Predicting DRAM-Caused Node Unavailability in Hyper-Scale Clouds

Pengcheng Zhang[†], Yunong Wang[‡], Xuhua Ma[‡], Yaoheng Xu[†], Bin Yao[†§*] , Xudong Zheng[‡], Linquan Jiang[‡]
[†]Shanghai Jiao Tong University, Shanghai, China, {zhangpc@, sjtu_xyh@, yaobin@cs.}sjtu.edu.cn
[‡]Alibaba Group, Hangzhou, China, {yunong.wyn, xuhua.mxh, xudong.zxd, linquan.jlq}@alibaba-inc.com
[§]Hangzhou Institute of Advance Technology, Hangzhou, China

***Abstract*—DRAM faults are major hardware sources of cloud node unavailability. To enable early preventive actions and mitigate DRAM fault impacts, prior studies focus on predicting DRAM uncorrectable errors (UEs) that typically cause immediate node unavailability. In our cloud with over half a million nodes, we firstly observe that the correctable error storm (numerous CEs occur in a short period) dominates 56% DRAM-caused node unavailability (DCNU). Therefore, we propose to predict DCNU that takes account into both UEs and CE storms. Observing that DCNUs have strong relevance to temporal statistics and spatial patterns of CEs, we design novel spatio-temporal features to train the prediction model. Considering the model's real effects cannot be evaluated by traditional metrics like F1-score, we propose a new metric NURR to quantify the node unavailability reduction and tune model hyperparameters with NURR. Our approach achieves over 40% better NURR than existing methods on historical data and runs stably in the production environment.**

I. Introduction

Today's software applications are increasingly built on cloud services and deployed on cloud platforms. IDG [22] reports that 81% of organizations leverage the cloud in their IT infrastructure. Cloud platforms such as Amazon Web Services (AWS) and Microsoft Azure are providing services to millions of users worldwide on a 24/7 basis. Therefore, high availability becomes an essential requirement for cloud computing systems. Many cloud service providers have promised exact availability levels in their Service Level Agreement. For example, for our public cloud service, Alibaba Cloud Elastic Compute Service (ECS), we promise that the availability of one multi-zone service should be no less than 99.995% [7]. It means the service is only allowed to be unavailable at most 130 seconds per month.

Plenty of efforts have been made to improve service availability of cloud platforms [17], [25], [28], [29], [42]. However, in reality, many hardware issues still exist in cloud computing systems, e.g., DRAM failures and CPU failures. These failures typically cause computing nodes to be unavailable, significantly impacting the overall service availability. In our cloud platform for ECS, a hyper-scale system composed of over half a million computing nodes, we define an unavailable node as an online node that fails to respond over a minute. DRAM faults are observed as the leading hardware sources of node unavailability.

Error Correction Code (ECC) has been developed to detect and correct DRAM errors and mitigate the crucial impacts of DRAM faults. For example, the single-bit error correction and double-bit error detection (SEC-DED) ECC can correct a single-bit error. The more complex Chipkill ECC [11] is able to correct any number of error bits in a single chip. Errors corrected by ECC are referred to as correctable errors (CEs). Usually, a few CEs have negligible impacts on the system. However, the error pattern may go beyond the capability of ECC. Such an uncorrectable error (UE) typically leads to a subsequent OS crash and node unavailability. As a result, dozens of studies are dedicated to predict DRAM UEs [2], [16], [18]. They commonly train a predictor from historical data and then perform UE predictions for further mitigation actions such as replacement of problematic Dual Inline Memory Modules (DIMMs) [16], job migration [2], etc.

In our hyper-scale cloud, besides node unavailability caused by UEs, we find that a live node may also fail to respond when a burst of CEs occurs in a short period (denoted as CE storm) and cause a kind of denial of service attack [10], [30]. Critically, we firstly observe that CE storm is the leading DRAM issue which dominates 56% DRAM-caused node unavailability (DCNU). Furthermore, we notice that the communications between a small fraction of DIMMs and the hardware systems are sometimes lost, under which condition the node will also be unavailable. Therefore, we propose to predict DCNU, which not only considers UEs, but also takes into account CE storms and DIMM communication losses.

To correlate DRAM errors and DCNUs, we conduct an empirical analysis on the logs collected from more than half a million nodes in ECS system over one year. We observe that the temporal statistics of CEs are highly relevant to DCNUs: CEs are frequently observed in the several hours before the nodes become unavailable. Furthermore, we find that the CE spatial distributions are also strong indicators of DCNUs. At the micro-level, a DRAM bank is structured as a two-dimensional cell array indexed by rows and columns. A faulty row/column is defined as a row/column that experienced CEs on at least two different spatial locations. Our analysis shows that faulty rows/columns are found on over 70% nodes before they become unavailable.

*Corresponding author
This work was supported by the NSFC (61922054, 61872235, 61832017, 61729202, 61832013), the National Key Research and Development Program of China (2020YFB1710200), the Science and Technology Commission of Shanghai Municipality (STCSM) AI under Project 19511120300 and Hangzhou Qianjiang Distinguished Expert Program.

Inspired by our statistical observations, we then design novel spatio-temporal features for DCNU prediction. Firstly, we design several DRAM error patterns (e.g., faulty row/column) to capture the spatial CE distributions and aggregate them on different DRAM components. Furthermore, we employ temporal aggregations to capture the important statistics such as sum and increment of the spatial features and other kernel events in both short-term and long-term time windows (e.g., the latest 3 hours and 4 weeks). The novel spatio-temporal features are combined with static configuration information such as CPU generation and DRAM capacity to train a machine-learning (ML) model for DCNU prediction. Considering the model's real effects cannot be evaluated by traditional metrics like F1-score, we propose the new metric NURR (Node Unavailability Reduction Rate) to quantify the node unavailability reduction and tune model hyperparameters with NURR.

We then develop XBrainM (M means Memory), an Artificial Intelligence for IT Operations (AIOps) system to predict and mitigate DCNUs in hyper-scale clouds. It ensembles both the ML model and traditional rule-based prediction, e.g., the number of recent CEs exceed a certain threshold, to predict DCNUs. When either the rules or the ML model reports a DCNU alert, the system then migrates virtual machines (VMs) hosted on the predicted faulty nodes to other healthy nodes. After that, the domain experts quickly locate the underlying DRAM issues, and finally, the problematic DIMMs get repaired or replaced.

The offline experiments on the one-year logs demonstrate the effectiveness of our approach: the predictor achieves 40% absolutely better NURR than existing methods, with more than 77% recall and 65% precision. XBrainM has also been integrated in the production environment of ECS system for over a year. Online results show that XBrainM significantly reduces node unavailable rate and node unavailable time (caused by DRAM faults) by 57% and 69%, respectively. Our main contributions are summarized as follows:

- We first propose to predict DCNU relevant to availability in clouds and successfully built XBrainM to predict and mitigate DCNU in hyper-scale clouds.
- We conduct statistic analysis on the field data and observe that: besides UEs, the CE storm is another major cause of node unavailability; faulty rows/columns are strong indicators of DCNUs.
- We use plenty of raw features and employ novel spatio-temporal feature engineering to achieve high prediction performance.
- Our predictor achieves more than 40% better NURR than existing methods on historical data. We also integrated XBrainM in the production environment of ECS system, effectively reducing 69% node unavailable time.

The rest of this paper is organized as follows: Section II introduces the background. Section III presents an exploratory study to correlate DRAM errors and DCNUs. Section IV introduces key components of our solution. Section V and VI detail our feature engineering methods and evaluation strategies, respectively. Section VII provides the offline and online results. The related work and conclusion are presented in Section VIII and Section IX.

II. Background and Problem Formulation

A. Terminology

DRAM fault and error. A fault refers to the underlying cause of an error, while an error is the symptom of a fault. Note that a fault may not manifest in errors if the faulty location is not accessed.

DRAM errors are reported when the bits returned differ from what has been written. Depending on whether ECC can correct them, DRAM errors are further characterized as correctable errors (CEs) and uncorrectable errors (UEs). A fault may cause many errors (CEs/UEs) if the faulty addresses are accessed frequently.

