Looking Back to Move Forward: Unveiling the Mysteries of HBM Errors to Predict Future Failures

SHUYUE ZHOU, XINBIN HU, RONGLONG WU, JIAHAO LU, ZHIRONG SHEN*, and ZIKANG XU, Xiamen Key Laboratory of Intelligent Storage and Computing, Xiamen University, China
YUE YU, Pengcheng Laboratory, China
YUZE JIANG, Department of Statistics, University of Michigan, USA
JIWU SHU, Xiamen University and Tsinghua University, China
KUNLIN YANG and FEILONG LIN, Huawei Technologies Co., Ltd, China
YIMING ZHANG, Xiamen Key Laboratory of Intelligent Storage and Computing, Xiamen University, China

High-bandwidth memory (HBM) is regarded as a promising technology for fundamentally overcoming the memory wall. It stacks up multiple DRAM dies vertically to dramatically improve the memory access bandwidth. However, this architecture also comes with more severe reliability issues, since HBM not only inherits error patterns of the conventional DRAM, but also introduces new error causes.
In this paper, we conduct the first systematical study on HBM errors, which cover over 460 million error events collected from nineteen data centers and span over two years of deployment under a variety of services. Through error analyses and methodology validations, we confirm that the HBM exhibits different error patterns from conventional DRAM, in terms of spatial locality, temporal correlation, and sensor metrics which make empirical prediction models for DRAM error prediction ineffective for HBM. We design and implement Calchas, a hierarchical failure prediction framework for HBM based on our findings, which integrate spatial, temporal, and sensor information from various device levels to predict upcoming failures. The results demonstrate the feasibility of failure prediction across hierarchical levels.

CCS Concepts: • **Computer systems organization** → **Reliability; Availability**; • **Computing methodologies** → *Machine learning approaches*;.

Additional Key Words and Phrases: high-bandwidth memory, memory error analysis, failure prediction

1 INTRODUCTION

The performance gap between the computing power and the memory bandwidth is continuously widening in modern computing systems (a.k.a. memory wall), which becomes one of the major obstacles in training ever-larger machine learning models. Extensive efforts have been made to mitigate the impact of the memory wall, including leveraging access locality in data prefetching

*Corresponding author: Zhirong Shen (shenzr@xmu.edu.cn).

This work is supported by the National Key R&D Program of China (Grant No. 2024YFB4505201), the Major Research Plan of the National Natural Science Foundation of China (Grant No. 92373114), the National Natural Science Foundation of China (Grant No. U22B2023, No. 62441220 and No. 624B2120), and Xiaomi Young Scholars. This journal submission presents substantial improvements over our previous version published in USENIX ATC 2024 [89]. Specifically, the journal submission incorporates new findings from spatial and temporal analyses, introduces a dual-model predictor to enhance UER detection, and presents extensive experimental evaluations to validate its effectiveness under various conditions.
Authors' addresses: Shuyue Zhou; Xinbin Hu; Ronglong Wu; Jiahao Lu; Zhirong Shen; Zikang Xu, Xiamen Key Laboratory of Intelligent Storage and Computing, Xiamen University, Xiamen, 361005, China, emails:{syzhouu,xinbinhu,rlwoo}@stu.xmu.edu.cn,lujhcoconut@foxmail.com,shenzr@xmu.edu.cn,xuzikang@stu.xmu.edu.cn; Yue Yu, Pengcheng Laboratory, Shenzhen, China, yuy@pcl.ac.cn; Yuze Jiang, Department of Statistics, University of Michigan, Ann Arbor, USA, jyzjyz@umich.edu; Jiwu Shu, Xiamen University and Tsinghua University, Beijing, China, jwshu@xmu.edu.cn; Kunlin Yang; Feilong Lin, Huawei Technologies Co., Ltd, Hangzhou, China, kunliny1997@gmail.com,linfeilong@huawei.com; Yiming Zhang, Xiamen Key Laboratory of Intelligent Storage and Computing, Xiamen University, Xiamen, China, sdiris@gmail.com.

2025. 1553-3077/2025/9-ART1 $15.00
https://doi.org/0000001.0000001

[9, 11, 41, 64, 76], exploring vectorization [32, 40], exploiting data compression [2, 4, 14, 73], and designing new system architectures (e.g., processing-in-memory architecture [37, 48, 81]).

Recently, *high bandwidth memory* (HBM) receives tremendous attentions and is considered as a promising technology to fundamentally overcome the memory bottleneck. For instance, the first 8-high 24 GB HBM3 Gen2 memory released by Micron can achieve the bandwidth greater than 1.2 TB/s [65], which reduces more than 30% of the training time for large language models and also the total cost of ownership. The rationale behind HBM is to stack up multiple planar DRAM chips vertically through *through-silicon vias* (TSVs) and microbumps [48, 49, 83]. HBM further provides several independent interfaces (called *pseudo channels*) to support parallel data access to different sets of banks, hence achieving high aggregated access bandwidth.

While being fast and power-efficient, HBM is more vulnerable to unexpected errors due to the following reasons. First, as HBM is constructed by stacking DRAM dies, it not only inherits the error characteristics of conventional DRAM, but also poses new error causes, such as more severe soft errors under the higher bit density [47] and TSV faults (including data TSV faults, command TSV faults, and power TSV faults [5, 50, 66]). Second, HBM usually equips weaker error correction codes than conventional DRAM due to cost and complexity considerations [59]. Therefore, understanding the error characteristics of HBM beforehand becomes extremely vital when designing techniques to guarantee the reliability of the stored data.

Extensive in-depth analyses of DRAM errors have already been conducted in recent years, which mainly study the error characteristics [6, 8, 15, 74, 86], identify the root causes of errors [16, 35, 43, 46, 56], and further attempt to predict the upcoming failures [7, 10, 18, 22, 24, 53, 93, 95]. However, after careful analyses and validations, we uncover that most of the findings and implications made from DRAM error analyses cannot be directly applied to HBM, since HBM has a more complex 3D-stacked architecture (e.g., system-in-package fabrication [77] and TSVs [38, 47]) that finally results in new fault modes and error patterns.

In this paper, we fill this blank by performing an in-depth data-driven analysis to study the error characteristics of HBM. We look into a large-scale dataset [1] collected from nineteen data centers over two years, which comprises error logs (e.g., *correctable errors* (CEs) and *uncorrectable errors* (UEs)), temperature logs (e.g., the temperature of *domain specific architecture* (DSA) devices), and power logs (e.g., the average power). We perform a series of analysis, in terms of the spatial analysis (e.g., spatial locality and structure analysis), temporal analysis (e.g., CE storm), power analysis (e.g., power trends of errors), and temperature analysis (e.g., temperature distribution analysis), which deliver sixteen findings in total. Based on the above analysis, we perform two unsuccessful attempts, which try to use the CE rate (defined as the number of CEs occurring within a monitored time interval) and historical CE information to predict the upcoming UEs. We finally design Calchas, which is a hierarchical, comprehensive, and non-intrusive failure prediction framework for HBM. Based on our analysis results, Calchas integrate information at different levels while considering the impact of various errors to predict upcoming failures.

More specifically, we make the following contributions:

(1) In-depth analyses for HBM errors collected from production clusters. We collect over 460 million error events in total from nineteen data centers with the time duration of over two years. We also collect auxiliary information, including temperature logs and power logs, to facilitate the analysis. We first perform the spatial analysis to study whether the emerging errors of HBM still follow the spatial locality and investigate the impact of the 3D-stacked architecture on the errors (§3.2). We then carry out temporal analysis to study the intervals between CEs and UEs to seek the

[1]The dataset is published at: https://github.com/wrl297/Calchas.

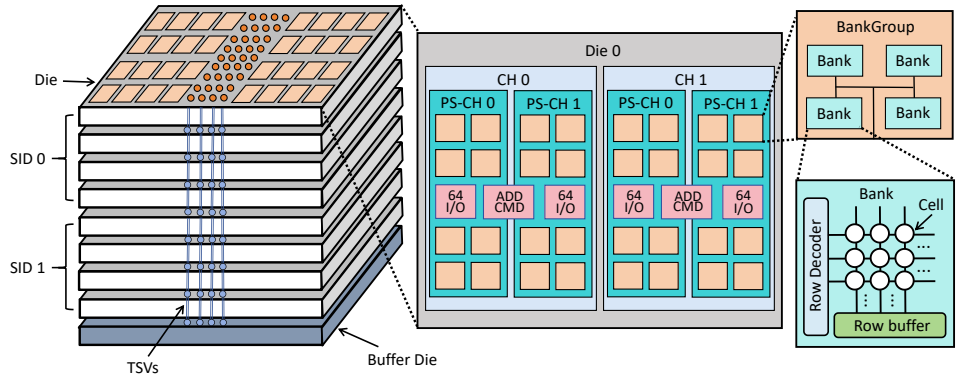


Fig. 1. The HBM architecture with eight DRAM dies (§2.1).

possibility of using historical CEs to predict upcoming UEs (§3.3). We finally study the impact of temperature and power on the error occurrence (§3.4).

To the best of our knowledge, Calchas is the first effort that systematically analyzes the error characteristics of HBM. Moreover, the dataset we study contains over 460 million error events, which is orders of magnitude more than the largest datasets of DRAM errors that have been released till now (e.g., the dataset with 75.1 million CEs collected from production data centers at Alibaba [15]).

(2) Lessons learned from unsuccessful attempts. We have tried two unsuccessful attempts to build prediction models based on empirical approaches in DRAM error prediction [15, 24]. We learn the following lessons from these attempts. First, the effectiveness of previous CE rate indicator [22, 62] in predicting UEs is questionable for HBM, as there is no clear correlation between the CE rate and the occurrence of UERs (§4.1). Second, the CE-based predictor trained by historical CEs is also impractical to predict the future occurrence of UERs, as most UERs tend to coincide with other UERs rather than with CEs (§4.2).

(3) A hierarchical failure prediction framework for HBM. With the lessons and insights gained from previous unsuccessful attempts, we finally design and implement Calchas, a hierarchical failure prediction framework based on random forest [12] for HBM. It is the first prediction model for predicting failures in HBM. It combines features specific to HBM (e.g., stack features) with those used in traditional DRAM (e.g., component features) for prediction (§5.1). Moreover, Calchas employs different prediction timing (i.e., period-based approach for server-level prediction and event-driven approach for micro-level prediction) to realize the elastic and adaptive failure prediction (§5.4).

2 BACKGROUND

2.1 High Bandwidth Memory

High bandwidth memory (HBM) has a specific 3D-stacked structure that sets it apart from traditional planar memory architectures. It offers significantly higher data transfer rates compared to conventional DRAM. For instance, HBM2 (the second-generation of HBM) can achieve a bandwidth of around 256 GB/s per stack [17, 39], which is far larger than the bandwidth in DDR4 (around 25 GB/s per channel [54]).

HBM can be constructed via either a 4Hi stack (with four DRAM dies) or an 8Hi stack (with eight DRAM dies). Under the 8Hi configuration, every four dies can be packed to form an SID. Figure 1 illustrates the architecture of HBM with an 8Hi stack, which comprises two SIDs with eight dies.

A DRAM die consists a number of *channels* (CH). In the pseudo-channel mode, a channel can be further divided into two *pseudo-channels* (PS-CH), which is composed of several *bank groups* (BG). A BG consists of four banks, each of which comprises multiple rows and columns.

Fundamentally, HBM utilizes *through-silicon vias* (TSVs) and microbumps to connect multiple layers of DRAM cells, which are then combined with a logic layer and a memory controller. At the bottom of the stack, there is a *buffer die* (or logic die) that serves as the control and communication hub for the entire HBM stack. The communication and data transfers among the stacked DRAM dies are then realized by the TSVs, which are vertical electrical connections that run through the silicon substrate, providing high-speed data pathways. While providing higher access bandwidth, HBM is limited in capacity (e.g., 5–10× smaller than DRAM [63]) due to cost and power considerations [48, 87].

In our platform, we pack multiple HBM chips within a single DSA device (e.g., GPU[1, 67, 68], TPU[44], and NPU [57]), and multiple DSA devices finally make up a server.

2.2 Terminology

To facilitate comprehension of the analyses conducted and approaches proposed in this paper, we provide a list of frequently used terminologies below.

Error and fault: We call an error occurs if finding that the data read from HBM is inconsistent with the originally stored data (usually detected by ECC [42]). A fault refers to the fundamental cause of an error, which is often induced by hardware faults (e.g., memory wear-out [79]) and external factors (e.g., cosmic rays [35]). Note that a fault can be *active* (causing errors) or *dormant* (not causing errors) [85]. For example, writing a bit '1' to a cell with a stuck-at-1 fault (i.e., a hardware defect where a bit is permanently set to '1') will not introduce an error when reading the stored bit from this cell.

Failure: Failure refers to the inability for HBM chips to support normal read and write operations, resulting in service unavailability. For example, DRAM failure is one of the major causes of server crashes [53]. In our dataset, HBM failures account for 44% of all observed DSA device failures.

ECC: *Error correcting code* (ECC) encodes data to generate additional parity bits, so that errors can be identified and corrected. ECCs have been extensively used in DRAM to resist bit errors, including SEC-DED [56], Chipkill [19], and SDDC [13]. It is reported that the Intel Stratix 10 HBM2 Controller supports SEC-DED, with 64-bits of data and 8-bits of ECC code [36].

CE and UE: We classify the errors into *correctable errors* (CE) and *uncorrectable errors* (UE) based on the number of bit errors and the correction capability offered by ECC. CE refers to the errors within the correction capability of ECC and hence they can be successfully restored, while UE refers to those that exceed the correction capability of ECC. We then take a step further to categorize UE into another two branches based on the necessity to take actions for addressing the appearing errors: (i) UER (**U**ncorrectable **E**rror action **R**equired), which implies that the UE occurs during the system's runtime and immediate actions (e.g., replacement of DRAM DIMM, row remapping, and dynamic page offlining) must be taken to prevent server crashes and service interruptions; and (ii) UEO (**U**ncorrectable **E**rror action **O**ptional), which is usually discovered by the periodic memory scrubbing over HBM chips and does not affect the system runtime. In particular, the UEO can be addressed by taking pages offline.

2.3 Data Collection

Dataset description: We collect the error information of HBM2 chips from nineteen data centers, which comprise over 60,000 HBM2 chips across approximately 15,000 DSA devices. The monitoring duration spans over two years (from 2021-03-22 to 2023-06-21). The data centers run a variety of

Total number of different errors			
	CE	**UEO**	**UER**
Error Count	466,236,831	49,297	3,334
Number of components with errors			
	CE	**UEO**	**UER**
HBM	808	64	229
SID	817	66	231
PS-CH	847	69	233
BG	909	76	354
Bank	993	87	555
Column	2,001	258	923
Row	8,279	35,543	2,140
Cell	10,817	37,010	2,192

Table 1. Dataset overview (§3.1). The total number of errors across different device levels does not equate to the total count of device levels with errors, as multiple errors may occur within the same device level.

workloads, including autonomous driving, structural biology analysis, high-performance computing, and public cloud services. We elaborate on the data collection in details as follows.

Collection methodologies: The HBM status information is periodically collected by the *baseboard management controller* (BMC), which is a specialized processor embedded on the motherboard of a server and is responsible for recording the operational statistics of various hardware components [28]. The BMC logs are finally collected by the on-site engineers once a day.

