2025 55th Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S) | 979-8-3315-1203-3/25/$31.00 ©2025 IEEE | DOI: 10.1109/DSN-S65789.2025.00045

Cordial: Cross-row Failure Prediction Method Based on Bank-level Error Locality for HBMs

Wenwei Gu*, Jiazhen Gu†§, Renyi Zhong*, Wenyu Zhang†, Ming Li†, and Michael R. Lyu*
*The Chinese University of Hong Kong, China. Email: {wwgu21, ryzhong22, lyu}@cse.cuhk.edu.hk
†Huawei Hong Kong Research Center, China. Email: {gu.jiazhen, epic.zhangwenyu, liming377}@huawei.com

***Abstract*—High Bandwidth Memory (HBM) is a promising solution for overcoming memory bottlenecks in high-performance computing, but it remains susceptible to memory errors. Our empirical study on high-performance computing platforms that serve large AI model training with over 80,000 HBMs, reveals that HBM errors have a high burst rate. As a result, conventional failure prediction methods that heavily rely on historical error data are ineffective. Through investigating the locality of uncorrectable errors (UCEs) of HBMs, we found that failure patterns at the bank level primarily exhibit aggregation tendencies, suggesting that errors in neighboring rows are often related. Based on these insights, we propose Cordial, a cross-row failure prediction method based on bank-level error locality. Cordial adopts a hierarchical approach: it first utilizes bank-level error information to predict the bank-level failure pattern. Our approach classifies bank-level failure patterns into three categories: double-row clustering, single-row clustering (these two are aggregation patterns), and scattered patterns. Then, it leverages spatial locality to guide cross-row predictions for aggregation patterns. Evaluation results show our method improves the F1-score by up to 90.7% and enhances the isolation coverage rate by 47.1%, demonstrating its practical applicability.**

***Index Terms*—High bandwidth memory; Memory failure prediction; Memory system reliability.**

I. Introduction

With the emergence of large language models (LLMs), *e.g.,* GPT and LLaMA, there has been a considerable increase in the scale of ultra-massive datasets, leading to a growing demand for rapid computation [1]. There is an exponentially growing trend of AI accelerators, *e.g.,* neural-network processing units (NPU), that focus on increasing computing performance [2]. However, the growing exponent for GPUs or NPUs is substantially larger than that for dynamic random-access memories (DRAMs) [3]. Thus, the latency of data movement between memory and AI accelerators is becoming the bottleneck in large-scale AI model training [4]. The increasing gap between the computing power and the memory bandwidth is known as the memory wall, especially in modern LLM training systems [5]. Several studies have delved into mitigating the impact of the memory wall, including utilizing bandwidth-effective DRAM cache [6], software [7], [8] or hardware prefetching [9], [10], and designing new system architectures like processing-in-memory architecture [11], [12]. Recently, High bandwidth memory (HBM) has gained tremendous attention as a promising solution to alleviate memory bottlenecks fundamentally [1], [13] and established itself as the memory solution for high-performance computing workloads like LLM training.

In modern LLM training systems, infrastructure failures are inevitable during the training process [14], which may cause devastating consequences. Specifically, in large-scale model training clusters with more than 10,000 NPUs, system failure or slowdown can significantly impede training efficiency and waste expensive NPU time, resulting in large revenue loss [15]–[17]. Hardware failures, especially memory failures, are among the most significant reasons for training job crashes or slowdowns [18], [19]. Unfortunately, due to its stacking structure, HBMs not only exhibit the errors of DRAM but also suffer from new errors, *e.g.,* TSV faults. Besides, conventional error correction codes (ECC) are insufficient to correct malfunctions of sub-wordline drivers (SWDs) in HBMs [20], a primary cause of errors, making HBM even more vulnerable to unexpected errors. Thus, it is of great significance to proactively predict HBM failures to ensure the reliability of the high-performance computing platform that serves for AI model training.

To mitigate HBM failures, isolating faulty regions is a common strategy to prevent continuous error propagation. Various hardware-sparing mechanisms exist at different micro-levels, like row sparing. Bank sparing is another mechanism, but it requires significantly higher hardware redundancy and incurs greater costs. Additionally, software-sparing mechanisms like page offlining in operating systems can be employed to avoid memory errors. Existing studies [21] highlight the importance of applying different correction strategies depending on fault rates, as interruptions during data copying can sometimes result in unsuccessful recovery when pages are locked. Thus, selecting appropriate recovery techniques is essential for maintaining system reliability.

