Examining Failures and Repairs on Supercomputers with Multi-GPU Compute Nodes

Amir Taherin[⋆], Tirthak Patel[⋆], Giorgis Georgakoudis[†], Ignacio Laguna[†], Devesh Tiwari[⋆]
[†]*Lawrence Livermore National Laboratory* [⋆]*Northeastern University*

***Abstract*—Understanding the reliability characteristics of supercomputers has been a key focus of the HPC and dependability communities. However, there is no current study that analyzes both the failure and recovery characteristics over multiple generations of a GPU-based supercomputer with multiple GPUs on the same node. This paper bridges that gap and reveals surprising insights based on monitoring and analyzing the failures and repairs on the Tsubame-2 and Tsubame-3 supercomputers.**

I. Introduction

HPC system reliability has been a major area of research for multiple decades. The primary driving factor has been the need to provide sustained reliability for long-running applications executing on multiple nodes. This line of research has resulted in making CPUs more reliable over time, and now GPUs too [1]–[6] – as they have become mainstream for supercomputing. While there have been multiple field studies about GPU and CPU errors [7]–[11], they are largely focused on a single production-scale supercomputer. There is no existing study that shares the experience and lessons learned from GPU-accelerated supercomputers over multiple generations. Furthermore, previous studies on GPU-accelerated supercomputers have included only one GPU per node and are limited to NVIDIA K80 or older GPUs [8]–[11]. In this study, we study two generations of Tsubame supercomputers (employing NVIDIA K20X and P100 GPUs); and importantly, each node has multiple GPU cards, which results in previously unobserved failure characteristics and creates opportunities for further innovation [8]–[14].

Additionally, this study also highlights the need for optimizing the time to recovery from failure – an aspect that has not received sufficient discussion and attention from previous field-studies. But, we show that the time to recovery is now becoming an important concern and figure of metric for system operations. Innovative solutions are needed to reduce the time to recovery, and in turn minimize the impact of failures on system operations. Overall, our major findings and implications include:

- As expected, GPUs are one of the most critical components in these GPU-accelerated supercomputers from the reliability point of view. Contrary to other GPU deployments [10], [11], [15], we find that the hardware reliability of NVIDIA GPUs has improved remarkably over the generations (up to $4\times$ improvement in overall system MTBF). But, GPU-related software and firmware failures (e.g., GPU driver issues) are still a concern and could benefit from further research investment from outside the GPU vendor/chip manufacturer. We also introduce a new term *"performance-error-proportionality"* to encourage systems community to jointly capture the effects of raw computing power and failure rate for benchmarking: "useful work done per failure-free period" (e.g., total FLOP per MTBF).

- We found that software failures are becoming the dominant failure type on these supercomputers. Alarmingly, the cause or type of a large fraction of these software failures is not known and are difficult to be reproduced.

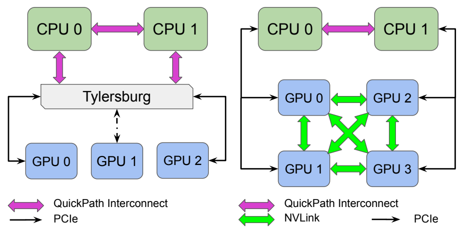
- As we move toward multi-accelerator-per-node supercomputers, our analysis reveals that system operators need to be wary of multiple GPUs failing simultaneously, and the failure distribution within a node being non-uniform and temporally correlated.

- While the mean time between two failures has improved drastically over the generations, we find that the mean time to recovery remains largely similar, i.e., the time to quickly heal from a failure is not improving at all. Each failure disrupts the system for roughly the same amount of time. Our failure type and seasonal analysis shows that the time to recovery trends vary across failure types and are not necessarily strongly correlated by the failure density in a particular time frame.

Our analysis tool and failure logs are available open-source at: http://doi.org/10.5281/zenodo.4606221.

II. Tsubame Supercomputer Background and Analysis Methodology

Tsubame is a supercomputer-class series of large-scale computing facilities housed at the Global Scientific Information and Computing Center (GSIC) at Tokyo Institute of Technology. Tsubame-1 was announced in 2006 as the then most powerful supercomputer in Japan. Tsubame-1 leveraged specific accelerators from ClearSpeed. Tsubame-2 was introduced in 2010 with 1408 nodes reaching theoretical peak (Rpeak) of 2.3 PFlop/s and power consumption of 1.4 MWatts. In 2017, Tsubame-3 was announced for Artificial Intelligence applications. It reached a theoretical peak (Rpeak) of 12.1 PFlop/s with power consumption of 792 kW. In terms of



(a) Tsubame-2 **(b) Tsubame-3**

Fig. 1. Tsubame-2 and Tsubame-3 node architecture.