Two approaches are employed to collect the error information with different preferences and hence we have two error logs, namely `ErrLog_Cycle` and `ErrLog_Occurrence`, respectively. Specifically, each CE entry in `ErrLog_Cycle` contains the physical address where the CE occurred, along with the accumulated times of CE occurrences at this address within the monitoring cycle. However, `ErrLog_Cycle` does not record the explicit timestamp when the CE occurred, but gives priority to the timestamp to collect the error and the error count. Hence, even if an address experiences multiple CEs in the monitoring cycle, it will be recorded as a single entry in the `ErrLog_Cycle`. On the other hand, each entry in `ErrLog_Occurrence` records a CE event captured by the BMC, which contains the explicit timestamp when the CE occurs and the corresponding physical address. For each UE, both `ErrLog_Cycle` and `ErrLog_Occurrence` record the occurrence timestamp and the physical address. However, due to the limited storage capacity of BMC, recording error entry in `ErrLog_Occurrence` may potentially be overwritten by subsequent errors, yielding error information loss, while `ErrLog_Cycle` does not miss any CE occurrences.

Sensor information: In addition to the error logs collected by BMC, we also collect the following sensor information to facilitate the identification of error root causes: (i) the temperature log, which collects the temperature every ten minutes using the temperature sensors embedded in the DSA devices; and (ii) the power log, which is collected by the power sensors embedded in the server and records the power information of the server every 10 minutes, including the peak power and the average power over the past ten minutes, as well as the transient power at the time of collection.

3 ANALYSIS

3.1 Dataset Overview

We first provide an overview about the spatial distributions of different error types (i.e., CE, UEO, and UER) during the monitoring period. The results presented in Table 1 deliver the following implications.

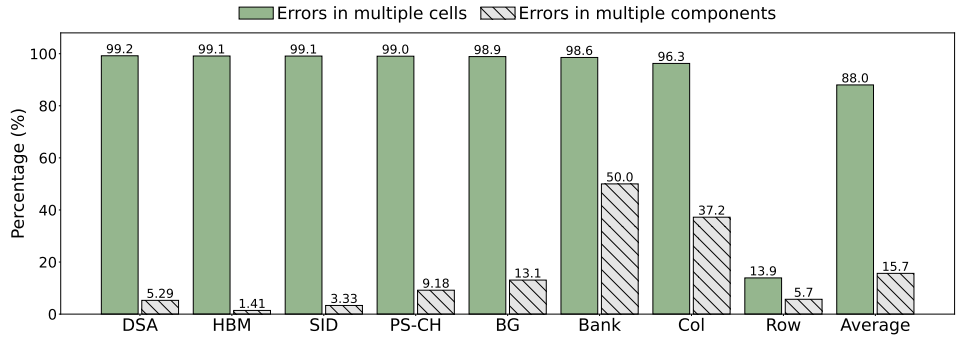


Fig. 2. Spatial locality (§3.2). This figure presents two metrics: (1) the percentage of errors occurring across multiple cells at different device levels (i.e., "Errors in multiple cells"), and (2) the percentage of errors occurring across multiple components (i.e., "Errors in multiple components").

First, we observe over 460 million CEs in HBM chips during the two-year monitoring duration, whose number is several times higher than that reported in existing studies (e.g., 75.1 million in [15]). Additionally, it is worth noting that the total number of UEOs (i.e., 49,297) is significantly higher than that of UERs (i.e., 3,334). The reason is that UEOs are observed during periodic memory scrubbing. Hence, if a component has a fault, its error information may be repeatedly collected for multiple times during the memory scrubbing.

Second, for each device level (e.g., HBM, SID, and PS-CH), the number of components exhibiting different types of errors varies. Specifically, the number of components experiencing CEs is significantly higher than those experiencing UEOs or UERs. For example, 808 HBM chips have encountered CEs, while only 64 HBM chips have experienced UEOs and 229 HBM chips have exhibited UERs. However, we observe a different distribution for row and cell levels. This is due to column failures (e.g., TSV failures [66]) being prevalent in HBM (see Finding 6 in §3.2), and when a bank experiences a column failure, performing memory scrubbing will uncover numerous cells or rows with UEOs.

3.2 Spatial Analysis

To investigate the spatial locality of HBM errors, we measure the number of errors occurring across hierarchical device levels. We then analyze the impact of specific structures (e.g., 3D-stacked and TSV architectures) on HBM errors, revealing differences in error modes between HBM and traditional DRAM.

Finding 1. *If an error occurs in one cell of a device level, there is a high probability of experiencing subsequent errors in another cell.*

Spatial locality. We start our spatial analysis by calculating the percentage of errors in multiple cells across different device levels. For each device level, we examine the proportion of devices that contain CE errors across multiple cells. Figure 2 illustrates that the majority of device levels experiencing errors are found across multiple cells. Specifically, only 12.0% of device levels experience errors within a single cell on average. At the row level, we also observe that 86.1% of errors in a row occur within a single cell (i.e., only 13.9% of errors are distributed across multiple cells), indicating that once a cell in a row experiences an error, it is likely to repeatedly experience subsequent errors. However, this observation does not hold for other device levels (excluding the row level), as we find that 98.6% of errors on average occur across multiple cells at these levels. This analysis indicates that there exists a strong correlation between the error mode and the device levels for HBM, which can be leveraged to guide the error prediction in §5.

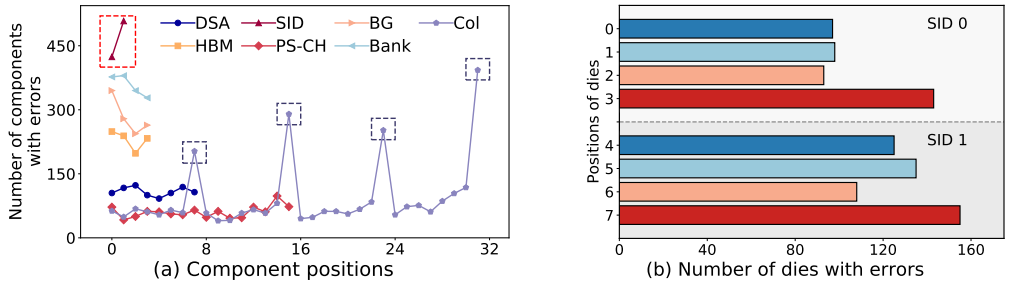


Fig. 3. Impact of HBM structure (§3.2). The figures depict (a) the distribution of components with errors across various positions and (b) the number of dies with errors in different positions.

Finding 2. *While errors are common across multiple cells in most device levels, only the bank exhibits errors in multiple components.*

We further investigate whether the errors are correlated among the components within the same device level. Given a device level, we measure the percentages of errors occurring in multiple components. For example, consider a DSA device with four HBM chips, we measure the proportion of DSA device that experience errors in multiple HBM chips. Figure 2 also shows that for most levels (e.g., DSA, HBM, and SID), they usually have only a component containing errors. This analysis implies that even though the errors are found in multiple cells, they usually manifest within a single component rather than spreading across multiple components. However, we observe a distinctive phenomenon at the bank level, where 50.0% of banks experience errors in multiple components (i.e., rows or columns). As a memory cell is organized into rows and columns, errors occurring in multiple rows or columns will inherently involve multiple cells. This is why we observe errors occurring in multiple cells for most levels, but they are typically confined to a single component.

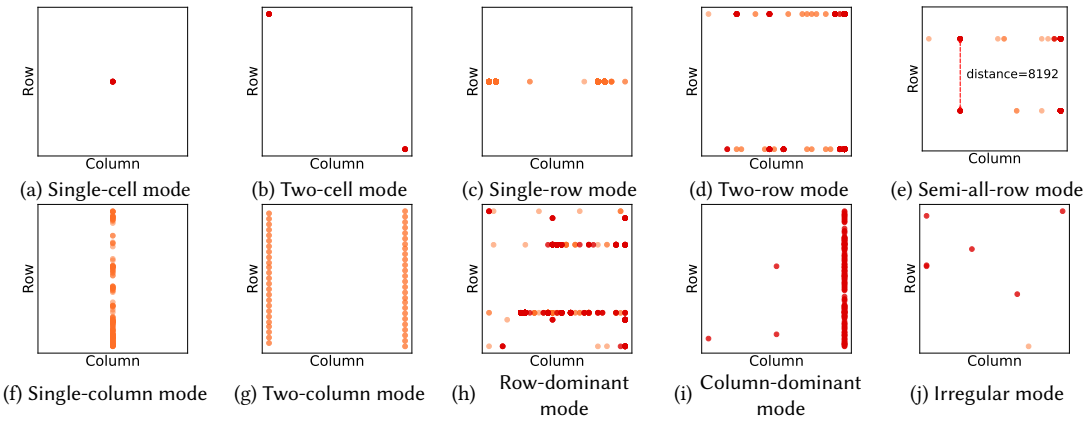
Hierarchical analysis. For HBM, we are interested in whether the 3D-stacked structure makes the errors vary across different components within the same device level. Here, we analyze the number of components experiencing errors at various device levels. For example, each BG consists of four banks, and we use $Bank_n$ (where $n \in \{0, 1, 2, 3\}$) to represent the position of a specific bank within a BG. We then count the number of banks with errors at different positions (i.e., $Bank_n$) across all BGs.

Finding 3. *The effects of crosstalk in the HBM may result in data loss, specifically in the 7-th, 15-th, 23-rd, and 31-st columns of a bank.*

Figure 3(a) shows the number of components with errors in various positions. Two key observations are made. First, for the columns with errors, those occurring in the 7-th, 15-th, 23-rd, and 31-st columns (marked in dashed boxes) are 338.4% higher on average than the errors in other column positions. After communicating with our device engineers, we suspect the reason is that crosstalk-induced faults [82] make errors more prevalent in specific columns. Moreover, we observe that, for the SIDs whose positions are closer to the buffer die (i.e., SID_1), exhibit a 20.0% higher susceptibility to errors compared to SID_0 (i.e., higher SID). We conjecture that the observed difference in error occurrence is due to poor heat dissipation in the lower SID (i.e., SID_1).

Finding 4. *Lower SIDs (i.e., SID_1) exhibit a higher susceptibility to errors. Each die in a lower SID (e.g., SID_1) has a greater probability of encountering errors compared to those in higher SIDs (e.g., SID_0).*

In Figure 3(b), we count the number of dies with errors in different positions to further demonstrate the impact of the 3D-stacked structure. When examining two dies located at the same positions but with distinct SIDs, such as Die_0 and Die_4, there is a higher probability of errors occurring in the die associated with the lower SID compared to the die linked to the higher SID.



(a) Single-cell mode (b) Two-cell mode (c) Single-row mode (d) Two-row mode (e) Semi-all-row mode

(f) Single-column mode (g) Two-column mode (h) Row-dominant mode (i) Column-dominant mode (j) Irregular mode

Fig. 4. Examples of error modes (§3.2). The figures show (a) single-cell, (b) two-cell, (c) single-row, (d) two-row, (e) semi-all-row, (f) single-column, (g) two-column, (h) row-dominant, (i) column-dominant, and (j) irregular error modes. Markers in orange (light) represent the occurrence of CEs, while the markers in red (dark) denote the occurrence of UEs.

For example, errors in Die_5 and Die_6 are 28.9% and 37.8% more prevalent than those in Die_1 and Die_2, respectively. These findings affirm that the lower SIDs in HBM are more susceptible to error occurrence.

Error modes. Our analysis reveals that multiple errors usually occur simultaneously within the same bank (see Finding 2). We further study the error modes at the bank level. We find that the errors emerged in the 1,430 banks exhibit the following nine modes. Figure 4 also plots these error modes for easy comprehension.

- Single-cell mode: Errors only occur in one cell of an HBM bank (Figure 4(a)).
- Two-cell mode: Errors only occur in two cells of an HBM bank, where the two cells are neither in the same column nor the same row (Figure 4(b)).
- Single-row mode: Errors occur in multiple cells within a single row (Figure 4(c)).
- Two-row mode: Errors occur in two rows, each of which contains multiple cells with errors (Figure 4(d)).
- Single-column mode: Errors are found in multiple cells within a single column (Figure 4(f)).
- Two-column mode: Errors are concentrated in two columns, each of which has multiple cells with errors (Figure 4(g)).
- Row-dominant mode: Errors manifest across various rows, with over 80% concentrated in specific ones (Figure 4(h)).
- Column-dominant mode: It is the dual mode of the row-dominant mode. In this mode, errors are identified across several columns, with over 80% of them concentrated in specific columns (Figure 4(i)).
- Irregular mode: It contains the modes that deviate from the aforementioned eight categories (Figure 4(j)).

Finding 5. *A special two-row error mode has been discovered, where each pair of rows is separated by 8192 rows (half of the total number of rows), possibly due to memory interleaving.*

We further examine the potential correlation of errors within these error modes. We measure the distance (i.e., the numerical difference in row or column numbers) between errors for each mode. Surprisingly, we find that the distances between a pair of rows in the two-row mode's banks are equal to half of the total number of rows within a bank. Specifically, each bank in our HBM chips contains 16,384 rows, and we observe that errors occur in the n-th rows (where n is less

Error Mode	Error Types						
	All Types	Single Error Type			Combinations of Error Types		
		CE	UER	UEO	CE&UER	UEO&UER	CE&UEO&UER
Single-cell	50.0%	55.9%	47.6%	8.05%	0%	2.27%	0%
Two-cell	4.69%	3.62%	11.3%	2.30%	9.38%	2.27%	0%
Single-row	0.629%	0.50%	0.180%	0%	2.50%	4.55%	0%
Two-row	2.94%	2.72%	0.721%	4.60%	20.6%	25.0%	50.0%
Single-column	23.7%	27.2%	16.2%	29.9%	10.0%	18.2%	12.5%
Two-column	2.65%	0.20%	1.80%	25.3%	4.38%	15.9%	3.13%
Row-dominant	1.40%	1.71%	0.721%	8.05%	10.0%	22.7%	21.9%
Column-dominant	2.87%	1.61%	3.96%	14.9%	6.88%	6.82%	9.38%
Irregular	11.1%	6.55%	17.5%	6.90%	36.3%	2.27%	3.13%

Table 2. Percentage of different error modes (§3.2). The table displays the percentage of the nine error modes across various error types. **All Types** gives the percentage of nine error modes for all banks experiencing errors (falling into one of the three error types).

than 8,192) and the $(n + 8192)$-th rows in banks identified as exhibiting the two-row mode. We name this special error pattern *semi-all row mode*. An example of the semi-all row mode is shown in Figure 4(e). Based on our analyses, we suspect that the root cause of the semi-all row mode is the memory interleaving algorithm, which results in simultaneous access to two rows that are separated by 8192 rows. Additionally, we analyze other error modes, but no distinct patterns are observed.

Since the semi-all-row mode is a special case of the two-row mode, we do not separately count the semi-all-row mode. Table 2 further gives the percentages of the nine error modes under different combinations of HBM errors, which delivers the following finding.

Finding 6. *Column-related error modes are more commonly observed in HBM, as opposed to the prevalent row-dominated error modes in conventional DRAM [8, 85].*

We find that 50.0% of faulty banks have errors in just one cell. Furthermore, 29.2% of faulty banks show the column-related modes (including the single-column, two-column, and column-dominant modes), while only 4.97% exhibit the row-related modes (including single-row, two-row, and row-dominant modes), implying that column-related error modes occur more frequently than row-related ones, deviating from the observations reported in prior analyses of DRAM errors [8, 85]. We suspect the root cause is TSV failures in HBM, which may manifest as column failures [66]. In addition, the sense amplifier failure may also exhibit the symptom of three error modes that closely relate to the column failures.