Row-level failure prediction emerges as a more practical solution in industrial scenarios due to its lower cost and more targeted approach. Existing HBM failure prediction frameworks, *e.g.,* Calchas [5], leverage hierarchical models that incorporate spatial, temporal, and sensor information from various device levels to anticipate failures. However, our analyses and validations indicate two main limitations that hinder their industrial application. Firstly, predicting failures without recommending corresponding mitigation strategies limits the actionable insights provided and does not fully align with practical industrial requirements. Secondly, these methods rely on historical error information to predict future errors in

§ Jiazhen Gu is the corresponding author.

2833-292X/25/$31.00 ©2025 IEEE
DOI 10.1109/DSN-S65789.2025.00045

the same rows, *i.e.,* in-row failure prediction. This can be ineffective in scenarios with high sudden error rates (*i.e.,* 95.61% in row-level) where useful historical data is lacking, which is precisely the challenge we face in our industrial scenarios.

In this paper, we introduce a new paradigm of cross-row failure prediction, which contrasts with the existing in-row failure prediction approach. Instead of predicting errors within the same row where historical error information may often be missing, cross-row prediction allows us to predict error rows based on error information from other rows. This is particularly advantageous in scenarios with high sudden error rates. We conducted a comprehensive empirical study that revealed a significant practical challenge: the high ratio of sudden errors in HBMs within the high-performance computing clusters renders existing in-row prediction methods ineffective. We further examine bank-level failure patterns, discovering that aggregation (clustering) patterns predominate. Additionally, our analysis of error locality indicates the feasibility of predicting cross-row errors in neighboring areas. Building on these insights, we propose the Cross-row Failure Prediction Method Based on Bank-level Error Locality (Cordial). Our framework begins by classifying bank-level failure patterns into three categories: double-row clustering, single-row clustering, and scattered patterns. For the first two patterns, we apply row-sparing techniques, while bank-sparing is used for the scattered pattern due to its extensive error distribution, which targets the first challenge. Leveraging the locality of errors in aggregation patterns, we perform cross-row failure prediction for the double-row clustering and single-row clustering patterns in the neighboring areas of current error rows, which effectively addresses the second challenge.

To sum up, we make the following contributions:

- We conduct a comprehensive empirical study focusing on the sudden error ratio, failure patterns at the bank level, and the locality of error rows using a large-scale dataset from real-world high-performance computing platforms that serve for LLM training, comprising over 10,000 NPUs and 80,000 HBMs.
- We propose the Cross-row Failure Prediction Method Based on Bank-level Error Locality, bridging the gap left by existing methods that struggle with sudden errors in the absence of historical error data. To the best of our knowledge, this is the first work that falls into the cross-row failure prediction paradigm.
- We evaluate our framework using an industrial dataset from large-scale, high-performance computing clusters. Our approach achieves up to a 90.7% improvement in F1-score and a 47.1% improvement in isolation coverage rate, demonstrating its superior performance in industrial settings.

II. Background

In this section, we provide an overview of the HBM organization and introduce the error types in HBM. Then, we introduce cross-row failure prediction, which is different from existing in-row prediction.

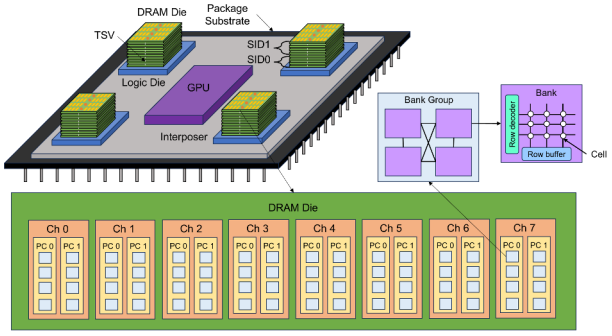


Fig. 1. The HBM2E Architecture with 8Hi stacks

A. HBM Organization

Unlike traditional planar memory architectures such as DRAM, HBM features a 3D-stacked design composed of multiple DRAM dies. This structure enables HBM to provide substantially higher data transfer rates. For instance, HBM2 can achieve a bandwidth of approximately 256 GB/s per stack [22], and this bandwidth is further increased to 512 GB/s in HBM2E [1]. In common LLM training platform scenarios, each compute node is equipped with 8 NPUs, with each NPU featuring two sockets for installing HBMs. HBMs are constructed via an 8Hi stack (with eight DRAM dies) [23]. Every four dies are packed together, forming two stack IDs (SIDs). A DRAM die in the stack typically consists of 8 channels (CH), while a channel can be further divided into two pseudo-channels (PS-CH). Each pseudo-channel is further constituted of 4 bank groups (BG), and four banks are distributed in each group. The banks are two-dimensional arrays consisting of cells indexed by rows and columns. Each cell contains multiple bits. Figure 1 illustrates the overall architecture of HBM. The fundamental structure of HBM is composed of a logic die at the bottom and stacked DRAM dies [24]. The logic die with I/O buffers serves as the control and communication module for the entire HBM [25]. The DRAM dies are connected by through-silicon vias (TSVs) and micro-bumps, which are vertical electrical connections that run through the silicon substrate, to offer high bandwidth pathways [5].