TABLE I. Tsubame-2 and Tsubame-3 node configurations.

	Tsubame-2	**Tsubame-3**
CPU	Intel Xeon X5670 (Westmere-EP, 2.93GHz)	Intel Xeon E5-2680 V4 (Broadwell-EP, 2.4GHz)
Cores/Threads per CPU	6 cores / 12 threads	14 cores / 28 threads
Num CPUs	2	2
Memory per Node	58GB	256GB
GPU	NVIDIA Tesla K20X (GK110)	NVIDIA Tesla P100 (NVlink-Optimized)
Num GPUs	3	4
SSD	120 GB	2TB
Interconnect	4X QDR InfiniBand - 2 ports	Intel Omni-Path HFI 100Gbps - 4 ports

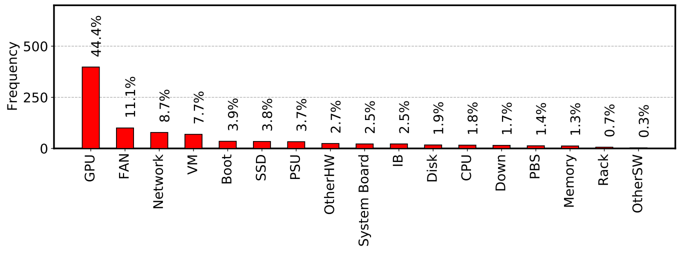
node structure, Tsubame-2 was designed with three GPUs per node, while Tsubame-3 has four GPUs per node (refer to Figure 1). Table I provides a high-level overview of the node specification of Tsubame-2, and Tsubame-3 [16].

Dataset. In this paper, we focus on the failures that are reported on Tsubame-2 and Tsubame-3. We used two failure logs from the Tsubame supercomputers: (i) Tsubame-2 failure log with 897 failures, and (ii) Tsubame-3 failure log with 338 failures. Tsubame-2 failure log includes the period between 1/7/2012, and 8/1/2013. The failure log on Tsubame-3 includes the period between 05/09/2017, and 02/22/2020. For each failure, the log includes the time of failure occurrence, the time to recovery from failure, and the category of failure. Table II lists the categories of failures reported in the logs. We focused on Tsubame-2 and Tsubame-3 with the goal of comparing our findings on two generations of supercomputers. In this work, we define a failure as an error that crashes the application (Table II provides a list of error types). These crashes can be fixed by rebooting or replacing the hardware, or updating the system software.

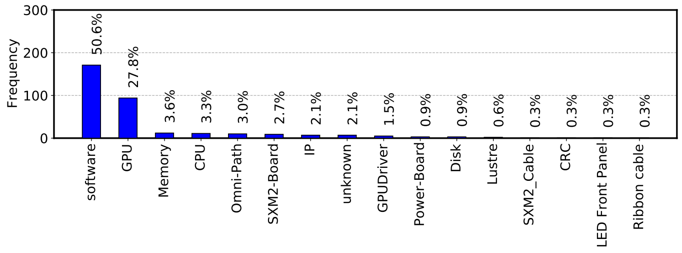
Limitations and Scope. We acknowledge that due to logging capability and business sensitivity of the study, this work has limitations. However, we ensure that these limitations are taken into account when drawing conclusions. For example, we do not focus on the root cause analysis for each failure because often there is a combination of root causes responsible for failures. Many times, not all of the contributing factors

TABLE II. Tsubame-2 and Tsubame-3 failure categories.