Besides, we also observe that the distribution of error modes is different when considering a single error type and combinations of error types. In particular, when considering a single error type within a bank, we find that column-related error modes (40.4% on average) appear more frequently than the row-related error modes (averaging 6.34%). However, when analyzing the banks with combinations of error types, we find that row-related error modes exhibit an average 23.3% higher probability than column-related error modes across all three combinations (i.e., CE&UER, UEO&UER, and CE&UEO&UER). Motivated by this observation, we perform predictions for upcoming UERs at various micro-level components (e.g., rows and columns) in §5.

Fault types. To analyze the faults related to different error mode, we study the fault types at the bank level for each error mode. We classify the faults into the following three types [8].

- *Transient fault*: The bank encounters a single error during the entire collection period. This fault is non-repeatable, may be caused by normal operation, and does not indicate device damage.

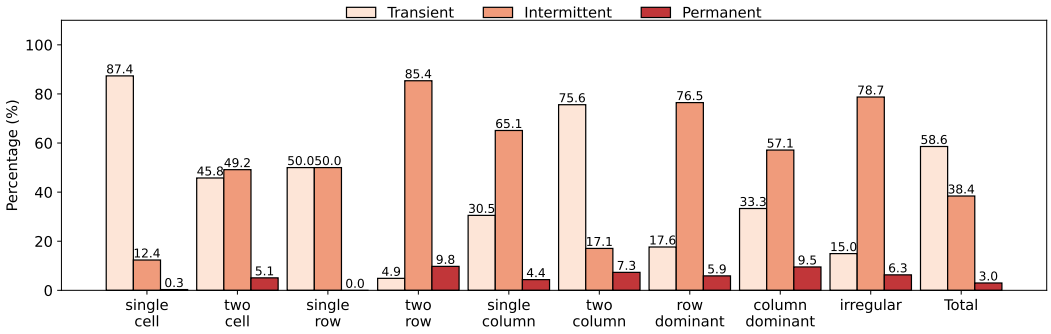


Fig. 5. Distribution of different fault types (§3.2). The figure demonstrates the distribution of different fault types for nine error modes.

- *Intermittent fault*: The bank encounters errors in at least two separate 24-hour intervals after the first error occurs, but not in every subsequent 24-hour interval. This fault typically arises under specific conditions (e.g., elevated temperature [85]) and generally indicates that the device has suffered damage or is malfunctioning.
- *Permanent fault*: The bank encounters errors in every subsequent 24-hour interval after the first error occurs. This fault indicates a high risk of hardware failure and requires immediate action (e.g., page offlining) to address the issue.

Finding 7. *Intermittent faults are common in most error modes, while transient faults are frequently observed in single-cell and two-column error modes.*

Figure 5 presents the distribution of different fault types across various error modes. We make the following observations. First, we find that intermittent faults occur more frequently in most error modes (except for single-cell and two-column modes). Therefore, it is necessary to perform memory testing for banks exhibiting these modes. Second, transient faults are more prevalent in single-cell and two-column modes. Specifically, 87.4% and 75.6% of the banks experience transient faults in single-cell and two-column modes, respectively. These results suggest that the errors are non-repeatable and not indicative of device damage in both modes. Third, permanent faults are rarely observed in HBM banks (only 3.0%), which is significantly lower compared to DRAM (e.g., 15.0% as reported by Google [8]). The reasons for this are twofold. On one hand, each bank in HBM is equipped with spare rows in hardware (e.g., 512 rows in the NVIDIA A100 [1]), which remaps rows with permanent faults to maintain the stability of HBM. Therefore, if the spare rows are not fully utilized, the permanent faults in the bank remain hidden. On the other hand, we perform memory testing on HBM before deployment to identify permanent faults and offline the corresponding rows, which further reduces the ratio of permanent faults during deployment.

3.3 Temporal Analysis

Although the CE dominates the errors that are captured in most component levels (see Table 1), the UER is the root cause to trigger HBM failures. In this analysis, we pay close attention to the properties of UERs. We first assess the impact of the number of cells with CEs on the occurrence of UERs at the row level. We then study the temporal correlation among UERs at the bank level. Our objective is to leverage the temporal correlation to guide the prediction designs (see §5), such that memory systems can take timely actions to cope with the upcoming UERs once receiving a UER alert. After that, we also investigate the temporal correlation among UERs and CEs, which is expected to figure out a way for the prediction of UERs based on historical CEs/UEs.

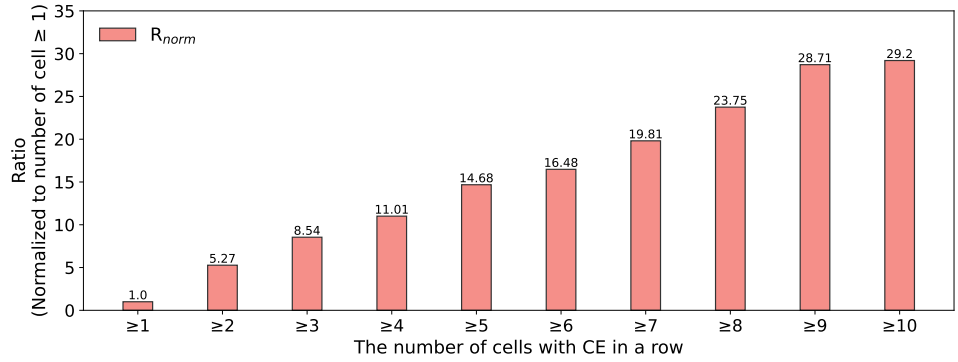


Fig. 6. Impact of the number of cells with CEs on the occurrence of UERs (§3.3). The figure shows the probability of UER occurrence in rows with different numbers of cells with CEs.

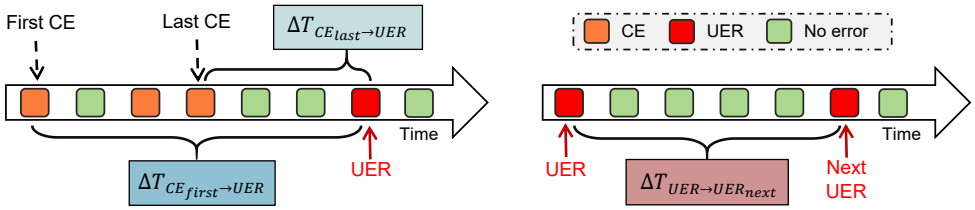


Fig. 7. Example of time between errors (§3.3). The figure illustrates the three intervals, including the time between the first/last CE and its subsequent UERs (i.e., $\triangle T_{CE_{\text{first}} \rightarrow UER}$ and $\triangle T_{CE_{\text{last}} \rightarrow UER}$), as well as the period between two successive UERs (i.e., $\triangle T_{UER} \rightarrow T_{UER_{\text{next}}}$).

Finding 8. *The more cells with CEs in a row, the greater the risk of encountering UERs.*

We first investigate the impact of the number of cells experiencing CEs (denoted as n_{CE}) on the probability of future UERs. Since banks that suffer from both CEs and UERs are more likely to exhibit row-related modes (33.1%) rather than column-related modes (21.26%), we focus primarily on the impact at the row level. We calculate the probability of UER occurrence when the number of n_{CE} in a row exceeds N (where N varies from 1 to 10). Figure 6 illustrates that as n_{CE} increases, the risk of encountering UERs also rises. The rationale behind this is that an increasing number of CE cells in a row leads to more complex error patterns, which the weakened ECC cannot correct, potentially resulting in UER occurrences. This analysis suggests that the number of CE cells at the row level can be used to assist in failure prediction (§5).

Bank with UERs: We then provide a statistical overview of the banks used in the analysis. Out of the 1,430 banks experiencing errors in total, about 91.0% (1,302 banks) have valid error events for analysis [2], where 525 banks have ever encountered UER(s). Among the 525 banks, we further identify that 219 banks only experience a single UER throughout the monitored time interval and hence are unqualified for the analysis. As a result, we conduct the correlation analysis over the remaining 306 banks, with 202 banks that contain UER(s) only and another 104 banks that emerge combinations of UER(s) and other error types.

Time between errors: After picking out the banks qualified for analysis, we start with the analysis on the time between failures. Recall that there are three error types (i.e., CE, UEO, and UER), where the occurrence time of the UEO is not accurately recorded, since it is captured by the periodic memory scrubbing over HBM chips (i.e., its occurrence time earlier than the recorded time). Hence,

[2]We exclude some error events because their timestamps fall outside the monitoring period.

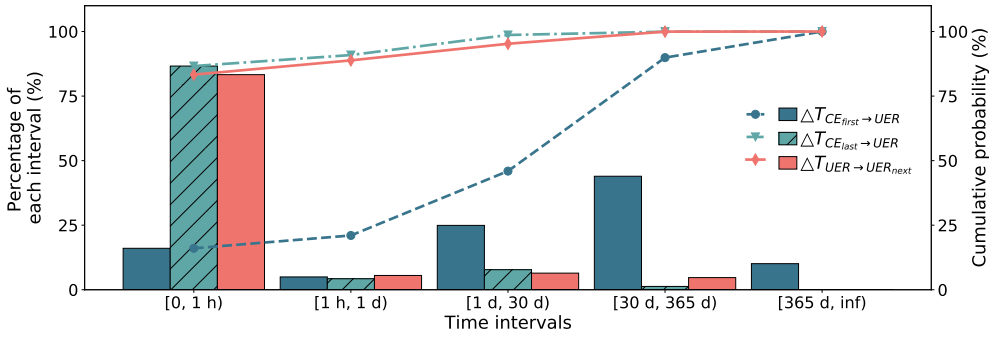


Fig. 8. Time between errors (§3.3). The figure illustrates the distribution and cumulative probability for each interval. The bars represent the distribution of $\triangle T_{CE_{\text{first}} \to UER}$, $\triangle T_{CE_{\text{last}} \to UER}$, and $\triangle T_{UER} \to T_{UER_{\text{next}}}$, while the lines depict their cumulative probability.

our analyses abandon the UEO information and aim to find the answers for the following questions: (i) how soon will the first CE [3] evolve to the first UER within a bank (i.e., $\triangle T_{CE_{\text{first}} \to UER}$ in Figure 7); (ii) how much time remains for the system administrator to prevent system crashes when the last CE occurs (i.e., $\triangle T_{CE_{\text{last}} \to UER}$ in Figure 7)? and (iii) how long will the next UER occur once encountering a UER (i.e., $\triangle T_{UER \to UER_{\text{next}}}$ in Figure 7)?

Finding 9. *There is a significant probability that two successive UERs occur within one hour.*

Figure 8 answers the above three questions by plotting the cumulative probability of different time intervals (i.e., $\triangle T_{CE_{\text{first}} \to UER}$, $\triangle T_{CE_{\text{last}} \to UER}$, and $\triangle T_{UER \to UER_{\text{next}}}$) between errors. We make the following observations.

First, two successive UERs within a bank may appear in one hour with a high probability (i.e., 83.3%). This observation indicates that once a UER is detected, the prediction algorithm should be frequently launched (e.g., performing the prediction within an hour) to infer the newly emerging UERs at the micro-level components (e.g., banks). Additionally, a remapping method (e.g., row-mapping [93]) is necessary to offline the page when a UER is observed and replicate data to a new page.

Second, we find that 16.1% of $\triangle T_{CE_{\text{first}} \to UER}$ and 86.6% of $\triangle T_{CE_{\text{last}} \to UER}$ are shorter than one hour. Besides, over 44.0% of $\triangle T_{CE_{\text{first}} \to UER}$ range from 30 days to 365 days. This phenomenon indicates that banks with both CEs and UERs may have encountered CEs as early as a long time ago (e.g., 365 days), and these banks may continue to experience CEs until the time approaches (e.g., one hour) the UER occurrence. Therefore, predicting UERs based on historical CEs seems like a promising approach. We make this attempt in §4.

Predictability of UERs. To further assess the predictability of UERs in HBM, we examine whether errors have occurred prior to the occurrence of UERs. We classify UERs into following two types.

- *Burst UER*: UERs occur without any preceding errors, and thus cannot be predicted using historical error logs.
- *Non-burst UER*: UERs occur after the occurrence of previous errors, suggesting the potential for prediction based on errors that have already occurred.

Finding 10. *A significant number of UERs occur following previous error events, suggesting that most UERs can be predicted using historical error logs.*

[3]Our analysis treats the first CE appeared in our dataset as the real first CE, but the real first CE may occur prior to the collection of the dataset.

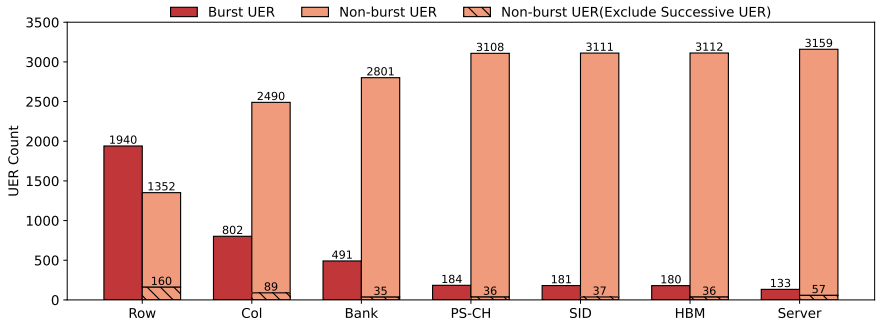


Fig. 9. Predictability of UERs (§3.3). The figure shows the number of burst UERs and non-burst UERs across different device levels. Furthermore, non-burst UERs are categorized into successive UERs and other non-burst UERs (with patterns shown in the figure).

Threshold	Servers with CE storms	Servers with no CE storm
10	26.47%	15.85%
20	21.73%	16.49%
30	37.50%	18.18%

Table 3. Impact of CE storm (§3.3). The table displays the percentage of servers experiencing UERs for those with CE storms compared to the servers without CE storms. The threshold values for identifying the occurrence of a CE storm are set between 10 and 30.

Figure 9 shows the number of UERs belonging to both types across different device levels. Specifically, burst UERs are more prevalent at the row level, while non-burst UERs are more common at other device levels. This can be attributed to the fact that column-related error modes are more commonly observed in HBM (Finding 6), implying that UERs tend to occur in the same column rather than the same row. Moreover, we observe that the same UER can be classified into different types across various device levels. For example, the number of burst UERs at the bank level is reduced by 311 compared to the column level, indicating that these 311 UERs have turned into non-burst UERs. Therefore, a hierarchical prediction approach is needed to identify UERs (§5.2). As non-burst UERs may occur after different error types, we count the successive UERs and the UERs that occur after CEs/UEOs, respectively. The results show that successive UERs account for 72.9–94.2% of non-burst UERs across different levels. Hence, statistics related to UERs (e.g., the number of UERs that have already occurred within a bank) should be used to predict future UERs (§5.1).

