A Rising Tide Lifts All Boats: How Memory Error Prediction and Prevention Can Help with Virtualized System Longevity

Yuyang Du Hongliang Yu Yunhong Jiang[§] Yaozu Dong[§] Weimin Zheng
Department of Computer Science and Technology, Tsinghua University
[§] *Intel Research and Development, Asia-Pacific*

Abstract

Memory is the most frequently failing component that can cause system crash, which significantly affects the emerging data centers that are based on system virtualization (e.g., clouds). Such environment differs from previously studied large systems and thus poses renewed challenge to the reliability, availability, and serviceability (RAS) of today's production site that hosts a large population of commodity servers. The paper advocates addressing this problem by exploiting memory error characteristics and employing a cost-effective self-healing mechanism. Specifically, we propose a memory error prediction and prevention model, which takes as input error events and system utilization, assesses memory error risk, and manipulates memory mappings accordingly (by page/DIMM replacement or VM live migration) to avoid potential damage and loss.

1 Introduction

Errors in Dynamic Random Access Memory (DRAM) [5, 14, 15] play an important role in system crash that causes major problems, such as service unavailability, data loss and corruption, etc. Although single occurrence of memory error is rare, to consider today's production site that hosts a large population of commodity servers (such as Google's server fleets), their accumulative effect is not insignificant. Studies [18, 19] show that memory errors make up the largest portion of system failures among all possible reasons (not only hardware faults) and take up almost the top place of component replacement.

Hardware fault tolerance [17] in general has been an active research area for a long time. To tolerate various faults, adding certain form of redundancy (by hardware or software) to the system is necessary. These techniques are primarily designed for mission critical applications that are unaware of the costs or orchestrated systems that can afford efforts to build.

However, in the emerging large data centers that are based on system virtualization (e.g., clouds [2]), the situation is different. The provisioning of computing resources often abides by "paying for what you use" model. The commercial contract between providers and consumers is expected to meet Service Level Agreement (SLA) with financial penalties, e.g., Percentage of Uptime [2]. Unfortunately, it is impossible for cloud providers to massively employ fault tolerance techniques due to cost-inefficiency. Despite SLA, consumers desire continuous availability and reliability but not the burden to accomplish it. Our paper aims to address this dilemma to an extent.

We take advantage of recent studies on the characteristics of hardware memory errors [13, 20] and employ a cost-effective self-healing mechanism [21] that diagnoses, predicts, and prevents memory errors to reduce their potential damage and loss at system level. We use the latest processor feature (i.e., Intel MCA [10]) that provides the ability to detect and report hardware errors. We propose a memory error prediction and prevention model, which takes as input error events and system utilization, assesses memory error risk, and manipulates memory mappings accordingly (by page/DIMM replacement or VM live migration [7]), such that the hypervisor is enabled to guard guest VMs against memory errors, increasing their uptime.

The contribution of this paper is a practical one. We designed effective and efficient memory error prediction and prevention mechanism in virtualized systems, which can provide scalability, lossless performance, and cost-efficiency "to the masses", therefore it will benefit from the scale effects in that a significant number of system failures can be saved expectably.

The remainder of this paper is organized as follows. In Section 2, we introduce the background and motivation. In Section 3, we propose our design model of the memory prediction and prevention mechanism. Then we present our implementation in Section 4. At last, we talk about related work in Section 5 followed by the conclusion and future work in Section 6.

2 Background and Motivation

In this section, we first introduce memory errors and their characteristics learnt from previous studies. Then we motivate our paper by discussing the extent to which memory error prediction and prevention can alleviate the threat faced by growing size of memory configurations and number of computers in system virtualization based large server farms. Finally, hardware requirement of this study is presented.

2.1 Memory Error Characteristics

Broadly speaking, the memory error we focus is hardware error. Further on, the memory errors can be categorized into soft errors, which randomly flip memory bits but without permanent physical damage (e.g., particles strike on the silicon chip); and hard errors, which repeatedly corrupt bits due to device defect (e.g., device wearout). The consequence of memory errors is dependent on their types. If the errors can be corrected by hardware (e.g., Error Correcting Codes), then software is oblivious to such events and can continue running; yet repeated correctable errors (i.e., one bit hard error) still lead to considerable performance loss [20]. Conversely, if the errors are uncorrectable, system failure is unavoidable.

