Understanding the Spatial Characteristics of DRAM Errors in HPC Clusters[*]

Ayush Patwari
Purdue University
patwaria@purdue.edu

Ignacio Laguna
Lawrence Livermore National Laboratory
ilaguna@llnl.gov

Martin Schulz
Lawrence Livermore National Laboratory
schulz6@llnl.gov

Saurabh Bagchi
Purdue University
sbagchi@purdue.edu

ABSTRACT

Understanding DRAM errors in high-performance computing (HPC) clusters is paramount to address future HPC resilience challenges. While there have been studies on this topic, previous work has focused on on-node and single-rack characteristics of errors; conversely, few studies have presented insights into the spatial behavior of DRAM errors across an entire cluster. Understanding spatial peculiarities of DRAM errors through an entire cluster is crucial for cluster temperature management, job allocation, and failure prediction. In this paper, we study the spatial nature of DRAM errors on data gathered in a large production HPC cluster. Our analysis shows that nodes with high degree of errors are grouped in spatial regions for time periods, suggesting that these "susceptible" regions are collectively more vulnerable to errors than other regions. We then use our observations to build a predictor, which identifies such regions given prior neighboring regions patterns.

CCS CONCEPTS

• **Computer systems organization → Reliability**; • **Hardware → Transient errors and upsets**;

ACM Reference format:
Ayush Patwari, Ignacio Laguna, Martin Schulz, and Saurabh Bagchi. 2017. Understanding the Spatial Characteristics of DRAM Errors in HPC Clusters. In *Proceedings of FTXS'17, Washington , DC, USA, June 26, 2017,* 6 pages.
https://doi.org/http://dx.doi.org/10.1145/3086157.3086164

1 INTRODUCTION

Dynamic Random Access Memory (DRAM) errors are a common source of failures in High-Performance Computing (HPC) clusters [5, 7–9]. With exascale machines estimated to have up to hundreds of petabytes of main memory, typically implemented as DRAM, this problem will only continue to grow. We therefore need a thorough understanding of DRAM errors in HPC systems to address the resilience challenges of future exascale systems.

While previous studies on DRAM errors in HPC systems exist, most of this work has been focused on on-node (e.g., chip-level) [7, 9] and single-rack error characteristics (e.g., top/bottom of the rack) [10]. However, no previous work has provided insights into DRAM error characteristics with respect to the physical layout of affected nodes and DRAM chips. Understanding such *spatial* characteristics of DRAM errors is of great value to HPC centers for room temperature management, job scheduling and allocation policies, failure prediction, and overall system resilience.

In this paper, we present a study of the spatial and temporal behavior of DRAM errors in a production cluster at the Lawrence Livermore National Laboratory (LLNL). The goal of this study is to provide insights into the spatial correlation of errors in DRAM modules and nodes across the entire cluster and across individual racks. In particular, we employ statistical and data analytics models that help us answer the following research questions: (a) given a node *n* that experienced a large number of DRAM errors in a given period of time *t*, what is the probability that another *neighbor* node *m*, at distance *d* from *n*, will also experience a large number of errors? and (b) can we identify spatial groups of nodes that experience a high degree of errors versus groups of nodes that do not experience such behavior? Our data comprises records of *correctable* DRAM errors of a 1296-node cluster at LLNL (Cab), which was gathered for period of 14 months (May/2013–July/2014).

In summary, our main contributions are:

- We develop a spatial analysis framework to visualize the errors as seen in the physical layout of the cluster. We show that several nodes, which report errors in the same time-frame, or *epoch*, are spatially grouped together. This phenomenon can be seen in several epochs.
- We introduced metrics for the *number of neighboring erroneous nodes* and the *minimum distance from a neighbor erroneous node.* A statistical analysis of these metrics suggests that error grouping is not due to a random occurrence, but rather due to external factors. This also reveals interesting insights into the probability of a node having errors given prior error patterns in neighboring regions.
- We design a classification model to predict whether a node will experience errors at any given point given prior history and build a model based on the above observations to classify data samples as to whether they would report a

[*] This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DEAC52-07NA27344 (LLNL-CONF-727473), and supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, program manager Lucy Nowell.