B. HBM Errors

We refer to HBM error as the situation in which an HBM delivers data to the memory controller that is inconsistent with the original data through the ECC [26]. According to the number of bit errors and the correction capability of the ECC, HBM errors can be divided into correctable errors (CE) and uncorrectable errors (UCE). CE refers to the errors within the correction capability of ECC that can be successfully recovered, *e.g.,* a single-bit error in HBM that can be corrected by ECC. Patrol scrubbing is employed as a proactive error detection and correction technique [5] in memory systems. It involves periodically scanning the memory to identify and

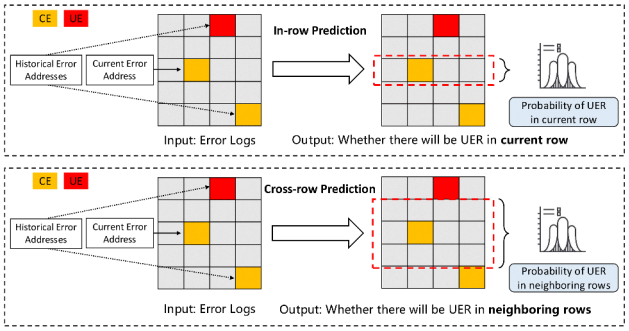


Fig. 2. An Illustration of Cross-row Failure Prediction

correct errors before these errors are accessed. Even though correctable, CEs can be accumulated over time to become uncorrectable [27]. UCEs refer to errors that exceed the correction capability of ECC and are not correctable [28]. They can be further categorized into two subtypes based on the necessity to take action to address the errors and the actual impact on the business of the errors: *Uncorrectable Error Action Optional* (UEO) and *Uncorrectable Error Action Required* (UER).

C. Cross-row Failure Prediction

Given the high cost associated with bank sparing, row sparing becomes a preferable option in HBMs of large-scale LLM training systems. As such, row-level prediction becomes essential in optimizing resource allocation and enhancing the overall reliability of these systems. Existing methods focus on forecasting UERs by analyzing patterns of prior errors within the same row. The assumption is that these historical errors can serve as precursors to future failures. However, in-row failure prediction methods often fall short in practical applications. Our empirical studies conducted on large-scale datasets in Huawei reveal that many UERs occur abruptly, without any prior error indications. This unpredictability renders in-row failure predictions inadequate for industrial use. Recognizing this gap, we propose cross-row failure prediction, which leverages the spatial and temporal information of errors across rows within the same HBM bank. The comparison of these two paradigms is illustrated in Figure 2.

III. Empirical Study

In this section, we present an empirical study to understand the characteristics of sudden UER, failure pattern and the locality of UER rows, which motivates our method design.

A. Sudden UER Ratio

As described in the literature [29], there are two types of UERs: sudden UERs, which result from component malfunctions that immediately corrupt data, and non-sudden UERs, which are predictable and initially appear as CEs and UEOs but evolve into UERs over time. Sudden UERs are typically considered unpredictable, meaning existing in-row failure prediction frameworks cannot effectively address them. In Table I, we present the ratio of sudden UERs observed in an industrial dataset, which includes more than 10,000 NPUs and 80,000 HBMs. Our analysis reveals that the behavior of sudden UERs in HBMs is markedly different compared to traditional DDR4 and DDR5 memory systems. Specifically, the ratio of sudden UERs increases drastically as we move from the NPU level to the row level, with sudden row UERs accounting for more than 95% of all UERs. This underscores the limitations of existing in-row prediction methods, rendering them impractical for managing sudden UERs at the row level.

TABLE I
In-row Predictable Ratio of UERs

Micro-level	Sudden UER	Non-sudden UER	Predictable Ratio
NPU	243	175	41.86%
HBM	246	175	41.56%
SID	260	180	40.91%
PS-CH	311	185	37.29%
BG	434	252	36.73%
Bank	760	314	29.23%
Row	4980	229	4.39%

