Investigating Memory Failure Prediction Across CPU Architectures

Qiao Yu*†, Wengui Zhang‡, Min Zhou‡¶, Jialiang Yu‡, Zhenli Sheng‡, Jasmin Bogatinovski†
Jorge Cardoso*§ and Odej Kao†
*Huawei Munich Research Center, Germany †Technical University of Berlin, Germany
‡Huawei Technologies Co., Ltd, China §CISUC, University of Coimbra, Portugal
{qiao.yu, zhangwengui1, zhoumin27, yujialiang, shengzhenli, jorge.cardoso}@huawei.com
{jasmin.bogatinovski, odej.kao}@tu-berlin.de

***Abstract*—Large-scale datacenters often experience memory failures, where Uncorrectable Errors (UEs) highlight critical malfunction in Dual Inline Memory Modules (DIMMs). Existing approaches primarily utilize Correctable Errors (CEs) to predict UEs, yet they typically neglect how these errors vary between different CPU architectures, especially in terms of Error Correction Code (ECC) applicability. In this paper, we investigate the correlation between CEs and UEs across different CPU architectures, including X86 and ARM. Our analysis identifies unique patterns of memory failure associated with each processor platform. Leveraging Machine Learning (ML) techniques on production datasets, we conduct the memory failure prediction in different processors' platforms, achieving up to 15% improvements in F1-score compared to the existing algorithm. Finally, an MLOps (Machine Learning Operations) framework is provided to consistently improve the failure prediction in the production environment.**

***Index Terms*—Memory, Failure prediction, Uncorrectable error, Reliability, Machine Learning**

I. Introduction

With the expansion of cloud computing and big data services, the challenge of maintaining the Reliability, Availability, and Serviceability (RAS)[1] of servers is intensifying, due to memory failures, which represent a significant fraction of hardware malfunctions [1]–[3]. These failures often occur as Correctable Errors (CEs) and Uncorrectable Errors (UEs). To tackle these issues, Error Correction Code (ECC) mechanisms such as SEC-DED [4], Chipkill [5], and SDDC [6] are utilized to detect and correct errors. For instance, Chipkill ECC is capable of correcting all erroneous bits from a single DRAM (Dynamic Random Access Memory) chip. However, its efficacy diminishes when errors span multiple chips, leading to system failures caused by UEs. Furthermore, the ECC mechanisms on modern Intel platform servers do not offer the same level of protection as Chipkill ECC, making them vulnerable to certain error patterns originating from a single chip [7]. Therefore, relying exclusively on ECC mechanisms for memory reliability proves inadequate, as memory failures remain a prevalent source of system failures.

To enhance memory reliability, numerous studies [8]–[14] have delved into the correlations between memory errors and failures, laying the groundwork for our research. Machine Learning (ML) techniques have been employed for predicting memory failures [15]–[24], using CEs information from large-scale datacenters to forecast UEs. These investigations have effectively exploited the spatial distribution of CEs to improve memory failure prediction. Additionally, system-level workload metrics such as memory utilization, read/write access have been considered for memory failure prediction in [25]–[27]. Results from [27] indicate that workload metrics play a minor role compared to other CE related features. Research in [28] focuses on CE storms (a high frequency of CEs in a brief timeframe) and UEs to predict DRAM-caused node unavailability (DCNU), highlighting the significance of spatio-temporal features of CEs. Furthermore, [7] explores specific error bit patterns and their association with DRAM UEs, developing rule-based indicators for DRAM failure prediction that vary by manufacturer and part number, adapted to the ECC designs of modern Intel Skylake and Cascade Lake servers. Moreover, [29], [30] examine the distribution of error bits and propose a hierarchical, system-level method for predicting memory failures, leveraging error bit characteristics. *However, the incidence of UEs is influenced not just by DRAM faults but also by differences across CPU architectures, due to the diverse ECC mechanisms in use, which can alter the patterns of memory failures observed.* Understanding and modeling these failure patterns across various CPU platforms and ECC types is essential for accurate prediction of UEs. This gap in research motivates us to undertake the first study of DRAM failures comparing X86 and ARM systems, specifically the Intel Purley and Whitley platforms and the Huawei ARM K920 (anonymized to protect confidentiality) processor. By analyzing the relationship between UEs and fault patterns across these processor platforms, we aim to create targeted memory failure prediction algorithms. Additionally, we acknowledge the dynamic nature of server configurations, CPU architectures, memory types, and workloads. To address these variables, we introduce an MLOps framework designed to accommodate such changes, thereby continuously improving failure prediction throughout the lifecycle of the production environment.

We make the main contributions of this paper are below:

- We present the first memory failure study between X86

¶Corresponding author: zhoumin27@huawei.com

[1]Reliability, Availability, and Serviceability are the foundational pillars used to measure the dependability of computer systems.

- and ARM systems, specifically focusing on Intel X86 Purley and Whitley, as well as Huawei ARM K920 processor platforms, in large-scale datacenters. Different fault modes within DRAM hierarchy are associated with memory failures across these platforms.
- We develop ML-based algorithms for predicting memory failures, leveraging identified DRAM fault modes to anticipate UEs on these platforms.
- We establish an MLOps framework of failure prediction, to facilitate the collaboration across teams within the organization and help manage production ML algorithms lifecycle.

