# Characterizing Modern GPU Resilience and Impact in HPC Systems: A Case Study of A100 GPUs

Shengkun Cui†*, Archit Patke†*, Ziheng Chen†*, Aditya Ranjan†*, Hung Nguyen†, Phuong Cao†,
Brett Bode†, Gregory Bauer†, Saurabh Jha¶, Chandra Narayanaswami¶, Daby Sow¶,
Catello Di Martino§, Zbigniew T. Kalbarczyk†, and Ravishankar K. Iyer†
†University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
§Nokia Bell Labs, Sao Paulo, Brazil
¶IBM Research, Yorktown Heights, NY 10598, USA

***Abstract*—In this study, we characterize GPU failures in *Delta*[1], the current large-scale AI system with over 1300 petaflops of peak compute throughput. Using three years of GPU error-recovery data (12.5M GPU hours), this study evaluates the resilience of NVIDIA A100's GPU hardware components to determine the vulnerability of different GPU components to failure and their impact on the GPU-node availability and on user jobs. Our key findings include: (i) A100 GPU node resilience in MTBE reduced slightly (23%) in the operational period compared to the pre-operational period due to more frequent GPU hardware errors primarily caused by higher GPU utilization. (ii) Contrary to common beliefs, GPU memory is 160× more reliable than GPU hardware in terms of MTBE (mean time between errors). (iii) The newly introduced GSP (GPU System Processor) is the most vulnerable GPU hardware component. (iv) NVLink errors did not always lead to user job failure, and we attribute it partly to the underlying error detection and retry mechanisms employed. (v) Hardware errors and insufficient recovery mechanisms cause frequent job failures and limit overall GPU node availability to 99.5% (7 minutes downtime per day), indicating the infrastructure is not yet ready for system-scale, long-running user jobs.**

***Keywords*—Large-scale AI/HPC System; Reliability Evaluation and Analysis; GPU Resilience; Application Impacts.**

## I. Introduction

This paper presents the results of a resilience study of *Delta* HPC, a large-scale GPU system consisting of 1,456 total modern NVIDIA GPUs. Specifically, we focus on characterizing the resilience of *Delta*'s A100 GPU nodes via three years of critical GPU error data, representing a total of 12.5 million GPU hours. In particular, we assess (a) the resilience of GPU hardware and memory components in the pre-operational and operational periods to identify the vulnerability of various GPU components to errors and their impact on GPU node availability and (b) the impact of GPU errors on user jobs. **Our findings include:**

(i) The per node MTBE (mean time between errors) of *Delta*'s A100 nodes saw a 23% degradation in the operational period compared to the pre-operational period, from 199 hours to 154 hours. We attributed this to increased GPU utilization in the operational period, which resulted in more frequent GPU hardware-related errors (e.g., GSP and PMU errors).

(ii) A100's GPU memory is 160× more resilient than GPU hardware, measured by per node MTBE. GSP (GPU System Processor), introduced to enhance performance by offloading CPU side driver workloads to Ampere GPUs, is the most vulnerable GPU hardware component due to its limited detection and recovery capabilities. GSP errors in A100s were highly impactful, with 100% of GSP errors leading to user job failures, A GSP error requires a node reboot. Further analysis shows that GSP error MTBE is 5.6× worst in the operational period than in the pre-operational period likely due to higher GPU utilization.

(iii) Communication errors with PMU (Power Management Unit), although not explicitly described in NVIDIA developer manual [1], can lead to issues such as the inability to change the GPU core clock frequency and memory clock frequency. Propagation of such errors can cause MMU (Memory Management Unit) errors, which then lead to user job failures 96% of the time.

(iv) NVLink (GPU-to-GPU communication fabric within a multi-GPU node) errors occur with a system-wide MTBE of 11 hours. An NVLink error results in job failure approximately 54% of the time. Conversely, about 46% of jobs experiencing NVLink errors are able to complete successfully. We attribute this to error detection mechanisms such as CRC (cyclic redundancy check) that detect errors and trigger message retransmission.

(v) In A100 GPUs, memory error-recovery mechanisms like memory row remapping and error containment reduce the impact of uncorrectable memory errors on user jobs while maintaining node availability. Our analysis shows that these mechanisms mitigate the impact of an uncorrectable memory error such as a DBE 100% of time in the operational period, and no row remapping failure (RRF) was observed. However, failure in error containment can result in bursty and persistent memory errors (e.g., we encountered a case of an uncontained memory error that persisted for 17 days during the pre-operational period from one faulty A100 GPU, presenting a resilience challenge.