Hard and soft fault/error. Faults are categorized into soft faults and hard faults. Soft faults occur randomly and may get recovered, such as strikes of high-energy particles or cosmic rays. Hard faults often refer to hardware damage. Errors caused by soft/hard faults are referred to as soft/hard errors. In practice, we follow a common rule of thumb and define hard errors as the repeated errors on the same cell [21].

Faulty row and column. A faulty row/column is defined as a row/column on which errors occur on at least 2 different locations. Usually, a faulty row/column is likely to trigger many errors (CEs/UEs) when it is frequently accessed.

Node unavailability refers to that an online node fails to respond for over a minute.

CE storm. Usually, when a CE is identified, processors send System Management Interrupt (SMI) or Corrected Machine Check Interrupt (CMCI) to the firmware or OS. Generally, a few SMI or CMCI events have negligible performance impacts. However, in the extreme cases of CE storms, the system error handling mechanism would exhaust the CPU resources for error handling and cause a node to become unresponsive [10], [30]. In our practice, we frequently observe node unavailable cases associated with the pattern that the number of CEs exceeds 500 in 1 minute. When a node become unavailable and the above pattern is satisfied, furthermore, we did not find UE and DIMM communication loss on the node, we regard the unavailability is caused by CE storms.

DIMM communication loss refers to the condition that the communication between the DIMM and hardware system is lost, which typically can be reported by the system log.

DRAM-caused node unavailability (DCNU) refers to node unavailability caused by UEs, CE storms, and DIMM communication losses: a memory UE typically leads to a system crash, and the node goes unavailable; a fraction of CE storms or communication losses may also cause node unavailability.

B. Cloud Computing Systems

A typical cloud computing system is composed of numerous computing nodes, and each hosts multiple VMs. Let us take ECS system as an example. ECS system is composed of more than half a million physical computing nodes. The nodes are

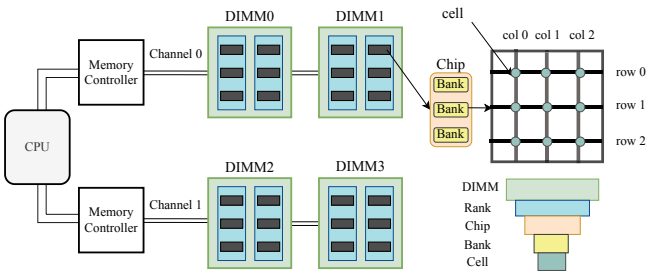


Fig. 1: DRAM components.

arranged into racks, and a group of racks forms a cluster. Usually, cloud computing systems use virtualization technology to provide a scalable and reliable runtime environment. However, software or hardware issues sometimes cause nodes to be unavailable, significantly impacting service availability provided by a VM, a node, even a large cluster.

Once a node is likely to be unavailable, it is better to migrate the workloads to other nodes promptly. Such a procedure is called live migration, which refers to moving a running VM between different nodes without disconnecting the clients or applications. Live migration is a powerful mechanism used in ECS system and other cloud computing systems. It enables rapid movement of workloads within clusters with low impact on running services.

C. Service Availability

High service availability is crucial to cloud computing systems. Cloud service providers have made every effort to assure the promised high service availability. Internally, they build strong infrastructures to monitor and log service status. Some clients are deployed to perform heartbeat probes periodically to assure node health. Moreover, they collect static configurations and monitoring data from operating system (OS) or Baseboard Management Controller (BMC). The typical monitoring data include error logs from kernel modules like mcelog and error detection and correction (EDAC), system resource usage (e.g., CPU and memory), and performance metrics (e.g., latency and throughput) from the computing nodes. Various alerts are developed based on the monitoring data to provide early warnings of potential failures, e.g., CE exceeds a predefined threshold. Then, preventive actions such as live migration can be performed to mitigate the failure impact.

Although tremendous efforts have been made to maintain high service availability, in reality, there are still many unexpected system issues causing node unavailability, e.g., software crashes, kernel panic, hardware errors, and memory/cache UEs. Once a node becomes unavailable, all VMs on it will fail to serve user requests. Among the failure causes, DRAM faults were observed as the leading hardware issues in the cloud and high-performance computing areas [24]. We found a similar observation in ECS system, which indicates that DRAM faults significantly impact service availability.

D. DRAM Components and Errors

DRAM components and errors. As shown in Fig. 1, in modern cloud servers, a CPU has several memory controllers. Each controller communicates with DIMMs through high-speed memory channels. Usually, a memory channel is shared by several DIMM slots. A DIMM has several ranks, and each is composed of several DRAM chips. For typical DDR4 DIMMs, a rank is composed of 16 chips for data bits and 2 additional chips for ECC bits. A chip consists of multiple banks, which enables the access parallelism. A DRAM bank is structured as a two-dimensional cell array indexed by rows and columns. At the micro-level, a cell can store multiple bits of data, and the number of data bits stored in a cell is called the data width of a chip, which is usually denoted as x4, x8, or x16, etc.

DRAM errors are the most commonly observed hardware errors in the cloud. They may occur on cells, rows, columns, even large blocks of the banks. The soft error usually occurs randomly and can hardly repeat. In contrast, hard errors are caused by hardware faults such as irreversible hardware wear-out and often repeat from time to time.

Error mitigation approaches. ECCs have been developed to detect and correct errors. The ECC algorithms used in modern CPUs are likely to correct almost any number of error bits in a single chip [11], [23]. If the error pattern goes beyond the correcting capability of the ECC, a UE will happen and lead to a system crash. The common strategy to avoid memory errors is to stop using the fault regions. Operating systems provide the mechanism of page offlining [8], [39] to avoid memory errors. However, it is not applicable to cloud services as unexpected large number of page offlining in peak service time may impact performance. The traditional method to repair hardware is based on a predefined error threshold. For example, a DIMM would be replaced when it reaches thousands of errors within 24 hours [16], [30].

III. Exploratory Study

We conduct an empirical study on the logs collected from over half a million nodes in ECS system over a year. The logs include the timestamp and error location of CEs, the occurrence of DCNUs. We perform considerable experiments on correlation analysis and present four major observations in this section.

Observation 1: CE storms are also major causes of DCNUs. Prior studies [12], [20], [35], [45] focus on UE analysis. We found that CE storms are also major causes of DCNUs: 56%, 42%, and 2% of DCNUs are caused by CE storms, UEs, and DIMM communication losses respectively. As described earlier, a UE typically leads to a node crash and a few CEs have negligible impacts. However, in reality, even live nodes may fail to respond to user requests when numerous CEs are continuously reported and handled. Our findings indicate that only UE prediction is not enough in our application context. Besides preventing UEs, we need to prevent node unavailability caused by CE storms.

TABLE I: CE temporal statistics in normal and failed nodes.

	Normal Nodes	Unavailable Nodes		
	Lifetime (months to years)	3h	1d	7d
1st quartile	5	15	24	28
Median	8	137	164	182
3rd quartile	20	944	1179	1257
Average	1041	8089	18201	23841

Observation 2: CEs are frequently observed before nodes become unavailable. For each DCNU occurrence, we aggregate the preceding CEs in 3 different time windows of 3 hours, 1 day, and 1 week before the DCNU. For simplicity, on normal nodes, we count the number of CEs throughout their lifetime (most nodes are running in the cloud for months to years). Table I shows the statistics of the data. Obviously, the number of CEs on the unavailable nodes are an order of magnitude higher than those on the normal nodes. The results indicate that CEs are indicators of DCNUs.

Observation 3: Faulty rows/columns are more frequently observed on unavailable nodes. Several prior studies have analyzed the spatial distribution of DRAM errors [1], [21], [38]. In case of many errors spanning in a large region of one row, the DIMM is likely to experience more errors in the future [21]. Here we further explore the relationship between DRAM spatial error patterns and DCNUs, which is not enclosed in previous studies.

Faulty rows and columns are commonly observed DRAM error patterns (defined in Section II-A). Fig. 2(a) illustrates the percentage of unavailable and normal nodes that have experienced faulty rows/columns, respectively. The results show that faulty rows or columns are more frequently observed on unavailable nodes than normal nodes. More than 70% of unavailable nodes have at least one faulty row or column. However, only no more than 5% normal nodes experienced faulty rows or columns. This observation indicates that spatial error patterns are predictive indicators of DCNUs.