The organization of this paper is as follows: Section II provides the background of this work. Section III describes the dataset employed in our data analysis. Section IV details the problem formulation and performance metrics. Section V uncovers high-level fault modes within the DRAM hierarchy and their relationship to UEs across various platforms. Section VI demonstrates the use of machine learning techniques for memory failure prediction. Section VII introduces our MLOps framework for failure prediction. Related work is shown in Section VIII. Section IX concludes this paper.

II. Background

A. Terminology

A *fault* in DRAM acts as the root cause for an error, which may arise from a variety of sources, including particle impacts, cosmic rays, or manufacturing defects.

An *error* occurs when a DIMM sends incorrect data to the memory controller, deviating from what the ECC [4]–[6], [31] expects, indicative of an underlying fault. Memory errors, depending on the ECC's correction capacity, are classified into two main types: Correctable Errors (CEs) and Uncorrectable Errors (UEs). Two specific types of UEs are well described in [15]. 1) *sudden UEs*, which result from component malfunctions that immediately corrupt data, and 2) *predictable UEs*, which initially appear as CEs but evolve into UEs over time. Sudden UEs occur without prior CEs, whereas predictable UEs may be forecasted through CEs using algorithms designed for failure prediction. In this study, our focus is on predicting *predictable UEs*, as they constitute the majority of memory failures, described in Section III.

B. DRAM Organization and Access

Fig. 1 illustrates the memory's hierarchical layout and its CPU interactions. In Figure 1(1), it shows a DIMM rank made up of DRAM chips organized by banks, rows, and columns, where data moves from memory cells to the memory controller, which can generally detect and correct CEs. Figure 1(2) shows the data transmission of x4 DRAM DDR4 chips via Data Bus (DQs) upon CPU requests, involving 8 beats of 72 bits (64 data bits plus 8 ECC bits). Implementing the ECC, the memory controller detects and corrects them in Figure 1(3). Note that the exact ECC algorithms are highly confidential and never exposed and ECC checking bits addresses can be decoded to locate specific errors in DQs and beats. Finally, all these logs including Corrected and Uncorrected errors, events, and memory specifications are recorded in Baseboard Management Controller (BMC)[2].

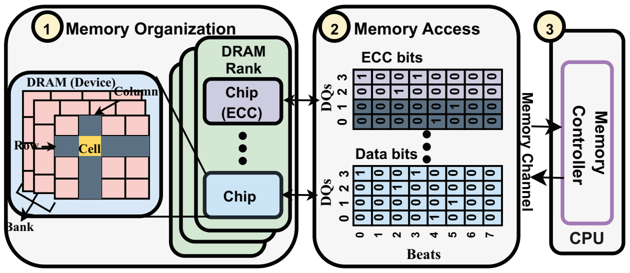

Fig. 1: Memory Organization.

C. Memory RAS Techniques

DRAM subsystems leverage RAS features for protection, including proactive VM migrations to minimize interruptions and CE storm[3] suppression to prevent service degradation. Advanced RAS techniques are designed to protect server-grade machines by avoiding faulty regions and employing sparing techniques like bit, row/column, and bank/chip sparing (e.g., Partial Cache Line Sparing (PCLS) [32], Post Package Repair (PPR) [33], Intel's Adaptive Double Device Data Correction (ADDDC) [34], [35]). Software-sparing mechanisms, such as the page offlining, mitigate memory errors [34], [36], [37]. However, these approaches may increase redundancy and overhead, affecting performance and limiting their universal applicability. Memory failure prediction plays a key role in foreseeing UEs and implementing specific mitigation strategies.

III. Dataset

Our dataset sourced from Huawei cloud datacenters, includes system configuration, Machine Check Exception (MCE) log, and memory events (CE storms, etc), focusing on DIMMs experiencing CEs and omitting those with sudden UEs due to the lack of predictive data. We examined error logs from approximately 250,000 servers across Intel Purley and Whitley platforms (including Skylake, Cascade Lake, and Icelake) as well as the Huawei K920 processor platform.

Table I describes an overview of our data, which includes over 90,000 DDR4 DIMMs from various manufacturers, spanning different CPU architectures, with CEs recorded from January to October 2023. Within Intel platforms, predictable UEs constitute 73% of the UEs on the Purley platform, surpassing the rate of sudden UEs. In contrast, the Whitley platform shows a higher incidence of sudden UEs than Purley, despite it having a lower total UE rate compared to Purley. Meanwhile, in ARM system with K920 processor platform, there's a significant predominance of predictable UEs over sudden UEs, showcasing a variance in ratios compared to the X86 systems, the overall rate of UEs in the K920 dataset is less than that of the Intel platforms. Note that these statistics are specific

[2]BMC is a specialized processor built into the server's motherboard, designed to supervise the physical status of computers, network servers, and additional hardware components.

[3]CE interruptions repeatedly occur multiple times, e.g., 10 times.

TABLE I: Description of Dataset.

