Improving DRAM Fault Characterization Through Machine Learning

Elisabeth Baseman*, Nathan DeBardeleben*, Kurt Ferreira†, Scott Levy‡,
Steven Raasch§, Vilas Sridharan¶, Taniya Siddiqua§ and Qiang Guan*
*Ultrascale Systems Research Center, Los Alamos National Laboratory
{lissa, ndebard, qguan}@lanl.gov
†Center for Computing Research, Sandia National Laboratories
kbferre@sandia.gov
‡Department of Computer Science, University of New Mexico
slevy@cs.unm.edu
§AMD Research, Advanced Micro Devices, Inc.
{Steven.Raasch,Taniya.Siddiqua}@amd.com
¶RAS Architecture, Advanced Micro Devices, Inc.
Vilas.Sridharan@amd.com

***Abstract*—As high-performance computing systems continue to grow in scale and complexity, the study of faults and errors is critical to the design of future systems and mitigation schemes. Fault modes in system DRAM are a frequently-investigated key aspect of memory reliability. While current schemes require offline analysis for proper classification, current state-of-the-art mitigation techniques require accurate online prediction for optimal performance. In this work, we explore the predictive performance of an online machine learning-based approach in classifying DRAM fault modes from two leadership-class supercomputing facilities. Our results compare the predictive performance of this online approach with the current rule-based approach based on expert knowledge, finding a 12% predictive performance improvement. We also investigate the universality of our classifiers by evaluating predictive performance using training data from disparate computing systems to achieve a 7% improvement in predictive performance. Our work provides a critical analysis of this online learning technique and can benefit system designers to help inform best practices for dealing with reliability on future systems.**

I. Introduction

As high-performance computing systems continue to grow in scale and complexity, reliability has become a great concern. If current predictions hold, future exascale systems expected in the early 2020s will see a hundred-fold increase in the amount of main memory (DRAM) and cache memory (SRAM) compared to current levels [1]. This places a great deal of pressure on system designers and operators to understand the failure characteristics of current systems in order to maintain reliability comparable to current systems. Due to this importance, a number of recent studies have focused on analyzing failures on current extreme-scale high performance computing systems, both within the U.S. national laboratory complex [2] and commercial data centers [3].

While these studies have contributed a number of significant insights on the characteristics needed for a reliable extreme-scale system, they have also exposed the significant challenges involved in this type of detailed analysis. In particular, these studies demonstrate the uncertainty involved and care needed in failure characterization. A truly accurate classification of faults typically requires the complete failure log dataset and therefore current studies have focused on offline analysis techniques. This is in contrast to current state-of-the-art failure mitigation techniques [4] that require accurate online prediction to react to these faults. Online analysis has other significant advantages over the current offline techniques; they have the potential to predict and diagnose faults on new systems in which no offline failure log currently exists, and can adapt to changing failure modes over time as a system ages. In addition, determination of fault modes can enable the system to take action to avoid this fault in the future, for example retiring that page in system memory [5].

In this paper, we present an analysis on the potential of one such online characterization technique. More specifically, we explore the predictive performance of an online machine learning-based approach in classifying memory fault modes from two leadership-class supercomputing facilities. This work provides a critical analysis of this online learning technique and can benefit system designers to help inform best practices for dealing with reliability on future systems.

II. Supercomputing Systems

Our study comprises data from two leadership-class production systems: Hopper, a supercomputer located at LBNL in Oakland, California; and Cielo, a supercomputer at LANL in Los Alamos, New Mexico. These two systems are very similar. Hopper contains approximately 6,000 compute nodes. Each *node* contains two 12-core AMD Opteron™processors, each with twelve 32KB L1 data caches, twelve 512KB L2 caches, and one 12MB L3 cache. Cielo, on the other hand, contains approximately 8,500 compute nodes. Each node contains two 8-core AMD Opteron™processors, each with eight 32KB L1 data caches, eight 512KB L2 caches, and one 12MB L3 cache.