Memory errors have intrigued research community for long, such that much work has been done to demystify their cause, impact, and behavior. We introduce two recent studies that motivate our work. Li et al [13] took measurement on over 300 machines from both production site and laboratory for up to 7 months. Schroeder et al [20] collected comprehensive data on Google's large server fleet over 2.5 years. We learn several lessons from their work:

- *Hard vs. Soft*. Hard errors overwhelm soft errors, which are unfortunately concerned by most previous studies ([13] only captured two soft errors that are also correctable, and all other errors are caused by hard errors). To this end, we concentrate on hard errors because soft errors not only account for a little portion, but also they are caused by unpredictable external factors, such as alpha-particles in cosmic rays [14].

- *Correctable Error Means*. There is strong correlation between correctable and uncorrectable errors. The chance that a DIMM who has a correctable error will have another correctable error in the same month is up to 228 times more than that of a DIMM seeing no correctable errors. Moreover, in 70-80% of the cases, an uncorrectable error is preceded by a correctable error in the same or previous month. In other words, the presence of correctable error increases the probability of uncorrectable error by factors up to 400. Since uncorrectable errors cause system crash, we leverage the correlation between correctable and uncorrectable errors to predict and prevent them from taking place.

- *Utilization Influences*. System utilization (CPU cycle rate and memory allocation) is strongly correlated to memory errors. The correctable error rate is by a factor of 2-3 higher for high system utilization than for low utilization. Thus, we take system utilization into account and balance the memory workload at three levels (page, DIMM, and server) to reduce memory error incidence.

In summary, memory errors are predictable as opposed to previous recognition. This finding can be used for system design to reduce their induced loss, e.g., intermittent correctable errors highly indicate hard errors, which will repeatedly cause soft errors at the same position and raise the probability of uncorrectable errors (if adjacent bits are flipped). [20] also shows that DIMM aging relates to memory errors, but we currently don't consider aging in this paper.

2.2 To What Extent Can Memory Error Prediction and Prevention Solve the Problem?

In contrast to previous studies, not only is serious memory error predictable, but also it is orders of magnitude higher than as anticipated [20] [1]. The FIT rates (Failures In Time per billion device hours) are 25,000 to 70,000 per Mbit. Put another way, more than 8% of DIMMs are affected yearly; the machine failure rates vary from 12% to 50% with different platforms since there are often multiple DIMMs per machine. Notably uncorrectable errors that typically bring about system crash affect 1.3% to 4% of the machines per year.

Because memory errors are not exceptional but are rather common place, we argue that memory errors need to be predicted and prevented: 1) They cause serious system crashes and costly component replacements if not

[1] Li et al [13] finds the opposite result. Here we choose [20] because their measurement period is longer; they have tested more servers with more DIMMs covering multiple vendors, capacities, and technologies; and their workloads are more stressful.

precisely prevented beforehand; 2) In today's data centers, they normally occur, so plenty of opportunities exist to predict them; 3) They have manifested themselves to be predictable through exploiting their characteristics. More importantly, since memory error self-healing mechanism is cost-effective, scalable, and of negligible overhead, it can be easily applied to data centers. The resulted scale effect ensures that the mechanism will significantly outweigh that without it, even if some unexpected errors can be overlooked.

2.3 Why Do We Focus on Virtualization?

We believe memory error prediction and prevention can apply to and benefit system design in general [21]. However, we take more interests in system virtualization, which consolidates multiple instances of operating systems (legacy or latest, proprietary or open source) into a single server for higher resource utilization, easier management, and lower energy consumption.

Modern large systems always adopt fault tolerance, such as Google search engines, however, for the emerging computing clouds that are based on system virtualization, their use model does not allow fault tolerance to be employed extensively at the level of application, operating system, or virtual machine, by either the providers or the consumers [8, 6].

In addition, the cloud infrastructure exposes significant vulnerability to memory errors: 1) Memory errors can be used to attack system security [9, 16]. As virtual machines sharing the same memory usually belong to different consumers or applications, this engenders the threat that the VM might be penetrated to violate customer confidentiality, which is proved to be susceptible to memory errors. 2) Virtualized systems naturally have higher utilization due to server consolidation, which in turn leads to an increase in memory error rate. 3) Since virtual machines are assigned to customers, the installed operating systems can be diverse. Many proprietary and legacy OSes can not handle memory errors, which can only rely on hypervisor to deal with them. 4) The "eggs in a basket" effect needs hypervisor to be more robust and resilient to system failures, to which memory errors contribute the most.