CPU Platform	DIMMs with CEs	DIMMs with UEs	Predictable UE DIMMs in %	Sudden UE DIMMs in %
Intel Purley	$> 50{,}000$	$> 2{,}000$	73%	27%
Intel Whitley	$> 10{,}000$	> 400	42%	58%
K920	$> 30{,}000$	> 600	82%	18%

to the datasets analyzed and the observed variations may be influenced by several factors, such as workload, server age, and distinct RAS mechanisms, etc. In particular, the ECC used in contemporary Intel platforms, which is integral for error correction and detection, is considered weaker than Chipkill. This is partly because some of the extra bits previously used by Intel ECCs are reallocated for other uses [7], such as to store ownership, security information, to mark failed areas of DRAM, etc. This suggests that the observed discrepancies in UE rates across different architectures could stem from the unique ECC mechanisms employed.

Finding 1. The UE and sudden UE rates show variation between X86 and ARM systems. This discrepancy could be attributed to the distinct ECC mechanisms implemented within these differing architectures.

IV. Problem Formulation and Performance Measures

The problem of predicting memory failures is approached as a binary classification task, following the methodology in [29], [30]. As illustrated in Figure 3, an algorithm at time t uses data from a historical *observation window* $\triangle t_d$ to predict failures within a future prediction period $[t + \triangle t_l, t + \triangle t_l + \triangle t_p]$, where $\triangle t_l$ represents the lead time [38] before a failure occurs, and $\triangle t_p$ is the duration of the prediction window. Event samples are recorded at intervals of $\triangle i_s$ (e.g., CE events are logged every minute), and predictions are made at intervals of $\triangle i_p$ (every 5 minutes). The observation ($\triangle t_d$) and *prediction validation windows* ($\triangle t_p$) are set to 5 days and 30 days, respectively, to facilitate early proactive strategies. Note that these parameters were optimized based on empirical data from the production environment. The *lead prediction time* $\triangle t_l \in (0, 3h]$, ranging up to 3 hours, designed to specific operational scenarios. A True Positive (TP) denotes an correctly predicted failure within the window. A False Positive (FP) represents an incorrect forecast. A False Negative (FN) describes a failure that happens without an earlier warning, and a True Negative (TN) is identified when no failures are anticipated or take place. The performance of the algorithm is evaluated using $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$ and $F1 = \frac{2 \times Precision \times Recall}{Precision+Recall}$.

VM Interruption Reduction Rate (VIRR). Prior works [7], [18], [20], [28]–[30] have introduced cost-aware models to assess the benefits of memory failure prediction. In this work, we emphasizes *VM Interruption Reduction Rate (VIRR)* [29] in Figure 2, as it more precisely reflects the effects on customer experience.

Understanding the VIRR involves considering V_a as the average number of VMs per server. Without predictive capabilities, the interruptions are calculated as ❶ $V = V_a(TP+FN)$

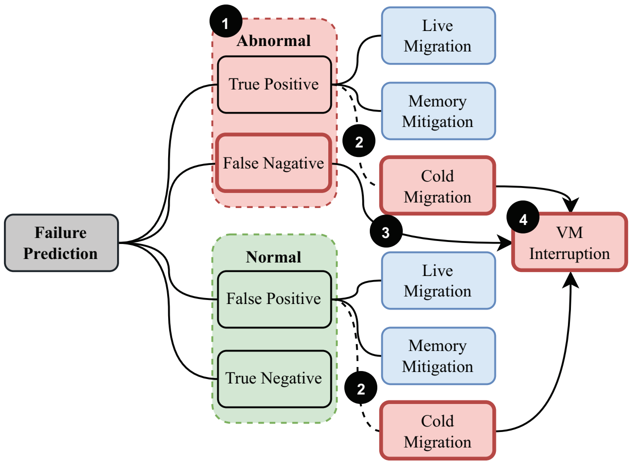

Fig. 2: VM Interruption under Failure Prediction.

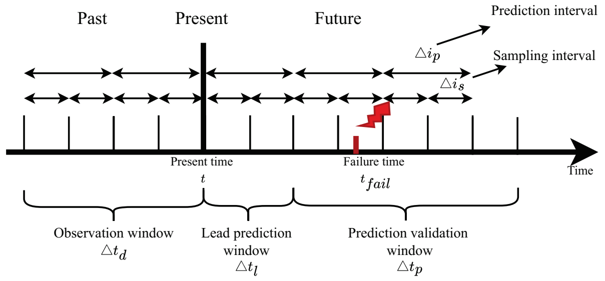

Fig. 3: Failure prediction problem definition [29].

as seen in the **Abnormal** of Figure 2. While proactive VM live migrations and memory mitigation techniques aim to minimize interruptions without service interruption, a notable fraction of VMs might still undergo cold migration, typically causing VM interruptions. Cold migrations are often the fallback when live migrations or memory mitigation are unfeasible, due to limited resources or unexpected failures, and are a common approach for VM reallocation and maintenance. The fraction of VMs under such migrations is denoted as y_c. As a result in Figure 2, we define ❷ $V_1' = V_a \cdot y_c(TP+FP)$ as the volume of interruptions from cold migrations triggered by accurate failure predictions (TP + FP). On the other side, missed failure predictions lead to increased interruptions, represented by ❸ $V_2' = V_a \cdot FN$. Considering the prediction algorithm, the total interruptions are ❹ $V' = V_1' + V_2'$. The VIRR formula is thus: $VIRR = \frac{V-V'}{V}$, simplifying to $(1 - \frac{y_c}{precision}) \cdot recall$, according to [29].