Both systems have eight 4GB DDR-3 registered DIMMs for a total of 32GB of DRAM per node.

In both Hopper and Cielo, each DDR-3 DIMM contains two *ranks* of 18 DRAM devices, each with four data (DQ) signals (known as an x4 DRAM device). In each rank, 16 of the DRAM devices are used to store data bits and two are used to store check bits. A *lane* is a group of DRAM devices on different ranks that shares data (DQ) signals. DRAMs in the same lane also share a strobe (DQS) signal, which is used as a source-synchronous clock signal for the data signals. A memory *channel* has 18 lanes, each with two ranks (i.e., one DIMM per channel). Each DRAM device contains eight internal *banks* that can be accessed in parallel. Logically, each bank is organized into *rows* and *columns*. Each row/column address pair identifies a 4-bit *word* in the DRAM device.

The primary difference between the two memory subsystems is that Cielo uses *chipkill-correct* ECC on its memory subsystem, while Hopper uses *chipkill-detect* ECC. Chipkill-detect ECC can detect but not correct any error in a single DRAM chip, while chipkill-correct can correct any error in a single DRAM chip.

III. Current Rule-Based System

Current methods for characterizing DRAM memory fault modes are offline, rule-based, and deterministic. This approach is useful for studying aggregate behavior, but not for online diagnosis and fault mitigation. The rule-based classification system runs on memory error data collected over the entire lifetime of the machine, and assumes that each DRAM will see at most one fault. Therefore, the rule-based system labels each DRAM that has produced errors with a fault mode, based on the entire stream of errors that DRAM has produced.

Note that because we cannot directly observe the fault patterns on the memory hardware to verify the fault modes, we must rely on this rule-based system run on data over all available time for our ground-truth labels in the learning process for our online system.

The DRAM fault definitions used in this rule-based system, defined by industry domain experts, are as follows:

- *Multi Rank*: Two DRAMs in the same lane with errors.
- *Single Bit*: DRAMs not in a Multi Rank Fault, errors all length one bit, and all same pin, row, column, and bank.
- *Single Word*: DRAMs without either of the above faults, and errors all same row, column, and bank.
- *Single Row*: DRAMS without any of the above faults, and errors all same row and bank.
- *Single Column* : DRAMS without any of the above faults, and errors all same column and bank.
- *Single Bank*: DRAMS without any of the above faults, and errors all same bank.
- *Multi Bank*: DRAMS without any of the above faults.

IV. Approach

Our goal is to classify memory fault modes as early in the stream of errors they produce as possible. Therefore, unlike the rule-based system, we account for the time variable.

We know that each memory fault produces a stream of one or more errors, and we assume that each DRAM experiences at most one fault. Then, we can use number of errors seen from the same DRAM as a proxy for time. In other words, we align each of the DRAM's error streams to start at time 0. We then subset the error data by truncating the error streams, first creating a dataset by taking all errors that occur first in their DRAM's error stream, then taking the first two errors of each DRAM's data stream, etc., and computing our features from these smaller datasets. Note that, as we progress in "time", the size of the dataset we consider decreases, as shown in Figure 1. In addition, the fault mode distributions differ across the various subsets, as shown in Figure 2.