Tsubame-2	**Tsubame-3**
Boot, CPU, Disk, Down, FAN, GPU, (Infiniband) IB, Memory, Network, OtherHW, OtherSW, Portable Batch System (PBS), Power Supply Unit (PSU), Rack, SSD, System Board, and Virtual Machine (VM)	CPU, Cyclic Redundancy Check (CRC), Disk, GPU, GPUDriver, IP Motherboard (IP), Led Front Panel, Lustre, Memory, Omni-Path, Power-Board, Ribbon Cable, Software, SXM2_Cable, SXM2-Board, and Unknown



(a) Tsubame-2 Failure Categories



(b) Tsubame-3 Failure Categories

Fig. 2. GPU failures are the most frequent on Tsubame-2, while software failures are the most common on Tsubame-3.

are observable or detectable. The root cause for hardware failures (e.g., GPU, CPU, SSD) is often localized to the component itself. However, accurate determination of root cause for software-related failures is more challenging. The effects of particular applications, and the impact of environmental factors (e.g., temperature and humidity) is not discussed due to business-sensitivity and limited availability of the information (e.g., information not persisted for long-term due to storage overhead). In general, we did not find any particular application experiencing noticeably more failures than its proportional share of computational resource usage.

III. Investigating Failure Characteristics and Their Implications

We begin our analysis by performing a high-level examination of the characteristics of the failure categories and GPU failures on Tsubame-2 and Tsubame-3. In particular, we ask the following questions.

RQ1: What is the distribution of most frequently occurring failure types? And, are they the same on both systems? Figure 2 shows the breakdown of failures for each reported category on Tsubame-2 (2(a)), and Tsubame-3 (2(b)). These results reveal several interesting trends. First, a few failure types dominate on both the supercomputers (e.g., GPU, fan, network, software), but the dominant failure types are different on both the systems. Second, GPU failures are significantly higher in number than CPU failures on both the systems.

As Figure 2(a) shows, 44.37% of the failures are incident on the GPUs on Tsubame-2. In contrast, only 1.78% of the failures are caused by or happen on CPUs. Figure 2(b) shows that on Tsubame-3, ∼28% of the failures are categorized as GPU failures, while only 3.25% of the failures are CPU failures. Even though prior works have noted GPUs being one of the major factors for failures [8], [15], the magnitude of the difference in failure rate between GPUs and CPUs is concerning. The higher rate of GPU failures is a result of two phenomena: (1) Increasingly, applications are spending a considerable amount of their runtimes on GPUs compared to CPUs [17], [18], and (2) Unlike CPUs, GPUs lack sophisticated error and failure mitigation and correction techniques [19], [20]. There has been a lot of progress recently in terms of both (a) making the GPUs more resilient from an architecture and design point of view [3]–[5], and (b) developing software solutions (e.g., checkpointing) to mitigate the GPU errors [21]–[23]. However, there is a significant opportunity for academia to continue investing more effort in this area and develop better software failure mitigation methodologies.

One significant difference between Figures 2(a) and 2(b) is that on Tsubame-2, the GPU category of failures has the highest occurrence rate (44.37%); however, on Tsubame-3, the software category has the highest share of failure (50.59%), and GPU comes second (27.81%). This increased rate of software failures from Tsubame-2 to Tsubame-3 points to these failures being potentially caused by the introduction of new artificial intelligence (AI) and machine learning (ML) applications. We dig deeper to find out the cause of this behavior by breaking down the software errors into the 171 reported root loci. Figure 3 shows the top 16 causes of software failures. We observed an interesting trend: ∼43% of software failures are "GPU Driver-related Problems". This is a result of frequent GPU driver updates/upgrades, software-driver mismatch, and applications being run with incorrect CUDA versions. For example, on Tsubame-3, the OmniPath driver was associated with GPU software failures. Also, because NVIDIA supported InfiniBand before it added support for OmniPath, GPU Direct also caused problems. Fortunately, these failures generally occur at the beginning of an application run and do not result in wasted runtime.

Summary. Our analysis shows that while GPU hardware has matured over time, still ∼28% of failures are GPU hardware failures. Furthermore, higher usage of GPUs results in more software errors that are a result of the GPU software stack being not well-developed. While hardware improvements are on the rise and have been effective, they are expensive and not needed for all market sectors. There is an opportunity to develop more resilient accelerator software stack for HPC applications.

Figure 3 reveals further interesting insights. First, we observe that a significant fraction of software failures (approx.

