Analyzing a Lifetime of Failures on a Cray XC40 Supercomputer

Kevin A. Brown
Argonne National Laboratory
Lemont, Illinois, USA
kabrown@anl.gov

Tanwi Mallick
Argonne National Laboratory
Lemont, Illinois, USA
tmallick@anl.gov

Zhiling Lan
University of Illinois Chicago
Chicago, Illinois, USA
zlan@uic.edu

Robert Ross
Argonne National Laboratory
Lemont, Illinois, USA
rross@anl.gov

Christopher D. Carothers
Rensselaer Polytechnic Institute
Troy, New York, USA
carotc@rpi.edu

Abstract

We analyze hardware errors over the seven-year lifetime of the Theta supercomputer, a large-scale Cray XC40 system at the Argonne Leadership Computing Facility. To ensure accurate interpretation of the logs, we leverage expert knowledge to clean the dataset and remove redundant information. Temporal and spatial analysis techniques are then used to expose how failures and errors trend over time and across components in the system. Additionally, we correlate hardware error logs to system downtime logs to capture the relationship between critical errors and outages over the lifetime of the system. The results in this work represent a state-of-the-practice report highlighting how severe error types vary over time and across different component types, such as on-node and off-node (network) components. We also demonstrate the effectiveness of our technique in simplifying log analysis by using a unified error classification across components from different vendors, providing valuable insights into normal and anomalous system behaviors.

CCS Concepts

• **General and reference** → Evaluation; • **Networks** → **Network reliability**; • **Hardware** → **Hardware reliability**.

Keywords

Supercomputing, log analysis, error analysis, failure analysis

ACM Reference Format:
Kevin A. Brown, Tanwi Mallick, Zhiling Lan, Robert Ross, and Christopher D. Carothers. 2018. Analyzing a Lifetime of Failures on a Cray XC40 Supercomputer. In *Proceedings of The 2025 Cray User Group Conference (CUG2025).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Modern high-performance computing (HPC) systems consist of thousands of individual components such as compute elements, network interfaces, and switching elements. These systems are often highly tuned with components tightly integrated to provide maximum overall performance for diverse scientific workloads. Failures across elements of the system are inevitable and can have varying impact on performance, productivity, and stability. Fortunately, systems such as the Cray XC40 Theta supercomputer at the Argonne Leadership Computing Facility (ALCF) have been architected to generate a wealth of logs for tracking utilization and stability, such as hardware error logs that capture failures in components and subcomponents [17].

Prior studies have shown that understanding failure patterns over the lifetime of a system can yield invaluable insight into its performance and stability characteristics [7, 14, 21]. This understanding can also expose how design, operational policies, and usage patterns can impact system stability. Such knowledge is critical for updating our models of failures on modern systems, guiding design efforts, and defining realistic stability goals and resiliency strategies for future deployments [6, 9].

Understanding the failure patterns over the lifetime of large systems poses challenges, however. These challenges stem from the complexity and heterogeneity of components in the infrastructure, the volume of the logs generated over the lifetime of the system, and the difference in timescales of error log records and user/workload activities that may trigger the errors. Therefore, extracting knowledge from these logs requires nontrivial data analysis efforts.

We analyze hardware errors over the seven-year lifetime of the Theta supercomputer, a large-scale Cray XC40 system used by thousands of researchers from across the globe. To ensure accurate interpretation of the logs, we leverage expert knowledge to clean the dataset and remove duplicate and redundant information that could lead to erroneous analysis. Once the dataset is prepared, we use various techniques to understand the occurrence, distribution, and impact of different types of errors on the system and its components. First, we define a common classification for errors from components of different vendors to enable uniform systemwide analysis across all types of hardware components. Second, temporal and spatial analysis techniques are used to expose how failures and errors trend over time and across components in the system. Third, we correlate hardware errors with system downtime logs to capture how error patterns relate to unplanned system outages.

The results in this work represent a state-of-the-practice report on the failure patterns at one of the U.S. Department of Energy's critical user computing facilities. It highlights how severe errors vary over time and across different component types, such as on-node and off-node (network) components. The insights from this

CUG2025, Jersey City, NJ

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

work can be used to improve system monitoring and stability. For example, our preprocessed data can be used to develop more accurate failure prediction models for improved predictive maintenance and job scheduling policies. Additionally, failure patterns across different component types can guide the design of more resilient infrastructure for future workloads.

2 Related Work

2.1 HPC Log Analysis

Recent research in HPC system log analysis has aimed to deepen understanding of system behavior. Park et al. [10] developed LogSCAN, a big data analytics framework leveraging Apache Spark and Cassandra to process multiyear logs from the Titan supercomputer, enabling pattern mining and correlation studies across spatial and temporal dimensions. Other works have introduced scalable architectures using distributed NoSQL databases and in-memory processing to support high-throughput log ingestion and anomaly detection [11]. Additionally, Hickman et al. [8] proposed techniques to link syslog messages back to their originating source code, enhancing the interpretability of log messages and enabling more effective root cause analysis in systems. Yoon et al. [22] analyzed job scheduler execution logs to identify resource-related delays in HPC systems and found that idle resource wait times significantly impact job execution. They proposed a backfilling algorithm to optimize resource utilization, which effectively reduced overall job execution time. The Cray XC40 systems have been the focus of many of these studies [18, 19]. Our work presents a data preprocessing methodology that effectively reduces hardware log volumes while still enabling meaningful log analysis.

2.2 HPC Failure Analysis

In parallel, several studies have focused specifically on failure characterization and prediction in HPC environments. El-Sayed and Schroeder [4] conducted a decade-long analysis of HPC failure logs from Los Alamos National Laboratory to examine the impact of various environmental, hardware, and usage factors on system reliability. Their study revealed key correlations between failures and external influences such as power quality, thermal conditions, and even cosmic radiation. Park et al. [12] analyzed detailed scheduler logs from a production supercomputer to identify key workload features associated with job failures. Using feature analysis and six machine learning models, they demonstrated that tree-based approaches provide high prediction accuracy and computational efficiency for forecasting HPC job failures. Other researchers have also examined failure patterns by analyzing HPC system logs [5, 13, 20]. For example, Schroeder and Gibson conducted pioneering research by analyzing large-scale system logs, identifying correlations between hardware faults and failure events at large HPC sites [15, 16]. Zheng et al. presented a methodology for preprocessing system logs to enhance failure analysis [23]. Building on this, Zheng et al. introduced a co-analysis approach, integrating RAS logs and job logs specifically for the Blue Gene/P system to achieve a deeper understanding of failures [24]. For another facility, Gupta et al. presented an extensive study of reliability characteristics across multiple large-scale HPC production systems [7]. These studies collectively demonstrate the importance of analyzing system logs for comprehensive failure characterization in HPC environments. We build on these efforts, emphasizing the importance of identifying abnormal activities such as maintenance windows when analyzing behaviors over the lifetime of a system.

3 The Theta System and Datasets

Theta was an 11.7-petaflops supercomputer deployed in the Argonne Leadership Computing Facility (ALCF) at Argonne National Laboratory. The system was used for workloads from a variety of scientific and engineering domains, and it peaked at position 16 on the Top500 list.

3.1 Cray XC40 Architecture

Theta was based on the Cray XC40 architecture and comprised 4,392 nodes distributed across 24 racks. Each node had Intel Xeon Phi 7230 processors, with a total of 281,088 cores across the system. Nodes were interconnected by using the Aries network architecture in a 2D dragonfly topology. The main components of this architecture were the Intel processors and associated memory hierarchy on the node, the network interfaces connecting the node to swithces in the network, and the Aries switches that interconnected switches and nodes in the system. The monitoring system also identified the Aries link control block (LCB) as a separate component in the logs. Each LCB manages a router-to-router connection and performs the actual transfer and network load sharing between routers [1].