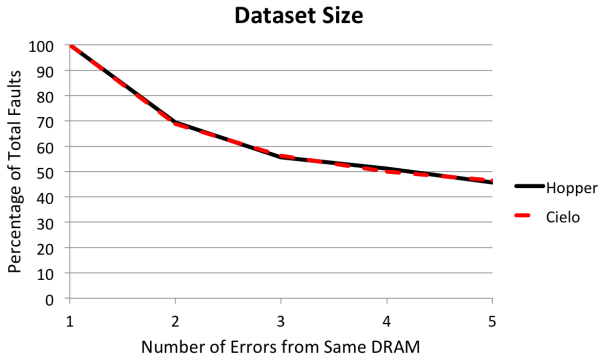


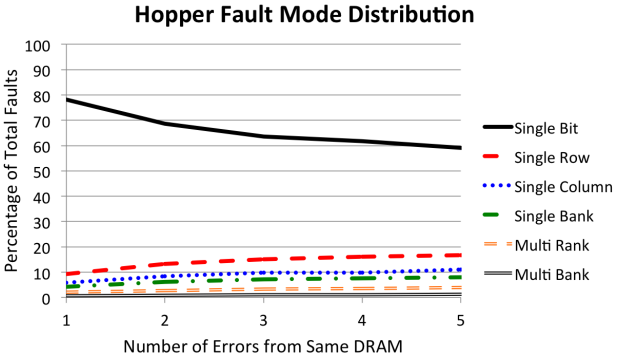
Fig. 1. Dataset subset size as a percentage of the total size of the dataset, as time progresses. Time is measured in number of errors produced by a DRAM.

Our work examines how early in these error streams we can identify the DRAM's fault mode with predictive performance greater than that of the rule-based system run on the same truncated error stream.

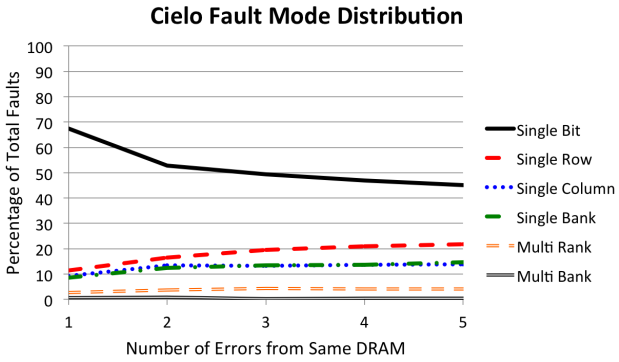
A fixed-time-interval memory scrubber reports raw feature information for each error it observes from a DRAM. The raw features collected for each error from the DRAM devices and translated by domain experts include a timestamp and location information (host, node, channel, lane, chip select, bank, row, column, and pin). For each error, we use the location information to engineer calculated features including total number of errors seen in the same bank up to the current timestamp, frequency of errors seen in the same bank since the first error in that bank, and time since the last error from the same bank, for each of the raw location features. This anonymizes away the specific location values, and focuses instead on relative error locations relevant to fault modes.

We apply four machine learning algorithms to the features calculated from error locations at each point in "time" (where time is measured in terms of number of errors produced by each DRAM). Our four learning algorithms are naïve Bayes [6], logistic regression [7], random forest [8], and gradient boosted random forest [9]. We then examine the predictive performance of each algorithm at each point in time, and compare the results to the predictive performance of the rule-based system run at each point in time.

We evaluate the predictive performance of our trained



(a) Distribution of fault modes in Hopper over time.



(b) Distribution of fault modes in Cielo over time.

Fig. 2. Distribution of fault modes in each supercomputing system, as determined by the rule-based system (ground truth), over time.

classification algorithms using one-versus-rest AUC (Area Under Receiver Operating Curve), and take the arithmetic mean across all fault modes [10]. AUC is a better evaluation metric than accuracy because it does not have to be considered relative to a baseline. In AUC terms, a (unitless) value of 0.5 is the same performance as random guessing, and a value of 1.0 is perfect predictive performance.

V. Experimental Setup

Using data extracted from two leadership-class supercomputing facilities, we can investigate predictive performance when trained and tested on data from the same supercomputing facility, as well as when trained on data from one facility and tested on the other. We calculate features as described above, and discard data for single word faults because they do not appear in the Hopper data, make up only 0.83% of faults in the Cielo data, and each single word fault in the Cielo data only causes a single error.