FTXS'17, June 26, 2017, Washington , DC, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5001-3/17/06…$15.00
https://doi.org/http://dx.doi.org/10.1145/3086157.3086164

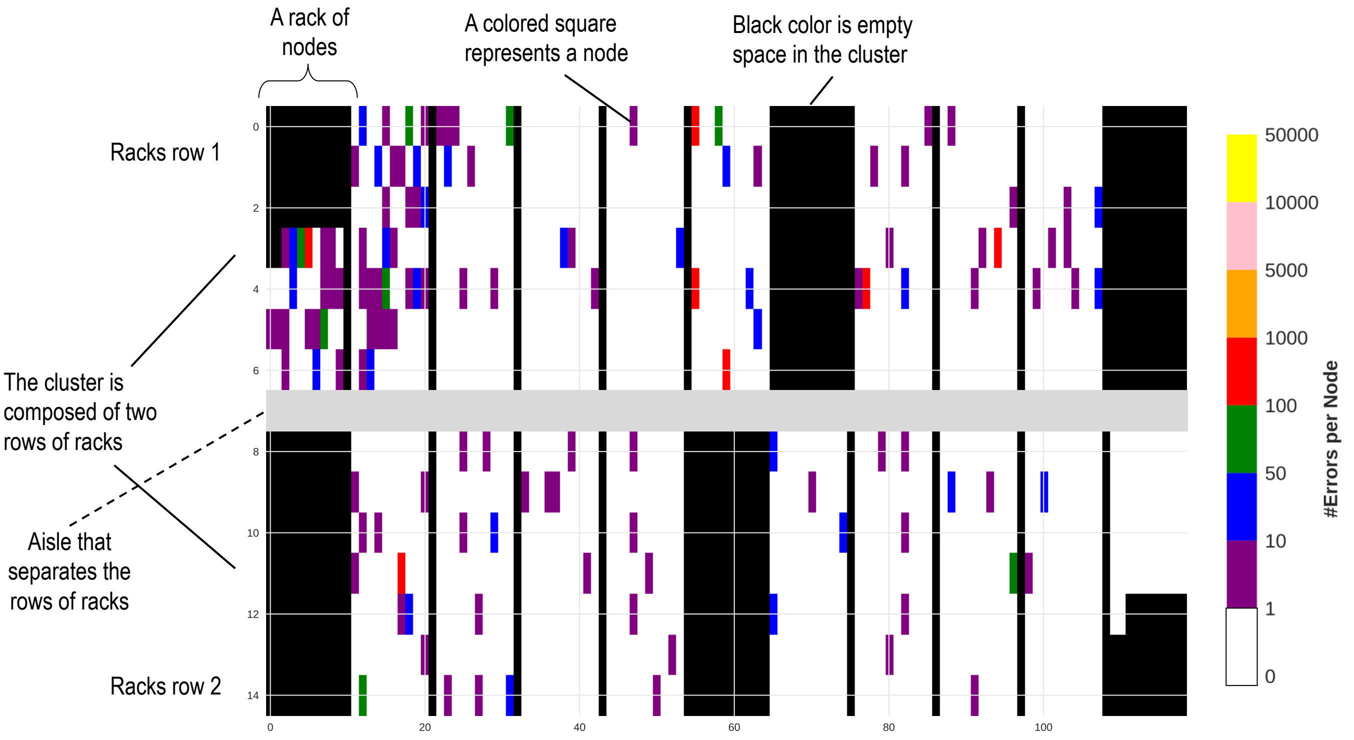

Figure 1: Snapshot of memory errors recorded between 5/20/2013 and 6/19/2013 on the Cab cluster displayed on the physical layout of the system. Nodes without errors are shown in white, while nodes with at least one error in the current epoch are colored. The black regions are either non-compute nodes or switches for the IB interconnect. The central horizontal gray block represents a hot aisle separating the two rows of racks.

new error on a node or not. This model is able to achieve a high F1-score (a metric that combines recall and precision) on a highly-unbalanced dataset (only 650 out of 251k samples report a new error).