3.2 System Lifetime

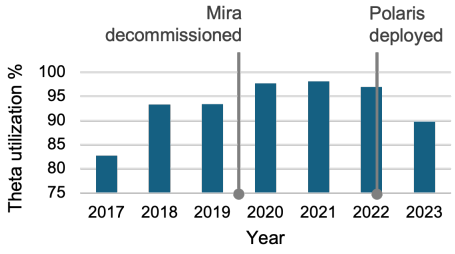


Figure 1: Theta's annual utilization. Users moved to and from Theta as resource availability changed in the facility.

Theta went into production on July 1, 2017, and was decommissioned at the end of 2023. During that time, the facility decommissioned and deployed other leadership-class systems alongside Theta. Mira, a 10-petaflop IBM Blue Gene/Q system, was decommissioned at the end of 2019; and Polaris, a 34-petaflop HPE Apollo Gen10+, was deployed in August 2022. As resource availability changed in the facility, so did user utilization patterns, as shown in Figure 1. Theta's utilization ramped up after deployment as more projects moved to the system and ramped down as it approached its end of life and users moved projects to new resources. Between Mira and Polaris, Theta was the primary system at the ALCF.

Theta underwent two major expansions. It was originally deployed with 3,624 KNL nodes and then increased to 4,392 KNL nodes at the end of 2017. Subsequently, 24 NVIDIA DGX nodes

were added to a ThetaGPU partition in 2020. ThetaGPU nodes are not included in our analysis.

As typical with HPC production systems, Theta underwent routine maintenance to update software and hardware and perform other preventive maintenance issues. For most of its lifetime, the system was taken offline every other Monday for a scheduled downtime window called "Big Run Mondays." During a Big Run, special application runs could be conducted alongside maintenance activities. Otherwise, the system was taken offline during unscheduled downtimes to address urgent issues such as hardware failures that interrupted the operations of the systems. During all downtime windows, regular users were not allowed to submit jobs. Therefore, understanding the normal behaviors of the system requires focusing on its activities outside of downtime periods.

3.3 Hardware Error Logs Dataset

The Theta hardware error dataset, detailed in the ALCF data dictionary, encompasses logs from the Theta supercomputer, capturing various types of hardware-related errors. This dataset records critical information for monitoring and diagnosing hardware performance and reliability issues, featuring attributes such as timestamp, failed component, error category, and error code. The dataset provides structured insights into error occurrences, facilitating analysis of error patterns and the frequency of particular issues across Theta's hardware components. The Theta hardware error dataset spans a significant duration, covering data from June 20, 2017, to January 1, 2024.

3.3.1 Error Categories. The Theta hardware error dataset includes a variety of error codes, each corresponding to specific hardware issues encountered within the system. These error codes are categorized to facilitate effective monitoring and diagnosis. The primary error categories are as follows:

- **Correctable memory errors**: These indicate memory issues within the Aries interconnect that are correctable, helping prevent data corruption without causing critical interruptions.
- **Machine Check Exception (MCE) errors**: These represent hardware errors detected by the processor's architecture, which may impact system stability.
- **Transaction errors**: These capture issues related to transactions within the Aries interconnect, which could affect data exchange between nodes.
- **Transient errors**: These are temporary, minor issues within the Aries interconnect that do not generally require immediate intervention.
- **Critical errors**: These denote severe errors within the Aries interconnect that necessitate prompt attention due to their potential impact on system operations.
- **Informational errors**: This category includes non-critical messages related to the Aries interconnect, primarily serving as informational logs that assist in performance monitoring without signaling urgent issues.

Each error entry in the dataset is associated with an error code, providing a numerical identifier for the specific error encountered.

3.3.2 Duplicated Error Records. For a given component, multiple errors may be reported for the same timestamp. Through discussion with the administrators, we determined this situation to be due to a single hardware failure (for example, a faulty memory channel) being encountered by multiple concurrent execution threads in an application, whereby each thread triggers a unique error record. When OS processes or other diagnostic tools encounter these errors, they may also report them in the log as well.

3.3.3 Persistent Errors over Time. When a given component fails, it may transition to a failed/error state until it is replaced or recovers from the error. Even though this situation is precipitated by a single failure, diagnostic tools that periodically probe the component's state will log an error every time the component is checked. Therefore, error logs will contain a stream of unique log records for the component over time. These should not be interpreted as unique failures or errors.

3.4 Machine Status Dataset

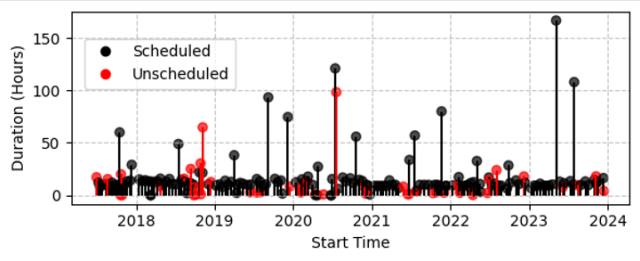


Figure 2: Timeline and durations of Theta's downtimes (outages).

The machine status dataset is maintained by the facility's operations team to record system downtimes. It has records of both complete system downtime, when all nodes are unavailable to users, and partial downtimes, when only some nodes are unavailable. Each entry describes a downtime period, indicating the start and end times of the period, the number of nodes unavailable, and whether the entire machine is down. For full system downtimes, the record will indicate whether the downtime is scheduled or unscheduled.

Figure 2 shows the records in the dataset. It indicates that most downtimes last for a few hours with few outliers. The longest downtime, recorded in May 2023, was due to facility-wide cooling and power issues.

4 Methodology

4.1 Redundancy Removal

The Theta hardware error log files contain two types of redundancies that can complicate analysis and obscure the actually patterns that exist within the data. The first type involves exact duplicate entries within each error category, where logs share identical attributes such as ERROR_CODE, EVENT_EPOCH_TS, and FAILED_COMPONENT. To address this, we applied an initial filtering step to identify and remove exact duplicates for each defined error type. By isolating entries and dropping rows with identical

Table 1: Unified Error Classification. Columns show the value of fields in the dataset used to determine the error class. The ERROR_MAP_JSON field contains a list of key-value pairs, with keys such as UC (uncorrected error), PCC (processor corruption context), and VAL (valid).

Unified Error Class	ERROR_CATEGORY_STRING	ERROR_MAP_JSON: UC	ERROR_MAP_JSON: PCC	ERROR_MAP_JSON: VAL
Critical	GHAL_ARIES_CRITICAL_ERRORS			
	MCE_ERROR	1	1	1
Major	GHAL_ARIES_TRANSACTION_ERRORS			
	MCE_ERROR	1	0	1
Intermediate	GHAL_ARIES_TRANSIENT_ERRORS			
Minor	GHAL_ARIES_CORRECTABLE_MEMORY_ERRORS			
	MCE_ERROR	0		1
No-considered	GHAL_ARIES_INFORMATIONAL			
	MCE_ERROR			0

combinations of timestamp, failed component, and error code, we streamlined the dataset and ensured that each unique error event was counted only once within its category. To quantify the impact, we calculated the reduction percentage for each error type, providing insight into the extent of redundant data removed.