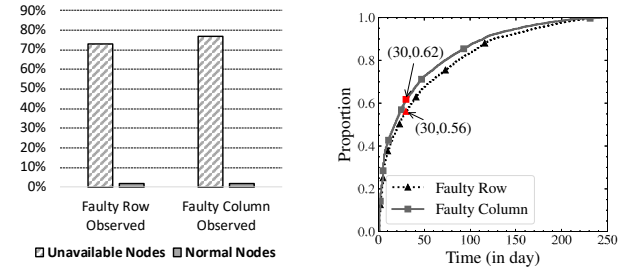
Observation 4: DCNUs have long-term relevance to faulty rows/columns. Fig. 2(b) visualizes the cumulative distribution function (CDF) of time intervals to the consequent DCNU after a new faulty row or column is observed. As we can see from the figure, 56% or 62% of DCNUs occur within a month once a new faulty row or column is observed. However, some nodes experience DCNU much later in the complicated environment. Therefore, we use four time windows within a month and the entire lifetime of a node as the observation windows to generate features (detailed in Section V).

IV. XBrainM: An AIOps System

We design XBrainM, an AIOps system integrated in ECS system, to predict and mitigate DCNUs. We briefly introduce the key components of XBrainM in this section.

A. System Overview

Fig. 3 shows the workflow of XBrainM. The system collects raw logs from multiple sources and feeds the cleaned data



(a) Percentage of unavailable/normal nodes on which faulty rows and columns are observed.

(b) Cumulative distribution function of time intervals (in day) to the consequent DCNU after a new faulty row or column is observed.

Fig. 2: Statistics on faulty rows and columns.

to the prediction module. The prediction module includes rule-based prediction and learning-based prediction. Rule-based prediction is based on simple heuristic rules capturing definitive signals of DCNUs, e.g., the number of recent CEs exceeds a certain threshold. Learning-based prediction can capture more complex signals that the rules cannot cover. It involves feature engineering, model learning, and prediction. The system periodically performs predictions to estimate whether ECS system nodes will experience DCNU in the near future. The system indiscriminately starts VM migration no matter the prediction is from the rule or ML model. Then stress testing (see Section IV-F) is used to validate DIMM health, and finally, the problematic DIMMs are repaired or replaced.

B. Raw Data Collection and Cleaning

ECS system has already supported the monitoring infrastructures as described in Section II-C. We collect the data from multiple sources, including hardware/BIOS/OS configurations, EDAC log, mcelog, kernel log, SEL, and BMC log. The

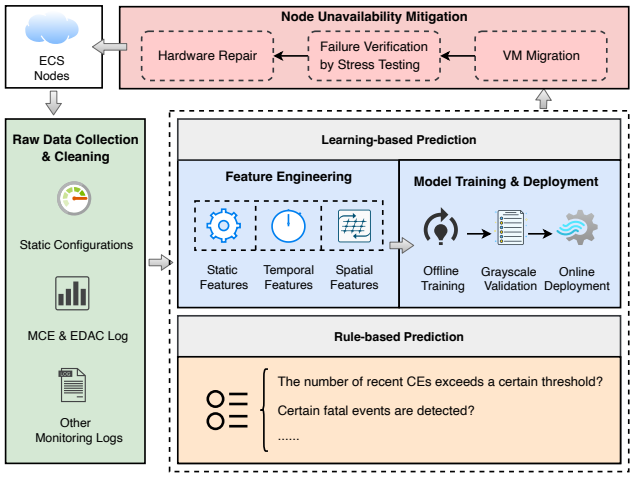


Fig. 3: The workflow of XBrainM.

static configurations are saved into a distributed database after node startup. The monitoring clients on nodes sample runtime system events and flush the data to the database. EDAC driver decodes the system address of the errors to the memory location, i.e., the socket, the memory controller, the channel, the slot, the rank, the bank group, the bank, the row, and the column. We clearly mark the source of the events and clean duplicates. When a node is detected to be unavailable, system experts on BMC, BIOS, and OS double-check the root cause according to the system logs and status, and mark the unavailability labels. As to online prediction, the runtime data is feed to online models through Flink [4], a streaming processing framework.

C. Rule-based Prediction

Several prior studies [16], [30] have explored the use of simple heuristic rules for predicting or mitigating DRAM errors. XBrainM applies total 18 heuristic rules to report early alerts of DCNUs. The rules are designed to capture definitive DCNU signals, e.g., the number of recent CEs exceed a certain threshold. These rules have been carefully tuned and running stably in ECS system, performing well in terms of interpretability and computational efficiency. However, a DCNU may have complex relevance to a large number of indirect signals, which cannot be well handled by the simple rule-based prediction.

D. Learning-based Prediction

To address the limitations of rule-based prediction, we further introduce learning-based prediction to XBrainM, which automatically learns the complex relevance between a large number of factors to DCNUs.

Prediction model. DCNU prediction is a typical binary classification problem. Normal nodes and unavailable nodes are considered as the negative and positive class in our system, respectively. Four machine learning models showing the state-of-the-art performance in binary classification are selected as candidates for evaluation, including extreme gradient boosting (XGBoost) [5], Random Forest (RF) [3], Support Vector Machine (SVM) [32], and Logistic Regression (LR) [40]. In practice, we expect that the model can be trained in parallel for performance consideration. In addition, the model is expected to quantify the importance of the features. Thus, data analysts can provide valuable suggestions based on the feature importance, such as model selection for DIMMs and servers. Furthermore, the model should be robust enough to be deployed in the hyper-scale cloud. To this end, we finally select XGBoost, which not only satisfies three basic requirements but also shows the best prediction performance in the experiments.

Prediction interval and prediction window. As described previously, DCNU prediction is performed periodically. A moderate prediction interval is desired to report DCNU alerts in time. In XBrainM, we find the day-level interval is insufficient to predict some DCNUs on time, while 1 minute introduces too much system overhead. Finally, we chose to perform predictions every 5 minutes. As to the prediction window, its length is set as three days. That is to say, we perform the prediction every five minutes to predict DCNUs in the future three days.

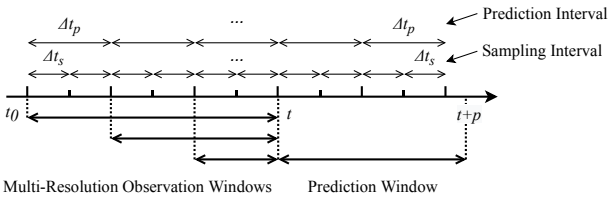


Fig. 4: Time windows in XBrainM.

Sampling interval and observation window. As shown in Fig. 4, in XBrainM, the sampling interval for runtime events is 1 minute, e.g., we save the CE counts of nodes in every minute. To generate more effective spatio-temporal features, we use multi-resolution observation windows to aggregate both the short-term and long-term error statistics from historical data. For example, in time t, we compute the sum and increment of CE counts in the recent 3 hours and 1 week, which are then used to predict DCNUs occur in the three days after t.

Feature engineering. Static node configurations and temporal error statistics are common features used in prior studies [2], [29]. Besides them, inspired by the observation that faulty rows and columns are predictive indicators of DCNUs (see Section III), we further explore micro-level spatial patterns of DRAM errors and then leverage multiple windows to aggregate the spatial statistics. More details will be discussed in Section V.

Model training and tuning. Software upgrades and workload variance may lead to data changes in the cloud. We retrain the model to adjust the environment change every 3 months, as the data in a 3-month period is relatively stable for such a large-scale cloud. Moreover, when plenty of new nodes are deployed online, or prediction results are far from expected, we also train a new model on demand. We employ Bayesian optimization [36] to optimize the model, seeking the best hyperparameters. Considering the model's real effects cannot be evaluated by traditional metrics like F1-score, we propose the new metric NURR to quantify the node unavailability reduction (detailed in Section VI). We always use the hyperparameter combination achieving the best NURR.

Deployment in the cloud. The online deployment is divided into 2 stages: (i) Grayscale validation: in this stage, we do not take any mitigation actions but only validate the performance. (ii) Online validation: after 3 months of running, we know the online performance of the model. If its performance is acceptable, it can be used to trigger DCNU mitigating actions (e.g., live migration) in the production environment.