In practical production settings, y_c remains a positive value since VMs may need cold migrated due to the failure of live migration or memory mitigation. When a model's precision falls below the percentage of cold migration ($precision < y_c$), the VIRR shifts to negative, indicating an increase in VM interruptions. Conversely, high-precision models achieve positive VIRR, amplified by their recall rate. Based on our observations, we've conservatively set $y_c = 0.1$ for our evaluation. Note that this value is already pessimistic, anticipating

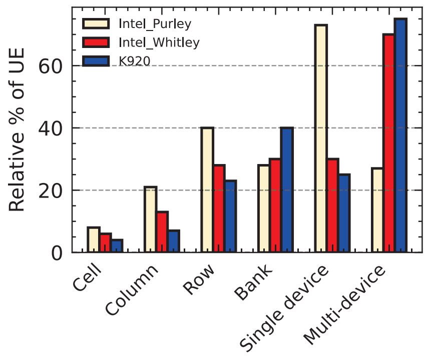

Fig. 4: Relative % of UE.

a reduction in y_c as cloud infrastructure evolves and expands.

V. Fault Analysis

Our analysis investigates the high-level fault modes in the DRAM hierarchy, and correlates them with UE rates across various platforms. We consider faults at the DRAM-level, including cell, column, row, bank faults as illustrated in Figure 1(1). A **Cell fault** occurs when CEs in a cell surpass a set threshold, while **Row** and **Column faults** are identified by exceeding thresholds across a row or column, respectively. **Bank faults** arise when thresholds for both row and column faults within a bank are exceeded. Additionally, when CEs affect a single device, this constitutes a **Single-device fault**. In contrast, if CEs extend across multiple devices, it is **Multi-device fault**. Further details on fault definitions and threshold settings can be found in [12], [29], [30]. The approach to calculating the relative UE rate depicted in Figure 4 follows previous studies [9], [11], [21], [29], [30], categorizing DIMMs according to distinct fault types (e.g., cell faults) and assessing the percentage of DIMMs that encounter UEs.

As shown in Fig. 4, the most UEs are attributed to faults in higher-level components, such as row and bank faults across all platforms. Specifically, on the Intel Purley platform, the primary source of UEs is single device faults. Conversely, in both the Whitley and K920 platforms, UEs predominantly arise from multi-device faults, possibly due to variations in ECC mechanisms.

Finding 2. The Intel Purley platform primarily experiences UEs due to single device faults, a trend that appears to diminish in the Whitley platform. Meanwhile, the K920 platform exhibits fewer single device faults, potentially attributed to the efficiency of its K920-SDDC.

Then we investigate the failure patterns of error bits in DQs and beats, similar to [30]. As shown in Fig. 5, we examined the counts and intervals of error DQs and beats in x4 bit width DRAM. On the Intel Purley platform, 2 error DQs and beats counts and a 4-beat interval are associated with significantly higher UE rates, in comparison to other error DQ and beat counts and intervals. By contrast, the Intel Whitley platform exhibited higher UE rates with 4 error DQs and 5 error beats counts. However, variations in UE rates were not significantly influenced by the intervals between DQs and beats from our observations. Thus, the failure patterns on Intel's more advanced Whitley platform are markedly different from those observed on the Purley platform.

Finding 3. At the bit-level within Intel architecture, distinct DQ and beat patterns emerge, highlighting the importance of formulating failure patterns designed to specific platforms due to the potential variations in their underlying ECC mechanisms.

VI. Failure Prediction

We develop our failure prediction using tree-based algorithms (Random Forest and LightGBM [28]–[30]) and deep learning FT-Transformer [39]. These ensemble learning techniques have been prevalently utilized in previous memory failure prediction literature [15], [18]–[20], [27], with the FT-Transformer considered as the leading algorithm for handling tabular data in the field of deep learning. The experimental design and feature generation follow the methodology in [30], which categorizes samples into two classes: Positive and Negative. DIMMs expected to experience at least one UE within the prediction window are categorized as **Positive**, while those expected not to have any UE are classified **Negative**. Samples labeling is based on the time interval between a CE and its subsequent UE, with specifics on interval settings available in [29], [30]. Features used in our models include DRAM characteristics such as manufacturer, data width, frequency, chip process, CE error rate, our conducted failure analysis, and memory events. The performance of these algorithms was evaluated using precision, recall, F1-score and VIRR as described in Section IV.

Comparison with the existing approach. We compare our algorithms with the existing reproduced Risky CE Pattern approach in [7], particularly for the Intel Skylake/Cascade (Purley platform) architecture. However, we noted a lack of dedicated memory failure prediction algorithms for the Intel Whitley platform and the Huawei ARM K920 Platform. Table II shows the superior performance of our method, achieving the high F1-score of 0.64 and VIRR of 0.65 using LightGBM on the Intel Purley platform, outperforming rule-based risky CE pattern algorithm. Additionally, it scores 0.50 F1-score on the Intel Whitley platform using the FT-Transformer, and 0.54 F1-score and 0.46 VIRR in K920 architecture with LightGBM. The LightGBM results overall outperformed than other machine learning methods including deep learning FT-Transformer algorithm, which agrees with the finding in [40].

Finding 4. Prediction efficacy varies across platforms; the Intel Whitley platform demonstrates comparatively weaker predictive performance than both the Intel Purley and K920 platform.