The second redundancy removal focuses on time-based filtering to address near-duplicate events within a specified time window (here, 70 seconds). We refer to these events as persistent errors. For each unique component in the dataset, the entries were sorted by EVENT_TIMESTAMP, and the code computed the time difference between consecutive logs. If an event occurred within the 70-second threshold and shared the same FAILED_COMPONENT and ERROR_CODE as another, it was marked as a duplicate. Preliminary exploration of the records revealed the time difference between near-duplicate errors was 60 seconds of less for each error category – an additional 10 seconds was added to this upper limit to account for slight variations observed at the boundary between days. The filtered dataset retains only non-duplicate entries, effectively removing repeated logs generated within a short timeframe, which could result from the same event being logged multiple times within a short interval. For each persistent error, a note of the duration of the error is included in its retained record.

Together, these two filtering techniques refine the data by first eliminating identical duplicates across specific fields and then addressing continuous or near-continuous entries, enhancing data quality for subsequent analysis. The resulting reduction in log volumes from the redundancy removal is presented in Section 5.1.

4.2 Hardware Error Classification

Records in the hardware error dataset, described in Section 3.3, include an ERROR_CATEGORY_STRING field that indicates the subsystem that generated the error. However, to understand the impact of the error on workload performance and system productivity, we must combine information from other fields in the error record with knowledge of how the operation of each subsystem impacts workloads and services.

Guided by vendor documentations, we classify hardware errors into four severity classes based on their likely impact on performance and productivity. These are outlined in Table 1 and discussed below.

Critical errors: These errors indicate failures that severely impact normal hardware operation and, by extension, the workloads and services. Recovery from critical failures typically requires resetting or replacing the faulty hardware and restarting the workload or service.

Major errors: These errors indicate persistent abnormal behaviors that usually cannot be resolved by retries but may *not* require restarting the node. These include failures that involve misconfigured application or service, such as an application attempting to use more than its allocated buffer size. Resolution of such errors usually requires application or service reconfiguration action.

Intermediate errors: These errors indicate non-persistent failures that are usually resolved by retrying the activity without needing to restart the application, service, or hardware. These errors include transmission timeouts due to temporary, unexpected congestion. The retries required to clear such errors would noticeably impact performance, such as network retransmission causing increased communication latencies.

Minor errors: These are warnings or failures that may be automatically corrected or ignored by the system without noticeable performance impact. These include soft errors that are automatically corrected by the error correction code or double bit flips detected in unconsumed data. Although minor errors may not immediately impact performance or productivity, their prolonged occurrence may eventually presage more severe failures that impact system stability.

4.2.1 Classification Challenges. While a best-effort attempt is made to correctly classify the likely impact of each error, the actual impact of all errors is difficult to determine based solely on the information captured in the logs. Understanding the actual impact of some error may require live or postmortem diagnosis of the hardware to determine the error's root cause and wider effects. In other cases, it may require feedback from users and system administrators on their experience with and response to the error. Since neither the system nor such feedback is available, we use system documentations and information in the logs to determine the likely impact of each error.

One specify type of error, MCE uncorrected recoverable (UCR) errors, could not be easily classified using documentation and log information. UCR errors are uncorrected but will not cause immediate failures because the processor context has not been corrupted [3]. However, they cause critical errors in the future if not resolved. When software error recovery support is present in the machine check architecture mechanism to respond to specific errors, the AR (Action Required) field is present in the records and set to '1' [3]. Since the AR field is absent in our dataset, we conclude that this

support is not present and mark UCR errors as major. The dataset has only 139 of these records.

4.2.2 Omitted Error Records. This study focuses on errors originating from the hardware during production utilization. Logs of temperature alerts and software asserted errors are present in the dataset but omitted from our current analysis. We consider these errors as environmental or external to the hardware. The system also generates errors due to diagnostics tools and other activities conducted while undergoing maintenance that do not reflect behaviors under normal utilization. Unless otherwise stated, we also omit these errors from our analysis. To identify the periods when the system is marked as down and available for regular operation, we use data from the machine status logs described in Section 3.4.

4.3 Correlation across Datasets

To better understanding the trends and impact of hardware failures, we need to consider the state of the system when the errors occurred. Specifically, we correlate hardware errors with system status for two reasons. First, we want to identify whether an error was generated during a time of normal usage or during a period outside of regular operations, such as a maintenance window. This information is necessary because we focus our study on how errors manifest themselves and impact normal operations. Second, we want to determine how error patterns change before and after system downtime. This information may be used to improve monitoring strategies for better failure detection and prediction.

To understand the state of the system when the errors occurred, we cross-reference errors logged in the hardware error dataset with downtimes reported in the machine status dataset. Each error has a timestamp of when the error occurred. We use records from the machine status dataset to check whether the machine was down or online to users at the time of the error. We refer to the period between the end of one downtime and the start of the next as an online window, i.e., the window of time when the system is online. Each window is given a unique ID, and all errors generated during the window are tagged with this ID. Errors without a window ID are removed from the dataset prior to our analysis since they were not generated during normal operations. A total of 6.1% of errors were removed by this process.

We augment error records with additional information about its window to provide more insights into errors leading to failures. A flag is added to indicate whether its window ended with a scheduled or unscheduled downtime. The start time and end time of the window are also used to determine the error's temporal relationship to the downtimes before and after its occurrence.

5 Results

5.1 Redundancy Removal

From June 2017 to January 2024, a total of approximately 479 million errors were recorded across all categories. This figure is based on an average of 5.99 million errors per month over a span of 80 months. The error data consumed an average of 4 GB of storage per month, resulting in an estimated total storage usage of 320 GB over the entire period. After applying redundancy removal techniques, the dataset was reduced to approximately 8 million unique error entries, occupying approximately 5 GB of storage, significantly reducing the data volume while preserving the integrity of the analysis.

5.2 Distribution of Failures

To analyze temporal patterns in error occurrences, we computed error counts across multiple time granularities. For yearly analysis, we aggregated errors according to the calendar year. Similarly, for monthly trends, we grouped errors by the combination of month and year, while for daily analysis, we used the exact calendar day of each error occurrence. To study time-of-day patterns, we grouped errors based on the hour in which they occurred, capturing variation across the 24-hour period. Similarly, to examine trends across the week, we categorized errors by the day of the week on which they occurred, from Monday through Sunday. We performed each of these aggregations separately for every error category, enabling a detailed understanding of how different types of errors vary across dimensions.

5.2.1 Temporal: Yearly. Figure 3 shows the distribution of error categories over the period from 2017 to 2024, revealing a consistent dominance of minor errors, which account for the vast majority of total errors each year. In most years, minor errors constitute over 90% of the total, reaching as high as 98.3% in 2023. Notably, intermediate, major, and critical errors remain relatively infrequent throughout the timeline. However, years such as 2018 and 2021 exhibit a more visible proportion of intermediate and critical errors, with intermediate errors peaking at 6.8% in 2018 and critical errors at 5.6% in 2021. Despite these fluctuations, the trend suggests a favorable error profile, with severe categories being rare and a gradual reduction in non-minor errors over time. This may indicate improvements in process quality or better mitigation strategies being implemented in recent years.

5.2.2 Temporal: Daily. The temporal analysis of system error logs reveals distinct patterns across the four severity classes: minor, intermediate, major, and critical. Minor errors are the most frequent, exhibiting a clear upward trend from 2017 to 2024. This sustained increase may be attributed to the increase in memory errors as the hardware ages due to regular utilization. In contrast, intermediate errors are less frequent but display sharp, irregular spikes, particularly around late 2018 and early 2022. These episodic surges correspond to specific operational events such as system updates or temporary misconfigurations. Major errors occur even less frequently and remain low in volume throughout the timeline, with occasional bursts. One of the most notable periods of increased activity occurs in 2022, potentially indicating localized issues or cascading failures. Critical errors are rare but exhibit pronounced spikes, including significant peaks in the 2018 and 2022 period, signaling isolated but severe disruptions. Facility reports confirm there were two major power outages due to a substation breaker tripping in the month of September 2018, and multiday parallel file system failures during February 2022 correlate with these spikes. The clear disparity in frequency and pattern across the severity levels underscores the importance of tailored monitoring and mitigation strategies. While minor errors require scalable logging and filtering, higher-severity errors demand rapid diagnostics and targeted intervention because of their potential systemwide impact.