(vi) Hardware errors and lack of robust recovery in hardware lead to job failures. Except for MMU and NVLink errors, none of the other hardware errors can be handled by application-level mechanisms. Hence, the reliability of the

*These authors contributed equally.

[1]*Delta* is an NCSA-operated HPC system at the University of Illinois Urbana-Champaign.

underlying GPU hardware needs to be improved, as relying on application-based recovery strategies is not feasible.

(vii) The overall availability per GPU node is 99.5%, which corresponds to a downtime of 7 minutes per day. Such a large downtime indicates that the infrastructure is not ready yet to meet the demands of critical user jobs that need to provide uninterrupted service.

## II. Background

This section provides information on (i) *Delta* specification, (ii) critical GPU error categories and error recovery mechanisms used in this study.

### A. Delta Specifications

*Delta* is an AI system equipped with 132 CPU-only nodes and 358 GPU-accelerated nodes, designed to run diverse scientific research, HPC, and machine learning (ML) workloads.

*Delta*'s *CPU-only* nodes are equipped with two 64-core AMD EPYC Milan CPUs. Out of the 358 *GPU-accelerated* nodes, 106 of them are A100 GPU nodes which are the subject of this study. The A100 GPU nodes are equipped with one 64-core AMD EPYC Milan CPU and 4-way (100 nodes) or 8-way (6 nodes) A100 GPUs; each GPU has 40 GB of HBM2e memory. Moreover, *Delta* is equipped with HPE Cray Slingshot 11 network that offers 400Gbps+ connectivity bandwidth between nodes, and Lustre file system for storage.

### B. NVIDIA GPU Error Categories & Recovery Mechanisms

NVIDIA GPU errors are logged as XID errors. We focus on those XID errors that are identified as high-impact based on descriptions from NVIDIA's Developer Manuals [1], [2], discussions in NVIDIA Developer Forums and Blogs, and input from *Delta* site reliability engineers (SREs), and collect their corresponding recovery events. Section §IV, Table I presents detail list of selected XID errors/events and their corresponding recovery action specified by [1]. Table I categorizes the selected GPU errors into (i) GPU hardware, (ii) NVLink interconnect, and (iii) GPU memory. It is important to note that we exclude GPU Software Errors (XID 13) and Reset Channel Verification Errors (XID 43) from our analysis despite significant numbers because they are typically triggered by user jobs and are not indicators of degraded GPU health [1].

**GPU Hardware Errors and Recovery.** The critical GPU hardware errors we studied include MMU[2] errors, GPU Fallen Off the Bus errors, GSP[3] RPC timeout errors, and PMU[4] communication errors. For most of these GPU hardware errors, manual GPU reset is required [1]. As such, *Delta* SREs set up node health checks to proactively monitor these errors and alert the SREs upon error discovery to minimize potential node downtime. Information on failure-recovery mechanisms of non-memory GPU hardware such as GSP, MMU, and PMU is limited and not publicly available.

**NVLink Interconnect Errors.** Errors in NVLink inter-GPU communication disrupt inter-GPU data transfer, reducing computational performance, instability, and user job failures. These errors are the result of defective NVLink hardware, connectors, or improper physical bridge installation. Resolving NVLink errors requires inspection of NVLink-related software and hardware, and GPU reset to clear [1]. To detect data corruption during transmission, NVLink employs Cyclic Redundancy Checks (CRCs) to monitor control and data packets. When a CRC checksum fails, the NVLink driver automatically retransmits packets starting from the last-know-good transmission.

**GPU Memory Errors.** GPU memory errors we study are uncorrectable double-bit errors (DBEs) that exceed the SECDED ECC correction capability. The exact number of SBEs are unknown as SBEs are automatically corrected by ECC mechanism and hence are not logged. A DBE triggers downstream error recovery mechanisms such as row-remapping and error containment, which are described below.

NVIDIA A100's memory is SECDED ECC protected. Since DBEs are uncorrectable, A100 GPUs employ memory *row-remapping* as the primary mechanism to mitigate DBEs. During row-remapping, the NVIDIA driver uses a spare row to replace the faulty memory row by altering the memory mapping table. Row-remapping event (RRE) is logged if a row is successfully marked for replacement. Conversely, a row remapping failure (RRF) is logged if all spare rows are exhausted [1], [2]. *Delta* site reliability engineers (SREs) actively track row-remapping failures and replace GPUs that repeatedly log RRFs.