Finding 11. *Servers that have experienced CE storms may increase the probability of encountering UERs.*

Impact of CE storm. The CE storm refers to the phenomenon that numerous CEs on a server occur within a short period (typically one minute [22, 95]). In this analysis, we try to infer if the CE storm has an impact on the occurrence of UERs. We first classify the servers into two categories based on the occurrence of CE storms: (i) the servers with CE storms; and (ii) the servers with no CE storm. We then calculate the ratio of the servers that encounter at least an UER for the two categories. We vary the *threshold value* (defined as the number of CEs suddenly occurred in one minute) to identify the occurrence of a CE storm from 10 to 30 [21, 22, 90], and show the results in Table 3. We can discover that the servers with CE storms have a higher probability (11.7% on

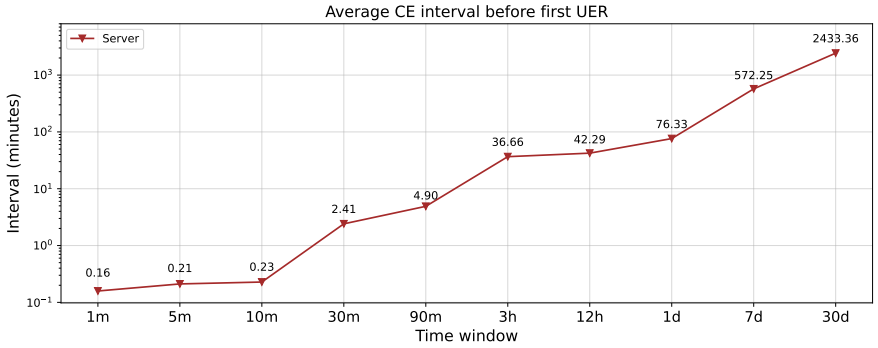


Fig. 10. The frequency of CE occurrences preceding UERs (§3.3). The figure depicts the average time interval between successive CE occurrences prior to the first UER on servers with varying time windows.

average) to produce UERs than the servers with no CE storm, indicating that the occurrence of CE storms can be leveraged for UER prediction.

In addition, we also find that configuring the threshold value to 30 can help identify the servers with UERs with the largest probability (37.50%), yet there is still a challenge to balance the trade-off between the accuracy (in predicting the upcoming UERs) and the sample size (i.e., the number of servers with CE storms) when selecting a threshold value to identify the CE storm. The rationale is that a larger threshold value will reduce the sample size, making the prediction accuracy more sensible to the variation of positive samples (i.e., the servers whose CE storms finally translate to UERs). For example, in our dataset, setting the threshold value to 30 makes the sample size smaller than 10; in this case, any change in the number of positive samples will introduce more than 10% of change to the ratio of the positive samples. Finally, in the dataset, we suggest configuring the threshold value to 10, which can achieve a considerable ratio (i.e., 26.47% in Table 3) with the most sample size.

Finding 12. *Servers tend to experience a higher frequency of CEs as UERs approach.*

We further analyze the frequency of CE occurrences prior to the first UER on servers. We define various time windows (ranging from 1 minute to 30 days) before the occurrence of UERs and calculate the average time interval between all pairs of successive CEs within each time window. Figure 10 illustrates the frequency of CE occurrences prior to the first UER under these different time windows. Our findings show that as the time window narrows towards the occurrence of the UER (i.e., smaller time windows), the average time intervals between successive CEs become shorter, indicating a more frequent occurrence of CEs. Specifically, the average time interval between successive CE occurrences is 2433.36 minutes within a 30-day window, but this decreases to 76.33 minutes within 1 day prior to the occurrence of UERs. Thus, there remains an opportunity to predict UERs at the server level by monitoring the frequency of CEs.

Finding 13. *The error modes change after the occurrence of the first UER.*

The previous findings in §3.2 introduced several error modes in banks. To further assess the change in error modes over time, we examine the error modes of banks before and after the occurrence of UERs. We only consider banks that encounter both CEs and UERs to draw more targeted conclusions. Table 4 shows that the distribution of error modes after the first UER occurrence is distinctly different from the error modes before encountering UERs. For example, banks that experienced single-row errors before the first UER occurrence are highly likely to transition to the two-row mode, rather than maintaining the original error mode. We suspect that the reason is that adjacent cells may begin to be affected by the error cells after the occurrence of the first UER. The

Pre \ Post	Single cell	Two cell	Single row	Two row	Single column	Two column	Row dominant	Column dominant	Irregular
Single cell	47.47%	10.34%	0.84%	3.80%	13.08%	2.11%	1.06%	3.80%	17.51%
Two cell	0%	0%	0%	0%	0%	0%	0%	0%	100.0%
Single row	0%	0%	12.50%	75.0%	0%	0%	12.50%	0%	0%
Two row	0%	0%	0%	100.0%	0%	0%	0%	0%	0%
Single column	0%	0%	0%	7.14%	50.0%	42.86%	0%	0%	0%
Two column	0%	0%	0%	0%	0%	100.0%	0%	0%	0%
Row dominant	0%	0%	0%	0%	0%	12.50%	75.0%	12.50%	0%
Column dominant	0%	0%	0%	0%	0%	0%	0%	100.0%	0%
Irregular	0%	0%	0%	50.0%	0%	10.0%	30.0%	0%	10.0%

Table 4. Percentage of error modes before and after the occurrence of the first UER (§3.3). The table presents the transformation of each error mode after the occurrence of the first UER.

results suggest that we should adjust the prediction model after banks encounter UERs (e.g., using a dual-model approach for prediction in §5.3).

3.4 Analysis of Sensor Information

In addition to the error logs, we also collect auxiliary information from sensors that report the power and temperature information every ten minutes. Prior studies have revealed that sensor information (e.g., temperature [45] and voltage [46]) has an impact on HBM errors and can be a benefit for failure prediction [30], we also analyze the impacts of both temperature and power on the error occurrence.

Impact of power: We first assess the impact of the power on the emergence of UERs. Each entry of the power log contains two pieces of information: (i) the average and the peak power in the last ten minutes; and (ii) the transient power at the time of the collection. Since the latter only reflects the power at a certain time point, we use the average and the peak power in the following analysis.

Finding 14. *The average and peak power of a server both exhibit a rapid increase when approaching the error occurrence.*

To learn the impact of the power, we first pick out the servers that have ever experienced either CEs or UERs. For each error event, we examine a specific time interval—seven days for the UER and thirty days for the CE— preceding the occurrence of the error. As we collect power information every ten minutes, we roughly depict the trends of peak power and average power for both CE events and UE events in Figure 11.

Figure 11(a) shows the power trends of servers encountering UERs. Both peak power and average power exhibit a noticeable increase as time approaches the occurrence of UERs. More specifically, we observe that the power starts to rise around the average power of the server 31 hours prior to the UER events and surpasses the average power of the server 15 hours ahead of the occurrence of

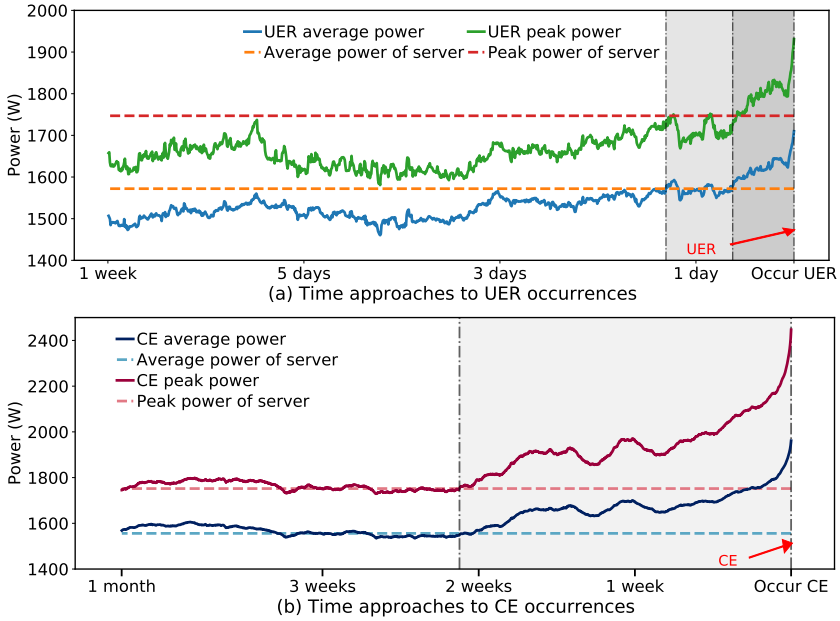


Fig. 11. Impact of power (§3.4). The figures show both peak power and average power as the time approaches (a) the occurrence of UERs and (b) the occurrence of CEs for the servers.

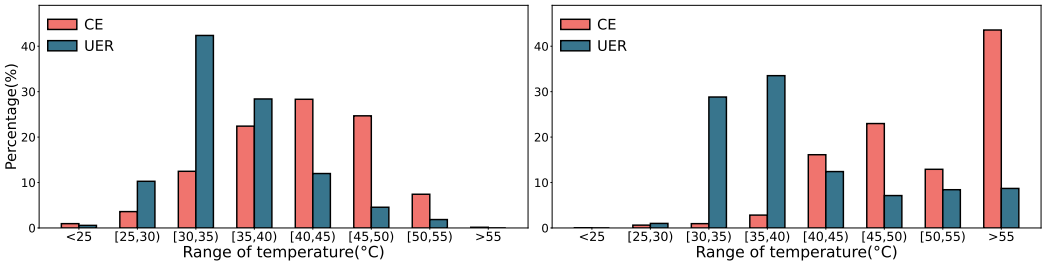


Fig. 12. Impact of temperature (§3.4). The figures illustrate the distribution of (a) average temperature and (b) maximum temperature within one day preceding the occurrence of CEs and UERs, respectively.

UERs. As the power may increase as the time approaches the occurrence of UERs, we can utilize this as a feature to predict potential UERs.

We also analyze the power trends of servers experiencing CEs in Figure 11(b). The power exhibits a more significant increase as time approaches the occurrence of CEs compared to UERs. We attribute this to the fact that CEs are typically triggered by soft errors resulting from memory-intensive access (e.g., RowHammer [43, 71]). In contrast, UERs are usually caused by hardware errors (e.g., TSV failures [66]). Given that memory-intensive access tends to increase server power [94], the average power of servers increases as time approaches the occurrence of CEs.

Finding 15. *The temperature distribution of CEs and UERs exhibits significant differences before their occurrence.*

Impact of temperature. We then analyze the influence of temperature on the occurrence of CEs and UERs. We examine the temperature distribution within one day preceding the error events

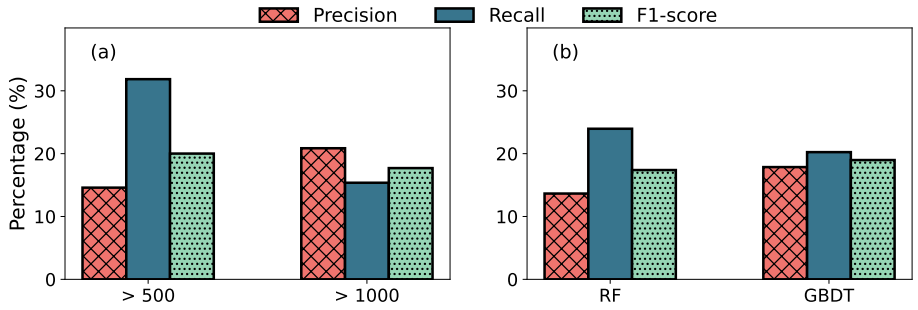


Fig. 13. Unsuccessful attempts. (§4). The figures show the precision, recall, and F1-score for (a) CE rate indicator and (b) CE-based predictor, respectively. We evaluate the CE rate indicator using two thresholds (i.e., 500 and 1000) and implement the CE-based predictor with RF and GBDT models.

for the DSA devices that have experienced errors. For each DSA device, we calculate both the average temperature and the maximum temperature one day before the occurrence of CEs or UERs. Figure 12 illustrates the temperature distributions for the DSA devices with errors. We observe that the average temperature for DSA devices with CEs is higher than that in DSA devices with UERs. Specifically, 60.57% of the DSA devices with CEs have an average temperature (refer to Figure 12(a)) exceeding 40℃ one day before the CE events, while the average temperature of DSA devices with UERs is more prevalent (70.76%) within the range of 30℃ to 40℃. Furthermore, DSA devices with CE events are more likely (58.92% higher) to experience a maximum temperature exceeding 40℃ compared to the DSA devices with UERs. We speculate that the reason is that higher temperatures increase charge leakage [79], which may lead to data loss but can be restored by the ECC (i.e., recorded as a CE). Hence, we observe a higher temperature as the time approaches the occurrence of CEs.

Summary: We observe that both temperature and power correlate with the occurrence of errors, though their contributions to HBM errors vary. We find that power does not directly contribute to error occurrences, and this conclusion arises from two primary observations. First, although power consumption increases rapidly as the error occurrence approaches, this increase exhibits a long-term trend (lasting several hours or even up to two weeks), indicating that the impact of power on errors is not straightforward. Second, frequent memory access can lead to elevated power consumption, which in turn increases the likelihood of soft errors that cause CEs. This observation aligns with our finding that the power level prior to CE occurrences is higher than that observed before UERs. In contrast, temperature increases directly correlate with error occurrence, as higher temperatures can lead to increased charge leakage [79], potentially resulting in errors.

4 UNSUCCESSFUL ATTEMPTS

In prior studies [15, 24], historical CEs have been utilized to predict future UERs. Our analysis in §3.3 also indicates the promise of using CEs for predicting potential UERs. As our objective is to precisely predict potential UERs, we employ two methods based on historical CEs in this section.

4.1 Attempt 1: CE Rate Indicator

We first employ a CE rate indicator (also utilized in previous studies [22, 23, 51, 62, 95]), which utilizes the CE rate to indicate whether a UER occurs in the near future. Specifically, the CE rate indicator monitors the number of CEs (denoted as N_{CE}) within the past one day. If N_{CE} exceeds a predefined threshold (e.g., 500 [24]), we consider that a UER may occur in the future. Given that

Level	Time window	(1) Average number of CEs		(2) Percentage of all CE/UER	
		CEs -> CE	CEs -> UER	CEs -> CE	CEs -> UER
Server	1h	9.21	0.48	95.79%	15 %
	1d	35.74	0.99	98.57%	15.74%
	7d	51.63	1.59	99.19%	16.8%
	30d	79.88	3.03	99.48%	17.5%
HBM	1h	6.99	0.43	95.75%	14.98%
	1d	28.04	0.88	98.54%	15.7%
	7d	43.92	1.35	99.15%	16.52%
	30d	69.65	2.49	99.42%	17.01%
Bank	1h	6.07	0.17	95.66%	14.88%
	1d	23.75	0.36	98.45%	15.52%
	7d	36.98	0.55	99.07%	16.19%
	30d	58.47	1.02	99.37%	16.59%

Table 5. Characterization of CE and UER occurrence (§4). The table displays (1) the average number of preceding CEs within four different time windows across three levels before the occurrence of CE and UER, and (2) the percentage of CEs and UERs that encounter CEs before their occurrence within four different time windows across three levels.

errors are common across multiple components within a bank (Finding 2 in §3.2), we focus the prediction at the bank level.

Limitation. Although we try our best effort to adjust the threshold of the CE rate indicator, its performance is still unsatisfactory (e.g., averaging 20.5% in Figure 13(a)). We suspect that the limited correlation between the CE rate and UERs is the underlying reason, given the significant gap between the total number of CEs (more than 460 million) and the total number of UERs (only 3,334).