VII. MLOps of Failure Prediction

After developing machine learning (ML) algorithms that accurately predict memory failures, it becomes crucial to both maintain and enhance these algorithms and to automate their operation within the data center. The MLOps framework

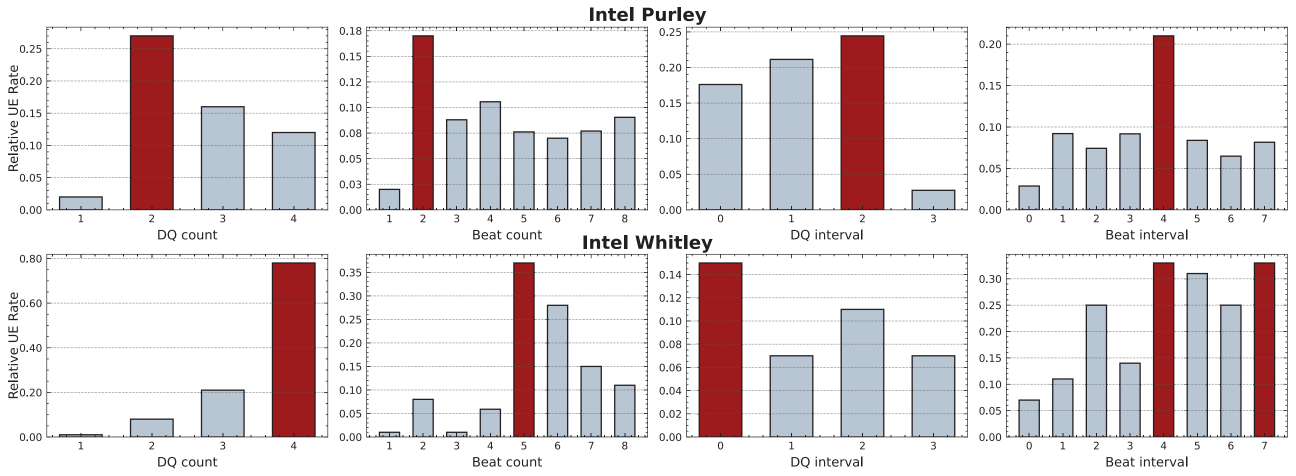

Fig. 5: Analyses of Error Bits in Intel Platforms: Highlighting The Highest Rate with Red Bar.

TABLE II: Overview of Algorithm Performance Comparisons. (X denotes the absence of prediction values)

Algorithm	Intel Purley				Intel Whitley				K920			
	Precision	Recall	F1	VIRR	Precision	Recall	F1	VIRR	Precision	Recall	F1	VIRR
Risky CE Pattern [7]	0.53	0.46	0.49	0.37	X	X	X	X	X	X	X	X
Random forest	0.61	0.62	0.61	0.52	0.34	0.46	0.39	0.32	0.44	0.51	0.47	0.39
LightGBM	0.54	0.80	**0.64**	**0.65**	0.46	0.54	0.49	**0.45**	0.51	0.57	**0.54**	**0.46**
FT-Transformer	0.49	0.74	0.59	0.58	0.53	0.49	**0.50**	0.40	0.40	0.54	0.46	0.41

is ideally suited for this purpose, ensuring the continuous accuracy and applicable of our memory failure prediction algorithms. Figure 6 illustrates an overview of the MLOps framework for memory failure prediction, with the workflow introduced in stages as follows:

Data Pipeline: The initial stage involves collecting raw data from various sources. For example, CE and UE logs are collected by the BMC, and are then processed and stored in the Data Lake, alongside other data sources such as runtime workload metrics (e.g., CPU utilizations) and environmental metrics (e.g., server locations, temperatures) using the Huawei Data Lake Insight (DLI) solution.

Feature Store: A feature store acts as a centralized repository for transforming, storing, cataloging, and serving features used in model training and inference. It ensures consistency between training and prediction phases and accelerate the development of machine learning models by making features readily accessible. This involves:

- **Transformation**: Converting raw data into features suitable for machine learning algorithms. This process is divided into batch and stream transformations for model training and online prediction, respectively.
- **Storage**: Once transformed, features are stored in an accessible format for training and prediction. They are cataloged with registry information to standardize features across all teams' models. For example, CEs are converted into temporal and spatial features within the DRAM hierarchy. This conversion includes the distribution of error bits across DQs and beats, the number of faults, within different time intervals (1min, 1h, 5d, etc) and memory configurations, such as manufacturers, DRAM processes are further encoded to static features.
- **Serving**: Feature store serves features for model training and inference, enabling Data Scientists to select features on demand for training different models based on varying requirements. For instance, Data Scientists might develop various models designed to distinct CPU architectures, utilizing unique features for each.

ML Deployment: This phase involves (1) **model training**, which contains algorithms selection, hyperparameters tuning, and the application of these configured algorithms on prepared datasets. This task can be performed manually by Data Scientists or through automated tools like AutoML. Once models are trained, and show substantial improvements in predefined benchmark evaluations, they advance to deployment in the production environment. This deployment leverages a (2) **Continuous Integration and Continuous Delivery (CI/CD) pipeline**, which automates the integration, testing, and deployment of ML algorithms, thereby ensuring models can be consistently updated and reliably released within the production environment. Subsequently, the successfully deployed models continue delivering (3) **online prediction** utilizing streaming data and the prediction results will be served to our cloud service.