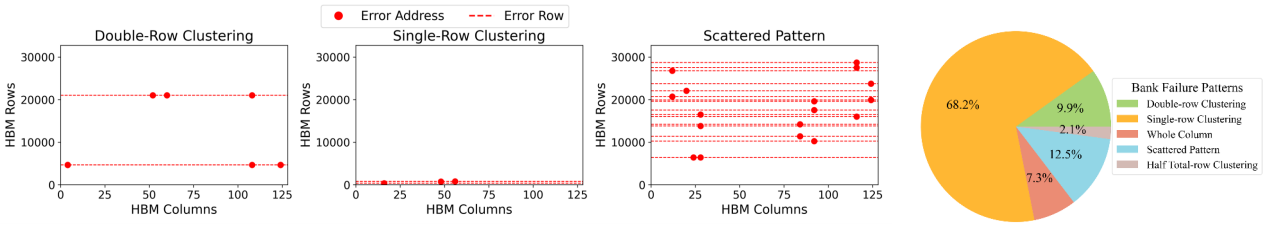
B. Bank-Level Failure Patterns

Our empirical analysis has identified several distinct failure patterns at the bank level: double-row clustering pattern (including half total-row clustering), single-row clustering pattern, scattered pattern, and a special case of scattered pattern known as the whole column pattern with the error dispread in nearly all the rows. Figure 3(a) illustrates examples of these failure patterns. The single-row clustering pattern, comprising 68.2% of observed UER banks, is characterized by errors concentrated within a contiguous, narrow area, facilitating easier failure prediction due to its spatial locality. Double-row clustering, which includes the half total-row clustering variant, accounts for 9.9% of UERs. This pattern features two clusters of UERs with a consistent interval between them, making prediction manageable by leveraging the predictable spacing between clusters. The scattered pattern is more complex, with UERs distributed irregularly across the bank, representing 12.5% of UERs. Within this category, column failure—a special case where UERs appear across all rows of a column—accounts for 7.3%. This pervasive distribution necessitates bank-sparing techniques, as the unpredictable nature of errors complicates row-level failure mitigation.

Our analysis of the industrial dataset, as depicted in Figure 3(b), reveals that while scattered patterns pose challenges, the prevalence of aggregation patterns (78.1% combined) indicates that cross-row failure prediction remains feasible for the majority of cases. This insight underscores the practical applicability of targeted prediction strategies in managing memory failures within high-performance computing systems.

C. Locality of Cross-row UER

Since the single-row clustering pattern is characterized by UERs concentrated within a narrow and contiguous area, this spatial concentration can be advantageous for cross-row UER prediction, as it suggests that subsequent UERs are likely to occur in the vicinity of the existing UER row. Thus, we explore



(a) Examples of Bank-level Failure Patterns (b) Bank Failure Pattern Distribution

Fig. 3. Comparison of Failure Patterns and Their Distribution in Banks

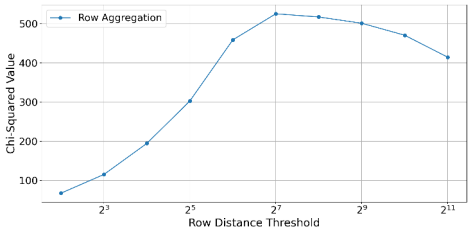


Fig. 4. Statistic Significance of Difference Distance Thresholds

the locality of cross-row UERs to determine the effective range within which predictions can be made. To quantify this locality, we compute the chi-square statistic of subsequent UERs occurring within various row distance thresholds from the current UER row. These thresholds range from 4 to 2048 rows. Our analysis indicates that the strongest statistical significance is achieved at a threshold of 128 rows, as shown in Figure 4. This suggests that predicting UERs within a 128-row range is both manageable and effective, enabling us to focus our prediction efforts and redundant resources on the most likely areas for subsequent UERs.

IV. Methodology

Based on insights from our empirical study, we propose the Cross-row Failure Prediction Method Based on Bank-level Error Locality (Cordial). Cordial is designed to predict failures in a cross-row manner, effectively addressing the limitations of existing in-row prediction methods.

A. Overview

The overall workflow of Cordial is illustrated in Figure 5, which consists of three stages: failure pattern feature extraction, failure pattern classification and cross-row failure prediction. We first collect the raw error log from the baseboard management controller (BMC) and then generate a set of spatial and temporal features (*e.g.,* the average row difference between two successive UER rows) with all CEs, UEOs and the first three UERs for each bank. Then, we use the generated features to train tree-based predictors and output the failure pattern of the current bank. We finally utilize the cross-row UER predictors to anticipate whether there are UER rows in the neighboring rows for aggregation patterns and conduct isolation for these predicted error rows. Otherwise, all banks with scattered row patterns will be isolated directly.

B. Failure Pattern Feature Extraction

Before training prediction models, we first generate representative features for both failure pattern classification and cross-row failure prediction. For each error event, the address of errors, the time of error occurrence, and the error types are recorded. Thus, we generate the following features for the predictors:

- *Spatial features.* Intuitively, the row number of errors and the row difference between errors can be indicators of whether there is error aggregation. Thus, we compute the minimum and maximum of CE, UEO, and UER rows and the minimum, maximum, and average row differences between two consecutive errors. These features help identify patterns that may indicate clustering or dispersion of errors within the data.
- *Temporal features.* From another perspective, there may be more frequent errors when there is error aggregation, as errors can soon propagate to nearby rows. Temporal features can provide insights into the frequency of error occurrences, enabling the prediction models to detect temporal patterns and trends that may signal impending failures. Thus, we compute the minimum and maximum occurrence time difference of CEs, UEOs, and UERs.
- *Count features.* When there are many CEs and UEOs in a bank, it can indicate a higher likelihood of having more UERs, suggesting a scattered pattern. Conversely, if there are not a very large number of errors, it may indicate a single-row clustering pattern. Therefore, we compute the total counts of CEs and UEOs before the first UER happens within the bank. These count features provide a measurement of error density.