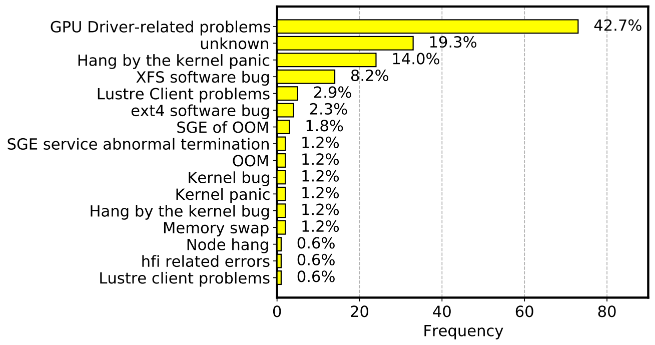
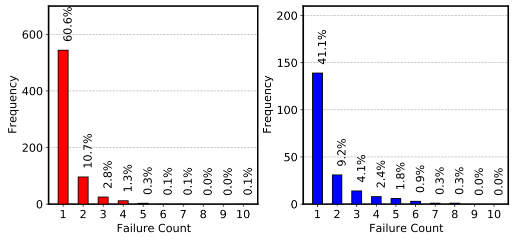


Fig. 3. Tsubame-3 software failures break down shows that most failure are GPU-driver-related.



(a) Tsubame-2 **(b) Tsubame-3**

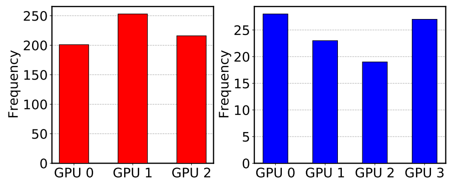
Fig. 4. On Tsubame-2, most nodes encounter only one failure. This is not the case for Tsubame-3.

20%) have no known cause and cannot be classified. This is an increasing problem and poses a significant challenge for operations where software failures cannot be diagnosed and the root-cause is not known. More academic effort is needed to identify non-reproducible bugs and their root causes. Second, this result also reveals that we need more effort in developing better mitigation techniques for GPU driver-related bugs. Finally, we note that, unlike previous works [9], [15], kernel panics and lustre bugs are relatively low – this is a testament to the years of efforts in making operating systems and lustre file systems hardened. Given some failure types being dominant, we ask whether certain nodes are affected more than others (i.e., they encounter more failures than others). More specifically:

RQ2: Are some nodes experiencing more failures than others on the Tsubame systems? If so, are these faulty nodes contributing to the majority of the failures on these HPC systems?

To answer these questions, we quantified the failure counts on each node to characterize how many failures an individual node has experienced. Figure 4 shows the results for Tsubame-2 (4(a)), and Tsubame-3 (4(b)). On Tsubame-2, ∼60% of the nodes experienced *only* one failure. Comparatively, on Tsubame-3, ∼60% of the nodes experienced more than one failure. More nodes experienced two or more failures on Tsubame-3 compared to Tsubame-2.

In terms of nodes with more than one failure, on both Tsubame systems, ∼10% of nodes experienced two failures. Nonetheless, the percentage of nodes that experienced three



(a) Tsubame-2 **(b) Tsubame-3**

Fig. 5. Different GPUs attached to a node experience different number of failures on both Tsubame systems.

failures on Tsubame-3 is ∼50% more than Tsubame-2. This is likely because each Tsubame-3 node has one additional GPU compared to each Tsubame-2 node. This shows that by increasing the number of GPUs per node in the system, the probability that a node experiences recurrent failures increases. Furthermore, considering nodes with more than 1 failure, on Tsubame-2, we observed 352 hardware failures and 1 software failure, and on Tsubame-3, we observed 104 hardware and 95 software failures. Thus, both hardware and software failures can occur multiple times on a node.

To investigate further, we inquire how the failures are spatially distributed within a node. Recall that each node has multiple GPUs and CPUs with different topologies on both supercomputers. Since GPU failures are more dominant, we focus on the GPU failures distribution within a single node (nomenclature of GPU 0, GPU 1, GPU 2, and GPU 3 is the same as shown in Fig. 1).

Figure 5 shows the failure distribution on GPUs. Based on Figure 5(a), GPU 1 has experienced ∼20% more failures than GPU 0 and GPU 2 per node on average. On Tsubame-3, GPU 0 and GPU 3 have experienced considerably more failures than GPU 1 and GPU 2 (Figure 5(b)). Therefore, we can conclude that the failure distributions among different GPUs are non-identical. While it has been difficult to pinpoint the exact reason for this behavior, several factors can be at play, including higher utilization of some GPUs as compared to others, manufacturing variability, and different distribution of hardware faults. An important implication of this finding is that HPC centers should inform and help end-users to take advantage of all the GPUs in a node in a load-balanced manner. Second, the operations staff could also mitigate this by rearranging the GPUs periodically during maintenance.