In addition, A100 GPUs employ additional resilience mechanisms: (i) dynamic page offlining and (ii) error containment to maintain GPU node availability [1], [2]. Dynamic page offlining isolates the faulty memory page by marking it as unallocable at runtime, helping to preserve GPU node availability without requiring a reset [2]. Error containment mechanisms prevent the spread of uncorrectable memory errors by terminating affected user processes that access corrupted memory regions [2]. When containment succeeds, the event is recorded as a Contained Memory Error; if containment fails, it is logged as an Uncontained Memory Error. Failures in either row remapping or error containment process can lead to GPU failures that require a GPU reset or a full node reboot to recover.

## III. Methodology

### A. Data Source

We conducted our data analysis on the error-recovery data collected from *Delta*'s A100 GPU nodes over an 1170-day period from January 2022 to March 2025. *Delta*'s SRE further divide this measurement period into pre-operational (bring-up and testing) period, from January 2022 to September 2022, and operational (production) period from October 2022 to March 2025. Below, we describe the data used in *Stage I* of Fig. 1.

[2]Memory management unit provide memory I/O functionalities [1].

[3]GPU system processor (GSP) is a co-processor on-board that offloads driver tasks from CPU [1].

[4]PMU regulates the frequency, voltage, and power of the GPU based on various factors such as temperature and power cap, also referred to as the "Performance Monitoring Unit" by NVIDIA [1].

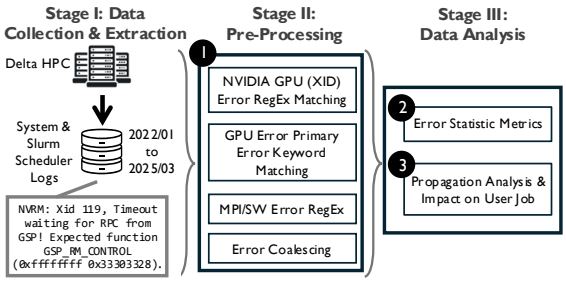


Figure 1: Data collection, processing, and analysis pipeline.

**System logs and Slurm database.** *Delta* system logs are collected from all compute nodes and record information, warning, and error events from all system components. The system logs are collected and consolidated per day for all nodes in the system. The pipeline additionally employs regular expression (RegEX) pattern-matching for filtering system logs to extract the selected XID error-recovery events, as shown in Fig. 1-(1). The set of extracted GPU error logs is the major data source of subsequent data analysis.

In addition, *Delta* HPC uses the Slurm Scheduler [3] for scheduling user jobs. The Slurm scheduler database consists of per-job submission records that were used to study the impact of GPU errors on job failures. The information includes per job submission time, start time and end time, resource requested, and scheduled node. In addition, the Slurm database also records job exit status and the run command line, which provides valuable information for more fine-grained analysis.

### B. Data Processing Pipeline

This subsection focuses on *Stage II and III* of the pipeline in Fig. 1, which processes the raw logs, computes error counts and mean time between errors, and characterizes error impact on user jobs.

**Error Coalescing.** The error coalescing step in Fig. 1-(1), stage ii, reduces duplicated errors. Error coalescing is an essential step for accurate error analysis because the same error can result in multiple, repeated log lines in close succession. Counting each log line as a separate error leads to over-counting, which significantly underestimates the GPU resilience. The error coalescing step mitigates this issue by combining identical error log lines from the same GPU in a short time window $\Delta t$ into a single error, i.e., only counting the first occurrence in $\Delta t$, as the same error likely causes subsequent identical log lines [4]–[7].

**Error Statistic Metrics.** The data analysis stage of the pipeline (in Fig. 1-(2)) analyzes the coalesced error logs and outputs standard reliability error statistics such as error counts and mean time between errors (MTBE), as in [6], [7]. The per-node MTBE, computed by multiplying the system-wide MTBE with the number of nodes, indicates the duration of a single *Delta* GPU node can function before encountering a GPU error. These statistics provide resilience assessments for GPU errors singly and collectively, e.g., a short MTBE highlights the weak links in system resilience to be improved.