E. Training and Prediction Overhead

XBrainM uses a few dedicated servers to train the prediction models and perform online predictions:

- Training ML models: 1 VM with 104 cores and 300 GB memory. Training with the 3-month data as input takes around 4 hours.
- Online prediction: 30 VMs, each with 4 cores and 8 GB memory. One-round feature extraction and prediction for over half a million nodes can be finished in 1 minute.

F. Mitigating DRAM-Caused Node Unavailability

When the rules or the learning model predict a node to be unavailable, we then perform live migration to migrate all VMs on the node to a healthy or vacant node. Generally, live migration can move VMs seamlessly without disrupting the running services, but it sometimes fail or cannot be applied due to limitations such as network boundaries. Cold (non-live) migration is suitable for more general cases, which isolates the VMs by shutting down or disconnecting from the network and then performs the migration. During cold migration, the original node is unavailable for a while.

After all VMs are successfully migrated, the predicted faulty node is then marked as offline, and stress testing is performed to reproduce DRAM errors. The testing tool scans through memory addresses with the popular algorithms used in memory testing, such as modified algorithmic test sequence (MATS) and march C- [33]. If the tested nodes go unavailable and DRAM faults are confirmed, we send a repair ticket to platform engineers to repair or replace problematic DIMMs.

V. Feature Engineering

Usually, predicting DRAM error/failure is quite challenging because the features extracted directly from the raw logs are either few or ineffective. For example, studies [2], [16] only use 3 and 20 features for prediction, which is not sufficient to achieve desirable performance in our evaluation. Richer features are included in our system. We employ novel feature engineering methods to extract static, spatial, and temporal features from the large-volume raw data.

A. Static Features

Static features represent fixed characteristics of a physical node or DRAM components, e.g., CPU generation and DRAM capacity. We perform one-hot encoding to transform these static, categorical configurations into binary vectors. We use total 11 static features and some typical features are:

- CPU Generation: different generations may use different ECC algorithms.
- DIMM Vendor: different manufacturing process may involve different reliability characteristics.
- DIMM Capacity: relevant to DRAM density.
- Server Model: though it is not directly relevant to DIMM quality, difference of hardware configuration and favorable workloads may trigger different fault patterns relevant to DCNU.
- Cluster: nodes in the same cluster are likely to run similar workloads and exhibit similar fault patterns.
- BIOS/OS Version: relevant to the behaviors of the system.

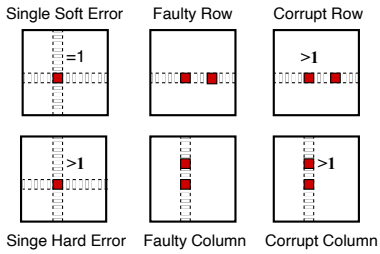


Fig. 5: Spatial error patterns.

B. Spatial Features

As introduced in Section II-D, a DRAM bank is structured as a two-dimensional cell array indexed by rows and columns. Considering the spatial error distributions (e.g., faulty rows/-columns) on the DRAM bank are relevant to DCNUs, we design a series of spatial features.

Bank-level features. Firstly, to infer the fault range and severity on a DRAM bank, we count the number of rows and columns that experienced CEs, then compute the sum, average, max of CEs and cells experienced CEs on them. For example, the feature "ce_count_col_max" represents the maximum number of CEs on rows.

Then we design several error patterns to capture the micro-level CE spatial distributions. Fig. 5 illustrates the spatial distributions of these patterns, which are defined as follows:

- Single soft error. A single error occurs in a new cell that is not in the faulty row/column.
- Single hard error. A cell not in a faulty row/column has experienced at least two errors.
- Faulty row/column, as defined in Section II-A.
- Corrupt row/column. A faulty row/column is observed, and at least 1 hard error is observed in the row/column.

The *single soft/hard error* is used to distinguish the transient soft fault from the permanent hard fault on a single cell. In addition, we use a *corrupt row/column* to check whether hard errors are observed on a faulty row/column, which typically indicates the faulty row/column is repeated accessed and likely to trigger more CEs.

DIMM/Node-level spatial features. We then aggregate those bank-level features to DIMM and node level by computing the max, sum, average, and standard deviation of each features from all banks that experienced CEs. For example, "single_hard_error_b_sum" means the total number of the single hard error pattern among all banks of a node.

As introduced in Section III, fault rows/columns are predictive indicators of DCNUs. With rich features relevant to faulty rows/columns, the ML model is able to learn the subtle correlation between fault rows/columns and DCNUs.

C. Temporal Features

Beside the spatial features, we also use relevant events from kernel logs, including CMCI Storm Detected (more than 16 CEs in second), MCE Killing (processes get killed due to

hardware memory corruption), CE Suppressed Notify (CEs are suppressed by a special suppressing mechanism), etc. We aggregate the historical occurrence of spatial feature as well as the kernel events to correlate them with DCNUs.

Since we observe that some DCNUs are more relevant to temporal error statistics, we select 3 hours, 1days, 1 week, 4 weeks, and the entire lifetime of a node as the different time window to aggregate the statistics of all features within the windows. For each time window, we calculate 7 types of popular statistics from the raw data with 1-minute granularity. Let t be a timestamp, x be an event, X be a sequence of minute-level event counts in the time window $(t-w, t]$, μ be the mean of X, σ be the standard deviation of X. The statistics at t are computed as follows:

- $Sum(x,t,w)$ and $Std(x,t,w)$ refer to the sum and standard deviation of event counts of x in the time window.
- $Diff$ means the difference of total errors in the first and second half of the time window: $Diff(x,t,w) = Sum(x,t,\frac{w}{2}) - Sum(x,t-\frac{w}{2},\frac{w}{2})$
- $Delta$ is the difference of total errors in recent 2 time windows: $Delta(x,t,w) = Sum(x,t,w) - Sum(x,t-w,w)$
- $Kurtosis$ is a statistical measure of the tail of the probability distribution. We use the standard measure of a distribution's kurtosis, a scaled version of the fourth moment of the distribution: $Kurtosis(x,t,w) = E\left[(\frac{X-\mu}{\sigma})^4\right]$
- $Skew$ differentiates extreme values in one versus the other tail. Let $\tilde{\mu}$ be the median of X: $Skew(x,t,w) = \frac{3(\mu-\tilde{\mu})}{\sigma}$
- $Entropy$ measures the average level of "uncertainty" inherent in the possible outcomes of a random variable. Assume that there are k different values observed in X, and let p_i denote the probability of i-th value, which equals to the ratio of the frequency of i-th value to the number of sampling periods. We have $Entropy(x,t,w) = -\sum_{i=1}^{k} p_i \log(p_i)$

Our feature engineering can produce more than 10,000 features. We then adapt recursive feature elimination [19] to further improve the feature set: (i) Train a XGBoost model; (ii) Eliminate the top-5% least important features ranked by XGBoost; (iii) Repeat step i˜ii until the prediction performance cannot be improved in recent three iterations. After that, about 2,500 features are reserved. They achieve the best prediction performance with limited computational cost (see Section IV-E), thus are used for regular training and prediction.

VI. Model Evaluation

Correctly evaluating the model's effects on node unavailability is vital before deploying a model online. XBrainM introduces sequential node-level evaluation strategies and a new metric NURR to estimate the node unavailability reduction and tune model hyperparameters with NURR.

A. Sequential Node-level Evaluation

To learn an accurate model, we need to split the error logs to simulate the real cases as much as possible. In our experiments, we keep the log in sequential order and split the data into 3-month pieces, using the earlier 3-month data as training data and the subsequent 3-month data for testing.

Since that prediction evaluation window is 3 days which is much longer than the prediction interval 5 minutes, once a DCNU belongs the 3-day prediction window of any prediction interval, the DCNU impact can be mitigated. Therefore, we aggregate the predictions to node level and evaluate the prediction performance only according to whether the DCNU is in the prediction window or not. In other words, a correct prediction is counted if DCNU occurs in the prediction window of any prediction which predicts the node to be unavailable. On the other hand, multiple incorrect predictions on a normal node are counted as a single false prediction.