2 TERMINOLOGY

We first introduce some definitions and notations used in the paper.

Correctable Error (CE). Errors on a DRAM DIMM (Dual In-line Memory Module) in a node, which are caused by transient, hard, or intermittent faults, but which can be corrected transparently using techniques like ECC.

Uncorrectable Error (UE). Errors on a DRAM DIMM in a node, which cannot be corrected by the memory system and typically lead to application aborts or node crashes.

Epoch. A parameter of our study that defines a period of time in which errors are analyzed. For all our analysis in this paper, an epoch is taken to be one month.

Erroneous Node. A node on which at least one correctable error was observed during the epoch being considered. Note that if a node had errors in a previous epoch, it will not be considered an erroneous node in the current epoch if it does not have reported *new* errors. Also note that we do not make a distinction between two nodes, one with a single error and one with many errors in an epoch—both would be considered erroneous nodes. The reason for this is that DRAM errors have the characteristic that they are latent, i.e., they are not detected until the affected cell in the DIMM is accessed. Because of this, many accesses of an affected cell may generate a large number of errors in a short period of time. Since we do not have memory access data for the nodes we cannot distinguish an erroneous node with few errors from one with many errors.

Number of errors. The total number of correctable errors observed on a node for the given epoch.

Timestamp. Record of the time at which a sample was collected.

Time since last reset. Uncorrectable errors are normally followed by a kernel panic and the node is rebooted. When this occurs, the error counters for the node are reset and error data between the beginning of the epoch and the hard reset is lost. This metric measures the time elapsed since the last time a reset occurred.

3 EXPERIMENTAL SETUP

We describe the experimental setup for this work and introduce the notations used in the following sections.

3.1 Data Gathering

As mentioned earlier, we use DRAM error data collected from the Cab cluster at LLNL over a period of 14 months from May 2013 to July 2014. Cab has a total of 1296 nodes connected with an InfiniBand (IB) network and each node has two Intel 8-Core Xeon E5-2670 processors. The physical layout can be seen in the Fig. 1.

The cluster uses the TLCC operating system, which is a derivate of RHEL, and has x4 DDR3 1600MHz memory running in S4ECD4ED.

EDAC (Error Detection and Correction). We used EDAC [1] to collect error data on the cluster. EDAC consists of a set of Linux kernel modules for handling hardware-related errors. It allows users to read ECC memory error information and, additionally, it also detects and reports PCI bus parity errors. EDAC errors are reported as counters stored in the Linux `/proc` virtual filesystem. Although EDAC has support for CEs and UEs, in this study we focus on CEs, as the number of UEs in the cluster are too small to make statistical inferences that are significant at a high confidence level.

EDAC offers the error data separated by memory controller (`mc`) system. Each `mc` controls a set of DIMM memory modules, which are laid out in a chip-select row (`csrowX`) and channel table (`chX`). Memory controllers have several csrows; the actual number is dependent on the physical configuration. The information is exported for each of `mcX` and `csrowX` directories in the form of separate files, which contain information on different attributes. The ones relevant to our analysis are `ue_count` (total uncorrectable errors on a csrow), `ce_count` (total correctable errors on a csrow), `mem_type` (DRAM type), `chX_ce_count` (total correctable errors on a DIMM on channel X).

User-level Jobs. To gather error data for all the above attributes, we submit user-level 256-node jobs periodically in the cluster (about 2 jobs per day). Note that since these jobs run at user level, they may have varying priorities and may stay in the job queues for some time before they get an allocation and execute. This affects our sampling rate making it not fully regular. However, to minimize this limitation, the duration of jobs is set to be very short, increasing the chance that jobs of being picked quickly by the job scheduler in comparison to long jobs.

It is worthwhile noting that the job scheduler does not give preferences to nodes (or groups of nodes) in the cluster for regular user-level jobs. As a result, the overall load of a node (expected value over a long period of time) is considered to be uniform.