**User Job Impact Analysis.** The user job impact analysis step (Fig. 1-(3)) characterizes the impact of GPU errors on user jobs by correlating GPU errors with user job failures. §V provides the detail of this analysis. Note that we only analyze impacts on user jobs in the operational period (from October 2022 onward) after the system has started to accept production user workloads.

## IV. Characterizing GPU Resilience

This section characterizes the resilience of *Delta*'s NVIDIA A100 GPU errors in three categories: (a) GPU hardware, (b) NVLink, and (c) GPU memory, as described in §II-B and Table I using error statistic metrics including counts and mean time between error (MTBE). The selected errors in Table I are critical because they propagate to the user job, as we show in §V. Below, we highlight key findings from our GPU error statistics analysis and compare error statistics between the pre-operational and operational periods to understand the benefits of pre-operational testing and improvements brought by refining operational procedures. Note that we do not compare *Delta* directly with Blue Waters [7] (NVIDIA Kepler), Titan [8] (NVIDIA Kepler), or Summit [6] (NVIDIA Volta) as those GPUs are one or several generation(s) behind NVIDIA A100 GPUs.

**The statistical highlights are as follows:**

(i) As shown in Table I, in the pre-operational period, uncontained memory errors were the predominant errors that accounted for 92% (38,900) of the total error (42,405). These uncontained memory errors originated from one faulty GPU that was replaced upon discovery, which substantially eliminated memory errors during the remaining pre-operational period. During the operational period, MMU errors, NVLink errors, and GSP RPC timeouts were the predominant errors, accounting for over 98% (14,642 out of 14,821) of the total GPU errors in that period. These errors could lead to a GPU error state, causing interruptions to user jobs.

(ii) *Delta*'s per-node MTBE slightly degraded from 199 hours[5] in the pre-operational period to 154 hours in the operational period, a 23% reduction. This was primarily due to the reduction in per node MTBE of GSP and PMU errors, likely due to the increased GPU utilization, as discussed below. That said, we observed large improvements in the MTBE of RRE, RRF, NVLink, and contained/uncontained memory errors, potentially due to the early replacement of defective GPUs and automatic node health checks that actively monitor GPU-related errors, prompting timely error-recovery process.

(iii) Contrary to common beliefs that memory is more prone to error, A100's GPU memory is 160× (per node MTBE of 155 hours vs 24,749 hours) more reliable regarding MTBE than GPU hardware during the operational period. Among the GPU hardware components, the newly introduced GSP is prone to error due to limited error detection and recovery mechanisms. *Delta* SREs indicated that GSP errors frequently

[5]We excluded those 38,900 uncontained memory errors when calculating the per-node MTBE for the pre-operational period, as those errors were generated by one faulty GPU and were considered an outlier by SREs.

Table I: *Delta* NVIDIA Ampere A100 GPU resilience statistics.

| Event Code | Abbr. | Category | Description | Recovery Action | Count | | Pre-op MTBE (hrs) | | Op MTBE (hrs) | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Pre-op | Op | System-wide | Per Node | System-wide | Per Node |
| XID 31 | MMU Error | Hardware | GPU memory management unit (MMU) error. | MMU error due to invalid memory access or driver/hardware bugs. | 1,078 | 8,863 | 6.1 | 649 | 2.4 | 257 |
| XID 48 | DBE | Memory | Double bit ECC memory error (DBE). | Triggers RRE; GPU reset or node reboot is needed to clear this error if RRE failed. | 0 | 1 | – | – | – | – |
| – | Uncorrectable ECC memory Errors | Memory | Multiple SBEs or a DBE at a memory location. | Triggers RRE; GPU reset or node reboot is needed to clear this error. | 46 | 34 | 143 | 15,208 | 632 | 66,967 |
| XID 63 | RRE | Memory | Row remapping event, triggered by 1 DBE or 2 SBE at the same memory address. | GPU reset needed for row remapping. | 31 | 34 | 213 | 22,568 | 632 | 66,967 |
| XID 64 | RRF | Memory | Row remapping failure of a row remapping event. | A GPU reset is needed to clear this error. | 15 | 0 | 440 | 46,640 | – | – |
| XID 74 | NVLink Error | Inter-connect | NVLink error, indicating connection issues between GPUs via NVLink interconnection. | GPU reset or SRE intervention required. | 2,092 | 1922 | 3 | 334 | 11 | 1185 |
| XID 79 | GPU Fallen Off the Bus Error | Hardware | GPU has fallen off the system bus and is not reachable, which is typically caused by driver or hardware errors. | GPU reset or SRE intervention required. | 4 | 10 | 1,650 | 174,900 | 2,184 | 227,688 |
| XID 94 | Contained Memory Error | Memory | Uncorrectable contained ECC error, indicating a successful uncorrectable error containment that prevents error propagation by terminating the affected processes. | Not specified. | 22 | 13 | 300 | 31,800 | 1,652 | 175,145 |
| XID 95 | Uncontained Memory Error | Memory | Uncontained memory error, indicating an unsuccessful uncorrectable error containment. | GPU reset or SRE intervention required. | 38,900 | 11 | 0.17 | 18 | 1,953 | 206,989 |
| XID 119/120 | GSP Error | Hardware | NVIDIA GPU Systems Processor (GSP) error. GSP is a coprocessor that manages GPU initialization and other tasks. | GPU reset or SRE intervention required. | 209 | 3857 | 32 | 3,347 | 5.6 | 590 |
| XID 122/123 | PMU SPI Error | Hardware | PMU SPI RPC read failure, indicating a failed communication with the PMU. | Not specified. | 8 | 77 | 825 | 87,450 | 279 | 29,569 |