2.4 Hardware Requirement

Since memory errors are strongly correlated and need to be taken seriously in virtualized systems, the missing piece of the puzzle is that we need hardware support to let us be aware of memory errors. Our work takes advantage of Intel Machine Check Architecture (MCA) [10] that is common in modern servers.

The Pentium 4, Intel Xeon, and P6 family processors implement MCA that provides a mechanism for detecting and reporting hardware errors. Two error events are related to memory: Corrected Machine Check Interrupt (CMCI), which will be delivered when a memory error is detected and corrected, and Machine Check Exception (MCE), a non-maskable event that informs system software that an uncorrectable memory error has occurred. CMCI also provides the ability to setup a threshold such that when it is exceeded by the number of corrected memory errors, a CMCI event will be delivered to system.

3 Design

We present the memory error prediction and prevention mechanism in this section. First, our model is proposed. Then we introduce memory error risk assessment and elimination. Finally, we discuss the situation when unexpected errors that are uncorrectable occur .

3.1 Model

The design model involves five components for memory error prediction and prevention, which is illustrated in Figure 1 and described below.

- **Event Collector** interfaces with the hardware support for memory error reporting (i.e., MCA [10]). It receives MCE and CMCI events, handles them (with error handler), and may also configure the hardware as necessary.

- **Error Log** records all the memory error events and detailed information regarding them, such as memory address, time stamp, and event type etc. This log will be retrieved at later time for further analysis.

- **Utilization Monitor** keeps track of the system utilization from hypervisor: CPU cycle rate and memory allocation (hence we combine CPU usage and memory allocation together to approximate memory utilization in stead of counting memory reads and writes).

- **Risk Assessor** takes as input the system utilization and memory error events, and makes an assessment of the risk that a hard or uncorrectable error may happen. It will also balance the memory workload of the server versus the exposed threat, and activate virtual machine live migration [7] when necessary. In addition, it has a bookkeeping of all the suspected memory addresses and historical operations.

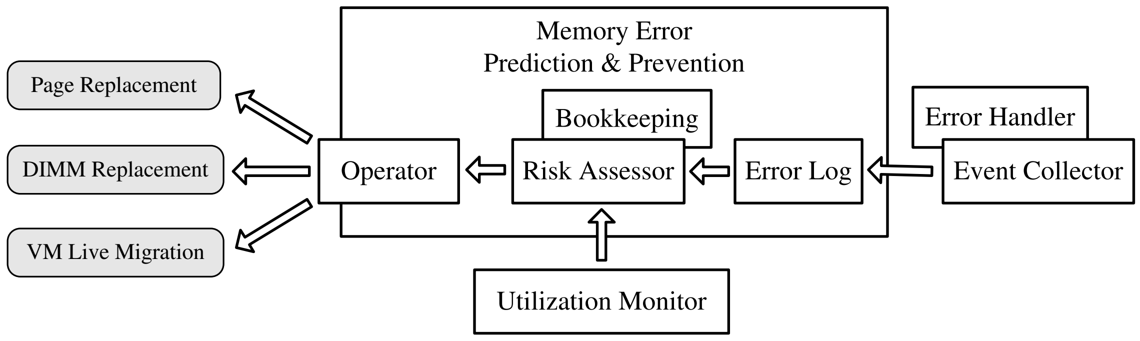

Figure 1: Memory error prediction and prevention: an overview

- **Operator** takes orders from risk assessor to perform page replacement, DIMM replacement, or virtual machine (VM) live migration to prevent the potential errors and loss.

3.2 Risk Assessment

Risk comes from hard and uncorrectable errors and can be foreshadowed by two symptoms with great probability: correctable errors and high system utilization, as is stated in Section 2.1. Therefore, we take two countermeasures to them respectively. One is error prediction and the other is hot spot avoidance.