Data Samples. In our analysis, each data sample has the following attributes that are relevant to our analysis: `nodename`, `timestamp`, `total_correctable_errors`, and `time_since_last_reset`. We gather approximately 251,000 data points for the cluster over the data gathering period.

Physical Layout. We use the physical layout with the exact dimensions of the cluster in our analysis. The inter-node distance between every two nodes of the cluster was measured and this distance varied between 0.2 feet to > 10 feet. We used these distances in our experiments.

3.2 Visualization and Exploratory Analysis

We develop a data analysis and visualization toolkit to understand the state of the cluster over time, with respect to erroneous nodes, with different granularity of epochs. It shows the cluster according to the layout seen by looking at the cluster from the rear. A sample snapshot of the collected error data between one epoch of 5/20/2013 and 6/19/2013 can be seen in Figure 1. The tool is developed in pyplot[1] and it can be configured to accommodate different configurations for cluster layouts and epoch granularity. After experimenting with different epoch time scales, we observed the following significant behaviors.

Spatial Node Grouping. By studying the erroneous nodes in our toolkit, we found that several of such nodes group together in many epochs. This phenomenon is shown visually in Figure 2: in epochs 0, 1 and 10 errors are clearly grouped (marked in light green). Although we have not found yet the reason for this phenomenon, we speculate that this could be due to external factors, such as irregularities in the room temperature in which the cluster is located or irregular workload distributions. Note that the DIMMs in this cluster are overall from the same vendor, so irregularities coming from vendor differences are unlikely.

Season-of-the-year Correlations. Further, we found a correlation of the degree of errors and the season of the year. While in some epochs, errors appear to be randomly distributed across the physical space, we observe that during the months of May–Aug in 2013 and Feb–Mar of 2014 the highest number of errors were observed. In contrast, the winter months of 9/13–1/14 relatively fewer errors were observed. The rise in number of erroneous nodes in February could be attributed to the rise in temperatures in the Livermore area, however, in absence of actual temperature data of the cluster room, this assumption could not be confirmed (see Table 1). For future work, we plan to perform time-based correlation between errors and other factors, such as temperature to possibly explain this behavior—although previous work has found no correlation between uncorrectable errors and temperature [4], these studies have not considered spatial relationships.

Random versus Non-random Grouping. We observe that with more erroneous nodes in an epoch, the grouping seems to increase, i.e., we tend to see more groups of erroneous nodes. This is intuitive, since, as more samples are added to the analysis, more of these samples will tend to be close to each other. In the next section, we use rigorous statistical analysis to reject the hypothesis that spatial grouping occurs by chance (or randomly).

4 EXPERIMENTS

We conduct a set of experiments to show that the `erroneous_nodes` do herd (or group) together to some extent by studying different metrics and statistically reject the hypothesis that these are due to random occurrence. Finally, we build a classification model to predict if a node would have a new error observed at the current time instance given previous observed history. We elaborate on these experiments in this section.

4.1 Neighbor Analysis

In this experiment, we are interested in understanding if `erroneous_nodes` do herd together in groups in an epoch. This analysis can help us to understand the spatial correlation in the error data for each `epoch`, which can be very useful in identifying regions which are error prone during a certain period.

Because herding could occur due to more and more erroneous nodes being sampled, our goal is to accept or reject this hypothesis. More formally, as a *null hypothesis*, we assume that the observed distribution of errors is due to random sampling (which means that there are no actual spatial groups in the data); the alternative hypothesis is assumed to be the opposite. Then, we perform statistical