*Error description and recovery action are derived from [1], [2].
*An NVIDIA Ampere A100 GPU supports page retirement and up to 512-row remapping; the previous generations support 64-page retirements and no row-remapping [2].
*Row remapping, Contained memory Error, and Uncontained memory Error are mechanisms introduced in the NVIDIA Ampere architecture for uncorrectable memory error management.
*Per node MTBE in hours is derived by multiplying system MTBE with the number of A100 GPU nodes, 106 nodes in total.
*Pre-op: pre-operational period; Op: Operational period; hrs: hours.

led to user job interruption and node outage, requiring manual node draining and reboot to recover, introducing significant overheads. The user job impact analysis (see §V) shows that 100% of the GSP errors lead to user job failure. Notably, the per node MTBE of GSP errors is $5.6\times$ shorter in operational period (590 hours) compared to pre-operational period (3,347 hours), indicating a reduction in GSP resilience potentially due to higher GPU utilization after entering production.

(iv) PMU SPI communication errors are frequent and exhibited high correlations with MMU errors, which then resulted in a job failure over 96% of the time, as shown in §V. Although PMU SPI communication error is a high-impact error from a user job perspective, it is not highlighted in NVIDIA's Developer's Manual [1]. We observed errors of this type in both pre-operational and operational period in our data, with $3\times$ worsen MTBE during the operational period, potentially due to higher GPU utilization.

(v) NVLink errors were more frequent than we anticipated, with 1,922 total NVLink errors in the operational period, which translates to a per-node of 1185 hours and system-wide MTBE of 11 hours, despite the usage of CRC error detection and other recovery mechanisms. Of those, 42% propagates two or more GPUs in the operational period, and overall, 54% NVLink errors led to a job failure. Apparently, GPU is more resilient to NVLink error than GSP error as the remaining 46% of the jobs ran to finish, meaning that the GPUs were still operational despite the NVLink errors. We conjuncture that this was because the NVLink was not in use when those errors happened, so the completion of the job was unaffected.

(vi) A100 GPUs memory were resilient to uncorrectable memory errors such as DBEs as we observed only one DBE during the 895 days operational period. All uncorrectable memory errors were mitigated by the row remapping process (RRE) and no RRF was observed in the operational period. Although not encountered during the operational period, failure in error containment manifests as uncontained memory errors that can be bursty and persisting in nature, presenting a resilience challenge. In one case, uncontained memory error persisted for 17 days (May $5^{th}$ to May $21^{st}$ 2022) in the pre-operational period without recovery, generating over a million duplicated log entries that were potentially disruptive to the GPU node's normal operation. *Delta* SREs use automatic health checks to actively monitor uncorrectable and uncontained memory errors for timely node recovery and GPU replacement to reduce node downtime.

## V. Propagation of GPU Errors to User Jobs

This section examines job-level fault tolerance and the impact of GPU-related failures on system availability. When a GPU encounters an error, two major consequences may arise:
1) *Job Termination:* If the error is unhandled or propagates beyond containment, the running job can crash.
2) *GPU Unavailability:* Hardware faults may necessitate resetting or physically replacing a GPU, resulting in a temporary period where the GPU cannot host any workload.