Finding 16. *Numerous consecutive CEs may occur without UER in HBM, whereas CEs typically before UER in DRAM.*

To confirm our conjecture, we analyze the number of CEs within different time windows before errors across various device levels. We vary the time window from 1 hour to 30 days and only consider errors that occur after CEs. Given that the occurrence time of the UEO is not accurately recorded (§2.3), we exclude UEOs from this analysis. Table 5 illustrates the number of CEs before different error types (i.e., CE and UER) and the percentage of these errors in the total. The results highlight two factors that contribute to the inefficiency of the CE rate indicator. First, although the frequency of CE occurrences increases as the time approaches UERs (Finding 12 in §3.3), the frequency is higher (i.e., the occurrence of more CEs) as the time approaches the occurrence of CEs. Specifically, the number of CEs prior to CEs is 36.4× higher on average compared to the number of CEs before UERs. This differs from DRAM failures, where the number of CEs in failure servers is an order of magnitude higher than that of normal servers [95]. Second, the performance of the CE rate indicator depends on the percentage of UERs that occur after prior CEs. However, only 17.5% of UERs occur after CEs, which limits the maximum number of UERs that can be predicted using the CE rate indicator. Therefore, utilizing the CE rate to predict subsequent UERs proves ineffective.

4.2 Attempt 2: CE-based Predictor

The limitation of our first attempt lies in solely considering the number of CEs. To address this, we employ a CE-based predictor that leverages component features related to CEs (including

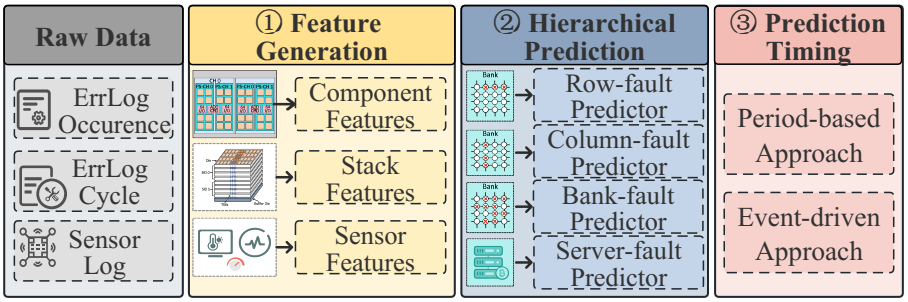


Fig. 14. The overview of Calchas (§5). Calchas extracts features from raw data (①) and feeds them into hierarchical predictors (②). Finally, Calchas employs different prediction timings at different levels (③).

numbers of components with CEs at different device levels) for predicting upcoming UERs. We utilize machine learning algorithms for the CE-based predictor, which is also applied in previous studies [10, 15, 20, 88]. The prediction is made at the bank level. For the predictor, we utilize the all features related to CEs to predict the upcoming UERs. Figure 13(b) presents the results of the CE-based predictor when employing Random Forest (RF) [12] and Gradient Boosting Decision Tree (GBDT) [29] models. On average, the precision, recall, and F1-score of the CE-based predictor are 22.2%, 15.4%, and 18.0%, respectively.

Limitation. The performance of CE-based predictors is disappointing in our dataset, even when considering different models (e.g., RF and GBDT) and various configurations (e.g., prediction windows). The reasons for this are twofold. First, we observe that only a small fraction of banks (11.2%) exhibit a combination of UERs and CEs. Second, the banks experiencing both CEs and UERs are commonly distributed across single or multiple rows. As a result, the CE-based predictor frequently mislabels normal banks. This suggests that we need to leverage additional information (e.g., UERs and UEOs) that has been recorded and consider row-level predictors for banks with multiple errors for failure predictions.

Summary: The approaches mentioned above indicate that considering only CE-related features fails to achieve effective prediction. Specifically, HBM does not frequently encounter CEs (or may not encounter CEs at all) before UER occurrences (Finding 16), making the use of only CE-related features insufficient for predicting future UERs in HBM. Therefore, although we apply the same method proposed for predicting UERs in traditional DRAM, it still does not perform well. This inspires us to reconsider the strategy for HBM failure prediction.

5 CALCHAS DESIGN

With the insights obtained from previous analyses and attempts, we propose Calchas, a hierarchical, comprehensive, and non-intrusive failure prediction framework for HBM. Calchas can use the row-level, column-level, and bank-level predictors to identify the upcoming UERs for micro-level components in time. It also employs the server-level predictor to periodically predict the potential server-level failures.

Overview of Calchas. Figure 14 illustrates the workflow of Calchas, which includes three successive steps. We first collect the raw error logs from the data centers and generate a sample of features (e.g., the number of cells with UERs) to perform prediction (§5.1). Next, we use the generated features to train four hierarchical predictors (i.e., row-level, column-level, bank-level, and server-level predictors §5.2). We finally select a suitable approach to trigger prediction (§5.4).

5.1 Feature Generation

Before applying prediction models, we first generate representative features for model training. Recall that the collected logs contain the following information: (i) the addresses of errors, (ii) the time of error occurrence, (iii) the error types, and (iv) the sensor logs (see §3.1). Here, we generate 43 features in total from the raw logs and classify them into three categories.

- *Component features*: We first count the number of components experiencing errors (falling into one of three error types) across nine device levels (i.e., cell, row, column, bank, BG, PS-CH, SID, HBM, and DSA). Since different error types within a component may exhibit distinct error modes (Finding 6 in §3.2), we then present statistics for the number of components with CEs, UEOs, and UERs, respectively. We finally assess whether the servers have encountered a CE storm in the past. In total, we consider 37 component features.
- *Stack features*: We have confirmed that the 3D-stacked structure of HBM has an impact on the error emergence, where the SID near the buffer die is more likely to appear errors (Finding 4 in §3.2). Hence, we convert the positions of SIDs into features using one-hot encoding [80] and generate two stack features.
- *Sensor features*: We obtain four sensor features based on the values of maximum temperature, average temperature, peak power, and average power. Given the significant differences in temperature and power before error occurrences (Findings 14 and 15 in Section 3.4), we generate two features from the temperature sensors and another two features from the power sensors.

We generate a collection of features when performing predictions for various predictors. Specifically, we choose component features and sensor features to predict potential UERs at the server level. Furthermore, we turn to use component features, sensor features, and stack features to predict errors at the micro-level components (e.g., rows, columns, and banks).

5.2 Hierarchical Prediction

After generating the features, we start the model training for prediction. Recall that the correlations between the features and the errors may vary at different levels. For example, we observe that errors are common across multiple components within a bank (Finding 2 in §3.2), whereas the CE storms may increase the likelihood of the emergence of UERs in a server (Finding 11 in §3.3). Hence, we establish hierarchical predictors to forecast the upcoming UERs as follows:

- *Row-level predictor*: Given that the row-related error modes are prevalent (52.4%) when multiple error types are observed within the same bank (§3.2), we build a row-level predictor to forecast UERs within a row.
- *Column-level predictor*: We have pointed out that the column-related error mode is more prevalent than the row-related error mode (Finding 6 in §3.2). Hence, we establish the column-level predictor to capture the potential UERs within a column.
- *Bank-level predictor*: We also build a bank-level predictor based on the observation that a bank, which has ever experienced an error before, is more likely to appear errors in the future (Finding 9 in §3.3). This bank-level predictor can work cooperatively with the row-level and column-level predictors to help uncover two-cell and irregular error modes of banks.
- *Server-level predictor*: We finally establish a server-level predictor that leverages the impact of the CE storms (Finding 11 in §3.3) and auxiliary sensor information (Findings 14 and 15 in §3.4), with the aim of capturing the symptom of server failure.

Specifically, if we detect a UER within a *prediction window* (i.e., the time interval for which a prediction is made) at a specific device level, we label the sample as positive; otherwise, it is

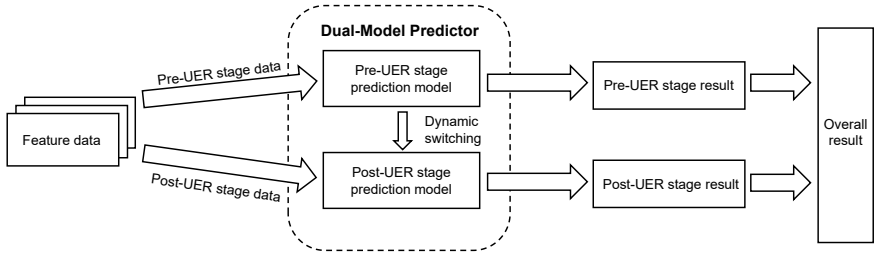


Fig. 15. The workflow of dual-model predictor (§5.3). The dual-model predictor employs a two-stage prediction approach: a pre-UER model for failure prediction before UERs occur, and a post-UER model for prediction after UERs are observed. It synthesizes the results of dual-model predictors from different levels to determine if a UER will occur.

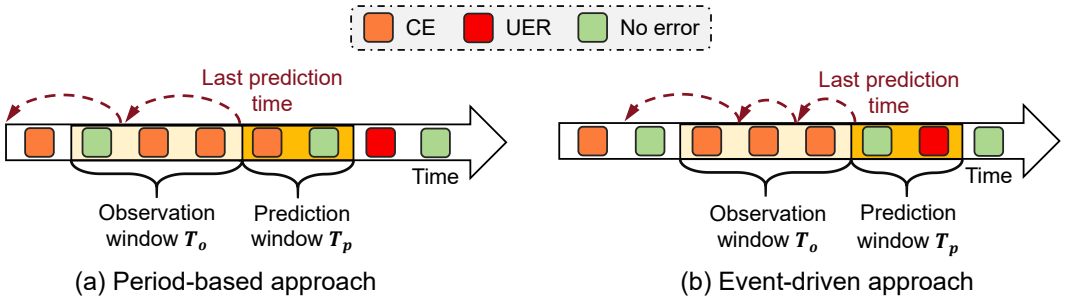


Fig. 16. The methods used to trigger predictions (§5.4). The figures depict (a) the period-based approach, which triggers prediction at fixed intervals ($\triangle T_p$) based on logs collected from the past observation window (i.e., $\triangle T_o$); and (b) The event-driven approach, which triggers prediction upon the observation of an error event, also utilizes logs recorded in the past observation window to generate features.

considered as negative. For each predictor, we initiate the process by generating a set of features and subsequently training a model for prediction.

5.3 Dual-model Predictor

Due to the changing characteristics of HBM errors (e.g., error modes and temporal locality in §3) after UER(s), we employ a two-stage prediction approach at each level to accommodate different error patterns. Specifically, Calchas extends each level's predictor from a single model to a dual-model: a pre-UER model for failure prediction before UERs occur, and a post-UER model for prediction after UERs are observed. For training the dual-model, we split the dataset into two shares: (i) a pre-UER dataset containing error logs before the first UER, with the first UER used to label the feature collections (§5.1); and (ii) a post-UER dataset containing logs after UERs have recorded. We then use these two shares of datasets to train the pre-UER model and the post-UER model, respectively. When triggering a prediction, we first generate a feature collection and check if a UER has been recorded. If a UER has observed, the post-UER model is used; otherwise, the pre-UER model is employed. Figure 15 illustrates the dual-model training and prediction process. For example, if a CE occurs in a column that has experienced UER(s), but the CE is not in the same row as the UER(s), we will use the pre-UER model for row-level prediction, as the row has not encountered a UER. For column-level prediction, the post-UER model will be applied instead.

5.4 Prediction Timing

While hierarchical predictors can be employed to predict potential UERs, a challenge arises: when is the appropriate time to make these predictions? To answer this question, we explore two approaches for triggering predictions as follows.

- *Period-based approach* [15, 93]: It refers to performing prediction periodically. Figure 16(a) shows the period-based approach, which comprises two windows: an *observation window* (represented by $\triangle T_o$), which denotes the duration of the past period under observation for prediction, while a prediction window (represented by $\triangle T_p$), which represents the time interval during which a failure can be predicted. To monitor each time interval until a UER alarm, we set the period of performing prediction the same as the prediction window.
- *Event-driven approach* [92, 93]: It refers to the prediction triggered by error events. As shown in Figure 16(b), the event-driven approach also utilizes the same two windows as the period-based method. Given the potential for frequent errors in HBM (such as CE storms in §3.3), we constrain the minimum interval between two predictions to alleviate the overhead of prediction.

The period-based approach relies on fixed intervals for feature generation and failure prediction, which benefits the capture of abnormal sensor metrics but lacks flexibility in handling irregular events [92]. Conversely, the event-based approach provides a rapid response to errors once they occur. However, it is ineffective when employed for a long-term prediction (e.g., a 16-hour observation window in [92]). We utilize the advantages of both approaches in Calchas. Specifically, we adopt the period-based approach for server-level predictor to monitor the fluctuation of sensor metrics. Meanwhile, we employ the event-driven approach to predict micro-level component failures for leveraging error events in time.

6 EVALUATION

We evaluate Calchas using a dataset collected over a monitoring period spanning two years. Our findings are summarized as follows.

- Calchas achieves an average precision of 58.0%, recall of 72.6%, and F1-score of 64.7% across different prediction levels (Exp#1).
- Calchas demonstrates better prediction performance when using the random forest model (Exp#2), which is thus adopted as the default model throughout the experiments.
- The dual-model predictors of Calchas enhance the effectiveness of UER detection (Exp#3) and keep effective in both the pre-UER and post-UER stages (Exp#4).
- The sizes of the observation window (Exp#5), prediction window (Exp#6), and the lead time (Exp#7) significantly affect prediction performance.

6.1 Experimental Setup

To assess the effectiveness of Calchas, we split the dataset (see §2.3) into a proportion of 7:3, with 70% used for training the model and the remaining 30% for testing. We measure the following accuracy metrics [33, 91].

- *Precision* : The ratio of correctly predicted UER events to the total number of events predicted as UERs.
- *Recall* : The ratio of correctly predicted UER events to the total number of actual UER events.
- *F1-score* : The harmonic mean of precision and recall [27], providing a balanced measure of both metrics.

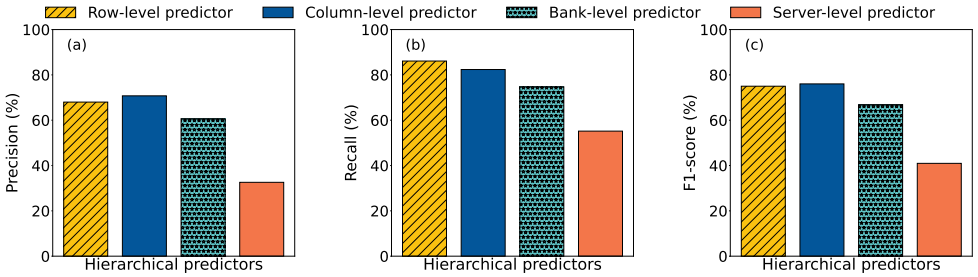


Fig. 17. Exp#1 (Performance of Calchas). The figures show the (a) precision, (b) recall, and (c) F1-score of Calchas.

We first study the performance for hierarchical predictors of Calchas (Exp#1, Figure 17). We then evaluate three machine learning algorithms (Exp#2, Figure 18), including Random Forest (RF) [12], Support Vector Machine (SVM) [61] and Gradient Boosting Decision Tree (GBDT) [29]. As the result in Exp#2 shows that RF has better performance, we eventually choose RF as default machine learning algorithm for Calchas. We also compare the performance of the dual-model predictor with the single-model predictor (Exp#3, Figure 19) and conduct a stage-wise breakdown of the dual-model predictor to demonstrate its effectiveness (Exp#4, Figure 20). Since the only difference between the two predictors lies in the use of two prediction stages, we leverage the single-model predictor when assessing the performance of Calchas (Exp#1) and the impact of machine learning algorithms (Exp#2) for a more concise explanation. These algorithms are implemented in Python (v3.6.8) using the scikit-learn library (v0.24.2) [75].

