[14] J. Ziegler et al. IBM Experiments in Soft Fails in Computer Electronics (1978-1994). *IBM Journal of Research and Development*, 40(1):3–18, Jan. 1996.

[15] J. Ziegler and W. Lanford. Effect of Cosmic Rays on Computer Memories. *Science*, 206(16):776–788, Nov. 1979.