### A. Job Statistics

Over the characterization period, users submitted 1,445,119 jobs to GPU nodes, achieving a success rate of 74.68%, and 1,686,696 jobs to CPU nodes, with a comparable success rate

| XID | GPU Error | # GPU-failed jobs encountering given XID | #Jobs encountering given XID | Job Failure Probability given an XID Error (%) |
|---|---|---|---|---|
| 31 | MMU Err. | 3206 | 3543 | 90.48 |
| 122 | SPI PMU RPC failure | 40 | 41 | 97.56 |
| 119 | GSP RPC Timeout | 31 | 31 | 100.00 |
| 74 | NVL Err | 43 | 80 | 53.75 |
| 94 | Contained ECC | 5 | 5 | 100.00 |

Table II: Distribution of GPU-failed jobs across the different GPU error types. The failure probability is calculated as (# GPU-failed jobs encountering that GPU error) / (# Jobs encountering that GPU error). The total number of GPU-failed jobs is 3,285 during the 2.5-year operational period.

of 74.90%. The majority of GPU jobs (69.86%) utilized a single GPU, while 27.31% leveraged 2–4 GPUs, and only 2.83% required 4 or more GPUs.

Since explicit labels indicating whether a job was machine learning (ML) related were unavailable, we approximated the fraction of ML jobs by analyzing job names and loaded modules or libraries. [6] For example, job names including keywords like model or train were considered indicative of ML workloads. Comprehensive statistics on GPU node usage and job durations for these categories are summarized in Table III.

### B. Job Failure Analysis

To investigate patterns of job failures, we categorize jobs into two groups: Completed" and GPU-Failed," based on their final status. Using this classification, we examine: (i) the types of GPU errors most commonly associated with job failures, and (ii) potential mitigation techniques such as checkpointing and exception handling.

*Job Classification Methodology:* Jobs are labeled according to their recorded exit status and the temporal correlation between GPU errors and job termination events. Exit statuses are retrieved from Slurm scheduler logs, as outlined in §III-A. A job is designated as "GPU-failed" if a GPU error is detected within a twenty-second window preceding the job's failure.

*Correlating GPU Errors and Job Failures:* We analyze GPU-failed jobs by categorizing them according to the GPU errors most likely responsible for the failure. Table II summarizes the probability of job failures associated with each type of GPU error. Since multiple errors can occur close to a job's termination, we attribute any GPU error detected within the twenty-second window before failure as a potential contributor.

In general, most GPU errors—such as ECC faults, GSP RPC timeouts, and PMU failures—tend to propagate and ultimately cause job termination. However, NVLink and MMU-related errors do not always result in job failure due to the following reasons:

1) *NVLink errors* may occur even when the affected link or GPU is idle or unused by active jobs, and
2) *MMU errors* may be masked at the application or library level. Besides true hardware faults, MMU errors can also arise from software bugs where illegal memory accesses are attempted but not mapped in the virtual address space. Such errors can often be mitigated using application-level exception handling. Modern machine learning libraries and frameworks [9], [10] provide support for handling these cases by skipping faulty training iterations, though at the potential cost of degraded model quality.

[6]Due to privacy policies, the exact submission scripts were not examined for classification.

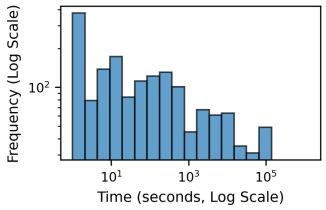


Figure 2: Unavailability Time Distribution.

| GPU Count | Count (%) | Elapsed Time (minutes) | | | GPU Hours (k) | |
|---|---|---|---|---|---|---|
| | | Mean | P50 | P99 | ML | Non-ML |
| 1 | 1,013,170 (69.86) | 175.62 | 10.15 | 2483.12 | 241.6 | 2724.0 |
| 2-4 | 396,133 (27.31) | 145.04 | 4.75 | 2880.03 | 344.6 | 3108.7 |
| 4-8 | 22,474 (1.55) | 133.89 | 2.70 | 2880.20 | 57.9 | 338.6 |
| 8-32 | 15,440 (1.07) | 270.40 | 73.73 | 2880.17 | 107.1 | 1332.7 |
| 32-64 | 2,054 (0.14) | 204.52 | 10.25 | 2817.08 | 161.9 | 226.4 |
| 64-128 | 913 (0.063) | 226.28 | 0.32 | 2211.94 | 25.1 | 322.3 |
| 128-256 | 82 (0.006) | 226.53 | 9.19 | 2785.29 | 0.0 | 52.4 |
| 256+ | 25 (0.002) | 32.12 | 20.40 | 120.14 | 0.0 | 4.5 |

Table III: Job distribution, elapsed time statistics (Mean, P50, P99), and GPU hours divided into ML and Non-ML categories for various GPU configurations.