We employ a one-day observation window and a one-hour prediction window for predicting the potential UERs in micro-level components to leverage the finding that two successive UERs may occur within a short-term period (Finding 9 in §3.3). As error events trigger the micro-level prediction (§5.4), we set a minimum interval to perform prediction as five minutes [15, 93]. For server-level prediction, the observation window and prediction window are set to thirty days and one day, respectively, aiming to better capture the long-term impact of power and temperature (§3.4). Additionally, we explore the impact of various sizes (ranging from one hour to thirty days) for observation windows in Exp#5 (Figure 21) and prediction windows in Exp#6 (Figure 22), respectively. Since memory-sparing techniques (e.g., row remapping and page offline [93]) require time to prevent server crashes after a failure is predicted, we further consider the time interval between performing failure prediction and the actual failure occurrence (i.e., lead time [92, 93]). We assess the impact of lead time by varying it from 0 to 30 minutes in Exp#7.

6.2 Evaluation Results

Exp#1 (Performance of Calchas). We first evaluate the performance (including precision, recall, and F1-score) of the four hierarchical predictors of Calchas. Figure 17 illustrates the precision, recall, and F1-score of predictors across different device levels. Specifically, we achieve an average precision of 58.0%, recall of 74.6%, and F1-score of 64.7% across various predictors. In Figure 17(a) and (c), we observe that the column-level predictor (i.e., the second bar) outperforms the other three predictors in the precision (70.8%) and F1-score (76.0%). These high precision and F1-score indicate that column-level predictors rarely mislabel normal columns as failures. The reason is that column-related faults are more prevalent in HBM banks (Finding 6 in §3.2). We can identify the column-related modes by counting the number of cells within the column (a piece of component features) to improve the prediction. Moreover, we notice that the row-level predictor (i.e., the first bar) provides a high recall (82.4%) in Figure 17(b). Given the prevalent use of row isolation methods

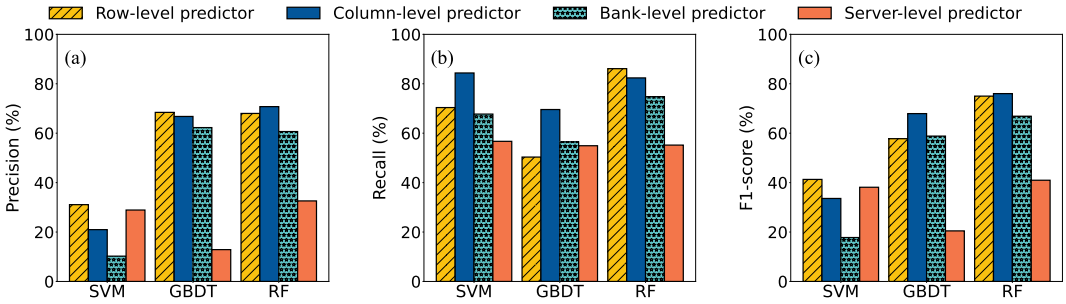


Fig. 18. Exp#2 (Impact of prediction models). The figures show the (a) precision, (b) recall, and (c) F1-score of four predictors when using the **SVM**, **GBDT**, and **RF**, respectively.

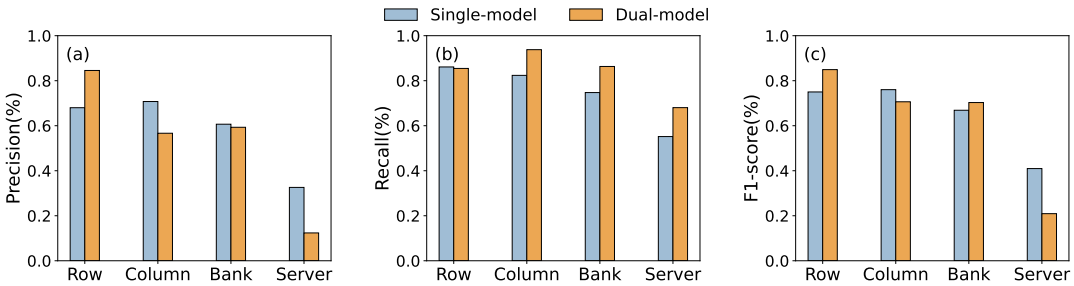


Fig. 19. Exp#3 (Effectiveness of the dual-model predictor). The figures show the (a) precision, (b) recall, and (c) F1-score of single-model predictor and dual-model predictor across four device levels.

in deployment systems [69, 93], the high recall can facilitate the advanced isolation of the faulty row.

Exp#2 (Impact of prediction algorithms). We then examine the impact of prediction models by implementing three classical machine learning algorithms, including RF [12], SVM [61], and GBDT [29]. Figure 18 shows the results of utilizing different machine learning models for each of the four hierarchical predictors. Our observations are as follows. First, while predictors based on SVM achieve high recalls (69.8% on average), their precision falls below 31.1%. Hence, the predictors based on SVM may bring a considerable probability of mislabeling the normal components and servers as failures. Second, the tree-based models (i.e., RF and GBDT) achieve a 25.3% higher F1-score for predicting the upcoming UERs in the micro-level components (i.e., rows, columns, and banks) compared with the SVM. Moreover, we also observe that RF achieves a high precision, recall, and F1-score. These results are aligned with the existing studies in predicting failure in traditional DRAM [10, 15]. Therefore, we select it as the default model for each predictor.

Exp#3 (Effectiveness of the dual-model predictor). To investigate the effectiveness of the dual-model predictor, we compare it with the single-model predictor across various device levels. Figure 19 shows the precision, recall, and F1-score for both predictors at different device levels. We make the following observations. On the one hand, the dual-model predictor achieves a higher recall at most levels (excluding the row level), outperforming the single-model predictor by an average of 12.0%. This indicates that the dual-model predictor is more effective in detecting UERs. The reason for this is that the dual-model predictor adapts to the changing characteristics of HBM errors after the first UER occurrence, allowing it to detect more UERs. On the other hand, we also observe that the precision of the dual-model predictor at the column, bank, and server levels is

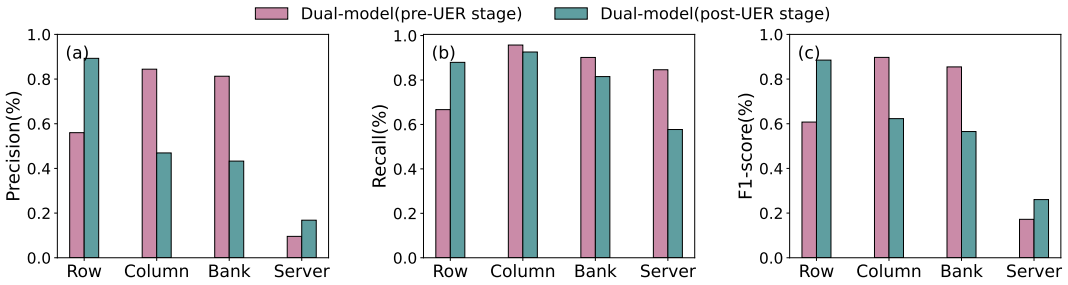


Fig. 20. Exp#4 (Breakdown of the Prediction Stages for the Dual-Model Predictor). The figures show (a) precision, (b) recall, and (c) F1-score for both the pre-UER and post-UER stages of the dual-model predictor across various device levels.

14.10%, 1.36%, and 20.23% lower than that of the single-model predictor. This can be attributed to the fact that the dual-model predictor inevitably introduces more mislabels as it expands coverage to accommodate a wider range of error characteristics.

However, since HBM is widely used in DSA devices (e.g., NVIDIA A100 [1] and H100 [70]), where failures can result in significant economic losses (e.g., 3 million dollars per month for a 10,000-GPU job [31]), the higher recall of the dual-model predictor reduces the probability of encountering failures in DSA devices, thereby decreasing the interrupt cost in deployment environments.

Exp#4 (Breakdown of the prediction stages for the dual-model predictor). We decompose the prediction stage of the dual-model predictor and measure the prediction performance at both the pre-UER and post-UER stages. Figure 20 shows the results for different device levels. We observe that both column and bank level predictions exhibit higher precision, recall, and F1-score at the pre-UER stage. Specifically, the pre-UER stage of the dual-model predictor achieves 37.50%, 5.88%, and 28.18% higher precision, recall, and F1-score compared to the post-UER stage.

Exp#5 (Impact of the observation window). We further study the impact of the observation window by varying its interval from one hour to thirty days. Figure 21 depicts precision, recall, and F1-score for different observation windows across various predictors, including single-model and dual-model predictors. We make the following observations. First, our results reveal that both single-model and dual-model predictors exhibit the highest F1-score when the observation window is set to one day at micro-level (i.e., row, column, and bank), indicating that one-day observation window is universally appropriate for micro-level. Thus, we employ an observation window interval of one day to achieve a better trade-off between precision and recall. Second, we observe that the performance of the server-level predictor significantly varies with the change of prediction windows. We suspect the reason is that the prediction at a server level requires a long-term monitor to capture the variation of sensor metrics. Besides, similar to the results observed in Exp#3, the recall of dual-model predictors in different levels significantly outperforms than that of single-model predictors across various observation windows.

Exp#6 (Impact of the prediction window). We next assess the impact of the prediction window. We vary it from one hour to thirty days for the four device level predictors. Figure 22 illustrates the performance of single-model and dual-model predictors with different prediction windows across hierarchical levels. A notable observation is that the performance of the single-model predictor in the row-level and column-level deteriorate when the prediction window is set to one day. We suspect the reasons are twofold. First, there is a slight probability (5.53%) of the occurrence of two successive UERs within the time interval ranging from one hour to one day. Second, the time intervals between CEs and their subsequent UERs are observed to be more prevalent within one

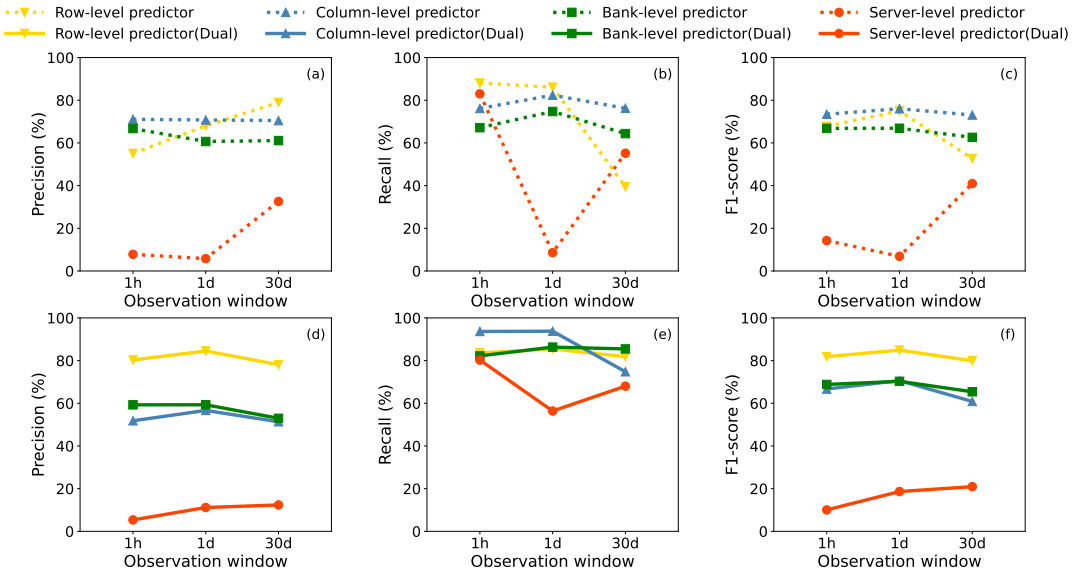


Fig. 21. Exp#5 (Impact of the observation window). The figures show the performance of different predictors, including single-model (figure (a), (b) and(c)) and dual-model (figure (d), (e) and (f)) when the observation window is set to one hour (**1h**), one day(**1d**), and thirty days (**30d**).

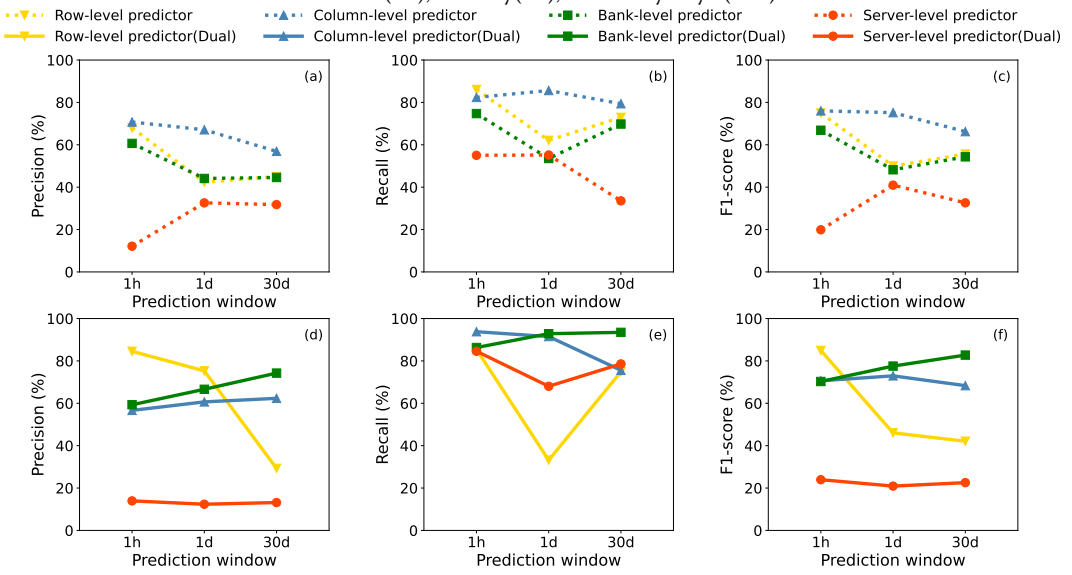


Fig. 22. Exp#6 (Impact of the prediction window). The figures show the performance of different predictors, including single-model (figure (a), (b) and(c)) and dual-model (figure (d), (e) and (f)) when the prediction window is set to one hour (**1h**), one day(**1d**), and thirty days (**30d**).

hour (51.3%) and more than one day (44.0%). Therefore, compared to a one-day prediction window, the performance improves when adopting a prediction window of one hour or thirty days.