C. Failure Pattern Classification

Tree-based machine learning approaches have been widely used in memory failure prediction literature [27] due to their fast learning and satisfactory performance. Thus, we leverage three tree-based machine-learning techniques for cross-row prediction: Random Forest, XGBoost, and LightGBM because they are lightweight, easy to deploy, and have low computation costs in industrial applications. If the classification reveals a single-row clustering or double-row clustering pattern, we proceed with cross-row prediction, as these patterns indicate a clustering distribution of UER rows. In contrast, if the pattern is identified as scattered, the UER rows are widely distributed across many rows, making it difficult to mitigate the failure

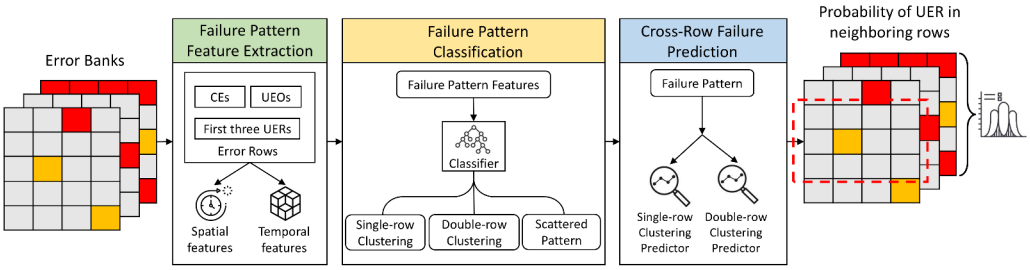


Fig. 5. The Overview of Our Proposed Method Cordial

through row isolation. In such cases, directly replacing the bank or NPU is more feasible, and cross-row failure prediction is not triggered.

It should be noted that our objective is to use as few error events as possible to determine the failure pattern, enabling us to select specific prediction models quickly. Early pattern identification is crucial for timely intervention. However, using all UERs for pattern classification can become impractical due to the need for rapid decision-making. On the other hand, when only a single UER is observed, it is challenging to distinguish between aggregation and scattered patterns. Similarly, differentiating between a double-row clustering and a scattered pattern can also be difficult with limited data. We use the first three UER information for failure pattern classification, providing a pragmatic trade-off.

D. Cross-row Failure Prediction

Based on our study of the locality of cross-row UERs, we focus on predicting the occurrence of UERs within a 128-row range, *i.e.,* 64 rows above and below the last UER row. To facilitate this prediction, we divide these 128 rows into 16 blocks, each containing 8 rows. Our goal is to predict whether there will be a UER in each of these blocks. In cross-row failure prediction, we leverage a variety of features for our tree-based models, including:

- *Spatial Features:* We incorporate the specific row numbers where CEs, UEOs, and UCEs occur to identify potential aggregation points within the memory architecture. Additionally, we analyze the differences in row numbers between consecutive errors to discern patterns of dispersion or clustering, which are critical for understanding the spatial dynamics of error propagation.
- *Temporal Features:* We examine the time intervals between consecutive occurrences of each error type to capture the temporal patterns underlying cross-row errors. We also measure the time difference from the last error event to highlight recent temporal trends.
- *Count Features:* We include the total counts of CEs, UEOs, UCEs, and all error types within the bank to provide a comprehensive overview of error density.

These features are utilized within tree-based models, such as Random Forest, to effectively predict the likelihood of UERs in each block. Our approach is tailored to address the specific challenges of cross-row UER prediction in HBM systems by focusing on this block-level prediction, leveraging the localized nature of errors to inform timely interventions.

TABLE II
A SUMMARY OF THE INDUSTRIAL DATASET

Micro-level	With CE	With UEO	With UER	Total Count
NPU	5497	327	418	5703
HBM	5944	330	421	6155
SID	6049	341	440	6277
PS-CH	6856	360	496	7136
BG	7571	423	686	7970
Bank	8557	537	1074	9318
Row	51518	4888	5209	60693

V. EVALUATION

In this section, we first present a summary of the datasets and the evaluation metrics. Finally, evaluation results are presented to demonstrate the effectiveness of Cordial.