Error Prediction. If correctable errors occur to the same memory address, it yields strong confidence that a hard error has appeared or an uncorrectable error will appear. According to how much risk the system would bear, the number of correctable errors may vary, for example, from 1 to more. If the risk is affirmative, the risk assessor will tell the operator to replace the affected page. If there are several DIMMs in the server, the risk assessor may even order the operator to replace the affected DIMM if many correctable errors have occurred. In such case, system administrator will also be alerted.

Suspected faulty memories will be put on probation for a period of time (e.g., 3 months) except those with hard error. Only when system is overloaded will the risk assessor reclaim the memory that is released from probation (but will only be allocated to low priority guests). This is done through consulting the error log and the bookkeeping that records the memory addresses previously replaced.

Hot Spot Avoidance. Utilization monitor will report the online usage for CPU and Memory to risk assessor. Risk assessor first tries to balance system utilization for each DIMM by exchanging guest VM's memory between DIMMs when appropriate. If virtual machine live migration is available, risk assessor would try to balance the load across servers in the site. Note that hot spot avoidance may lead to thrashing (not at the virtual memory level in OS) or violating energy saving policy (which concentrates workload as much as possible and turns off the unused server), however, this is a tradeoff to make and these topics will be further studied and are beyond the scope of this paper.

3.3 Unexpected Errors

If an uncorrectable memory error is encountered, the error handler in event collector (see Figure 1) first checks which DIMM owns the affected memory address. The memory is probably owned by guest VMs, because hypervisor has very small footprint. In such a case, error handler either destroys the guest VM and restarts it, or invokes guest's error handler to deal with the error by passing a simulated MCE event to the guest. By all means, the error event will be finally reported to the risk assessor. When uncorrectable errors occur often, the assessor may replace all the guest VMs affected to other DIMMs, or live migrate them to other servers.

If the hypervisor owns the faulty memory, however, error handler will probably have no choice but reboot the system. So this is a hole in our system. In fact, our work does not aim to recover uncorrectable memory errors, since it is not a complete fault tolerance mechanism. We are motivated to predict memory errors and prevent them from affecting system reliability, availability, and serviceability. More importantly, our concern is scalability, cost-efficiency, and minimal overhead, so it can be deployed to production sites for "economies of scale". Since hypervisor is small and has low system usage, we can apply fault tolerance to hypervisor. However, this is not the focus of our paper.

4 Implementation

Our work is based on Xen project [3] (an open-source virtualization platform). The hardware requested is Intel latest processors with Machine Check Architecture support [10].

Guest Physical Address (GPA) is translated to Host Physical Address (HPA) in virtualization environment. Replacing a physical page for a guest VM requires the updating of all the translations accordingly. This may include M2P (machine to pseudo-physical) and P2M (pseudo-physical to machine) address tables that are managed by hypervisor, hardware translation table (e.g., Expanded Page Table) that is used by CPU in hardware assisted mode, and the translation from guest linear address to host physical address in the direct paging mode and shadow page table. In this process, when MCA (Machine Check Architecture) interrupt happens, the hypervisor may update all the address translations. Or at later time, upon the operator's (see Figure 1) request, the hypervisor first uses an IPI (inter-processor interrupt) to interrupt the execution of guest VM to trigger the guest's VM exit, then the hypervisor captures that exit and updates all the memory address translations, and at last resumes the guest VM's execution. Shared page handling hereby is elided due to space limit. Currently, a large portion of our implementation is already included in mainline source code in Xen project.

The overhead of page replacement is very small, especially when Expanded Page Table (EPT) is used. DIMM replacement can be viewed as many successive page replacements. VM live migration is well tested in [7]. For a thorough test of our work, it would take a lot of efforts, and we plan to do that in the future.

5 Related Work

The most relevant work is MPR [21], which automatically retires pages that suffer from memory errors in Solaris 10 Operating System. Our work differs in that we focus on virtualization and take system utilization into account in balancing memory workloads to predictively reduce memory error risk. PAI [11] and Duplication Cache [1] investigate hardware methods to recover failed memory. In virtualization area, Remus [8] and [6] provide high availability to VMs by state replication, and record and replay respectively. Many studies [4, 5, 14, 15, 18, 19, 12] take various ways to demystify memory error cause, impact, or behavior. Li et al [13] and Schroeder et al [20] measured and analyzed memory errors in large scale and for long term. Their work provides the basis for our memory error prediction and prevention mechanism.