[1] http://matplotlib.org/api/pyplot/api.html

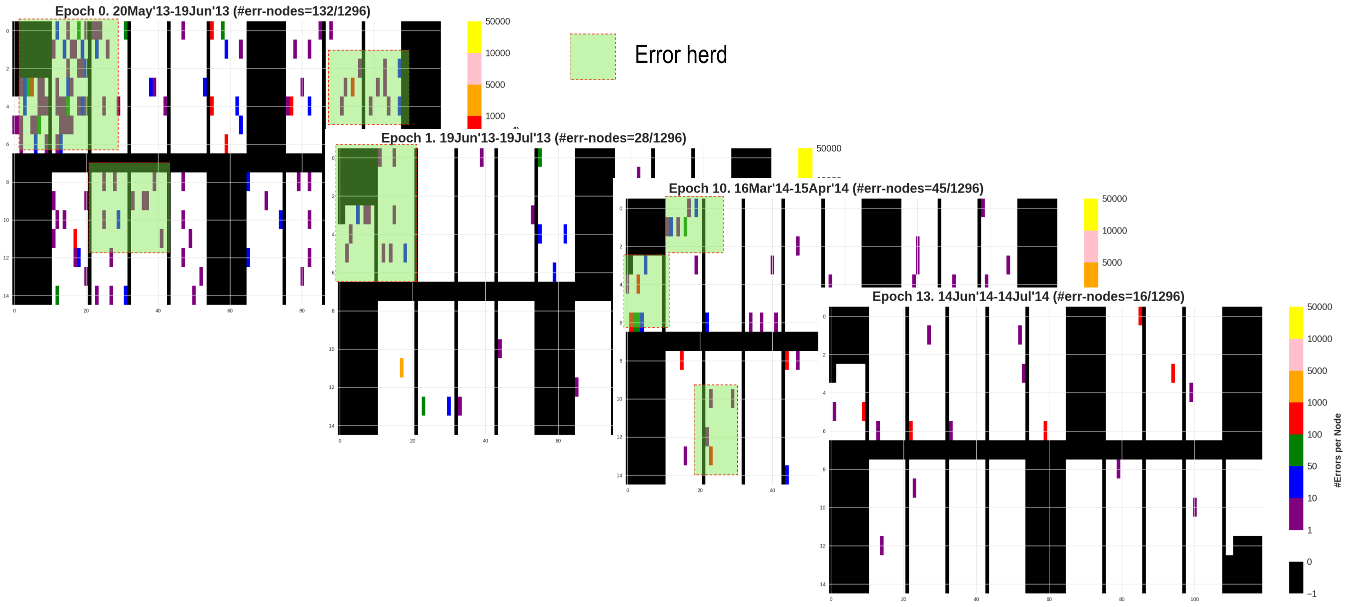

Figure 2: Snapshot of several epochs. Although we can observe herds of erroneous nodes appearing in several epochs, this could be due a random phenomenon unless rejected by a statistical test

Distribution of **erroneous_node**s across epochs

Epoch	20May	19Jun	19Jul	18Aug	17Sep	17Oct	16Nov	16Dec	15Jan	14Feb	16Mar	15Apr	15May
EN	132	28	32	23	18	22	23	12	12	52	45	20	23
AMT (C°)	26	27	31	31	32	23	18	13	13	16	18	18	22

Table 1: Epochs are represented here only by start date. Epoch size is one month. EN is number of erroneous nodes in an epoch. AMT is the Average Maximum Temperature in the particular month observed in Livermore, CA [2]

testing to verify that the null hypothesis can be rejected in favor of the alternative hypothesis.

For each epoch, we do the following:

(1) Define a neighborhood of $M \times N$ sq. units where 1 unit = 0.2 feet, which is the minimum distance between two nodes as calculated from the physical layout. Here, we use $M = 4, N = 2$.
(2) Determine the number of erroneous_nodes in the neighborhood for each node.
(3) Generate a frequency distribution of num_of_nodes versus erroneous_neighbors.
(4) Generate a random sampling of erroneous_nodes for the epoch using the same error rate of the epoch and generate the expected frequency distribution (as above).
(5) Perform a chi-squared test to test the null hypothesis, using a significance level of 5%. Chi-squared tests are used to determine whether there is a significant difference between the expected frequencies and the observed frequencies in one or more categories—here, the categories are observed data and randomly-generated data of erroneous nodes.