Cloud Service: The alarm is triggered in the **Cloud Alarm System** upon predicting memory failures. Depending on different use cases, the **memory RAS techniques** are then implemented to mitigate the failures, with VMs being migrated automatically or manually as required.

Monitoring: Throughout the MLOps workflow, each phase is continuously monitored through dashboards. This includes monitoring data collection rates, feature importance, and algorithm performance, as well as VM migrations and service interruptions. To enhance algorithm accuracy and ensure fair-

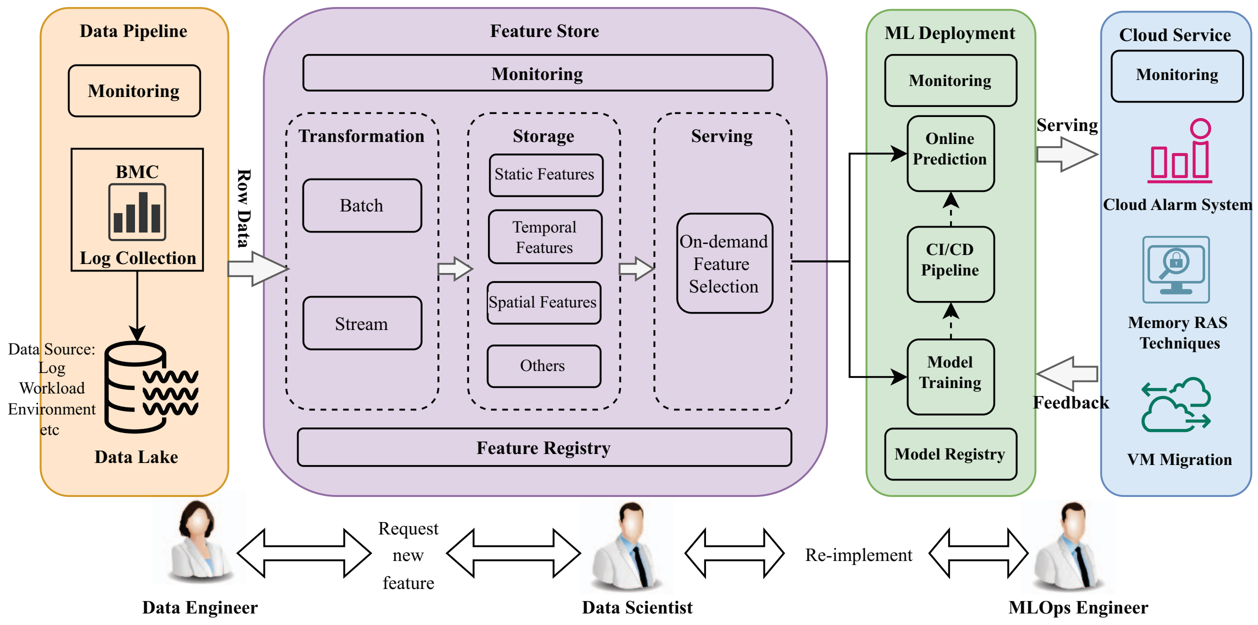

Fig. 6: The MLOps Framework of Failure Prediction.

ness in predictions, feedback is proactively gathered from the cloud service. Dashboards are implemented in both testing and production settings to closely observe algorithm performance, as well as the rates of false negatives and positives. This dual-environment monitoring facilitates the ongoing refinement of algorithms.

In our memory failure prediction development, collaboration across various teams is essential, this collaborative effort spans from Data Engineers, who are responsible for processing new data and integrating it into the Data Lake in response to Data Scientists' requests. Data Scientists analyze this data, develop predictive algorithms, and specify requirements for operational deployment. To the MLOps Engineers, who take on the critical role of re-implementing and deploying newly developed algorithms by Data Scientists into the production environment.

VIII. Related Work

Empirical research [8]–[12] has laid the groundwork in the study of memory errors, focusing on correlation analyses and failure modes. Their works serves as foundational elements for developing memory failure prediction algorithm. This section highlights significant contributions in memory failure prediction.

The ensemble learning approaches [15], [17], [18], [20] have constantly improved memory failure prediction by leveraging correctable errors, event logs, sensor metrics. The node/server-level memory unavailability prediction methods are proposed in [21], [28], considering both UE and CE storm/CE-driven prediction. Li et al. [7] explored correlations between CEs and UEs using error bit information and DIMM part numbers, creating a new risky CE indicator for UE prediction across different manufacturers and part numbers. Peng et al. [23] designed DRAM failure forecasters by utilizing different UCE types. Yu et al. [29], [30] further examined the distribution of error bits and proposed a hierarchical, system-level approach for predicting memory failures by utilizing the error bits features.

However, the literature mentioned does not examine failure patterns across various processors' platforms, nor does it engage in the development of ML models specifically designed for distinct CPU architectures to improve prediction. In our previous work [41], we explored memory failure patterns across various CPU architectures. In this extended version, we further expand the fault analysis, present 4 findings, and establish an MLOps framework to continuously improve failure prediction models throughout their lifecycles.