6 Conclusion and Future Work

DRAM errors can cause system failures that lead to service unavailability and data corruption. In today's large production sites with lots of servers, their effects become significant. This paper aims to predict memory errors and prevent them from affecting system reliability, availability, and serviceability (RAS) by using memory error characteristics.

Hardware fault tolerance in general adds redundancy to the system. These techniques are suited for mission critical applications or large orchestrated systems, however, the emerging cloud infrastructure that is based on system virtualization often lacks the vantage to deploy them. Therefore, we take special interests in this respect. In addition, latest processor mechanism (such as Intel MCA) provides the ability to take finer-grained actions to memory errors. In this paper, we designed a memory error prediction and prevention framework, which takes as input error events and system utilization, assesses memory error risk, and correspondingly manipulates memory mappings (page replacement, DIMM replacement, and VM live migration). Our work has been largely implemented on Xen hypervisor and included in the mainline source codes. We are currently on the way to fulfill our work and collaborate with industry to take quantitative measurements to fine-tune the predictive model and evaluate the effects.

Acknowledgments

We thank the anonymous reviewers for their insightful comments that helped improve this paper. We also thank the Intel Linux OS and VMM team for their in-depth suggestion and generous support. The work is supported by the Major State Basic Research Development Program of China (973 Program) under Grant No. 2007CB311100 and Huawei Research Fund under contract No. YBCB2009032.

References

[1] N. Aggarwal, J. Smith, K. Saluja, N. Jouppi, and P. Ranganathan. Implementing high availability memory with a duplication cache. In *MICRO 2008*.

[2] Amazon Elastic Compute Cloud. `http://aws.amazon.com/ec2/`.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *SOSP*, 2003.

[4] R. Baumann. Soft errors in advanced computer systems. *IEEE Design & Test of Computers*, 22(3):258–266, 2005.

[5] P. Bernadat, D. Mannaru, et al. Susceptibility of commodity systems and software to memory soft errors. *IEEE Transactions on Computers*, 53(12):1568, 2004.

[6] T. Bressoud and F. Schneider. Hypervisor-based fault tolerance. *ACM Transactions on Computer Systems*, 14(1):107, 1996.

[7] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *USENIX NSDI*, 2005.

[8] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield. Remus: High availability via asynchronous virtual machine replication. In *The 5th USENIX NSDI*, 2008.

[9] S. Govindavajhala and A. Appel. Using memory errors to attack a virtual machine. In *IEEE SP 2003*.

[10] Intel Machine Check Architecture (MCA). `http://www.intel.com/products/processor/manuals/`.

[11] J. Kim, J. Smolens, B. Falsafi, and J. Hoe. PAI: A lightweight mechanism for single-node memory recovery in DSM servers. In *The 13th IEEE PRDC 2007*, pages 298–305. IEEE Computer Society.

[12] X. Li, M. Huang, K. Shen, and L. Chu. A realistic evaluation of memory hardware errors and software system susceptibility. In *USENIX ATC*, 2010.

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

[14] T. May and M. Woods. Alpha-particle-induced soft errors in dynamic memories. *IEEE Transactions on Electron Devices*, 26(1):2–9, 1979.

[15] T. O'Gorman, J. Ross, A. Taber, J. Ziegler, H. Muhlfeld, C. Montrose, H. Curtis, and J. Walsh. Field testing for cosmic ray soft errors in semiconductor memories. *IBM Journal of Research and Development*, 40(1):41–41, 1996.

[16] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and Communications Security*, pages 199–212. ACM, 2009.

[17] F. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):319, 1990.

[18] B. Schroeder and G. Gibson. A large-scale study of failures in high-performance computing systems. In *DSN 2006.*, pages 249–258, 2006.

[19] B. Schroeder and G. Gibson. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *The 5th USENIX FAST*, 2007.

[20] B. Schroeder, E. Pinheiro, and W. Weber. Dram errors in the wild: A large-scale field study. In *the Eleventh SIGMETRICS*. ACM, 2009.

[21] D. Tang, P. Carruthers, Z. Totari, M. Shapiro, S. Inc, and M. View. Assessment of the effect of memory page retirement on system RAS against hardware faults. In *DSN 2006*, pages 365–370, 2006.