A. Experimental Settings

We collected the MCE log and memory events from the BMC of HBMs in a large-scale LLM training system. All the CE, UEO, UER events are recorded in MCE log [27], including details about memory error addresses (*e.g.,* server number, bank, row). We examine error logs from the large-scale LLM training platform containing more than 10,000 NPUs and 80,000 HBMs, where the summary is shown in Table II. We split the dataset into a proportion of 7:3, with 70% used for training the model and the remaining 30% for testing [5]. We assess the precision, recall and F1 score for both the failure pattern classification and cross-row failure prediction. Additionally, we assess the Isolation Coverage Rate (ICR) to gauge the practical effectiveness of deploying our prediction framework. This metric measures the proportion of UER rows that can be preemptively isolated based on our cross-row failure predictions, thereby preventing potential failures. Through this evaluation metric, we aim to demonstrate the tangible benefits of our framework in real-world applications.

B. Evaluation Results

We employed three tree-based models, *i.e.,* LightGBM, XGBoost, and Random Forest, for failure pattern classification. The performance of these models is shown in Table III. Among them, Random Forest demonstrated the best overall performance. This can be attributed to its ensemble learning approach, which effectively reduces variance and improves

TABLE III
PERFORMANCE OF FAILURE PATTERN CLASSIFICATION

Pattern	LightGBM			XGBoost			Random Forest		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Double-row Clustering	0.600	0.474	0.529	0.611	0.289	0.393	0.633	0.500	0.559
Single-row Clustering	0.921	0.972	0.946	0.881	1.000	0.937	0.921	0.981	0.950
Scattered Pattern	0.672	0.629	0.650	0.698	0.597	0.643	0.696	0.629	0.661
Weighted Average	0.833	0.844	0.837	0.803	0.835	0.813	**0.842**	**0.859**	**0.854**

TABLE IV
PERFORMANCE OF DIFFERENT FAILURE PREDICTION METHODS

Methods	Precision	Recall	F1 Score	ICR (%)
Neighbor Rows	0.322	0.393	0.347	13.31%
Cordial-LGBM	0.642	0.504	0.563	18.60%
Cordial-XGB	0.732	0.509	0.591	18.87%
Cordial-DT	**0.806**	**0.580**	**0.662**	**19.58%**

accuracy by averaging the predictions of multiple decision trees. LightGBM and XGBoost, both based on boosting techniques, inherently accumulate errors over iterations as they sequentially build models to correct the errors of previous models. While this can enhance performance in some cases, it can also lead to overfitting, which might explain their relatively lower performance compared to Random Forest.

Regarding the failure pattern classification, the single-row clustering pattern proved to be the most effectively classified, achieving the highest precision, recall, and F1 scores. This is because when the first three UER rows are neighboring each other, a single-row clustering pattern is strongly indicated. This proximity of the initial errors allows for easier identification of the pattern. The classification of the other two patterns, double-row clustering and scattered, is slightly more challenging but still satisfactory. If the three UER rows are scattered across different areas of the bank, it suggests a scattered pattern. Conversely, if one UER row is distant from the other two, which are clustered together, a double-row clustering pattern is more likely. This approach of using only the first three UER rows is strategic, leveraging the minimal but necessary error information to achieve effective classification.

As previously mentioned, the sudden UER ratio can reach as high as 95.61% at the row level, indicating that existing methods are ideally capable of predicting only 4.39% of UERs. Therefore, to ensure a fair comparison, we benchmark our approach against an industrial baseline. This baseline method isolates the eight rows adjacent to an identified UER row, aiming to prevent further propagation of errors within the immediate vicinity. Compared to this baseline, our method demonstrates significantly improved performance across various metrics, including weighted precision, recall, and F1 score of all prediction blocks, as well as in practical industrial evaluation metrics such as Isolation Coverage Rate (ICR), which is shown in Table IV. Notably, the strong performance of Random Forest aligns with the results of the failure pattern classification, reinforcing its effectiveness. Though our method achieves a 19.58% isolation coverage rate due to the inherent randomness of UER rows that adds complexity to the problem, it is substantially higher compared to traditional in-row failure prediction (19.58% over 4.39%). This randomness underscores the challenge of accurately predicting failure patterns, yet our framework's intelligent block selection strategy effectively addresses these difficulties. By adopting our new paradigm of cross-row failure prediction, we believe industries can achieve more robust and proactive management of memory failures, ultimately improving system stability and performance.