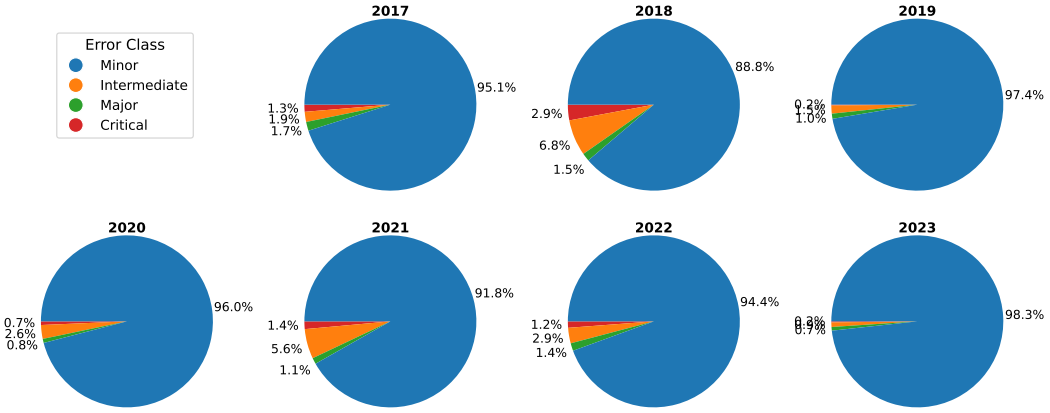


Figure 3: Annual distribution of error categories from 2017 to 2023. Each pie chart represents the proportional breakdown of minor, intermediate, major, and critical errors for a given year. Minor errors consistently dominate across all years, while the share of more severe error types (intermediate, major, critical) remains relatively low, with notable deviations in 2018 and 2021.

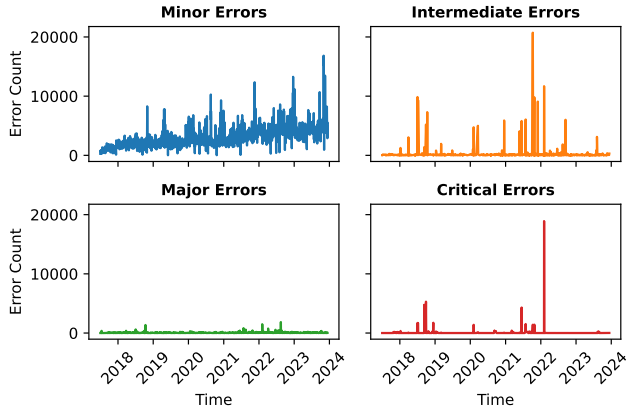


Figure 4: Daily count of system errors from 2017 to 2024 across four error categories: minor, intermediate, major, and critical. Minor errors exhibit a steadily increasing trend with regular fluctuations. Intermediate errors show sporadic spikes, including a sharp peak in early 2022. Major errors remain relatively low and infrequent, with occasional localized increases. Critical errors are rare but exhibit a prominent spike around late 2022.

5.2.3 Temporal: Monthly. Figure 5 presents the monthly distribution of errors throughout the system's lifetime. Box and whisker plots show the distribution across all years and line plots show the trends for 2021 and 2022. High numbers of minor errors occur each month, but their increase as the system ages is only partially visible in the 2021 and 2022 trend lines. These trends show minor error rates increasing from February to July and then the 2022 rates trend downward from August onward while the 2021 rates increase after August. Box and whisker plots show inconsistent variations across years, with months like November showing wide ranges and months like May showing smaller variations. These inconsistencies in the month-by-month trends and distributions indicate that the age of the system is not a sufficient indicator to predict the number of minor errors that will occur. Other factors such as utilization, the type of workloads, and the maintenance schedules will likely impact wear and tear and should be considered in predicting minor error rates.

Critical, major, and intermediate errors are infrequent but show clear month-specific spikes as well as some common monthly trends across years. These spikes are due to isolated high-impact events. For these errors, the spike during February 2022 was likely due to multiple parallel file systems failures throughout that month. The other significant spike in critical errors occurred in September 2018 due to multi-day power outages at the facility. The major error spike during August 2022 did not coincide with any documented hardware outages, suggesting system software or application issues as the likely cause. Hence, additional data must be considered in the analysis of these errors. The intermediate error spike during October 2022 coincides with two incidents of cabinets going down. In one case, a redundant cabinet went down and resulted in the system remaining online but in a degraded state for 10 days. In the second case, the cabinet with the scheduler resource manager went down and affected all running jobs. While we can correlate these spikes with incidents reported by system administrators, further analysis incorporating workload and operational data is needed to confirm causation. The other common trend observed for critical, major, and intermediate errors is the consistently small distributions of errors for the months of April, May, June, and November. Even with the occasional outliers, the number of errors produced in these months does not vary significantly across the years. Nonetheless, these error patterns highlight that while minor errors form a consistent backdrop, higher-severity error types exhibit more episodic and month-dependent behavior, underlining the importance of granular, month-level monitoring for early detection and mitigation of anomalies.

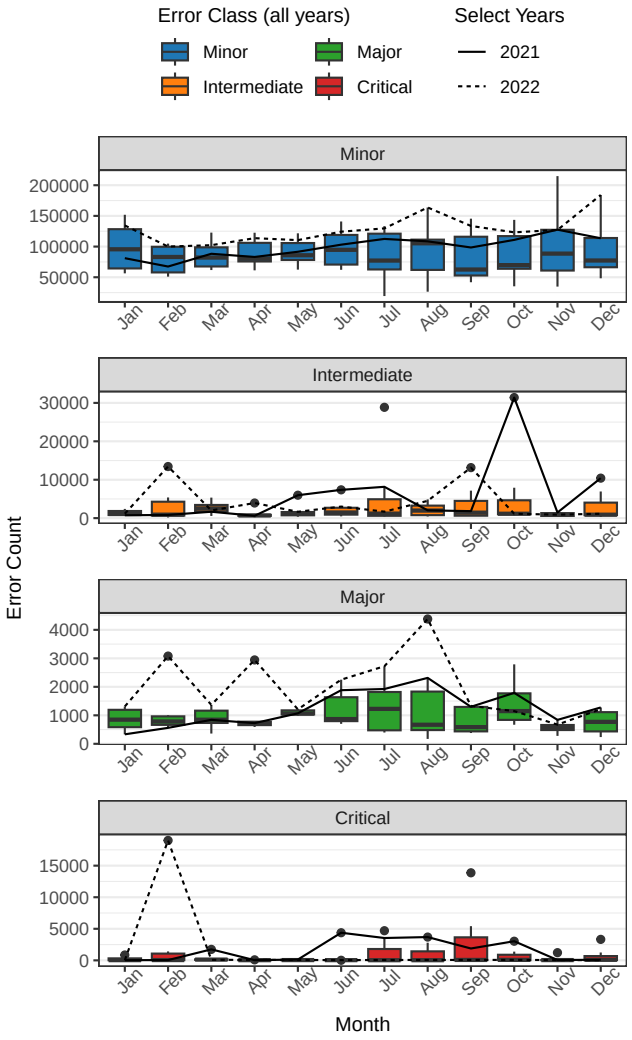


Figure 5: Distribution of errors generated during each month over the life of the system. Box and whisker plots show the monthly errors across all years while the line plots highlight the errors for years 2021 and 2022.