B. Node Unavailability Reduction Rate (NURR)

Traditional evaluation metrics for binary classification include recall, precision, F1-score, etc. Let TP, FP, and FN denote the total number of true positives, false positives, and false negatives, respectively. Precision, TP/(TP+FP) is defined as the ratio of true positives to total alerts. A low precision value indicates that many useless migrations would be taken. Recall, TP/(TP+FN), is defined as the ratio of true positives to total DCNUs observed. A low recall value indicates we fail to identify a large portion of DCNUs to happen in the future. In many cases, a high precision value is likely to be associated with a low recall value and vice versa. F1-score is the harmonic average of precision and recall to penalize a bias to any of the two metrics.

These metrics are useful for evaluating the prediction accuracy. However, they cannot measure the model's real effects on node unavailability. In this subsection, we propose a new metric, Node Unavailability Reduction Rate (NURR). NURR is directly relevant to node unavailability. We use NURR instead of F1-score to tune the model hyperparameters to reach lower unavailability in our cloud.

Let t_u denote the average duration of node unavailability that fails to be predicted. Without a prediction model, the total node unavailable time is:

$$T = t_u(TP + FN) \tag{1}$$

In XBrainM, the direct node unavailability is greatly reduced by early VM migration. As described in Section IV-F, VMs on most nodes are moved by live migration transparently. A small fraction of nodes experience cold migration, during which process the node is marked as unavailable for a while. Let y denote the percentage of cold migration, t_m denote the average node unavailable time during cold migration. Considering the impact of DCNU prediction and VM migration, there would be three DCNU cases as illustrated in Fig. 6:

- Case ❶: Correct prediction (TP) and cold migration is performed. The total unavailable time under this case is $yt_m \cdot TP$.
- Case ❷: Prediction missed (FN). The average duration of this case is t_u, and the total unavailable time is $t_u \cdot FN$.

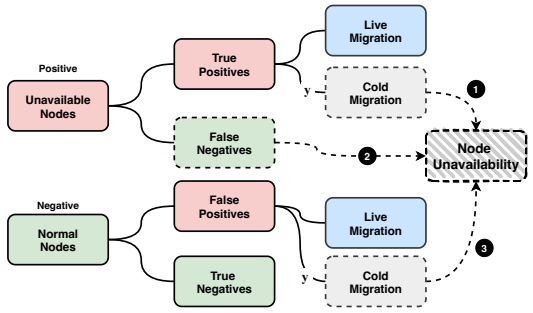


Fig. 6: DCNU cases with a prediction model.

- Case ❸: False DCNU prediction (FP) and cold migration is performed. As described before, the nodes will be considered as unavailable for a while if the cold migration is performed. The total unavailable time is $yt_m \cdot FP$.

Therefore, considering the prediction and VM migration impact, the new total unavailable time is:

$$T' = yt_m \cdot TP + t_u \cdot FN + yt_m \cdot FP \tag{2}$$

The node unavailability reduction rate is calculated as:

$$NURR = \frac{T - T'}{T} \tag{3}$$

$$= \frac{t_u(TP + FN) - (yt_m \cdot TP + t_u \cdot FN + yt_m \cdot FP)}{t_u(TP + FN)} \tag{4}$$

$$= \frac{t_u TP - yt_m \cdot TP - yt_m \cdot FP}{t_u(TP + FN)} \tag{5}$$

$$= \frac{TP}{TP + FN} - y\frac{t_m}{t_u}\frac{TP + FP}{TP + FN} \tag{6}$$

$$= \frac{TP}{TP + FN} - y\frac{t_m}{t_u}\frac{\frac{TP}{TP+FN}}{\frac{TP}{TP+FP}} \tag{7}$$

$$= recall - y\frac{t_m}{t_u}\frac{recall}{precision} = (1 - \frac{y\frac{t_m}{t_u}}{precision}) \cdot recall \tag{8}$$

Since $y\frac{t_m}{t_u}$ is always positive, precision determines DCNU prediction is beneficial or not. If $precision < y\frac{t_m}{t_u}$, the NURR becomes negative, which indicates that inaccurate predictions may even cause more nodes to be unavailable. In contrast, a high-precision model can always achieve positive NURR, and recall then amplifies the unavailability reduction magnitude.

VII. Experiments

A. Experiment Setup

Dataset. The dataset for offline evaluation is collected from more than half a million nodes in ECS system in one year and covers the multiple CPU generations, and the DIMMs of multiple vendors, generations and capacities, etc. These nodes incorporate different hardware specifications, e.g., varying CPU generation, DIMM vendor, memory capacity, etc. We process the raw data and generate more than 1TB structured data, each sample with 2500+ features. We divide the dataset evenly into 4 parts, each with 3 consecutive months of the data. Then we reorganize the 4 parts to 3 datasets, each with 3 months of the training data and the testing data in the following 3 months. Table II shows the evaluation period of 3 datasets.

TABLE II: Experimental datasets, where t_0 is the timestamp that Dataset1 starts with, and m is the time unit in month.

Name	Training Period	Test Period
Dataset1	$[t_0, t_0 + 3m)$	$[t_0 + 3m, t_0 + 6m)$
Dataset2	$[t_0 + 3m, t_0 + 6m)$	$[t_0 + 6m, t_0 + 9m)$
Dataset3	$[t_0 + 6m, t_0 + 9m)$	$[t_0 + 9m, t_0 + 12m)$

Evaluation metrics. We use the traditional metrics *recall*, *precision*, and *F1-score* to evaluate the accuracy improvement of our approach over existing solutions. Furthermore, we use NURR to evaluate the predictors' effects on service availability, which is beyond the capability of traditional metrics.

B. Comparison with Existing Methods

Since the definition of our problem is totally different from prior studies, there is no existing methods for reference and comparison. The most relevant work is UE prediction [2], [16]. Hence, we adapt two UE predictors in our environments to predict DCNUs: *CE Threshold* [16] and UE predictor based on random forest [2]. We also explore the use of simple heuristic rules based on threshold of faulty rows and columns. Finally, four baselines are included in our evaluation.

1. CE Threshold reports a DCNU alert when the CE count reaches a certain threshold in recent 24 hours [16]. We carefully tune the threshold and report the best NURR.

2. RF-based UE Predictor uses random forest to predict UEs with a set of basic memory error statistics [2]. We use DCNUs instead of UEs to train the model and carefully tune the hyperparameters to achieve the best NURR.

3. Faulty-Row Threshold. A node will be predicted to be unavailable if the number of faulty rows accumulated in the history exceeds a threshold.

4. Faulty-Column Threshold. Similar to Faulty-Row Threshold.

As to XBrainM, we evaluate the individual performance of its rule-based prediction (XBrainM-Rule), XGBoost-based prediction (XBrainM-XGB), and the hybrid approach actually used in XBrainM that predicts a DCNU if any of the rule or XGBoost predict a DCNU.

Table III shows the comparison results. Firstly, as the rules used in XBrainM can be considered as a superset of the threshold-based baselines, XBrainM-Rule achieves slightly-better performance (2%˜9% higher NURR) than the baselines. Secondly, the ML model of XBrainM, i.e., XBrainM-XGB, consistently outperforms the baselines in the three datasets. With more effective features, XBrainM-XGB can achieve at least 30% absolutely higher F1-score and 37% absolutely higher NURR than the baselines. Last but not least, the hybrid approach demonstrates the best F1-score and NURR that are more than 40% absolutely better than the baselines.

TABLE III: Recall / Precision / F1-socre / NURR of different prediction approaches.