Exp#7 (Impact of the lead time). We measure the impact of lead time by varying its interval from 0 to 30 minutes. Figure 23 shows the performance of the two predictors with different lead times. We

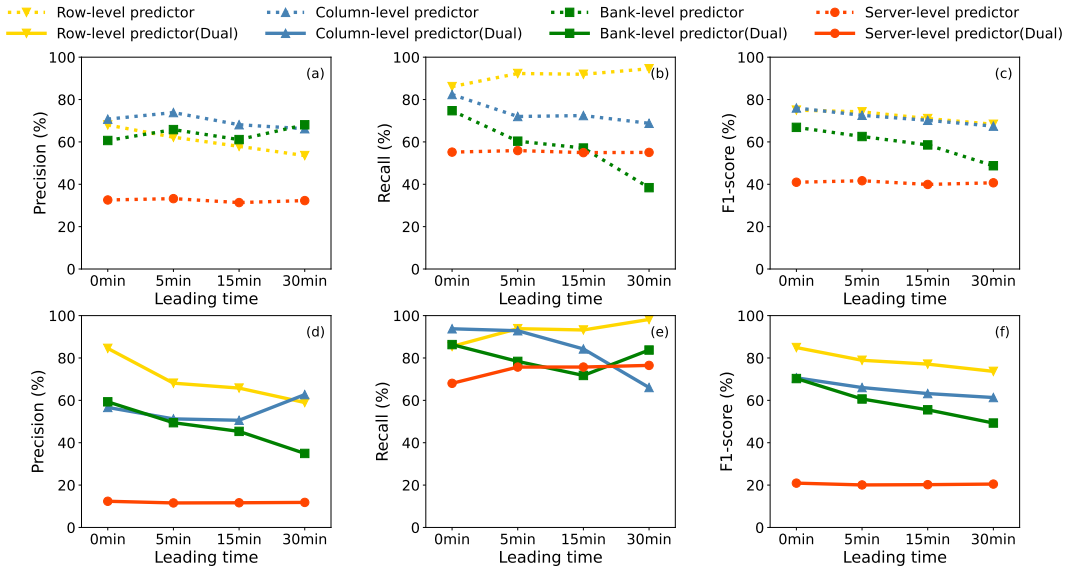


Fig. 23. Exp#7 (Impact of the lead time). The figures show the prediction performance of different predictors, including single-model (figure (a), (b) and(c)) and dual-model (figure (d), (e) and (f)) when the leading time is set to zero minute (**0min**), five minutes (**5min**), fifteen minutes (**15min**), and thirty minutes (**30min**).

find that as the lead time increases, the F1-score of both the single-model and dual-model predictors at the micro-level components (i.e., rows, columns, and banks) decreases, indicating that longer lead times degrade overall prediction performance. Additionally, we observe that the prediction performance at the server level remains stable across different lead times for both predictors. We suspect that this stability is due to the server-level prediction approach, which is period-based and triggers predictions at regular intervals (e.g., every 24 hours), making it less sensitive to the variations introduced by lead time.

7 DISCUSSION

7.1 Fault Tolerate

We offer an in-depth analysis of errors in HBM and introduce Calchas for predicting the upcoming UERs at different device levels. In this section, we will explore effective strategies to ensure reliability and availability following prediction.

EC-based operator. To enhance the reliability of micro-level components predicted to experience future UERs, we suggest utilizing EC-based operators, which integrate operators with erasure coding [34, 58]. Specifically, if Calchas predicts an upcoming UER in a specific memory region, we continuously monitor this region and employ EC-based operators when writing data to it. This approach allows us to avoid suspending the training task and enables data recovery using the parity of EC when UERs actually occur. We have implemented EC-based operators on servers with CPUs, observing only a 3% performance overhead in the CPU. Given that DSA devices possess stronger parallel computing capabilities [1] and higher bandwidth for data access [39] than CPUs, the overhead of EC-based operators may be even lower than that on CPUs. Therefore, implementing EC-based operators in DSA devices is feasible and has a negligible performance degradation.

Transparent migration. The DSA devices in multiple servers are interconnected using specific techniques (e.g., NVLink [52]). In the event of a server failure, the loss of parameters may occur across all servers. Therefore, it is essential to implement a reliability scheme once potential failures in the servers are predicted. As Calchas achieves a long-term monitor (i.e., thirty days) for prediction at the server level, we have enough time to deal with the upcoming failures. Therefore, we suggest adopting an elastic scaling approach, initiating an upward scale (i.e., increasing the number of servers for training tasks) upon predicting a potential failure. Following the scaling operation, we identify and remove servers that may occur failures. This strategy ensures that the training performance remains unaffected, even in cases where Calchas mislabels a normal server.

7.2 Limitations

Collection challenges. As a service provider, we face challenges in obtaining detailed information about the workload, a factor known to influence failures according to existing studies [25, 30]. To address this gap, we explore the impact on power and identify it as a viable metric for assessing the workload (§3.4). Real-time collection of sensor information for each error event is impractical due to hardware overhead; thus, we collect sensor data collection every ten minutes. Fortunately, our observation (§3.4) reveals that both temperature and power exhibit long-term (e.g., one day in Finding 9) effects on UERs. As a result, collecting sensor information every ten minutes can also be beneficial for predicting upcoming UERs.

Generalizability. The findings are derived from the analysis of HBM2 in our data centers; therefore, their generalizability is limited to HBM standards. Although HBM2 is commonly utilized in commercial DSA devices (e.g., NVIDIA V100[68], NVIDIA P100 [67], and AMD MI50 [3]), it may not be applicable in cases where HBM2E is considered [1]. We specifically focus on analyzing the impacts of common features (3D-stacked structures and sensor metrics) of different HBM standards on failures. Similar findings may be applicable across various HBM standards.

8 RELATED WORK

HBM failure study. With HBM being extensively employed in commercial DSA devices, both academia and industry have focused their attention on exploring the impacting factors of HBM failures. Several studies [45, 46] pay close attention to exploring the impact of operating conditions; they find the rising temperature causes increment of retention errors [45] and a higher rate of bit flips when reducing the voltage [46]. Researchers [38, 47, 49] from SK Hynix unveil that package joints may affect the reliability of HBM. Moreover, TSV structure also brings new faults [5, 50, 66], with potential impacts on large portions of memory [66]. Similar to DRAM dies, HBM dies also face severe disturbance issues (e.g., RowHammer [43, 71], RowPress [60, 72]) that degrade reliability. Furthermore, disturbance errors may vary among different dies and channels [71, 72].

Differing from existing research that studies the characteristics of HBM to guide hardware design, our work provides an in-depth analysis based on datasets collected from deployed DSA devices. Additionally, we propose prediction methods to assist service provider in identifying failures in advance.

DRAM failure analysis. Given the escalating severity of faults and errors in DRAM, substantial efforts have been made to analyze the root causes of faults [16, 35, 55, 56]. Memory faults can be caused by various factors, such as particle strikes [56], cosmic ray [35], and defects in the memory circuit [55]. Prior work [8, 62, 78, 79] also analyses the DRAM errors within real production systems to study the characteristics and uncover memory fault modes [8, 85] . Sridharan et al. [78, 79] find that failures caused by DRAM errors accounted for a significant proportion at high-performance computing sites. Meza et al. [62] show that an increase in chip density is associated with a higher

failure rate. [84--86] study the reliability impact of position, vendors and hardware resilience schemes. [26, 35, 74] revealed differences in errors between various components and between different memory regions.

However, our analysis differs from previous researches in that we not only consider fault modes for different error types but also consider the impact of temperature and power on UERs.

DRAM failure prediction. For identifying the potential failures in advance, existing studies [15, 93] use historical error logs to predict the failures in data centers. Du et al. [25] find that system utilization and intermittent CEs rasie the probability of UEs and utilize these information to predict the UEs. Giurgiu [30] futher detect the predictive power of sensor metrics and adopt sensor metrics and CEs to pretict the UERs. Zhang et al. [95] and Cheng et al. [15] concentrate on the correlation between DRAM and server failuresin Alibaba data centers. Both studies analyze influencing factors (e.g., CE storm [95] and DRAM configurations [15]) on server failures. Baseman et al. [7] and Costa et al. [18] both focus on predicting fault at the page level. The researchers [20, 23, 24] from Intel further perform a failure prediction at the micro-level and propose an ensemble predictor for adapting to diverse environments [22]. Meanwhile, they also introduce a new risky correctable error indicator based on error-bit information for predicting future uncorrectable error occurrences [53]. Moreover, some studies focus on enhancing the effectiveness of prediction by employing a deep learning algorithm [88], leveraging yet-to-be-consumed uncorrectable errors [93], and adopting new metrics [10]. Calchas distinguishes from them by combining spatial, temporal, and sensor features for predicting potential UERs across various device levels of HBM.

9 CONCLUSIONS

This paper presents the first in-depth analysis of HBM errors based on the large-scale dataset, which is collected from nineteen data centers for over two years. We perform the spatial and temporal analyses to understand the properties of HBM errors. We then make two attempts that aim to predict imminent UEs based on the CE rate and the historical CEs, which are proved to be unsuccessful because of the new error causes introduced by the 3D-stacked architecture of HBM. We finally present Calchas, an approach that utilizes spatial locality, temporal correlation, and sensor information to predict upcoming UERs. Calchas achieves accurate and comprehensive predictions at various device levels.

REFERENCES

[1] 2020. NVIDIA A100 Tensor Core GPU. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/a100-80gb-datasheet-update-nvidia-us-1521051-r2-web.pdf

[2] Alaa R Alameldeen and David A Wood. 2004. Adaptive cache compression for high-performance processors. *ACM SIGARCH Computer Architecture News* 32, 2 (2004), 212.

[3] AMD. 2020. AMD RADEON INSTINCT™ MI50. https://www.amd.com/system/files/documents/radeon-instinct-mi50-datasheet.pdf.

[4] Alexandra Angerd, Angelos Arelakis, Vasilis Spiliopoulos, Erik Sintorn, and Per Stenström. 2022. GBDI: Going beyond base-delta-immediate compression with global bases. In *Proceedings of the 2022 International Symposium on High-Performance Computer Architecture (HPCA)*.

[5] Kwanho Bae and Jongsun Park. 2020. Efficient TSV fault detection scheme for high bandwidth memory using pattern analysis. In *Proceeding of the 2020 International SoC Design Conference (ISOCC)*.

[6] Elisabeth Baseman, Nathan DeBardeleben, Kurt Ferreira, Scott Levy, Steven Raasch, Vilas Sridharan, Taniya Siddiqua, and Qiang Guan. 2016. Improving dram fault characterization through machine learning. In *Proceedings of 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*.

[7] Elisabeth Baseman, Nathan Debardeleben, Kurt Ferreira, Vilas Sridharan, Taniya Siddiqua, and Olena Tkachenko. 2017. Automating dram fault mitigation by learning from experience. In *Proceedings of 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*.

[8] Majed Valad Beigi, Yi Cao, Sudhanva Gurumurthi, Charles Recchia, Andrew Walton, and Vilas Sridharan. 2023. A Systematic Study of DDR4 DRAM Faults in the Field. In *Proceedings of 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*.

[9] Rahul Bera, Anant V Nori, Onur Mutlu, and Sreenivas Subramoney. 2019. Dspatch: Dual spatial pattern prefetcher. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*.

[10] Isaac Boixaderas, Darko Zivanovic, Sergi Moré, Javier Bartolome, David Vicente, Marc Casas, Paul M Carpenter, Petar Radojković, and Eduard Ayguadé. [n. d.]. Cost-aware prediction of uncorrected DRAM errors in the field. In *Proceedings of the 2020 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*.

[11] Peter Braun and Heiner Litz. 2019. Understanding memory access patterns for prefetching. In *Proceedings of the 2019 International Workshop on AI-assisted Design for Architecture (AIDArc)*.

[12] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5--32.

[13] Chin-Long Chen. 1986. Error-correcting codes for byte-organized memory systems. *IEEE transactions on information theory* 32, 2 (1986), 181--185.

[14] Xi Chen, Lei Yang, Robert P Dick, Li Shang, and Haris Lekatsas. 2009. C-pack: A high-performance microprocessor cache compression algorithm. *IEEE transactions on very large scale integration systems* 18, 8 (2009), 1196--1208.

[15] Zhinan Cheng, Shujie Han, Patrick PC Lee, Xin Li, Jiongzhou Liu, and Zhan Li. 2022. An in-depth correlative study between DRAM errors and server failures in production data centers. In *Proceedings of the 2022 41st International Symposium on Reliable Distributed Systems (SRDS)*.

[16] Hyungmin Cho, Chen-Yong Cher, Thomas Shepherd, and Subhasish Mitra. 2015. Understanding soft errors in uncore components. In *Proceedings of the 52Nd Annual Design Automation Conference (DAC)*.

[17] Young-kyu Choi, Yuze Chi, Jie Wang, Licheng Guo, and Jason Cong. 2020. When hls meets fpga hbm: Benchmarking and bandwidth optimization. *arXiv preprint arXiv:2010.06075* (2020).

[18] Carlos HA Costa, Yoonho Park, Bryan S Rosenburg, Chen-Yong Cher, and Kyung Dong Ryu. 2014. A system software approach to proactive memory-error avoidance. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*.

[19] T Dell. 1997. A white paper on the benefits of chipkill-correct ECC for PC server main memory. IBM Microelectronics Division. *Technical Report* (1997).

[20] Xiaoming Du and Cong Li. 2018. Memory failure prediction using online learning. In *Proceedings of the 2018 International Symposium on Memory Systems (MEMSYS)*.

[21] Xiaoming Du and Cong Li. 2019. Combining error statistics with failure prediction in memory page offlining. In *Proceedings of the International Symposium on Memory Systems (MEMSYS)*.

[22] Xiaoming Du and Cong Li. 2021. Predicting uncorrectable memory errors from the correctable error history: No free predictors in the field. In *Proceedings of the 2021 International Symposium on Memory Systems (MEMSYS)*.

[23] Xiaoming Du, Cong Li, Shen Zhou, Xian Liu, Xiaohan Xu, Tianjiao Wang, and Shijian Ge. 2021. Fault-aware prediction-guided page offlining for uncorrectable memory error prevention. In *Proceedings of IEEE 39th International Conference on Computer Design (ICCD)*.

[24] Xiaoming Du, Cong Li, Shen Zhou, Mao Ye, and Jing Li. 2020. Predicting uncorrectable memory errors for proactive replacement: An empirical study on large-scale field data. In *Proceedings of 2020 16th European Dependable Computing Conference (EDCC)*.

[25] Yuyang Du, Hongliang Yu, Yunhong Jiang, Yaozu Dong, and Weimin Zheng. 2010. A rising tide lifts all boats: how memory error prediction and prevention can help with virtualized system longevity. In *Proceedings of the Sixth international conference on Hot topics in system dependability (HotDep)*.

[26] Kurt B Ferreira, Scott Levy, Joshua Hemmert, and Kevin Pedretti. 2022. Understanding memory failures on a petascale Arm system. In *Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*. 84--96.

[27] Peter Flach and Meelis Kull. 2015. Precision-recall-gain curves: PR analysis done right. *Advances in neural information processing systems* 28 (2015).

[28] Jessie Frazelle. 2020. Opening up the baseboard management controller. *Commun. ACM* 63, 2 (2020), 38--40.

[29] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189--1232.

[30] Ioana Giurgiu, Jacint Szabo, Dorothea Wiesmann, and John Bird. 2017. Predicting DRAM reliability in the field with machine learning. In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference: Industrial Track (Middleware)*.

[31] Tanmaey Gupta, Sanjeev Krishnan, Rituraj Kumar, Abhishek Vijeev, Bhargav Gulavani, Nipun Kwatra, Ramachandran Ramjee, and Muthian Sivathanu. 2024. Just-In-Time Checkpointing: Low Cost Error Recovery from Deep Learning Training Failures. In *Proceedings of the 19th European Conference on Computer Systems (EuroSys)*.

[32] Ameer Haj-Ali, Nesreen K Ahmed, Ted Willke, Yakun Sophia Shao, Krste Asanovic, and Ion Stoica. 2020. Neurovectorizer: End-to-end vectorization with deep reinforcement learning. In *Proceedings of the 18th ACM/IEEE International*

Symposium on Code Generation and Optimization (CGO).