### C. Impact of GPU Downtime on Jobs

Beyond compute time lost to failed jobs, additional node hours are consumed during the recovery process for affected GPU nodes, either through a node reset or hardware replacement. Resetting a node typically involves draining it, allowing any active job to be completed, followed by a reboot. If the node passes post-reboot health checks, it returns to service and resumes scheduling new jobs. Otherwise, if the reset fails, the node remains marked as failed until the GPU hardware is physically swapped.

To estimate average system downtime, we measure the period during which the GPU remains unavailable, primarily encompassing drain and reboot times. Fig. 2 presents the distribution of unavailable durations observed during the characterization period. On average, servicing a failed node takes approximately 0.88 hours, resulting in a cumulative loss of 5,700 node hours due to GPU downtime. Using observed downtime and failure rates, we estimate GPU node availability as $\frac{MTTF}{MTTF+MTTR} = 99.5\%$, where the Mean Time To Failure (MTTF) is 162 hours[7] and the Mean Time To Repair (MTTR) is 0.88 hours.

## VI. Related Work

Existing work has analyzed GPU resilience at the microarchitecture- and system level. This work extends previous work by comprehensive analyses of GPU error characteristics during the pre-operational and operational periods and their impact on HPC/ML workloads.

[7]The MTTF estimate is derived from the GPU's Mean Time Between Errors (MTBE), under the conservative assumption that all GPU errors cause node interruption.

Prior work [11]–[14] has primarily focused on the resilience of individual GPUs at the microarchitecture level, e.g., for older generations such as NVIDIA G80/GT200/Fermi. None of those works evaluate modern GPUs' resilience in HPC settings. Various works have analyzed the resilience of GPUs in HPC systems [15]–[19] e.g, the NVIDIA Tesla K20X GPUs in various supercomputers [5], [7], [8], [20]–[24]. In particular, the studies on Blue Waters, Titan, and Summit supercomputers [6]–[8], [23] have examined node and GPU failures. Complementary those works, we are the first to study the resiliency of modern NVIDIA A100 GPUs in an HPC system.

## VII. Conclusion

This paper described the results of a resilience study of *Delta*'s 448 A100 GPUs. The study uses three years of data (12.5 million GPU hours) on critical GPU errors collected across those GPUs. We assesses the resilience of GPU components and their impact on node availability and user jobs, and compared statistics in pre-operational (testing) period with operational (production) period. Future work extends this analysis to the NVIDIA Grace Hopper systems that are equipped with H100 GPUs.

## References


[1] NVIDIA. (2024) GPU Deployment and Management. [Online]. Available: https://docs.nvidia.com/deploy/pdf/XID_Errors.pdf

[2] NVIDIA Corporation, *NVIDIA GPU Memory Error Management*, May 2024, release r555. [Online]. Available: https://docs.nvidia.com/deploy/a100-gpu-mem-error-mgmt/index.html

[3] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60.

[4] D. Tiwari, S. Gupta, J. Rogers, D. Maxwell, P. Rech, S. Vazhkudai, D. Oliveira, D. Londo, N. DeBardeleben, P. Navaux *et al.*, "Understanding gpu errors on large-scale hpc systems and the implications for system design and operation," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2015, pp. 331–342.

[5] ——, "Understanding GPU errors on large-scale HPC systems and the implications for system design and operation," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2015, pp. 331–342.

[6] V. Oles, A. Schmedding, G. Ostrouchov, W. Shin, E. Smirni, and C. Engelmann, "Understanding gpu memory corruption at extreme scale: The summit case study," in *Proceedings of the 38th ACM International Conference on Supercomputing*, 2024, pp. 188–200.

[7] C. Di Martino, Z. Kalbarczyk, R. K. Iyer, F. Baccanico, J. Fullop, and W. Kramer, "Lessons learned from the analysis of system failures at petascale: The case of Blue Waters," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2014, pp. 610–621.