While different GPUs on the same node can have a different number of failures, can a GPU failure affect multiple GPUs on the same node? In particular, we ask:

RQ3: Can multiple GPUs within a node fail simultaneously? If so, what is the probability, and does that probability change across the two supercomputers? Table III shows that on Tsubame-2, in ∼30% of the failures, only one GPU was involved; however, in ∼70% of the failures more than one GPU was affected at the same time and needed action (Table III). On Tsubame-3 however, more than 92% of the failures only affected one GPU. In fact, no failure affected all four GPUs attached to a node. This is surprising

TABLE III. Number of GPUs involved in node failures.

#GPUs	Tsubame-3	Tsubame-2
1	75 (92.6%)	112 (30.44%)
2	4 (4.95%)	128 (34.78%)
3	2 (2.45%)	128 (34.78%)
4	0 (0%)	N/A
Total	**81 (100%)**	**368 (100%)**

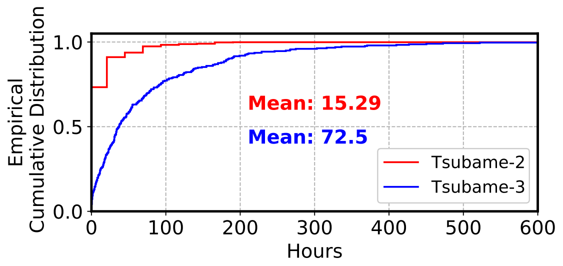
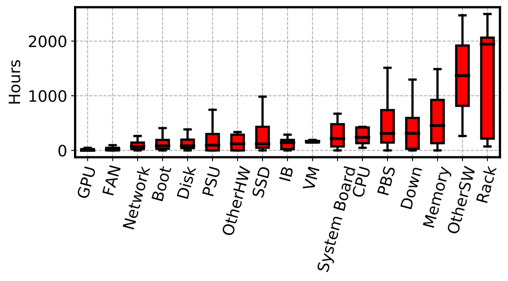


Fig. 6. Cumulative distribution of time between two failures. The mean time between failures (MTBF) is much higher on Tsubame-3 than Tsubame-2.

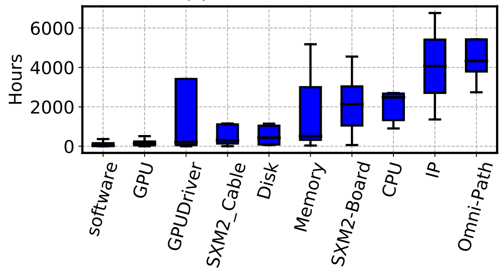
considering that more GPUs per node should lead to more multi-GPU failures. However, the counter-intuitive trend is a result of Tsubame-3 operational practices learned from the Tsubame-2 experience: more health-tests for multi-GPU cards on the same node and proactive replacements. Users have also become more informed and ensure that their multi-GPU jobs are debugged more rigorously to avoid the possibility of multiple GPUs failing simultaneously. This finding has an important implication for system administrators since the number of GPUs per node is likely to increase [24], [25]. The primary mode for simultaneous multi-GPU failures has been "fallen off the bus" errors, temperature-related failures, and simultaneous correlated reboots.

Summary. Our results reveal two novel insights: (1) the spatial distribution of GPU failures within a node is non-uniform for both the systems, and (2) each failure may affect multiple GPUs simultaneously on the same node. We recommend that HPC systems facilitate data collection on different failures involving GPUs for further investigation. An important implication is that HPC centers should inform and help end-users take advantage of all the GPUs in a node in a load-balanced manner, change the scheduler design when co-locating multiple jobs on the same node for increased utilization, and develop better testing for simultaneous multi-GPU failure mode.

Next, we analyze the temporal characteristics of failures. In particular, we investigate two key metrics: *time between two failures* (TBF), and *time to recovery* (TTR). Time between two failures simply refers to the elapsed wall clock time between two failure instances on the system. Time to recovery refers to the time taken to completely repair the failure and come back to the normal operational status (e.g., time taken to replace/restart a failed GPU).