VI. RELATED WORK

With the increasing demand for fast and high-capacity memory due to the emergence of large language models (LLMs), HBM has gained tremendous attention and is considered a promising technology to overcome the memory bottleneck [1], [30]. Though the memory access bandwidth is enhanced via stacking up multiple DRAM dies through TSVs, new reliability issues are introduced for HBM due to the vertical architecture. Specifically, some studies [31] reveal that poor-quality micro-bump joints caused by thermal compression bonding (TCB) affect the reliability of HBM. Besides, TSVs are also prone to faults as micro-bumps with micrometer size must be precisely aligned with the TSVs, where shocks may occur and lead to the potential damage of TSVs [32]–[34]. Some studies also investigate reliability issues under inappropriate operating conditions, for example, a higher rate of bit flips occurs when reducing the voltage [35]. HBM also shares similar reliability degradation caused by read disturbance vulnerability (*e.g.,* RowHammer and RowPress) with DRAM [25]. Apart from the studies on the hardware characteristics of HBM, a hierarchical HBM failure prediction framework, Calchas, that utilizes spatial, temporal, and sensor information at different device levels has been proposed in [5].

VII. CONCLUSION

In this work, we conduct a comprehensive empirical study that highlights the challenges of sudden error and the failure patterns in HBMs of high-performance computing platforms. Motivated by these findings, we propose a cross-row failure prediction method based on bank-level error locality, effectively addressing the limitations of traditional in-row prediction methods in high sudden error rate scenarios. The evaluation results demonstrate significant improvements in prediction accuracy and isolation coverage, demonstrating the practical usefulness of our framework.

VIII. ACKNOWLEDGMENTS

The work was supported by the Research Grants Council of the Hong Kong Special Administrative Region, China (No. CUHK 14206921 of the General Research Fund).

REFERENCES

[1] K. Kim and M.-j. Park, "Present and future, challenges of high bandwith memory (hbm)," in *2024 IEEE International Memory Workshop (IMW)*. IEEE, 2024, pp. 1–4.

[2] J. Lee, J. M. Lee, Y. Oh, W. J. Song, and W. W. Ro, "Snakebyte: A tlb design with adaptive and recursive page merging in gpus," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2023, pp. 1195–1207.

[3] A. Gholami, Z. Yao, S. Kim, C. Hooper, M. W. Mahoney, and K. Keutzer, "Ai and memory wall," *IEEE Micro*, 2024.

[4] F. Karimzadeh, M. Imani, B. Asgari, N. Cao, Y. Lin, and Y. Fang, "Memory-based computing for energy-efficient ai: Grand challenges," in *2023 IFIP/IEEE 31st International Conference on Very Large Scale Integration (VLSI-SoC)*. IEEE, 2023, pp. 1–8.

[5] R. Wu, S. Zhou, J. Lu, Z. Shen, Z. Xu, J. Shu, K. Yang, F. Lin, and Y. Zhang, "Removing obstacles before breaking through the memory wall: A close look at hbm errors in the field," in *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, 2024, pp. 851–867.

[6] J. Hong, S. Cho, G. Park, W. Yang, Y.-H. Gong, and G. Kim, "Bandwidth-effective dram cache for gpu s with storage-class memory," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024, pp. 139–155.

[7] S. Ainsworth and T. M. Jones, "Software prefetching for indirect memory accesses: A microarchitectural perspective," *ACM Transactions on Computer Systems (TOCS)*, vol. 36, no. 3, pp. 1–34, 2019.

[8] N. Talati, K. May, A. Behroozi, Y. Yang, K. Kaszyk, C. Vasiladiotis, T. Verma, L. Li, B. Nguyen, J. Sun *et al.*, "Prodigy: Improving the memory latency of data-indirect irregular workloads using hardware-software co-design," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2021, pp. 654–667.

[9] H. Li, K. Liu, T. Liang, Z. Li, T. Lu, H. Yuan, Y. Xia, Y. Bao, M. Chen, and Y. Shan, "Hopp: Hardware-software co-designed page prefetching for disaggregated memory," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2023, pp. 1168–1181.

[10] G. Fu, T. Xia, Z. Luo, R. Chen, W. Zhao, and P. Ren, "Differential-matching prefetcher for indirect memory access," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024, pp. 439–453.

[11] X. Xie, Z. Liang, P. Gu, A. Basak, L. Deng, L. Liang, X. Hu, and Y. Xie, "Spacea: Sparse matrix-vector multiplication on processing-in-memory accelerator," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2021, pp. 570–583.

[12] X. Zhang, S. L. Song, C. Xie, J. Wang, W. Zhang, and X. Fu, "Enabling highly efficient capsule networks processing through a pim-based architecture design," in *2020 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 542–555.

[13] C. Zhang, H. Sun, S. Li, Y. Wang, H. Chen, and H. Liu, "A survey of memory-centric energy-efficient computer architecture," *IEEE Transactions on Parallel and Distributed Systems*, 2023.

[14] Q. Hu, Z. Ye, Z. Wang, G. Wang, M. Zhang, Q. Chen, P. Sun, D. Lin, X. Wang, Y. Luo *et al.*, "Characterization of large language model development in the datacenter," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 709–729.