A histogram plot for some of the epochs are show in Figure 3. In plot (a), we can see that there is significant difference in the histograms for the observed and random distributions with the same error rate. Specifically, we see that there are several nodes which have 5, 6, or 7 erroneous_neighbors whereas these numbers are significantly smaller for a random sampling. A similar yet less profound difference is seen in plot (d). However, we observe that in other plots this difference cannot be easily seen.

We perform chi-squared tests only for epochs that have 25 or more erroneous_nodes to make sure we have enough data samples for each test. In 5 out of 7 epochs, the p-value was less than 0.05, which allow us to reject the null hypothesis in these 5 cases. This shows that in most of the cases, the spatial grouping of erroneous_nodes is not due to random occurrence and rather it is an effect of other physical or operating factors.

4.2 Nearest Distance Error Node

In this experiment, we want to analyze how close erroneous_nodes are with respect to each other when they occur in the same epoch. For each epoch, we calculate the distance of the nearest erroneous_neighbor for each erroneous_node. We then plot a cumulative distribution function as shown in Figure 4. We show the epochs with at least 25 erroneous_nodes. In all such epochs, at least 35% of erroneous_nodes have at least one erroneous_neighbor within 0.8 feet. Epoch 0 (May/13–Jun/13) has greater than 70% of such nodes. The corresponding distributions for random sampling (dotted lines) show much smaller percentages for

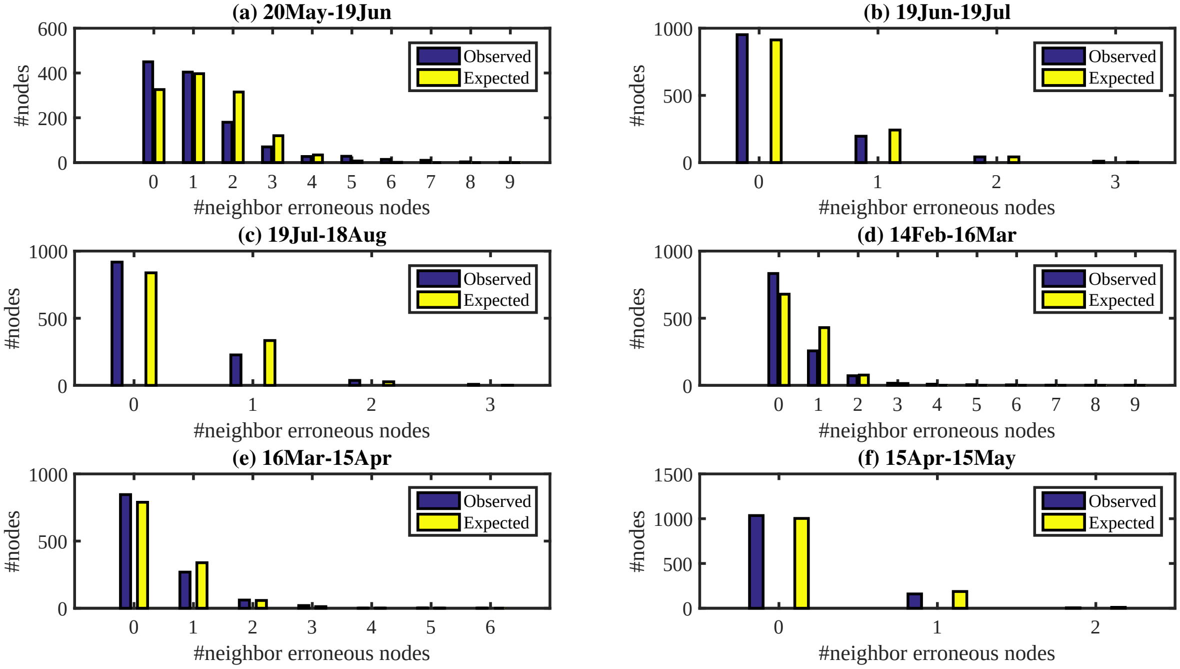

Figure 3: Distribution of number of neighboring erroneous nodes for the nodes in Cab for selected epochs. The expected distribution is generated after generating erroneous nodes randomly across the physical layout (total nodes in Cab are 1296).

distances smaller than 1 feet. This is also intuitive as a random sampling would create samples that are relatively close but not very close to each other.