IX. Conclusion

We present the first comprehensive analysis of DRAM failures spanning both X86 and ARM systems across various platforms in large-scale datacenters. From our analytical and predictive modeling work, we report 4 findings: 1) UE and sudden UE rates differ between X86 and ARM systems. 2) Fault modes vary across architectures. 3) Bit-level failure patterns of DRAM are architecture-dependent. 4) Prediction accuracy differs by platforms. Utilizing datasets from production environment, our approach showcased a 15% enhancement in F1-score compared to the method in [7], specifically within the Intel Purley platform. Moreover, we executed initial experiments on the Intel Whitley and ARM-based platforms, achieving F1-scores of 0.50 and 0.54, along with VIRR of 0.45 and 0.46 respectively. Finally, we present our MLOps framework for memory failure prediction, implemented in the production environment, This framework is designed to ensure the continuous enhancement and maintenance of failure prediction performance.

Acknowledgement

We thank the anonymous reviewers from DSN'24 for their constructive comments.

REFERENCES

[1] "Intel MCA+MFP Helps JD Stable and Efficient Cloud Services." [Online]. Available: https://www.intel.com/content/dam/www/central-libraries/us/en/documents/2023-12/mca-mfp-helps-jd-stable-and-efficient-cloud-services.pdf

[2] G. Wang, L. Zhang, and W. Xu, "What can we learn from four years of data center hardware failures?" in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2017, pp. 25–36.

[3] P. Notaro, Q. Yu, S. Haeri, J. Cardoso, and M. Gerndt, "An optical transceiver reliability study based on sfp monitoring and os-level metric data," in *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2023, pp. 1–12.

[4] M. Y. Hsiao, "A class of optimal minimum odd-weight-column sec-ded codes," *IBM Journal of Research and Development*, 1970.

[5] T. J. Dell, "A white paper on the benefits of chipkill correct ecc for pcserver main memory," in *Computer Science*, 1997. [Online]. Available: "https://asset-pdf.scinapse.io/prod/48011110/48011110.pdf"

[6] "Intel® e7500 chipset mch intel® x4 single device data correction (x4 sddc) implementation and validation." [Online]. Available: https://www.intel.com/content/dam/doc/application-note/e7500-chipset-mch-x4-single-device-data-correction-note.pdf

[7] C. Li, Y. Zhang, J. Wang, H. Chen, X. Liu, T. Huang, L. Peng, S. Zhou, L. Wang, and S. Ge, "From correctable memory errors to uncorrectable memory errors: What error bits tell," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '22. IEEE Press, 2022.

[8] B. Schroeder, E. Pinheiro, and W.-D. Weber, "Dram errors in the wild: A large-scale field study," in *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 193–204. [Online]. Available: https://doi.org/10.1145/1555349.1555372

[9] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, "Revisiting memory errors in large-scale production data centers: Analysis and modeling of new trends from the field," in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Rio de Janeiro, Brazil: IEEE, 2015, pp. 415–426.

[10] V. Sridharan and D. Liberty, "A study of dram failures in the field," in *SC '12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, 2012, pp. 1–11.

[11] 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*, ser. ASPLOS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 297–310. [Online]. Available: https://doi.org/10.1145/2694344.2694348

[12] M. V. Beigi, Y. Cao, S. Gurumurthi, C. Recchia, A. Walton, and V. Sridharan, "A systematic study of ddr4 dram faults in the field," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 991–1002.

[13] M. Patel, T. Shahroodi, A. Manglik, A. G. Yaglikci, A. Olgun, H. Luo, and O. Mutlu, "A case for transparent reliability in dram systems," 2022.

[14] J. Jung and M. Erez, "Predicting future-system reliability with a component-level dram fault model," in *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 944–956. [Online]. Available: https://doi.org/10.1145/3613424.3614294

[15] I. Giurgiu, J. Szabo, D. Wiesmann, and J. Bird, "Predicting DRAM reliability in the field with machine learning," in *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Industrial Track*, ser. Middleware '17. New York, NY, USA: Association for Computing Machinery, Dec. 2017, pp. 15–21, 00026. [Online]. Available: https://doi.org/10.1145/3154448.3154451

[16] X. Du and C. Li, "Memory failure prediction using online learning," in *Proceedings of the International Symposium on Memory Systems*, ser. MEMSYS '18. New York, NY, USA: Association for Computing Machinery, Oct. 2018, pp. 38–49, 00000. [Online]. Available: https://doi.org/10.1145/3240302.3240309

[17] X. Du, C. Li, S. Zhou, M. Ye, and J. Li, "Predicting Uncorrectable Memory Errors for Proactive Replacement: An Empirical Study on Large-Scale Field Data," in *2020 16th European Dependable Computing Conference (EDCC)*, Sep. 2020, pp. 41–46, 00003 ISSN: 2641-810X.

[18] I. Boixaderas, D. Zivanovic, S. Moré, J. Bartolome, D. Vicente, M. Casas, P. M. Carpenter, P. Radojković, and E. Ayguadé, "Cost-aware prediction of uncorrected DRAM errors in the field," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '20. Atlanta, Georgia: IEEE Press, Nov. 2020, pp. 1–15, 00005.