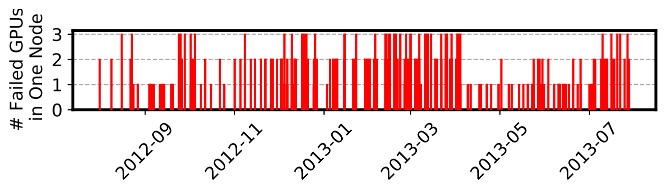


(a) Tsubame-2

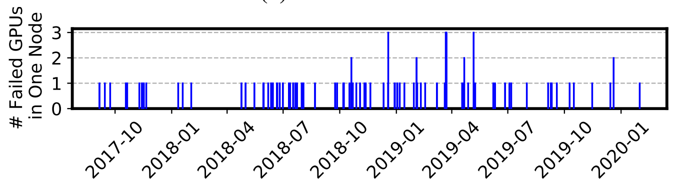


(b) Tsubame-3

Fig. 7. Distribution of the time between two failures for different failure types (sorted by mean time between two failures).



(a) Tsubame-2



(b) Tsubame-3

Fig. 8. Temporal distribution of GPU failures within node.

RQ4: How do the characteristics of the "time between two failures" change from one system to another and across different types of failures?

Figure 6 shows the distribution of the time between two failures for Tsubame-2 and Tsubame-3. We make two important observations. First, the distribution is significantly different for the two systems. Tsubame-2 has a steeper curve and Tsubame-3 has a longer tail. This indicates that there are longer failure-free periods on Tsubame-3. Such long failure-free periods are relatively fewer on Tsubame-2. In fact, 75% of the failures on Tsubame-2 occur within 20 hours of each other. In contrast, on Tsubame-3, 75% of the failures occur within 93 hours.

Second, the mean time between failures (MTBF) is much higher for Tsubame-3 than Tsubame-2. The MTBF on Tsubame-2 is $\sim$15 hours, but it is more than 70 hours on Tsubame-3 (more than 4$\times$ improvement). However, it is important to understand that the MTBF across two systems cannot be compared trivially. A meaningful comparison involves taking two factors into account: (1) the system's computing capability, and (2) its size.

Tsubame-3 has $\sim$8$\times$ more computing power than Tsubame-2. The MTBF, however, is $\sim$4$\times$ better. This shows that more useful work is done on Tsubame-3 than on Tsubame-2 even when interrupted by failures. As the performance of the systems increases, useful work done per failure should be used as a benchmarking metric for systems to account for reliability. We term this as *performance-error-proportionality* which can be expressed as maximum useful computation during failure-free period (e.g., total FLOP per MTBF).

With respect to the system size, Tsubame-3 has fewer number of nodes and hence, it could be argued that it is not surprising that Tsubame-3 has higher MTBF. The total number of CPU and GPU components in the system are: 7040 for Tsubame-2 and 3240 for Tsubame-3 (less than 2.5$\times$ difference). So the improvement in MTBF is not simply a side-effect of the reduced number of components. Furthermore, we calculated the MTBF for GPU and CPU related failures.

We estimated that the MTBF for GPU failures is 226.48 hours for Tsubame-3, but 21.94 hours for Tsubame-2. This relative increase in MTBF is $\sim$10$\times$, which is interesting because the number of GPUs has decreased by only 2$\times$. Similarly, the MTBF for CPU failures is 1593.6 hours for Tsubame-3, but 537.6 hours for Tsubame-2. CPU reliability has also increased ($\sim$3$\times$), but note that the number of CPUs also has decreased by $\sim$3$\times$.

Next, we dig deeper to understand the time between failures characteristics for different failure types. Figure 7 plots the distribution of time between two failures for different failure types. We make a few observations. First, as expected, not all failures have similar distribution of failure inter-arrival times on both the systems. Some failures have a lower median and spread (difference between the 75^{th} percentile and 25^{th} percentile) than others – and this is true for both the systems. For example, GPU-related hardware failures and software failures have the least median time between two failures. Second, memory- and CPU-related failures have a much higher median time between failures on both the systems and their relative spread is also higher compared to GPU failures.

Finally, Figure 8 shows the temporal distribution of GPU failures. This results reveals that failures that involved multiple GPUs failing within the same node often tend to happen close-by in time. That is, a failure where multiple GPUs within a node failed at the same time is likely to be followed by another such failure in close-by time. This is suspected due to interaction between application, GPU hardware, and operating conditions (e.g., temperature). An implication of this trend is how one can proactively schedule GPU nodes and provision for spare resources.