5.2.4 Temporal: Day of Week. Figure 6 illustrates the distribution of system errors by day of the week across the four error classes. The weekly distribution reveals distinct behavioral patterns for each error type. Minor errors are by far the most frequent and remain relatively stable throughout the week, with a slight increase in peaks observed midweek from Wednesday through Saturday. This pattern suggests that routine system activity or background processes may consistently contribute to these low-severity issues, regardless of the day. Intermediate errors display noticeable outliers on Sunday and more bounded but varied distributions on other days, potentially indicating more consistent elevated system stress or workload on days besides Sundays. Major errors occur at a much lower frequency but show a relatively even distribution across the

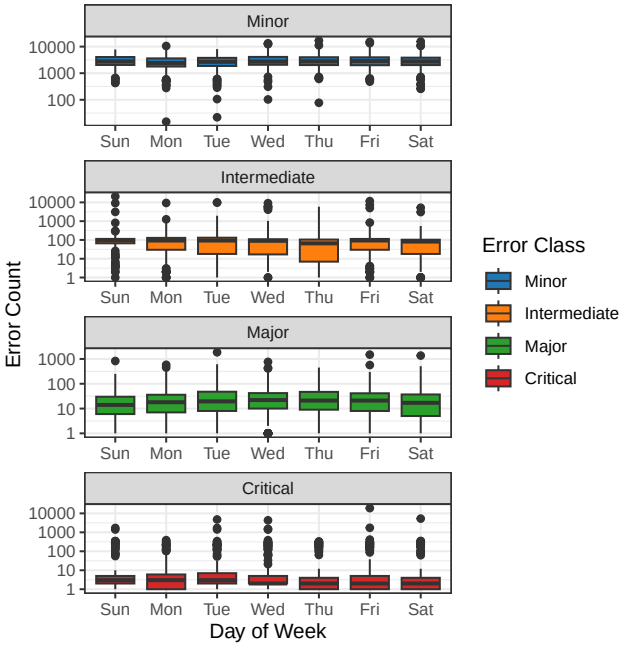


Figure 6: Distribution of system errors by day of the week across the four error classes.

week suggesting no strong weekday dependency. Critical errors, while rare, exhibit lower peak occurrences on Mondays and Thursday, which likely corresponds to the weekly planned maintenance activities carried out on Monday and potentially other operational activities scheduled around Thursdays.

5.2.5 Temporal: Time of Day. Figure 7 illustrates the distribution of system errors by hour of the day, segmented across the four error classes: minor, intermediate, major, and critical. As observed in previous analyses, minor errors overwhelmingly dominate across all hours, maintaining a consistently high count of above 100 errors per hour. This consistency suggests that minor errors occur independently of time, likely driven by regular system processes. Intermediate errors, while significantly lower in volume, do not exhibit any distinct hourly pattern throughout the day besides the occasional spikes. Major errors occur infrequently and uniformly throughout the day, without any significant hour-specific concentration. Critical errors, though rare and prone to extreme spikes, do not display any hourly patterns for their average rates. The only common trend observed across error classes is that intermediate and critical error spikes do not occur before 7:00 AM. This display of time-of-day dependency may indicate that the major faults resulting in these spikes are often due to human-in-the-loop activities when users and administrators are awake.

5.2.6 Spatial: Daily Component Type. Figure 8 shows the daily distribution of errors across the different types of components. By decomposing daily distributions in this manner, we expose that the aggregated trends seen in Figure 4 do not reflect the trends experienced by all types of components. Rather, the monotonically

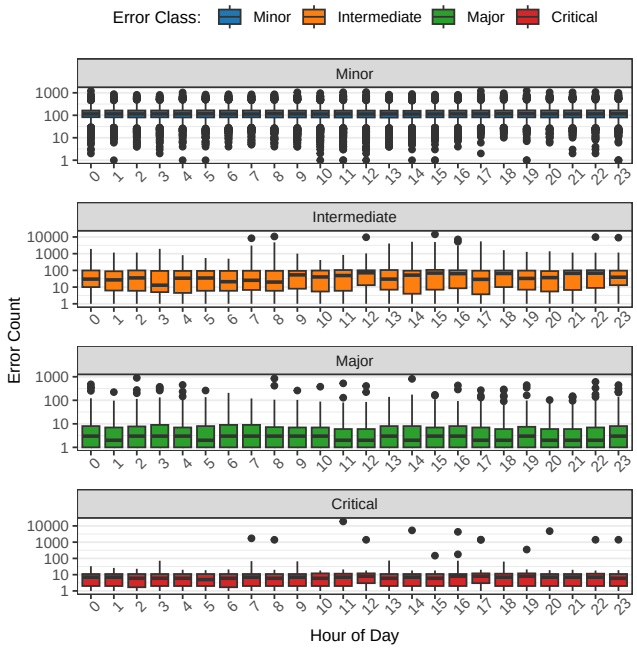


Figure 7: Distribution of system errors by hour of the day for each error class. Minor errors remain consistently high throughout the 24-hour period, while intermediate errors display time-dependent peaks during mid-morning and afternoon.

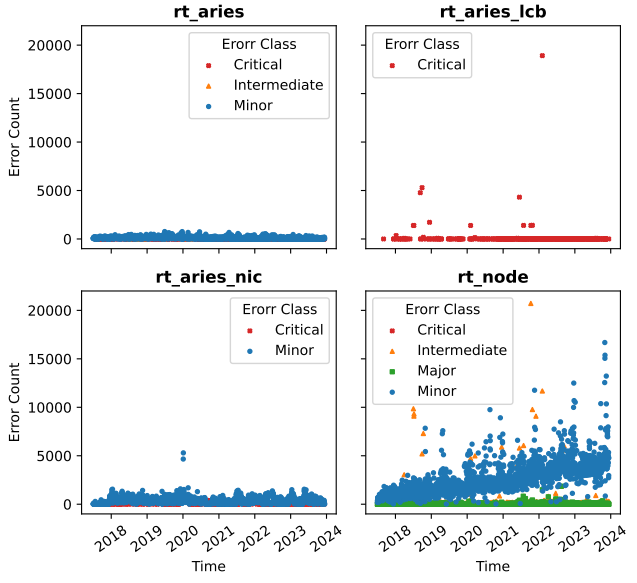


Figure 8: Daily distribution of errors across the different types of components.

increasing rate of minor errors occurs only on rt_nodes and not on other component types. Furthermore, the rates of minor errors on the rt_aries and rt_aries_nic components generally decrease or stay steady throughout the lifetime of the system.

Another notable observation in this breakdown is the type of errors reported by different components based on our error classification technique. rt_node components report all four classes of errors, while rt_aries_lcb reports only critical errors, and major errors are reported only from rt_nodes. Furthermore, the highest rates of critical and intermediate errors are reported from rt_aries_lcb and rt_node components, respectively.

Administrators, armed with this awareness of the different sources and relative impact of each class of error, can view high-level aggregated information and still understand the potential impact and origins of the observed trends. For example, looking back at Figure 4, one could correctly infer the following: (i) the intensity of spikes in critical and intermediate errors is likely due to rt_aries_lcb and rt_node components, respectively, and (ii) all major errors are from rt_node components.