	Dataset1	Dataset2	Dataset3
CE Threshold	0.24 / 0.34 / 0.28 / 0.17	0.36 / 0.17 / 0.23 / 0.15	0.36 / 0.17 / 0.23 / 0.15
RF-based UE Predictor	0.50 / 0.18 / 0.26 / 0.22	0.46 / 0.16 / 0.23 / 0.17	0.39 / 0.22 / 0.28 / 0.21
Faulty-Row Threshold	0.41 / 0.25 / 0.31 / 0.23	0.37 / 0.23 / 0.28 / 0.19	0.35 / 0.18 / 0.24 / 0.14
Faulty-Column Threshold	0.48 / 0.20 / 0.28 / 0.22	0.40 / 0.17 / 0.24 / 0.16	0.81 / 0.15 / 0.26 / 0.26
XBrainM-Rule	0.30 / 0.62 / 0.41 / 0.25	0.33 / 0.58 / 0.42 / 0.28	0.35 / 0.60 / 0.44 / 0.29
XBrainM-XGB	0.75 / 0.51 / 0.61 / 0.60	0.77 / 0.48 / 0.59 / 0.61	0.79 / 0.49 / 0.60 / 0.63
XBrainM-Rule+XGB	0.77 / 0.68 / **0.72** / **0.65**	0.83 / 0.65 / **0.73** / **0.70**	0.78 / 0.67 / **0.72** / **0.66**

TABLE IV: Performance of different ML models in XBrainM.

	Recall	Precision	F_1	NURR
XBrainM-XGB	79.34%	48.65%	**60.31%**	**63.03%**
XBrainM-RF	68.57%	49.26%	57.33%	54.65%
XBrainM-LR	51.30%	41.23%	45.72%	38.86%
XBrainM-SVM	55.73%	29.32%	38.42%	36.72%

With the simple rules and effective learning model, XBrainM consistently achieves over 77%, 65%, 72%, and 65% of recall, precision, F1-score and NURR, respectively.

C. Comparison of Various Prediction Models

We then explore the use of different ML models in XBrainM and perform head-to-head comparisons among XGBoost, RF, SVM, and LR. Table IV shows the recall, precision, F1-score (denoted as F_1), and NURR with 4 different models. SVM and LR perform relatively well but are still inferior to RF and XGBoost. XGBoost achieves the highest F1-score and NURR, which are 60.31% and 63.03%, respectively. It means when the model is deployed online, it can roughly reduce 63.03% node unavailability caused by DRAM faults. So, we use XGBoost as the default learning-based prediction model in XBrainM.

D. Feature Evaluation

Feature importance. To understand the underlying causes of DCNU, we expect to quantify the importance of each feature. Fig. 7 illustrates the top 20 important features relevant to DCNU, including 10 spatial (beginning with 's_') and 10 temporal features respectively. It indicates that both spatial and temporal features are important indicators of DCNU. The most important spatial features are s_single_hard_error_b_max, and s_single_hard_error_b_sum; they are the max and sum of single hard errors from the fault banks of a node. For temporal features, the events in 3 hours t_mcelog_ce_count_3h and t_mce_killing_3h contribute most to the prediction. It indicates that the temporal error statistics in the 3-hour time window are strongly relevant to DCNUs.

Effectiveness of feature engineering. We also examine the importance of static, spatial, and temporal[1] features respectively by excluding them from the feature list. The experimental results are shown in Table V. As we can see from the table, the static features are less important, and the NURR decreases no more than 7%. Both the temporal and spatial features contribute a lot to the final results as the NURR decreases over 11% and 13% respectively, without including those features.

[1]CE counts, kernel events, and their temporal aggregations.

TABLE V: Performance on different feature combinations.

	Recall	Precision	F_1	NURR
All Features	79.34%	48.65%	**60.31%**	**63.03%**
Excluding Static	70.08%	50.84%	58.93%	56.29%
Excluding Temporal	67.01%	43.01%	52.39%	51.43%
Excluding Spatial	67.35%	38.57%	49.05%	49.89%
Top-20 Only	72.64%	37.22%	49.22%	53.12%

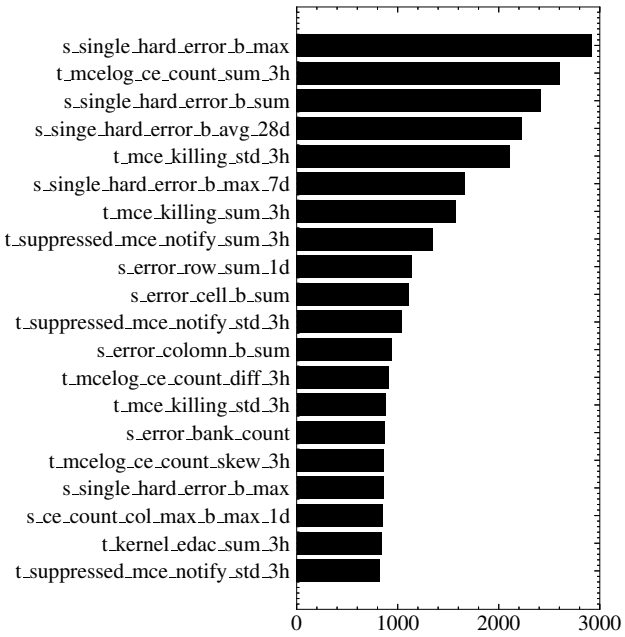
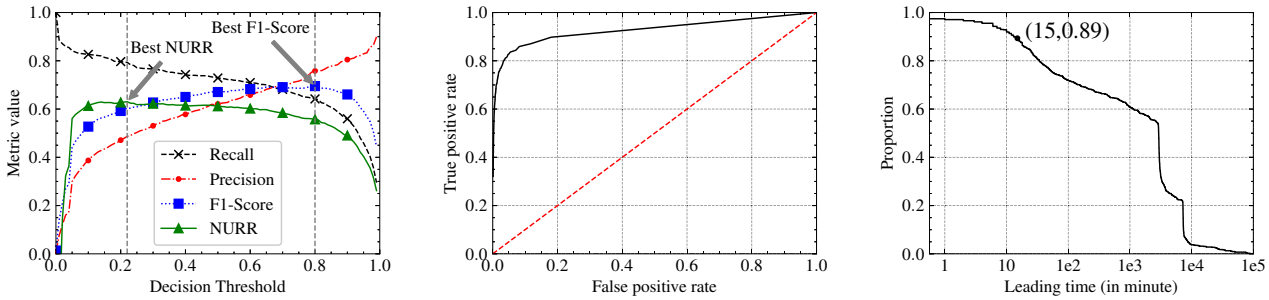


Fig. 7: Feature importance of the XGBoost model.

The spatial features are most important in prediction, and it again indicates that DCNU occurrences are relevant to micro-level spatial fault characteristics [16], [21]. Furthermore, to evaluate whether we need so many features, we train the XGBoost model with only the top-20 important features. Table V shows that the NURR decreases roughly 10%. In such a large-scale cloud, 10% node unavailability reduction impacts a lot and is quite valuable, especially considering that the

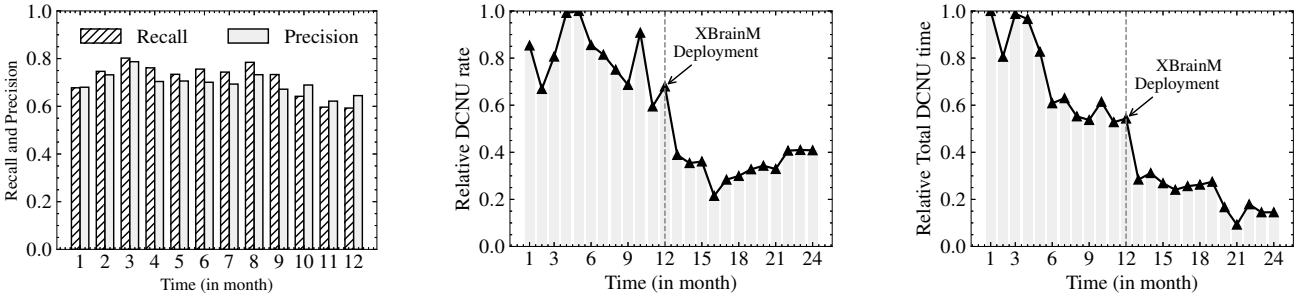


(a) Offline evaluation, varying decision threshold.

(b) ROC curve in offline evaluation (AUC = 0.927).

(c) Leading time versus proportion of leading time .

Fig. 8: Parameter tuning and leading time.



(a) Prediction recall and precision of XBrainM in the production environment.

(b) Monthly DCNU rate before and after XBrainM deployment. Y-axis is in relative scale due to confidential reasons.

(c) Monthly toal DCNU time before and after XBrainM deployment. Y-axis is in relative scale due to confidential reasons.