[15] Z. Jiang, H. Lin, Y. Zhong, Q. Huang, Y. Chen, Z. Zhang, Y. Peng, X. Li, C. Xie, S. Nong *et al.*, "{MegaScale}: Scaling large language model training to more than 10,000 {GPUs}," in *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, 2024, pp. 745–760.

[16] W. Gu, J. Gu, J. Liu, Z. Chen, J. Zhang, J. Kuang, C. Feng, Y. Yang, and M. R. Lyu, "Adamas: Adaptive domain-aware performance anomaly detection in cloud service systems," in *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2025, pp. 621–621.

[17] W. Gu, J. Liu, Z. Chen, J. Zhang, Y. Su, J. Gu, C. Feng, Z. Yang, Y. Yang, and M. R. Lyu, "Identifying performance issues in cloud service systems based on relational-temporal features," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 3, pp. 1–31, 2025.

[18] Y. Xiong, Y. Jiang, Z. Yang, L. Qu, G. Zhao, S. Liu, D. Zhong, B. Pinzur, J. Zhang, Y. Wang *et al.*, "{SuperBench}: Improving cloud {AI} infrastructure reliability with proactive validation," in *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, 2024, pp. 835–850.

[19] Z. Cheng, S. Han, P. P. 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)*. IEEE, 2022, pp. 262–272.

[20] Y. Moon, S. H. Shin, S. Jang, D. Won, and S. Kang, "A novel prediction-based two-tiered ecc for mitigating swd errors in hbm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2024.

[21] D. Kline, J. Zhang, R. Melhem, and A. K. Jones, "Flower and fame: A low overhead bit-level fault-map and fault-tolerance approach for deeply scaled memories," in *2020 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 356–368.

[22] A. K. Kabat, S. Pandey, and V. T. Gopalakrishnan, "Performance evaluation of high bandwidth memory for hpc workloads," in *2022 IEEE 35th International System-on-Chip Conference (SOCC)*. IEEE, 2022, pp. 1–6.

[23] C. Tsai, T. Ku, M. Chen, W. Chiou, C. Wang, and D. Yu, "Low-temperature soic™ bonding and stacking technology for 12/16-hi high bandwidth memory (hbm)," in *2020 IEEE Symposium on VLSI Technology*. IEEE, 2020, pp. 1–2.

[24] S. Ha, S. Lee, G. Bae, D. Lee, S. Kim, B. Woo, N. Lee, Y. Lee, and S. Pae, "Reliability characterization of hbm featuring hk+mg logic chip with multi-stacked drams," in *2023 IEEE International Reliability Physics Symposium (IRPS)*. IEEE, 2023, pp. 1–7.

[25] A. Olgun, M. Osseiran, A. G. Yağlıkçı, Y. C. Tuğrul, H. Luo, S. Rhyner, B. Salami, J. G. Luna, and O. Mutlu, "Read disturbance in high bandwidth memory: A detailed experimental study on hbm2 dram chips," in *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2024, pp. 75–89.

[26] 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 *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2022, pp. 01–14.

[27] 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)*. IEEE, 2023, pp. 216–228.

[28] W. Li, M. Zhang, T. Gui, Z. Fang, C. Xie, and F. Wu, "Improving dram reliability using a high order error correction code," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2024.

[29] 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, 2023, pp. 01–09.

[30] C. Pohl, K.-U. Sattler, and G. Graefe, "Joins on high-bandwidth memory: a new level in the memory hierarchy," *The VLDB Journal*, vol. 29, no. 2, pp. 797–817, 2020.

[31] H. Jun, J. Cho, K. Lee, H.-Y. Son, K. Kim, H. Jin, and K. Kim, "Hbm (high bandwidth memory) dram technology and architecture," in *2017 IEEE International Memory Workshop (IMW)*. IEEE, 2017, pp. 1–4.

[32] K. Bae and J. Park, "Efficient tsv fault detection scheme for high bandwidth memory using pattern analysis," in *2020 International SoC Design Conference (ISOCC)*. IEEE, 2020, pp. 19–20.

[33] Y. Lee, D. Han, and S. Kang, "Tsv built-in self-repair architecture for improving the yield and reliability of hbm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 4, pp. 578–590, 2023.

[34] D. Han, D. Won, S. Kim, and S. Kang, "Tsv built-in self-repair architecture for lifespan reliability enhancement of hbm," *IEEE Transactions on Reliability*, 2024.

[35] S. S. N. Larimi, B. Salami, O. S. Unsal, A. C. Kestelman, H. Sarbazi-Azad, and O. Mutlu, "Understanding power consumption and reliability of high-bandwidth memory with voltage underscaling," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 517–522.