Like the last experiment, chi-squared tests were performed for each of the above epochs and the null hypothesis was rejected with a p-value of < 0.05 for all epochs. This shows that the distance between `erroneous_nodes` in an epoch is not very high and it is not by chance in majority of the cases.

4.3 Classifying Data Samples Experiment

Being able to predict future erroneous nodes would be of great value for job allocation, job migration, and overall system management[2]. In this experiment, we are interested in building a classification model that can predict if a node will be erroneous at a given timestamp t, given the history of the state of its neighborhood, for time $[t-1, t-2, \ldots, t-n]$.

4.3.1 Dataset. We consider data samples that have at least one new error reported for the given node from the last count of errors. For example, if a node had 35 errors till time t, and at $t+1$ a sample reports the node has 35 errors (the same number), this sample is labeled as not erroneous. In contrast, if the node has 55, the sample is labeled as erroneous since 20 new errors were reported.

Not surprisingly, only 640 of such data samples are marked as reporting new errors. Note that the features we use in the model (described below) depend only on the history and not on the current sample so we could effectively predict the state of a node at any time given its recorded history.

4.3.2 Features. From our observations in previous sections, we define six features which would help us build a robust prediction model: (1) `Prev_error_count` (number of samples in which a new error was reported for this node in the past history); (2) `Neighbor_error_count` (number of neighbors that had at least one error in the past history); (3) `Total_error_count` (total number of times a new error was reported for this node since start of data collection); (4) `Max_error_value` (maximum number of errors accumulated for this node in the past history); (5) `Prev_error_value` (number of new errors reported in the last sample); (6) `Week_of_year` (the week number of the year corresponding to the current timestamp).

4.3.3 Results. Because our data is highly imbalanced, we trained our model using ensemble learning techniques (Random Forest and Adaboost algorithm). We selected the top k features using ANOVA and performed a 3-fold cross-validation.

To measure the quality of the predictor, we used the F1-Score, a metric that weights together classification precision and recall; F1-Score reaches its best value at 1 and worst at 0. Our model achieved a F1-Score of 0.63 for the minority class (samples which report error; see Figure 5). For the majority class, the F1-Score is always very close to one, hence discussion omitted for brevity. The most important feature was `Prev_error_count` which is intuitive since nodes which had errors in the past would be more likely have errors in the current time [5, 7]. These results are encouraging and show that the spatial location of erroneous nodes can be predicted with a moderate degree of recall and precision, which may help in building proactive resilience techniques.

[2] Prediction of UEs has more impact than prediction of CEs; however, the techniques presented here using CEs can also be applied to UEs.

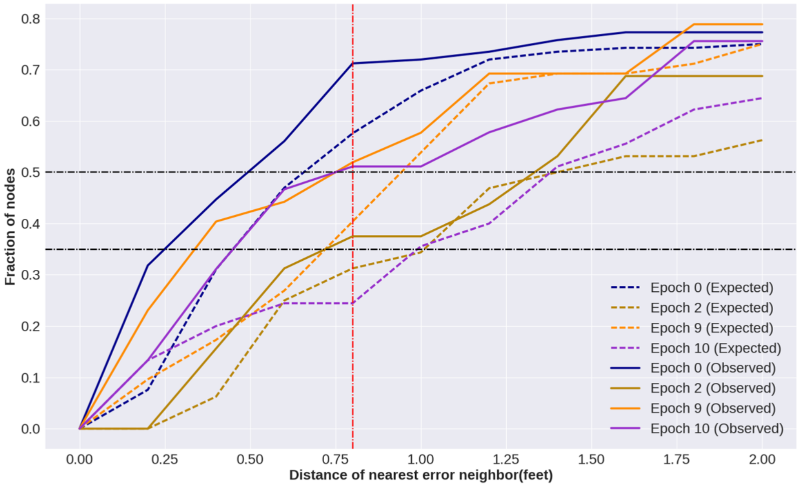

Figure 4: CDF plots for fraction of nodes having a `erroneous_neighbor` within a given distance. The bold line represent the observed data for particular epochs whereas the dotted lines represent the randomly sampled data for the same epoch.