Fig. 9: Online results.

overhead with 2500+ features is negligible.

E. Parameter Tuning and Leading Time

We next perform a set of experiments to evaluate the impact of different parameters and show the leading time of the successful predictions with our model.

Impact of decision threshold. For binary classification, the decision threshold decides the confidence level of the alerts. As shown in Fig. 8(a), a strict decision threshold is likely to be associated with a low recall and a high precision and vice versa. By carefully tuning the decision threshold, we can strike a good balance between precision and recall. Fig. 8(a) shows the variance of recall, precision, F1-score, and NURR with the varying threshold. As shown in the figure, when we reach the best NURR 63%, the recall and precision are 79% and 49%. As to F1-score, the threshold 0.8 achieves the best F1-score 70% with 64% recall and 75% precision. Tuning the threshold with F1-score is 8% worse NURR than tunning with NURR directly. That is to say, a model tuned by NURR can reduce 8% more node unavailability when it is deployed online.

AUC (Area Under The Curve)-ROC (Receiver Operating Characteristics) curve is one of the most popular evaluation metrics for checking the performance of the classification model. It measures the quality of predictions irrespective of what classification threshold is chosen. As shown in Fig. 8(b), our model is accurate and robust as the curve is close to the upper left corner, and the percentage of AUC to 92.7%.

Impact of the prediction interval. We select prediction interval from a set of candidates {5 minutes, 1 hour, 3 hours, 1 day}. Generally, a shorter prediction interval can reach a higher recall, but it may involve too much system overhead. On the other hand, a longer window brings less overhead, but it fails to predict some failures promptly. Table VI shows the prediction performance with different prediction intervals. As we can see from the table, precision is relatively stable, but recall keeps decreasing as the prediction interval increases. In the list, a 5-minute prediction interval is most suited to DCNU prediction in XBrainM.

Leading time. Fig. 8(c) shows the distribution of the leading time of DCNUs which are correctly predicted. The X-axis

TABLE VI: Performances on different Prediction Interval.

Prediction Interval	Recall	Precision	F_1	NURR
5 minutes	79.34%	48.65%	**60.31%**	**63.03%**
1 hour	57.57%	48.41%	52.59%	45.68%
3 hours	46.69%	48.13%	47.40%	36.99%
1 day	39.75%	48.28%	43.60%	31.51%

is the log-scale time-axis measured in minute, while Y-axis represents the fraction of the predictions whose leading time exceeds the value of the X-axis. As we can see from the figure, more than 89% of all leading time is longer than 15 minutes, leaving sufficient time for VM migration. Actually, in our cloud, more than 95% of the leading time is sufficient for VM migration as our system specially optimizes the process of live migration.

F. Online Results

XBrainM has been deployed in the production environment of ECS system for over a year and significantly improved the overall service availability.

Online recall and precision. As described earlier, we use stress testing to confirm whether the predicted faulty nodes will become unavailable or not. Then, we compute the online recall and precision based on the stress testing results. As shown in Fig. 9(a), XBrainM consistently achieved more than 60% monthly recall and precision in one year.

DCNU rate and total DCNU time. In the production environment, DCNU rate is the percentage of nodes that experienced DCNU among all nodes, and the total DCNU time is used to evaluate the total unavailability caused by DRAM faults. Fig. 9(b) and 9(c) illustrate the monthly DCNU rate and monthly total DCNU time in one year before and after XBrainM deployment. The Y-axis is in relative scale due to business confidential reasons. As shown in the figures, both the DCNU rate and total DCNU time have a significant (over 40% relative) reduction after XBrainM being deployed, which indicates XBrainM effectively avoids a considerable number of DCNUs. As to the yearly average, XBrainM significantly improves the availability and reliability of our cloud by reducing 57% of DCNU rate and 69% of total DCNU time.

VIII. Related Work

During the past years, many studies have been carried out on failure prediction in real production systems. Li et al. [26], [27] performed empirical and realistic evaluation of memory errors on an Internet service farm. They observed different failures on cells, rows, columns, and whole chips. Zivanovic et al. [45] presented the memory failure characteristics from a large-scale field study in Google's server fleet. It was observed that the probabilities of both CEs and UEs increase in the presence of prior errors. Hwang et al. [21] presented an expanded study on the error data collected from Google's server fleet and the Blue Gene clusters. It was found that along those cells in the band of nearby rows and columns, the probability of error occurrence increases significantly. Sridharan et al. [37], [38] argued that DRAM failure rate is a better indicator of DIMM health than error rate.

Dattatraya et al. [10] studied the memory error data for more than 1 billion compute node-hours over 8 years. It was observed that MCEs from memory and cache constitute a large fraction of total hardware failures. Patwari et al. [34] performed an investigation on spatial behaviors of DRAM errors across an entire cluster. It was shown that some susceptible regions are more vulnerable to errors than other regions. Meza et al. [30] performed an investigation on memory errors in the servers at Facebook over fourteen months. It was observed that the hardware and software overheads to handle such errors cause a kind of denial of service attack on some servers. However, this work only quantifies the importance of the features, no prediction results are given. Du and Li [13] proposed a method to predict DRAM errors in micro-level components, such as cell rows and columns. The method is based on a kernel function that measures the similarity between the current observation and a certain previous observation in history.

Several studies proposed to use machine learning to predict DRAM UEs before they occur [2], [16], [18]. They typically train an ML model from historical data and then perform predictions for further actions such as job migration to mitigate the failure impact. Giurgiu et al. [18] processed the raw data using feature selection, pre-branching, and data imputation, finally built a sliding window classification model with random forests and achieved decent performance in experiments. However, the best recall reported with the model is only 31%. Boixaderas et al. [2] used random forest to predict DRAM uncorrected errors in the MareNostrum3 supercomputer. Instead of using the traditional metrics such as precision, recall, and F1-score, this work proposed the cost-aware prediction to measure the system cost with a low precision (no more than 2%, our precision is much higher).

Recently, building AI solutions to solve system problems has become a hot topic. A few studies proposed to enhance system availability in cloud or HPC by predicting node failures in advance [6], [9], [28], [29], [43]. They apply the most popular machine learning models in their system, such as LSTM, random forest, etc. Failure prediction on other important components of the computing nodes or systems were also studied, such as disk failure prediction [17], [41], [42], switch failures prediction [44] and GPU error prediction [31]. Differing from those studies, our task is to prevent DCNUs.

To mitigate the memory errors proactively, the software solution of memory page offlining was studied in [8], [14], [21], [30], [39]. As a feature on Intel ICE Lake platform, Partial Cache Line Sparing was studied in [15].

IX. Conclusion

In this paper, we proposed to predict DCNU relevant to UEs, CE storms, and DIMM communication losses. We designed novel spatio-temporal features based on the observation that DCNUs have strong relevance to temporal statistics and spatial patterns of CEs, to train a ML model. We then developed XBrainM, which ensembles the ML model and traditional rule-based approaches to predict DCNUs. Offline results show that our prediction approach achieves over 40% more node unavailability reduction rate than existing methods. Online results in the production environment show that XBrainM significantly reduces 57% of DCNU rate and 69% of total DCNU time.

REFERENCES

[1] Further memory fault modeling. https://link.springer.com/content/pdf/bbm%3A978-0-306-47972-4%2F1.pdf.

[2] Isaac Boixaderas, Darko Zivanovic, Sergi Moré, Javier Bartolome, David Vicente, Marc Casas, Paul M Carpenter, Petar Radojković, and Eduard Ayguadé. Cost-aware prediction of uncorrected dram errors in the field. In *SC*, pages 1–15, 2020.

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4), 2015.

[5] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[6] Yujun Chen, Xian Yang, Qingwei Lin, Hongyu Zhang, Feng Gao, Zhangwei Xu, Yingnong Dang, Dongmei Zhang, Hang Dong, Yong Xu, et al. Outage prediction and diagnosis for cloud service systems. In *WWW*, pages 2659–2665, 2019.

[7] Alibaba Cloud. Elastic compute service (ecs) service level agreement, 2021. https://www.alibabacloud.com/help/en/doc-detail/42436.htm.

[8] Carlos HA Costa, Yoonho Park, Bryan S Rosenburg, Chen-Yong Cher, and Kyung Dong Ryu. A system software approach to proactive memory-error avoidance. In *SC*, pages 707–718, 2014.