[33] Zhulin Hao, Jianqiang Du, Bin Nie, Fang Yu, Riyue Yu, and Wangping Xiong. 2016. Random forest regression based on partial least squares connect partial least squares and random forest. In *Proceeding of the 2016 International Conference on Artificial Intelligence: Technologies and Applications (ICAITA)*.

[34] Yuchong Hu, Liangfeng Cheng, Qiaori Yao, Patrick PC Lee, Weichun Wang, and Wei Chen. 2021. Exploiting combined locality for {Wide-Stripe} erasure coding in distributed storage. In *Proceedings of 19th USENIX Conference on File and Storage Technologies (FAST)*.

[35] Andy A Hwang, Ioan A Stefanovici, and Bianca Schroeder. 2012. Cosmic rays don't strike twice: Understanding the nature of DRAM errors and the implications for system design. *ACM SIGPLAN Notices* 47, 4 (2012), 111--122.

[36] Intel. 2023. Intel® Stratix® 10. https://www.intel.com/content/www/us/en/docs/programmable/683189/21-3-19-6-1/hbm2-in-intel-stratix-10-devices.html

[37] Joe Jeddeloh and Brent Keeth. 2012. Hybrid memory cube new DRAM architecture increases density and performance. In *Proceedings of the 2012 symposium on VLSI technology (VLSIT)*.

[38] Hongshin Jun, Jinhee Cho, Kangseol Lee, Ho-Young Son, Kwiwook Kim, Hanho Jin, and Keith Kim. 2017. Hbm (high bandwidth memory) dram technology and architecture. In *Proceedings of the 2017 IEEE International Memory Workshop (IMW)*.

[39] Amit Kumar Kabat, Shubhang Pandey, and Venkatesh Tiruchirai Gopalakrishnan. 2022. Performance evaluation of High Bandwidth Memory for HPC Workloads. In *Proceedings of the 35th International System-on-Chip Conference (SOCC)*.

[40] Vasileios Karakasis, Georgios Goumas, and Nectarios Koziris. 2009. Exploring the effect of block shapes on the performance of sparse kernels. In *Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*.

[41] Jinchun Kim, Seth H Pugsley, Paul V Gratz, AL Narasimha Reddy, Chris Wilkerson, and Zeshan Chishti. 2016. Path confidence based lookahead prefetching. In *Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*.

[42] Jungrae Kim, Michael Sullivan, and Mattan Erez. 2015. Bamboo ECC: Strong, safe, and flexible codes for reliable computer memory. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*.

[43] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. Flipping bits in memory without accessing them: an experimental study of DRAM disturbance errors. In *Proceeding of the 41st annual international symposium on Computer architecuture (SIGARCH)*.

[44] Haklin Kimm, Incheon Paik, and Hanke Kimm. 2021. Performance comparision of TPU, GPU, CPU on Google colaboratory over distributed deep learning. In *Proceedings of 2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*.

[45] Junhyeong Kwon, Shi-Jie Wen, Rita Fung, and Sanghyeon Baeg. 2022. Temperature Estimation of HBM2 Channels with Tail Distribution of Retention Errors in FPGA-HBM2 Platform. *Electronics* 12, 1 (2022), 32.

[46] Seyed Saber Nabavi Larimi, Behzad Salami, Osman S Unsal, Adrián Cristal Kestelman, Hamid Sarbazi-Azad, and Onur Mutlu. 2021. Understanding power consumption and reliability of high-bandwidth memory with voltage underscaling. In *Proceedings of the 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*.

[47] Chang Yeol Lee, Sungchul Kim, Hongshin Jun, Kyung Whan Kim, and Sung Joo Hong. 2014. TSV technology and challenges for 3D stacked DRAM. In *Proceedings of the 2014 Symposium on VLSI Technology: Digest of Technical Papers (VLSIT)*.

[48] Dong Uk Lee, Kyung Whan Kim, Kwan Weon Kim, Hongjung Kim, Ju Young Kim, Young Jun Park, Jae Hwan Kim, Dae Suk Kim, Heat Bit Park, Jin Wook Shin, et al. 2014. 25.2 A 1.2 V 8Gb 8-channel 128GB/s high-bandwidth memory (HBM) stacked DRAM with effective microbump I/O test methods using 29nm process and TSV. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE.

[49] Dong Uk Lee, Kang Seol Lee, Yongwoo Lee, Kyung Whan Kim, Jong Ho Kang, Jaejin Lee, and Jun Hyun Chun. 2015. Design considerations of HBM stacked DRAM and the memory architecture extension. In *Proceedings of the 2015 IEEE Custom Integrated Circuits Conference (CICC)*.

[50] Jong Chern Lee, Jihwan Kim, Kyung Whan Kim, Young Jun Ku, Dae Suk Kim, Chunseok Jeong, Tae Sik Yun, Hongjung Kim, Ho Sung Cho, Sangmuk Oh, et al. 2016. High bandwidth memory (HBM) with TSV technique. In *Proceeding of the 2016 International SoC Design Conference (ISOCC)*.

[51] Scott Levy, Kurt B Ferreira, Nathan DeBardeleben, Taniya Siddiqua, Vilas Sridharan, and Elisabeth Baseman. 2018. Lessons learned from memory errors observed over the lifetime of Cielo. In *Proceedings of the 2018 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*.

[52] Ang Li, Shuaiwen Leon Song, Jieyang Chen, Jiajia Li, Xu Liu, Nathan R Tallent, and Kevin J Barker. 2019. Evaluating modern gpu interconnect: Pcie, nvlink, nv-sli, nvswitch and gpudirect. *IEEE Transactions on Parallel and Distributed Systems* 31, 1 (2019), 94--110.

[53] Cong Li, Yu Zhang, Jialei Wang, Hang Chen, Xian Liu, Tai Huang, Liang Peng, Shen Zhou, Lixin Wang, and Shijian Ge. 2022. From correctable memory errors to uncorrectable memory errors: what error bits tell. In *Proceedings of the 2022 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*.

[54] Shang Li, Dhiraj Reddy, and Bruce Jacob. 2018. A performance & power comparison of modern high-speed dram architectures. In *Proceedings of the International Symposium on Memory Systems (MEMSYS)*.

[55] Xin Li, Michael C Huang, Kai Shen, and Lingkun Chu. 2010. A realistic evaluation of memory hardware errors and software system susceptibility. In *Proceeding of 2010 USENIX Annual Technical Conference (USENIX ATC)*.

[56] Xin Li, Kai Shen, Michael C Huang, and Lingkun Chu. 2007. A Memory Soft Error Measurement on Production Systems.. In *Proceedings of the 2007 USENIX Annual Technical Conference (USENIX ATC)*.

[57] Heng Liao, Jiajin Tu, Jing Xia, and Xiping Zhou. 2019. DaVinci: A Scalable Architecture for Neural Network Computing.. In *Proceedings of 31st Hot Chips Symposium (HCS)*.

[58] Shiyao Lin, Guowen Gong, Zhirong Shen, Patrick PC Lee, and Jiwu Shu. 2021. Boosting {Full-Node} Repair in {Erasure-Coded} Storage. In *Proceedings of 2021 USENIX Annual Technical Conference (USENIX ATC)*.

[59] Wenjie Liu, Shoaib Akram, Jennifer B Sartor, and Lieven Eeckhout. 2021. Reliability-aware garbage collection for hybrid HBM-DRAM memories. *ACM Transactions on Architecture and Code Optimization (TACO)* 18, 1 (2021), 1--25.

[60] Haocong Luo, Ataberk Olgun, Abdullah Giray Yağlıkçı, Yahya Can Tuğrul, Steve Rhyner, Meryem Banu Cavlak, Joël Lindegger, Mohammad Sadrosadati, and Onur Mutlu. 2023. RowPress: Amplifying Read Disturbance in Modern DRAM Chips. In *Proceedings of the 50th Annual International Symposium on Computer Architecture (ISCA)*.

[61] David Meyer, Friedrich Leisch, and Kurt Hornik. 2003. The support vector machine under test. *Neurocomputing* 55, 1-2 (2003), 169--186.

[62] Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu. 2015. Revisiting memory errors in large-scale production data centers: Analysis and modeling of new trends from the field. In *Proceeding of 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.

[63] Hongyu Miao, Myeongjae Jeon, Gennady Pekhimenko, Kathryn S McKinley, and Felix Xiaozhu Lin. 2019. Streambox-hbm: Stream analytics on high bandwidth hybrid memory. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[64] Pierre Michaud. 2016. Best-offset hardware prefetching. In *Proceedings of the 2016 22nd International Symposium on High Performance Computer Architecture (HPCA)*.

[65] Inc Micron Technology. 2023. HBM3 Gen2. https://investors.micron.com/node/45591/pdf.

[66] Prashant J Nair, David A Roberts, and Moinuddin K Qureshi. 2016. Citadel: Efficiently protecting stacked memory from tsv and large granularity failures. *ACM Transactions on Architecture and Code Optimization* 12, 4 (2016), 1--24.

[67] NVIDIA. 2016. NVIDIA Tesla P100. https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf

[68] NVIDIA. 2017. NVIDIA Tesla V100 GPU Architecture. https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

[69] NVIDIA. 2024. NVIDIA GPU Memory Error Management. https://docs.nvidia.com/deploy/a100-gpu-mem-error-mgmt/index.html

[70] NVIDIA. 2024. NVIDIA H100 Tensor Core GPU. https://www.nvidia.com/en-us/data-center/h100/

[71] Ataberk Olgun, Majd Osseiran, A Giray Yağlıkçı, Yahya Can Tuğrul, Haocong Luo, Steve Rhyner, Behzad Salami, Juan Gomez Luna, and Onur Mutlu. 2023. An experimental analysis of rowhammer in hbm2 dram chips. In *Proceedings of the 2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*.

[72] Ataberk Olgun, Majd Osseiran, Abdullah Giray Yaglikci, Yahya Can Tugrul, Haocong Luo, Steve Rhyner, Behzad Salami, Juan Gomez Luna, and Onur Mutlu. 2023. Understanding Read Disturbance in High Bandwidth Memory: An Experimental Analysis of Real HBM2 DRAM Chips. *arXiv preprint arXiv:2310.14665* (2023).

[73] Sungbo Park, Ingab Kang, Yaebin Moon, Jung Ho Ahn, and G Edward Suh. 2021. Bcd deduplication: Effective memory compression using partial cache-line deduplication. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*.

[74] Ayush Patwari, Ignacio Laguna, Martin Schulz, and Saurabh Bagchi. 2017. Understanding the spatial characteristics of DRAM errors in HPC clusters. In *Proceedings of the 2017 Workshop on Fault-Tolerance for HPC at Extreme Scale (FTXS)*.

[75] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine Learning research* 12 (2011), 2825--2830.

[76] Seth H Pugsley, Zeshan Chishti, Chris Wilkerson, Peng-fei Chuang, Robert L Scott, Aamer Jaleel, Shih-Lien Lu, Kingsum Chow, and Rajeev Balasubramonian. 2014. Sandbox prefetching: Safe run-time evaluation of aggressive prefetchers. In *Proceedings of the 20th International Symposium on High Performance Computer Architecture (HPCA)*.

[77] Suresh Ramalingam. 2016. HBM package integration: Technology trends, challenges and applications. In *2016 IEEE Hot Chips 28 Symposium (HCS)*.
[78] Bianca Schroeder and Garth A. Gibson. [n. d.]. A Large-Scale Study of Failures in High-Performance Computing Systems. *IEEE Transactions on Dependable and Secure Computing* 7, 4 ([n. d.]), 337--350.
[79] Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. 2009. DRAM errors in the wild: a large-scale field study. *ACM SIGMETRICS Performance Evaluation Review* 37, 1 (2009), 193--204.
[80] scikit-learn. 2023. Python One-hot Encoder. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
[81] Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A Kozuch, Onur Mutlu, Phillip B Gibbons, and Todd C Mowry. 2017. Ambit: In-memory accelerator for bulk bitwise operations using commodity DRAM technology. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*.
[82] Seyed Mohammad Seyedzadeh, Alex K Jones, and Rami Melhem. 2018. Mitigating wordline crosstalk using adaptive trees of counters. In *Proceedings of 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE.
[83] Keeyoung Son, Subin Kim, Hyunwook Park, Seongguk Kim, Keunwoo Kim, Shinyoung Park, Boogyo Sim, Seungtaek Jeong, Gapyeol Park, and Joungho Kim. 2020. A novel through mold plate (TMP) for signal and thermal integrity improvement of high bandwidth memory (HBM). In *Proceeding of the 2020 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO)*.
[84] Vilas Sridharan, Nathan DeBardeleben, Sean Blanchard, Kurt B Ferreira, Jon Stearley, John Shalf, and Sudhanva Gurumurthi. 2015. Memory errors in modern systems: The good, the bad, and the ugly. *ACM SIGARCH Computer Architecture News* 43, 1 (2015), 297--310.
[85] Vilas Sridharan and Dean Liberty. 2012. A study of DRAM failures in the field. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*.
[86] Vilas Sridharan, Jon Stearley, Nathan DeBardeleben, Sean Blanchard, and Sudhanva Gurumurthi. 2013. Feng shui of supercomputer memory: Positional effects in DRAM and SRAM faults. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*.
[87] JEDEC Standard. 2013. High bandwidth memory (hbm) dram. *Jesd235* 16 (2013).
[88] Xiaoyi Sun, Krishnendu Chakrabarty, Ruirui Huang, Yiquan Chen, Bing Zhao, Hai Cao, Yinhe Han, Xiaoyao Liang, and Li Jiang. 2019. System-level hardware failure prediction using deep learning. In *Proceedings of the 56th Annual Design Automation Conference (DAC)*.
[89] Ronglong Wu, Shuyue Zhou, Jiahao Lu, Zhirong Shen, Zikang Xu, Jiwu Shu, Kunlin Yang, Feilong Lin, and Yiming Zhang. 2024. 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)*. USENIX Association, Santa Clara, CA, 851--867.
[90] XFUSION. 2023. https://www.xfusion.com/cn/service/fusioncare-basic-service
[91] Shiyang Xuan, Guanjun Liu, Zhenchuan Li, Lutao Zheng, Shuo Wang, and Changjun Jiang. 2018. Random forest for credit card fraud detection. In *Proceeding of the 2018 IEEE 15th international conference on networking, sensing and control (ICNSC)*.
[92] Li Yu, Ziming Zheng, Zhiling Lan, and Susan Coghlan. 2011. Practical online failure prediction for blue gene/p: Period-based vs event-driven. In *Proceedings of 2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*.
[93] Qiao Yu, Wengui Zhang, Paolo Notaro, Soroush Haeri, Jorge Cardoso, and Odej Kao. 2023. HiMFP: Hierarchical Intelligent Memory Failure Prediction for Cloud Service Reliability. In *Proceedings of the 2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.
[94] Kaiqiang Zhang, Dongyang Ou, Congfeng Jiang, Yeliang Qiu, and Longchuan Yan. 2021. Power and performance evaluation of memory-intensive applications. *Energies* 14, 14 (2021), 4089.
[95] Pengcheng Zhang, Yunong Wang, Xuhua Ma, Yaoheng Xu, Bin Yao, Xudong Zheng, and Linquan Jiang. 2022. Predicting DRAM-caused node unavailability in hyper-scale clouds. In *Proceedings of the 2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.

Received XXX; revised XXX; accepted XXX