For each of four machine learning algorithms — naïve Bayes, logistic regression, random forest, and gradient boosted random forest — we run 200 train/test iterations on each time-partitioned dataset using a 60/40 train/test split. The random forest and gradient boosted random forest are both initialized with 200 estimators. The logistic regression uses an LBFGS solver with an L2 penalty. We report the arithmetic average of the resulting AUC values from each set of 200 iterations

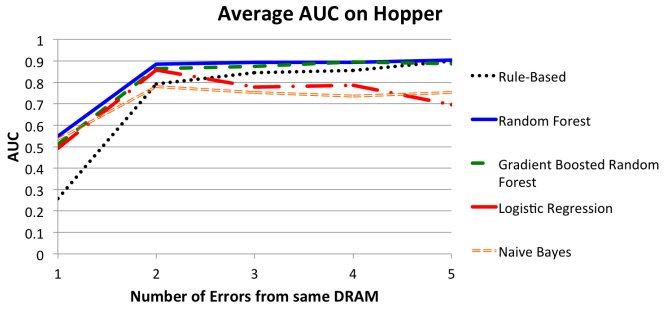


Fig. 3. AUC averaged across all fault modes on Hopper.

for each fault mode, and report the average of all fault mode AUCs for each algorithm at each point in time.

We repeat this experiment scheme for each of three train/test pairings: train/test on Hopper, train/test on Cielo, and train on Hopper test on Cielo. The results indicate not only whether our approach performs well, but also whether the approach may be more universal and apply across systems.

VI. Classification Results

Figure 3 shows AUC averaged over all fault modes for each learning algorithm trained and tested on Hopper, and Figure 4 shows the same analysis on Cielo. Standard error bars are not shown due to negligible size. We find that the random forest classifier achieves the highest predictive performance on both Hopper and Cielo, out-performing the rule-based system at all but one point in time for the first five timesteps. The greatest performance improvement, approximately 12% on Hopper and 9% on Cielo, over the current rule-based system is achieved after only two errors have been seen from each DRAM.

We present results for the best-performing learning algorithm in the cross-system experiments. Figure 5 shows average AUC results for random forests trained on Hopper data and tested on Cielo data, as well as AUC results for each individual fault mode in these cross-system experiments. While performance improvement varies by fault mode, on average we see a 7% increase in predictive performance of a random forest over the rule-based system, after the second error is seen from each DRAM. The highest performance improvement, of approximately 27%, is again seen at the second point in time, for single bank faults. The rule-based system never out-performs the random forest classifier at the second point in time. This implies that a machine learning system could be trained on an older supercomputing system and then immediately applied online to a newer facility.

VII. Conclusion

We find that trained classifiers can identify memory fault modes earlier and more accurately than the current deterministic rule-based system in both Hopper and Cielo, two leadership-class supercomputing facilities. The random forest classifier shows the greatest improvement in predictive performance, with a maximum increase of 12% when trained and tested on the same system, and a predictive performance

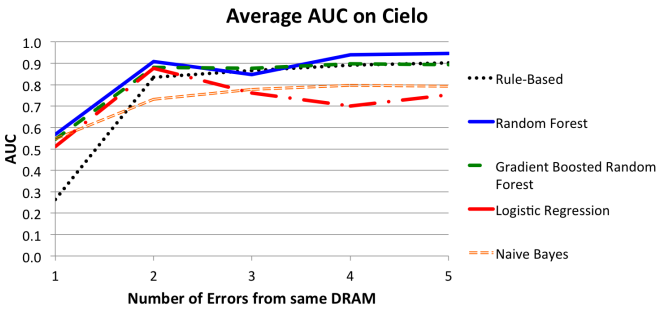
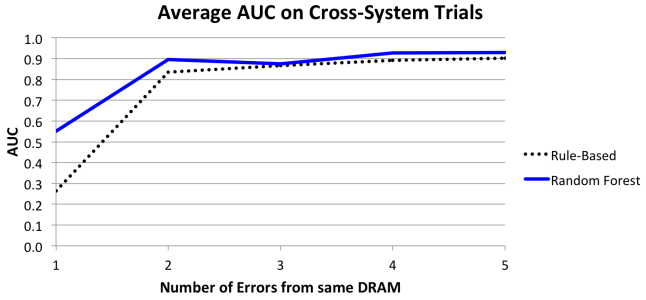
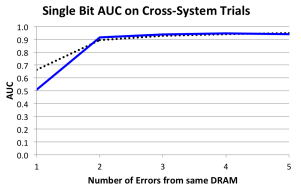


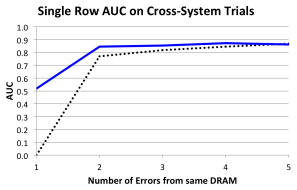
Fig. 4. AUC averaged across all fault modes on Cielo.