[9] Anwesha Das, Frank Mueller, and Barry Rountree. Aarohi: Making real-time node failure prediction feasible. In *IPDPS*, pages 1092–1101, 2020.

[10] Harish Dattatraya Dixit, Fan Lin, Bill Holland, Matt Beadon, Zhengyu Yang, and Sriram Sankar. Optimizing interrupt handling performance for memory failures in large scale data centers. In *ICPE*, pages 193–201, 2020.

[11] Timothy J Dell. A white paper on the benefits of chipkill-correct ecc for pc server main memory. *IBM Microelectronics division*, 11:1–23, 1997.

[12] Catello Di Martino, Zbigniew Kalbarczyk, Ravishankar K Iyer, Fabio Baccanico, Joseph Fullop, and William Kramer. Lessons learned from the analysis of system failures at petascale: The case of blue waters. In *DSN*, pages 610–621, 2014.

[13] Xiaoming Du and Cong Li. Memory failure prediction using online learning. In *MEMSYS*, pages 38–49, 2018.

[14] Xiaoming Du and Cong Li. Combining error statistics with failure prediction in memory page offlining. In *MEMSYS*, pages 127–132, 2019.

[15] Xiaoming Du and Cong Li. Dpcls: Improving partial cache line sparing with dynamics for memory error prevention. In *ICCD*, pages 197–204, 2020.

[16] Xiaoming Du, Cong Li, Shen Zhou, Mao Ye, and Jing Li. Predicting uncorrectable memory errors for proactive replacement: An empirical study on large-scale field data. In *European Dependable Computing Conference*, pages 41–46, 2020.

[17] Sandipan Ganguly, Ashish Consul, Ali Khan, Brian Bussone, Jacqueline Richards, and Alejandro Miguel. A practical approach to hard disk failure prediction in cloud platforms: Big data model for failure management in datacenters. In *International Conference on Big Data Computing Service and Applications*, pages 105–116, 2016.

[18] Ioana Giurgiu, Jacint Szabo, Dorothea Wiesmann, and John Bird. Predicting dram reliability in the field with machine learning. In *Middleware*, pages 15–21, 2017.

[19] Baptiste Gregorutti, Bertrand Michel, and Philippe Saint-Pierre. Correlation and variable importance in random forests. *Statistics and Computing*, 27(3):659–678, 2017.

[20] Saurabh Gupta, Tirthak Patel, Christian Engelmann, and Devesh Tiwari. Failures in large scale systems: long-term measurement, analysis, and implications. In *SC*, pages 1–12, 2017.

[21] Andy A Hwang, Ioan A Stefanovici, and Bianca Schroeder. Cosmic rays don't strike twice: Understanding the nature of dram errors and the implications for system design. *ASPLOS*, 47(4):111–122, 2012.

[22] IDG. 2020 cloud computing survey, 2020. https://www.idg.com/tools-for-marketers/2020-cloud-computing-study/.

[23] Intel. Intel e7500 chipset mch intel x4 single device data correction (x4 sddc) implementation and validation, 2002. https://www.intel.com/content/dam/doc/application-note/e7500-chipset-mch-x4-single-device-data-correction-note.pdf.

[24] Scott Levy, Kurt B Ferreira, Nathan DeBardeleben, Taniya Siddiqua, Vilas Sridharan, and Elisabeth Baseman. Lessons learned from memory errors observed over the lifetime of cielo. In *SC*, pages 554–565, 2018.

[25] Sebastien Levy, Randolph Yao, Youjiang Wu, Yingnong Dang, Peng Huang, Zheng Mu, Pu Zhao, Tarun Ramani, Naga Govindaraju, Xukun Li, et al. Predictive and adaptive failure mitigation to avert production cloud vm interruptions. In *OSDI*, pages 1155–1170, 2020.

[26] Xin Li, Michael C Huang, Kai Shen, and Lingkun Chu. An empirical study of memory hardware errors in a server farm. In *HotDep*, 2007.

[27] Xin Li, Michael C Huang, Kai Shen, and Lingkun Chu. A realistic evaluation of memory hardware errors and software system susceptibility. In *ATC*, pages 75–88, 2010.

[28] Yangguang Li, Zhen Ming Jiang, Heng Li, Ahmed E Hassan, Cheng He, Ruirui Huang, Zhengda Zeng, Mian Wang, and Pinan Chen. Predicting node failures in an ultra-large-scale cloud computing platform: an aiops solution. *ACM Transactions on Software Engineering and Methodology*, 29(2):1–24, 2020.

[29] Qingwei Lin, Ken Hsieh, Yingnong Dang, Hongyu Zhang, Kaixin Sui, Yong Xu, Jian-Guang Lou, Chenggang Li, Youjiang Wu, Randolph Yao, et al. Predicting node failure in cloud service systems. In *ESEC/FSE*, pages 480–490, 2018.

[30] Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu. Revisiting memory errors in large-scale production data centers: Analysis and modeling of new trends from the field. In *DSN*, pages 415–426, 2015.

[31] Bin Nie, Ji Xue, Saurabh Gupta, Tirthak Patel, Christian Engelmann, Evgenia Smirni, and Devesh Tiwari. Machine learning models for gpu error prediction in a large scale hpc system. In *DSN*, pages 95–106, 2018.

[32] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[33] Muddapu Parvathi, N Vasantha, and KSatya Parasad. Modified march c-algorithm for embedded memory testing. *International Journal of Electrical and Computer Engineering*, 2(5):571, 2012.

[34] Ayush Patwari, Ignacio Laguna, Martin Schulz, and Saurabh Bagchi. Understanding the spatial characteristics of dram errors in hpc clusters. In *ACM Workshop on Fault-Tolerance for HPC at Extreme Scale*, pages 17–22, 2017.

[35] Bianca Schroeder and Garth A Gibson. A large-scale study of failures in high-performance computing systems. *IEEE Transactions on Dependable and Secure Computing*, 7(4):337–350, 2009.

[36] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[37] Vilas Sridharan, Nathan DeBardeleben, Sean Blanchard, Kurt B Ferreira, Jon Stearley, John Shalf, and Sudhanva Gurumurthi. Memory errors in modern systems: The good, the bad, and the ugly. *ASPLOS*, 43(1):297–310, 2015.

[38] Vilas Sridharan and Dean Liberty. A study of dram failures in the field. In *SC*, pages 1–11, 2012.

[39] Dong Tang, Peter Carruthers, Zuheir Totari, and Michael W Shapiro. Assessment of the effect of memory page retirement on system ras against hardware faults. In *DSN*, pages 365–370, 2006.

[40] Raymond E Wright. Logistic regression. 1995.

[41] Jiang Xiao, Zhuang Xiong, Song Wu, Yusheng Yi, Hai Jin, and Kan Hu. Disk failure prediction in data centers via online learning. In *ICPP*, pages 1–10, 2018.

[42] Yong Xu, Kaixin Sui, Randolph Yao, Hongyu Zhang, Qingwei Lin, Yingnong Dang, Peng Li, Keceng Jiang, Wenchi Zhang, Jian-Guang Lou, et al. Improving service availability of cloud systems by predicting disk error. In *ATC*, pages 481–494, 2018.

[43] Yang Yang, Jing Dong, Chao Fang, Ping Xie, and Na An. Fp-ste: A novel node failure prediction method based on spatio-temporal feature extraction in data centers. *Computer Modeling in Engineering & Sciences*, 123(3):1015–1031, 2020.

[44] Shenglin Zhang, Ying Liu, Weibin Meng, Zhiling Luo, Jiahao Bu, Sen Yang, Peixian Liang, Dan Pei, Jun Xu, Yuzhi Zhang, et al. Prefix: Switch failure prediction in datacenter networks. *Proceedings of the ACM on Measurement and Analysis of Computing System*, 2(1):1–29, 2018.

[45] Darko Zivanovic, Pouya Esmaili Dokht, Sergi Moré, Javier Bartolome, Paul M Carpenter, Petar Radojković, and Eduard Ayguadé. Dram errors in the field: a statistical approach. In *MEMSYS*, pages 69–84, 2019.