[19] F. Yu, H. Xu, S. Jian, C. Huang, Y. Wang, and Z. Wu, "Dram failure prediction in large-scale data centers," in *2021 IEEE International Conference on Joint Cloud Computing (JCC)*. Los Alamitos, CA, USA: IEEE Computer Society, aug 2021, pp. 1–8. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/JCC53141.2021.00012

[20] X. Du and C. Li, "Predicting uncorrectable memory errors from the correctable error history: No free predictors in the field," in *The International Symposium on Memory Systems*, ser. MEMSYS 2021. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3488423.3519316

[21] Z. Cheng, S. Han, P. P. C. Lee, X. Li, J. Liu, and Z. Li, "An in-depth correlative study between dram errors and server failures in production data centers," in *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*, 2022, pp. 262–272.

[22] J. Bogatinovski, O. Kao, Q. Yu, and J. Cardoso, "First ce matters: On the importance of long term properties on memory failure prediction," in *2022 IEEE International Conference on Big Data (Big Data)*, 2022, pp. 4733–4736.

[23] X. Peng, Z. Huang, A. Cantrell, B. H. Shu, K. K. Xie, Y. Li, Y. Li, L. Jiang, Q. Xu, and M.-C. Yang, "Expert: Exploiting dram error types to improve the effective forecasting coverage in the field," in *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, 2023, pp. 35–41.

[24] Z. Liu, C. Benge, Y. Dagli, and S. Jiang, "Stim: Predicting memory uncorrectable errors with spatio-temporal transformer."

[25] X. Sun, K. Chakrabarty, R. Huang, Y. Chen, B. Zhao, H. Cao, Y. Han, X. Liang, and L. Jiang, "System-level hardware failure prediction using deep learning," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, Jun. 2019, pp. 1–6, iSSN: 0738-100X.

[26] L. Mukhanov, K. Tovletoglou, H. Vandierendonck, D. S. Nikolopoulos, and G. Karakonstantis, "Workload-aware dram error prediction using machine learning," in *2019 IEEE International Symposium on Workload Characterization (IISWC)*, 2019, pp. 106–118.

[27] X. Wang, Y. Li, Y. Chen, S. Wang, Y. Du, C. He, Y. Zhang, P. Chen, X. Li, W. Song, Q. xu, and L. Jiang, "On workload-aware dram failure prediction in large-scale data centers," in *2021 IEEE 39th VLSI Test Symposium (VTS)*, 2021, pp. 1–6.

[28] P. Zhang, Y. Wang, X. Ma, Y. Xu, B. Yao, X. Zheng, and L. Jiang, "Predicting dram-caused node unavailability in hyper-scale clouds," in *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2022, pp. 275–286.

[29] Q. Yu, W. Zhang, P. Notaro, S. Haeri, J. Cardoso, and O. Kao, "Himfp: Hierarchical intelligent memory failure prediction for cloud service reliability," in *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2023, pp. 216–228.

[30] Q. Yu, W. Zhang, J. Cardoso, and O. Kao, "Exploring error bits for memory failure prediction: An in-depth correlative study," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, Oct. 2023. [Online]. Available: http://dx.doi.org/10.1109/ICCAD57390.2023.10323692

[31] "Memory ras configuration user's guide." [Online]. Available: https://www.supermicro.com/manuals/other/Memory_RAS_Configuration_User_Guide.pdf

[32] X. Du and C. Li, "Dpcls: Improving partial cache line sparing with dynamics for memory error prevention," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*, 2020, pp. 197–204.

[33] C.-S. Hou, Y.-X. Chen, J.-F. Li, C.-Y. Lo, D.-M. Kwai, and Y.-F. Chou, "A built-in self-repair scheme for drams with spare rows, columns, and bits," in *2016 IEEE International Test Conference (ITC)*, 2016, pp. 1–7.

[34] X. Du, C. Li, S. Zhou, X. Liu, X. Xu, T. Wang, and S. Ge, "Fault-aware prediction-guided page offlining for uncorrectable memory error prevention," in *2021 IEEE 39th International Conference on Computer Design (ICCD)*, 2021, pp. 456–463.

[35] X. Jian and R. Kumar, "Adaptive reliability chipkill correct (arcc)," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, 2013, pp. 270–281.

[36] X. Du and C. Li, “Combining error statistics with failure prediction in memory page offlining,” in *Proceedings of the International Symposium on Memory Systems*, ser. MEMSYS ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 127–132. [Online]. Available: https://doi.org/10.1145/3357526.3357527

[37] D. Tang, P. Carruthers, Z. Totari, and M. Shapiro, “Assessment of the effect of memory page retirement on system ras against hardware faults,” in *International Conference on Dependable Systems and Networks (DSN’06)*, 2006, pp. 365–370.

[38] K. A. Alharthi, A. Jhumka, S. Di, L. Gui, F. Cappello, and S. McIntosh-Smith, “Time machine: Generative real-time model for failure (and lead time) prediction in hpc systems,” in *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2023, pp. 508–521.

[39] Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko, “Revisiting deep learning models for tabular data,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 932–18 943, 2021.

[40] L. Grinsztajn, E. Oyallon, and G. Varoquaux, “Why do tree-based models still outperform deep learning on typical tabular data?” *Advances in Neural Information Processing Systems*, vol. 35, pp. 507–520, 2022.

[41] Q. Yu, J. Cardoso, and O. Kao, “Unveiling dram failures across different cpu architectures in large-scale datacenters,” in *2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2024, pp. 1–2, accepted.