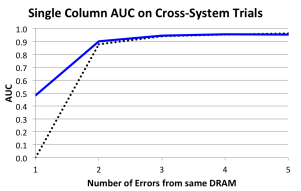
(a) AUC averaged over all fault modes for random forests trained on Hopper and tested on Cielo.



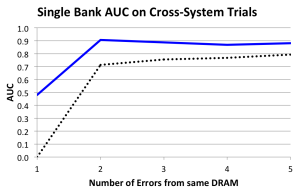
(b) Single bit fault average AUC.



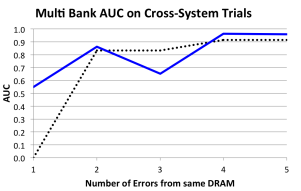
(c) Single row fault average AUC.



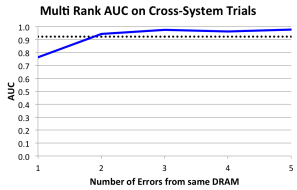
(d) Single column fault AUC.



(e) Single bank fault average AUC.



(f) Multi bank fault average AUC.



(g) Multi rank fault average AUC.

Fig. 5. Fault mode classification performance results for a random forest trained on Hopper and tested on Cielo.

increase of 7% when trained and tested on disparate systems. These promising results indicate that machine learning algorithms can exploit universal aspects of DRAM faults, applicable across computing systems.

Future work includes evaluating learning algorithms trained on supercomputer "burn-in" time, rather than on a disparate system, as well as investigating unsupervised learning methods to uncover anomalies and unexpected fault modes.

ACKNOWLEDGMENTS

The authors thank John Shalf from Lawrence Berkeley National Laboratory, who provided the Hopper field data. Please see "Memory Errors in Modern Systems: The Good, The Bad, and The Ugly" for more on the LANL/SNL/LBNL/AMD collaboration on supercomputer memory errors. This work was performed at the Ultrascale Systems Research Center (USRC) at Los Alamos National Laboratory, supported by the U.S. Department of Energy contract AC52-06NA25396. The publication has been assigned the LANL identifier LA-UR-16-22693 and the Sandia identifier SAND 2016-3270C. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. AMD, the AMD Arrow logo, AMD Opteron, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

REFERENCES

[1] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep*, vol. 15, 2008.

[2] V. Sridharan, J. Stearley, N. DeBardeleben, S. Blanchard, and S. Gurumurthi, "Feng shui of supercomputer memory positional effects in DRAM and SRAM faults," in *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for*. IEEE, 2013, pp. 1–11.

[3] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM errors in the wild: a large-scale field study," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1. ACM, 2009, pp. 193–204.

[4] M. Gamell, D. S. Katz, H. Kolla, J. Chen, S. Klasky, and M. Parashar, "Exploring automatic, online failure recovery for scientific applications at extreme scales," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2014, pp. 895–906.

[5] 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 *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*. IEEE, 2006, pp. 365–370.

[6] K. P. Murphy, "Naive Bayes classifiers," *University of British Columbia*, 2006.

[7] D. W. Hosmer Jr and S. Lemeshow, *Applied logistic regression*. John Wiley & Sons, 2004.

[8] T. K. Ho, "Random decision forests," in *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, vol. 1. IEEE, 1995, pp. 278–282.

[9] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning*, vol. 40, no. 2, pp. 139–157, 2000.

[10] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 17, no. 3, pp. 299–310, 2005.