5.3 Hardware Error Correlation with Downtime

Critical and major hardware errors can signal faults that necessitate urgent remedial action, potentially needing an unscheduled downtime to correct the fault. In such scenarios, early fault detection and rapid development of resolution strategies are paramount for minimizing the negative impacts to normal operations. We correlate the errors with system downtime events to observe error patterns prior to system outages and compare these patterns for scheduled and unscheduled downtimes.

Table 2: Theta Lifetime Status Overview

Total Online Duration:	2231.04 Days
Total Downtime Duration:	125.69 Days
Total Online:	94.67%
Scheduled Downtime:	4.41%
Unscheduled Downtime:	0.92%

5.3.1 System Downtime Statistics and Trends. Table 2 provides a summary of the system's state throughout its lifetime. The system was down for 5.33% of its lifetime, encompassing 179 scheduled and 24 unscheduled downtimes. We identify the periods between successive downtime sessions during which the machine was online and accessible to users for normal operations. Figure 9 shows the distribution of durations for when the machine was online and offline (down).

The figure shows that approximately 90% of all downtimes last for less than 24 hours. However, scheduled downtimes are likely to last longer than unscheduled downtimes. This is expected because scheduled downtimes may be used to undertake major hardware and software upgrades, whereas unscheduled downtimes are focused on resolving the specific, immediate issues that interrupt system operations.

The online duration subplot shows a spike in probability around 330 hours. This indicates that the system is more likely to be online for this duration than others. The ALCF scheduled routine biweekly

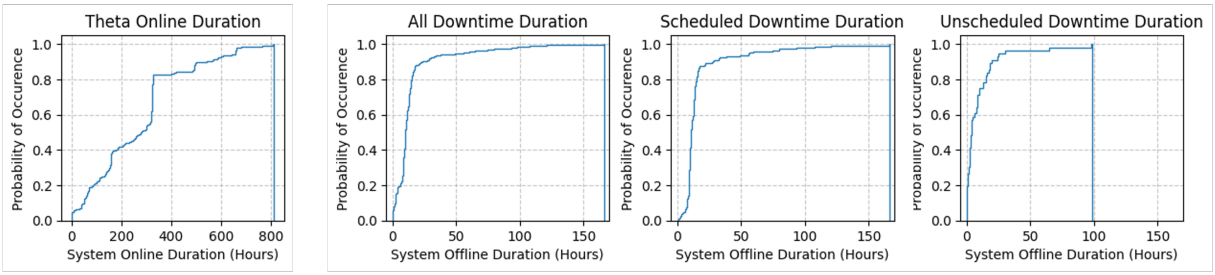
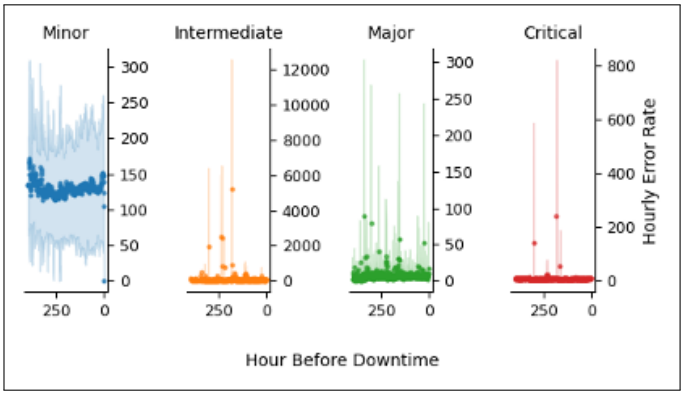
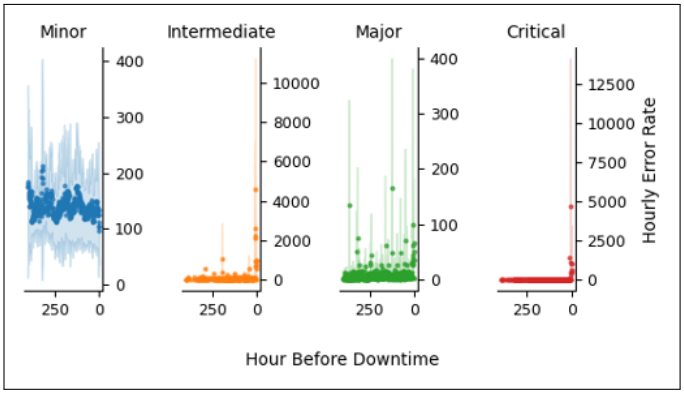


Figure 9: Cumulative distributions of durations Theta spent online and offline.

maintenance windows where the system was taken offline every other Monday for planned work. As a result, the online duration is most frequently two weeks or approximately 330 hours.



(a) Scheduled downtime.



(b) Unscheduled downtime.

Figure 10: Hourly rate of errors reported prior to downtimes.

5.3.2 Error Rates and Outages. To examine the relationship between errors and outages, we matched each error to the next downtime session after the error's occurrence, noting the time offset between the event's timestamp and the start of the downtime. By observing the variations in hourly error rates prior to downtimes, we can more fairly compare across all system outage events. We

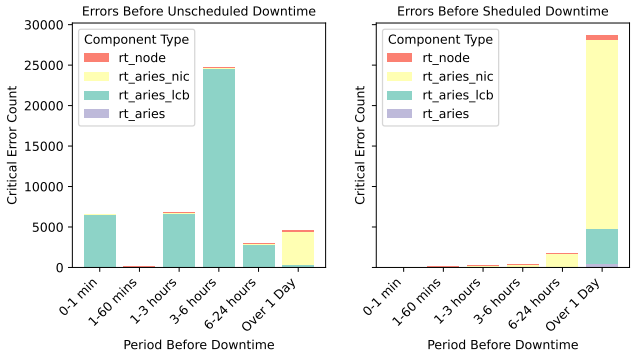


Figure 11: Count of critical errors reported by components before downtime.

calculated these hourly rates over the lifetime of the system as follows: Given a one-hour period, for example between the 6^{th} and 5^{th} hour before the system is taken down, we count the errors during this period before each outage and average the values across all similar periods.

Figure 10 reports the average rates of each class of error over time prior to the downtimes. Each dot represents the average number of errors reported during the same relative one-hour span preceding all downtime events, and the shaded area denotes the standard deviation. Rates prior to scheduled downtimes are presented in 10a, and rates prior to unscheduled downtimes are presented in 10b. Comparing the scheduled and unscheduled downtime plots, we find that the most significant difference appears between the trends for critical errors. Being mindful that the y-axis scales are not the same across plots, we see that the critical error rate is usually low and steady in both cases except for the few hours before unscheduled downtimes. Then, the rate spikes substantially as it approaches the zeroth hour before the downtimes. Multiple dots are observed rising above 1,000 errors per hour, with the highest approaching 5,000 errors per hour. Similarly, for intermediate and major errors, more dots can be seen with relatively higher rates close to the zeroth hour on the x-axis than during other periods. This situation is unique to these three classes of errors and only occurs prior to unscheduled downtimes. Hence, this technique can effectively demonstrate the relationship between the error classes and unscheduled outages. Additionally, this view of the data can highlight how the system

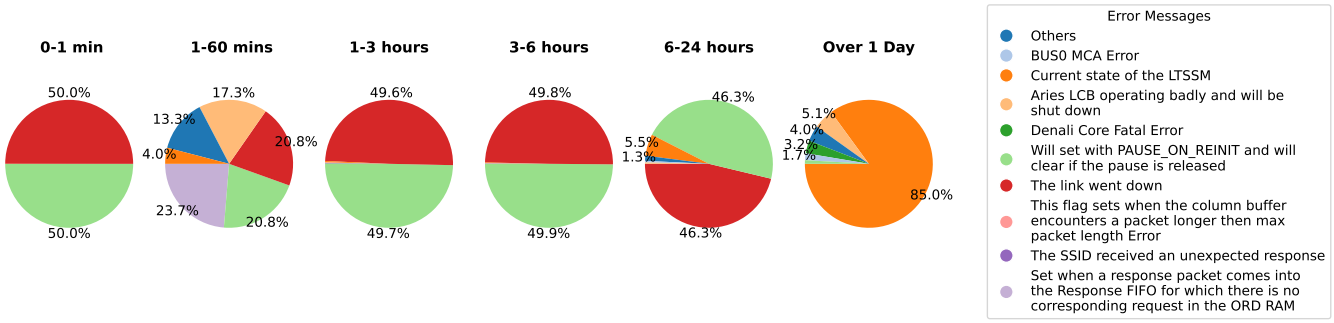


Figure 12: Distribution of critical error messages during each period before unscheduled downtimes.

degrades after failures are reported and the responsiveness of the facility's strategies to detect and address failures.