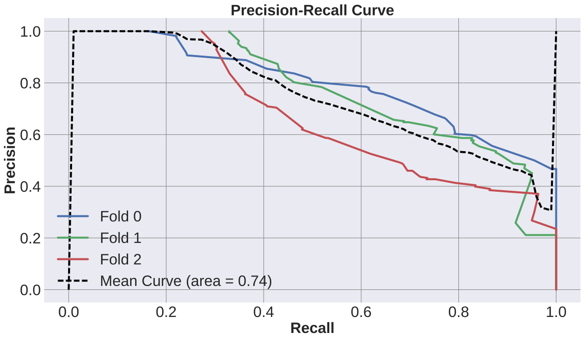

Figure 5: Precision-Recall curve for the 3 folds of cross-validation for the classification task. The values shown are only for the error class (minority). For the majority class precision and recall are always very close to 1.

5 RELATED WORK

DRAM errors at large scale in HPC and data center clusters have been studied previously from various perspectives. Schroeder et al. [7] present a study of DRAM errors from samples of Google's servers. Detailed studies of the characteristics of DRAM errors in HPC clusters have been presented in [3, 6, 9, 10]. The study presented in [8] shows that counting errors instead of faults can have a high impact when measuring system reliability. Characteristics of DRAM locations with respect to bottom, middle and top positions in a rack are shown in [10]. Spatial correlations of other system failures are presented in [5].

6 SUMMARY AND FUTURE WORK

In this experience paper, we show insights into the spatial correlation of DRAM errors in HPC clusters at a fine granularity and show that this phenomenon is not due to random occurrence. In particular, we show that nodes that experience a high degree of correctable errors can be spatially correlated in certain racks of the cluster. This can help to provide information of possible factors that could cause this spatial grouping, such as non-uniform temperatures inside the cluster room, or non-uniform workloads. We also show that, using machine learning, erroneous nodes can be predicted with moderate recall and precision, which can help in proactive resilience techniques. As future work, we will correlate spatial groups and epochs with temperature data, workload data, and to perform the same spatial analysis on other clusters.

REFERENCES

[1] EDAC. https://www.kernel.org/doc/Documentation/edac.txt.

[2] Weather Underground. https://www.wunderground.com, 2017. [Online; accessed 15-March-2017].

[3] N. DeBardeleben, S. Blanchard, V. Sridharan, S. Gurumurthi, J. Stearley, K. Ferreira, and J. Shalf. Extra Bits on SRAM and DRAM Errors–More Data from the Field. In *IEEE Workshop on Silicon Errors in Logic-System Effects (SELSE)*, 2014.

[4] N. El-Sayed, I. A. Stefanovici, G. Amvrosiadis, A. A. Hwang, and B. Schroeder. Temperature Management in Data Centers: Why Some (Might) Like It Hot. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '12, pages 163–174, 2012.

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

[6] A. A. Hwang, I. A. Stefanovici, and B. Schroeder. Cosmic rays don't strike twice: understanding the nature of dram errors and the implications for system design. In *ACM SIGPLAN Notices*, volume 47, pages 111–122. ACM, 2012.

[7] B. Schroeder, E. Pinheiro, and W.-D. Weber. DRAM errors in the wild: a large-scale field study. In *ACM SIGMETRICS Performance Evaluation Review*, volume 37, pages 193–204. ACM, 2009.

[8] V. Sridharan, N. DeBardeleben, S. Blanchard, K. B. Ferreira, J. Stearley, J. Shalf, and S. Gurumurthi. Memory Errors in Modern Systems: The Good, The Bad, and The Ugly. In *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, pages 297–310, New York, NY, USA, 2015. ACM.

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

[10] 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*, pages 1–11. IEEE, 2013.