[8] B. Nie, D. Tiwari, S. Gupta, E. Smirni, and J. H. Rogers, "A large-scale study of soft-errors on gpus in the field," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2016, pp. 519–530.

[9] PyTorch Contributors. (2018) Discussion: How to Handle Exception in DistributedDataParallel. Accessed: 2024-11-26. [Online]. Available: https://discuss.pytorch.org/t/how-to-handle-exception-in-distributeddataparallel/42026

[10] PyTorch Lightning Contributors. (2021) PyTorch Lightning Discussion: Handling Exceptions in Forward Pass. Accessed: 2024-11-26. [Online]. Available: https://github.com/Lightning-AI/pytorch-lightning/discussions/15188

[11] A. Vallero, S. Tselonis, D. Gizopoulos, and S. Di Carlo, "Multi-faceted microarchitecture level reliability characterization for nvidia and amd gpus," in *2018 IEEE 36th VLSI Test Symposium (VTS)*. IEEE, 2018, pp. 1–6.

[12] D. Sartzetakis, G. Papadimitriou, and D. Gizopoulos, "gpufi-4: A microarchitecture-level framework for assessing the cross-layer resilience of nvidia gpus," in *2022 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2022, pp. 35–45.

[13] S. K. S. Hari, T. Tsai, M. Stephenson, S. W. Keckler, and J. Emer, "Sassifi: Evaluating resilience of gpu applications," in *Proceedings of the Workshop on Silicon Errors in Logic-System Effects (SELSE)*, 2015.

[14] L. Yang, G. Papadimitriou, D. Sartzetakis, A. Jog, E. Smirni, and D. Gizopoulos, "Gpu reliability assessment: Insights across the abstraction layers," in *2024 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2024, pp. 1–13.

[15] I. S. Haque and V. S. Pande, "Hard data on soft errors: A large-scale assessment of real-world error rates in gpgpu," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE, 2010, pp. 691–696.

[16] N. DeBardeleben, S. Blanchard, L. Monroe, P. Romero, D. Grunau, C. Idler, and C. Wright, "Gpu behavior on a large hpc cluster," in *Euro-Par 2013: Parallel Processing Workshops: BigDataCloud, DIHC, FedICI, HeteroPar, HiBB, LSDVE, MHPC, OMHI, PADABS, PROPER, Resilience, ROME, and UCHPC 2013, Aachen, Germany, August 26-27, 2013. Revised Selected Papers 19*. Springer, 2014, pp. 680–689.

[17] A. Kokolis, M. Kuchnik, J. Hoffman, A. Kumar, P. Malani, F. Ma, Z. DeVito, S. Sengupta, K. Saladi, and C.-J. Wu, "Revisiting reliability in large-scale machine learning research clusters," *arXiv preprint arXiv:2410.21680*, 2024.

[18] H. S. Gunawi, R. O. Suminto, R. Sears, C. Golliher, S. Sundararaman, X. Lin, T. Emami, W. Sheng, N. Bidokhti, C. McCaffrey *et al.*, "Fail-slow at scale: Evidence of hardware performance faults in large production systems," *ACM Transactions on Storage (TOS)*, vol. 14, no. 3, pp. 1–26, 2018.

[19] N. Cini and G. Yalcin, "A methodology for comparing the reliability of gpu-based and cpu-based hpcs," *ACM Comput. Surv.*, vol. 53, no. 1, Feb. 2020. [Online]. Available: https://doi.org/10.1145/3372790

[20] D. Tiwari, S. Gupta, G. Gallarno, J. Rogers, and D. Maxwell, "Reliability lessons learned from gpu experience with the titan supercomputer at oak ridge leadership computing facility," in *Proceedings of the international conference for high performance computing, networking, storage and analysis*, 2015, pp. 1–12.

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

[22] S. Gupta, T. Patel, C. Engelmann, and D. Tiwari, "Failures in large scale systems: long-term measurement, analysis, and implications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[23] B. Nie, J. Xue, S. Gupta, C. Engelmann, E. Smirni, and D. Tiwari, "Characterizing temperature, power, and soft-error behaviors in data center systems: Insights, challenges, and opportunities," in *2017 IEEE 25th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2017, pp. 22–31.

[24] G. Ostrouchov, D. Maxwell, R. A. Ashraf, C. Engelmann, M. Shankar, and J. H. Rogers, "Gpu lifetimes on titan supercomputer: Survival analysis and reliability," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020, pp. 1–14.