Minor errors do not generate system faults and have no impact on immediate outages. From our earlier results we confirmed that the aggregate error rate for this class steadily increases in time after the system is brought online. Therefore, since we are essentially looking backwards in time while comparing aggregating data from different periods of the system's lifetime, any trend seen for minor errors in these plots is a random artifact.

5.3.3 Failed Components and Error Messages. For a more detailed view of the critical errors related to downtimes, we identify the sources of these errors and track how they manifest within the minutes and hours before outages. The average critical errors generated by the type of component is presented in Figure 11. Each bar shows the count of errors during a period of time not including the time covered by adjacent bars. These results verify that a significant number of critical errors are reported within the 24 hours prior to an unscheduled downtime, with most errors coming from the rt_aries_lcb component. This component generated an average of approximately 25,000 errors during the 3- to 6-hour period before the system was taken offline. All other periods for the unscheduled downtime plot showed a noticeable count of critical errors except for the 1- to 60-minute period. While the 0- to 1-minute windows average over 5,000 critical errors, system administrators have explained that restarting nodes can cause some components to throw critical errors that are unrelated to the source of the actual failures. One example is when routers are restarted and the connected router-to-router ports report loss of connectivity. Hence, we surmise that most of the critical errors that actually cause outages occur over one hour before administrators complete arrangements to take the system offline for maintenance. The chart of the scheduled downtimes reveals that the system is relatively quiet during the hour before a scheduled downtime. The high count of errors for the period over a day before the outage can be attributed to online windows before scheduled outages being longer than those before unscheduled outages. The system was online for an average of 12.3 days before scheduled downtimes while it averaged 7.8 days before unscheduled outages.

Since unscheduled downtimes are undertaken only for unexpected issues, viewing the types of critical errors reported before these windows can provide insight into which errors are most likely related to the downtimes. Such insight may enhance failure predictions and guide preventive maintenance activities. We focus on the periods before unscheduled downtimes, and in Figure 12, we chart the error messages logged during these windows. Starting from the rightmost chart, LTSSM (Link Training Status State Machine) errors are the most frequent errors during the days before an outage. This message signals that a NIC encountered an issue negotiating the connection with its downstream port. On the other hand, "The link went down" and PAUSE_ON_REINIT errors are the most common messages during the breakdown of the 24 hours before the outage. This combination of errors signals that a network link has failed and the connection will not be retried. While root cause analysis of failures is beyond the scope of this work, these results demonstrate that our process of redundancy removal and error classification can enable fine-grain analysis of errors during periods of interest without extraneous and overwhelmingly redundant data. They definitively show the difference in error messages and patterns by component types when the system is in anomalous states that require unscheduled outages compared with when it is operating normally.

6 Discussion

6.1 Variations in Stability over Lifetime

The task of understanding and predicting system stability from hardware logs is becoming more complex for leadership-class systems in facilities such as the ALCF. The various perspectives of Theta's stability, spatial and temporal, over its lifetime show failures due to component aging and non-recurrent incidents. Figure 13 shows the daily errors generated by the system. One may have expected to observe a bathtub curve showing high "infant mortality" failures in its early life, relatively lower constant failures during the midlife period, and high end-of-life failures due to wear and tear [2]. External factors such as routine maintenance to minimize the impact of wear and tear and severe outages caused by facility upgrades will distorted these trends [25].

6.2 Failure Prediction

While failures due to isolated incidents are not easily captured in long-term forecasts, minor errors on Theta show a clear trend as the system ages (see Figure 4). However, even minor errors become harder to accurately predict over smaller timescales because of the

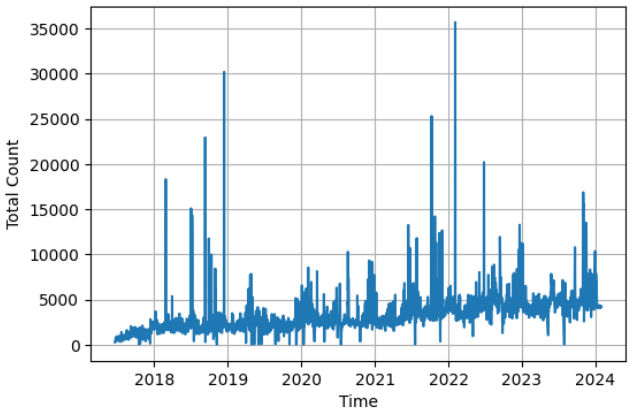


Figure 13: Count of total daily errors. The expected bathtub curve is not readily visible in this plot because of external factors impacting system reliability, such as frequent preventive maintenance and changes in user patterns and operational policies.

impact of external factors such as changes in user patterns and operational policies. And user patterns are, in turn, impacted by changes in resource availability in the facility, such as the deployment or decommissioning of other resources, the changes in the type of workloads that stress different components in the system, and deadlines for producing results and utilizing resource allocations. Modern efforts to understand and predict hardware failures must therefore be enhanced by cross-referencing data regarding operational policies and incident reports as well as users and workloads. Preemptively identifying which information is relevant and defining a systematic relationship between the data from the varied types of sources remain key challenges to be addressed.

6.3 Scalable Error Logging

Hardware vendors classify and log errors in different ways. However, operational and maintenance efforts must support general system stability and, therefore, consider the trends and impacts of all types of errors from all components. The disparate approaches to classifying errors from vendors complicate efforts to understand systemwide failure patterns. Until all vendors adopt a standardized approach to classifying and reporting errors, facilities can augment their monitoring infrastructure to define a unifying classification that captures the relative impact of each type of error. Our analysis using a unifying classification across Intel MCE and Cray Aries errors confirms the efficacy of this approach in exposing anomalous system behaviors related to severe failures. Furthermore, this classification enables efficient redundancy removal without impeding data analysis efforts. We posit that they may be used to improve log collection pipelines in production where effective monitoring is required but storage and bandwidth capacities are limited. Well-instrumented architectures such as Slingshot that can generate a wealth of hardware logs stand to benefit from this approach.

7 Conclusion

Throughout its seven-year lifetime, Theta reported approximately 479 million errors for 8 million unique component failure events. Our work demonstrates that useful insights can be gained from the logs by uniformly classifying errors across different component types based on the errors' potential impact on system and workload stability. We have confirmed that this classification technique can remove log redundancy by a factor of 60X without loss of the key information needed to understand the system's stability trends. Our results also illustrate that stability analysis on modern leadership-class systems cannot be effectively done without considering factors external to the hardware. For example, while most errors were correctable memory errors, signaling increased wear as the system aged, various major incidents caused spikes in errors of all severity. In both cases for Theta, the frequency and pattern of failure were impacted by preventive maintenance policies, changes in user and workload patterns, and changes in resources deployed at the facility. Future work will investigate how to efficiently identify and connect information about the architecture, operational policies, and user jobs to improve understanding and predictability of errors.

Acknowledgments

We thank Doug Waldron, Jon Bouvet, and the ALCF staff as well as Bob Alverson of HPE for consulting on the Theta system and the Cray XC40 architecture.

This work was supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research, under Contract No. DE-AC02-06CH11357. This work also used data generated at and resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

References

[1] Bob Alverson, Edwin Froese, Larry Kaplan, and Duncan Roweth. 2012. Cray XC series network. *Cray Inc., White Paper WP-Aries01-1112* (2012).

[2] John Cooper. 2024. Implementation of the Product Reliability Program. In *2024 Pan Pacific Strategic Electronics Symposium (Pan Pacific)*. 1–6. doi:10.23919/PanPacific60013.2024.10436446

[3] Intel Corporation. 2016. Intel® 64 and IA-32 Architectures Software Developer's Manual Volume 3B: System Programming Guide, Part 2". https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html

[4] Nosayba El-Sayed and Bianca Schroeder. 2013. Reading between the lines of failure logs: Understanding how HPC systems fail. In *2013 43rd annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 1–12.

[5] Ana Gainaru, Franck Cappello, and William Kramer. 2012. Taming of the Shrew: Modeling the Normal and Faulty Behaviour of Large-scale HPC Systems. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium*. 1168–1179.

[6] Jiangang Gao, Fang Zheng, Fengbin Qi, Yajun Ding, Hongliang Li, Hongsheng Lu, Wangquan He, Hongmei Wei, Lifeng Jin, Xin Liu, et al. 2021. Sunway supercomputer architecture towards exascale computing: analysis and practice. *Science China Information Sciences* 64, 4 (2021), 141101.

[7] Saurabh Gupta, Tirthak Patel, Christian Engelmann, and Devesh Tiwari. 2017. 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, SC 2017, Denver, CO, USA, November 12–17, 2017*, Bernd Mohr and Padma Raghavan (Eds.). ACM, 44.

[8] Megan Hickman, Dakota Fulp, Elisabeth Baseman, Sean Blanchard, Hugh Greenberg, William Jones, and Nathan DeBardeleben. 2018. Enhancing HPC system log analysis by identifying message origin in source code. In *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 100–105.

[9] Esteban Meneses, Xiang Ni, Terry Jones, and Don Maxwell. 2015. Analyzing the interplay of failures and workload on a leadership-class supercomputer. *Computing* 2, 3 (2015), 4.

[10] Byung H Park, Yawei Hui, Swen Boehm, Rizwan A Ashraf, Christopher Layton, and Christian Engelmann. 2018. A big data analytics framework for HPC log data: Three case studies using the Titan supercomputer log. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 571–579.

[11] Byung H Park, Saurabh Hukerikar, Ryan Adamson, and Christian Engelmann. 2017. Big data meets HPC log analytics: Scalable approach to understanding systems at extreme scale. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 758–765.

[12] Ju-Won Park, Xin Huang, and Chul-Ho Lee. 2024. Analyzing and predicting job failures from HPC system log. *The Journal of Supercomputing* 80, 1 (2024), 435–462.

[13] Antonio Pecchia, Domenico Cotroneo, Zbigniew Kalbarczyk, and Ravishankar K. Iyer. 2011. Improving Log-based Field Failure Data Analysis of multi-node computing systems. In *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*. 97–108.

[14] Elvis Rojas, Esteban Meneses, Terry Jones, and Don Maxwell. 2019. Analyzing a five-year failure record of a leadership-class supercomputer. In *2019 31st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, 196–203.

[15] Bianca Schroeder and Garth A Gibson. 2007. Understanding failures in petascale computers. *Journal of Physics: Conference Series* 78, 1 (July 2007), 012022. doi:10.1088/1742-6596/78/1/012022

[16] Bianca Schroeder and Garth A. Gibson. 2010. A Large-Scale Study of Failures in High-Performance Computing Systems. *IEEE Transactions on Dependable and Secure Computing* 7, 4 (2010), 337–350. doi:10.1109/TDSC.2009.4

[17] J Schutkoske. 2014. Cray XC System Level Diagnosability: Commands, Utilities and Diagnostic Tools for the Next Generation of HPC Systems. *Proceedings of the Cray User Group (CUG)* (2014).

[18] Shilpika Shilpika, Bethany Lusch, Murali Emani, Filippo Simini, Venkatram Vishwanath, Michael E. Papka, and Kwan-Liu Ma. 2022. Toward an In-Depth Analysis of Multifidelity High Performance Computing Systems. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. 716–725. doi:10.1109/CCGrid54584.2022.00081

[19] Shilpika Shilpika, Bethany Lusch, Murali Emani, Filippo Simini, Venkatram Vishwanath, Michael E. Papka, and Kwan-Liu Ma. 2024. A Multi-Level, Multi-Scale Visual Analytics Approach to Assessment of Multifidelity HPC Systems. In *2024 IEEE 24th International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. 478–488. doi:10.1109/CCGrid59990.2024.00060

[20] Marc Snir, Robert W Wisniewski, Jacob A Abraham, Sarita V Adve, Saurabh Bagchi, Pavan Balaji, Jim Belak, Pradip Bose, Franck Cappello, Bill Carlson, Andrew A Chien, Paul Coteus, Nathan A Debardeleben, Pedro C Diniz, Christian Engelmann, Mattan Erez, Saverio Fazzari, Al Geist, Rinku Gupta, Fred Johnson, Sriram Krishnamoorthy, Sven Leyffer, Dean Liberty, Subhasish Mitra, Todd Munson, Rob Schreiber, Jon Stearley, and Eric Van Hensbergen. 2014. Addressing failures in exascale computing. *Int. J. High Perform. Comput. Appl.* 28, 2 (May 2014), 129–173.

[21] Amir Taherin, Tirthak Patel, Giorgis Georgakoudis, Ignacio Laguna, and Devesh Tiwari. 2021. Examining Failures and Repairs on Supercomputers with Multi-GPU Compute Nodes. In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. 305–313. doi:10.1109/DSN48987.2021.00043

[22] JunWeon Yoon, TaeYoung Hong, ChanYeol Park, Seo-Young Noh, and HeonChang Yu. 2020. Log analysis-based resource and execution time improvement in HPC: a case study. *Applied Sciences* 10, 7 (2020), 2634.

[23] Ziming Zheng, Zhiling Lan, Byung-Hoon Park, and Al Geist. 2009. System log pre-processing to improve failure prediction. In *Proceedings of the 2009 IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2009, Estoril, Lisbon, Portugal, June 29 - July 2, 2009*. IEEE Computer Society, 572–577.

[24] Ziming Zheng, Li Yu, Wei Tang, Zhiling Lan, Rinku Gupta, Narayan Desai, Susan Coghlan, and Daniel Buettner. 2011. Co-analysis of RAS Log and Job Log on Blue Gene/P. In *25th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2011, Anchorage, Alaska, USA, 16-20 May, 2011 - Conference Proceedings*. IEEE, 840–851.

[25] Xiaoyan Zhu, Jagadish Cherukuri, and Tao Yuan. 2016. Failure and maintenance analysis of supercomputers. In *2016 Annual Reliability and Maintainability Symposium (RAMS)*. 1–6. doi:10.1109/RAMS.2016.7448082