Summary. Our results reveal GPU hardware has become significantly more reliable over generations, and the corresponding increase in MTBF is more than the

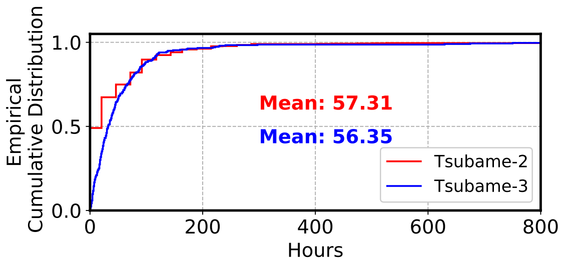
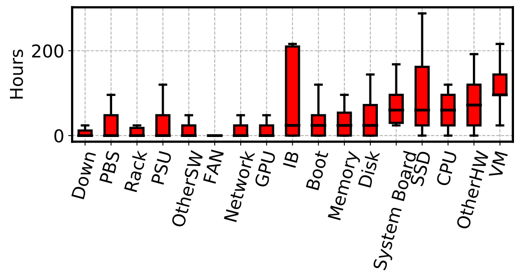
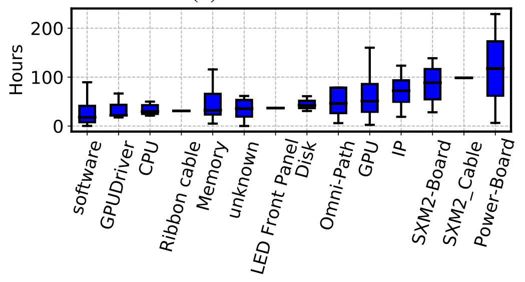


Fig. 9. Cumulative distribution of the time to recovery. The mean time to recovery (MTTR) is roughly the same for Tsubame-3 and Tsubame-2.



(a) Tsubame-2



(b) Tsubame-3

Fig. 10. Distribution of time to recovery for different failure types (sorted by mean time to recovery)

decrease in the number of components. However, we also observed that the resilience-proportionality does not scale at the same speed as the raw computing power. Hence, resilience-proportionality should be considered as a design factor to allow for the reliability of the system to increase at the same rate as the computing power.

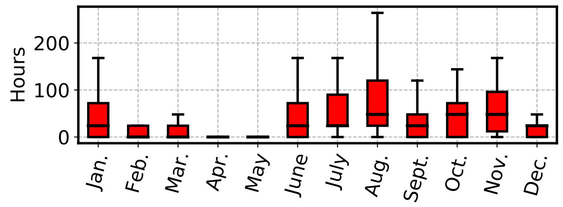
RQ5: How do the "time to recovery" characteristics change between two systems and across different failute types?

Figure 9 shows the distribution of the time to recovery from failures for both the systems: Tsubame-2 and Tsubame-3. Interestingly, the mean time to recovery (MTTR) is very similar (approx. 55 hours) for both systems. In fact, the distribution shape is very similar for both the systems. This is particularly interesting in the context that the MTBF and distribution of the time between failures is quite different for both the systems, as we observed earlier (Figure 6). When analyzed together, these trends have a number of important implications.

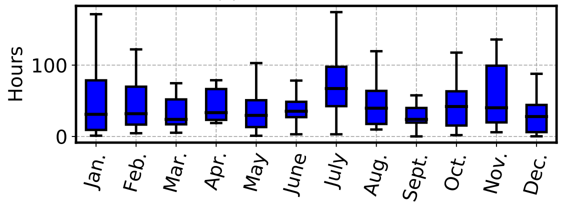
While the MTBF has improved significantly over the generations, the time to recovery has not. MTTR remains around 55 hours for both the system. An improvement in the time to recovery can be concluded if the distribution was much steeper for Tsubame-3. But, we observe that the distribution shape remains roughly the same. That means the strategies to improve repair time have not been as prevalent and effective as much as one would like.

In general, the time to recovery has not received as much attention in the academic literature as the MTBF and the efforts to reduce MTTR have not been as intense as reducing the MTBF [7], [9], [26]. However, our results show that the MTTR should receive similar attention for two reasons: (1) the MTTR is very comparable to MTBF and hence, it is likely that multiple concurrent failures might impact the handling/repair of previous failures. (2) the time to recovery directly quantifies the impact of the failure on the operations of the system - amount of time that a component is unavailable to the jobs.

Next, we investigate the time to recovery distribution for different failure types (Figure 10). We make a few important observations. As expected, the time to recovery distribution varies significantly across failure types and this is true for both systems. However, in general, hardware-related failures (GPUs, system board, power delivery failures) tend to have a higher spread in the recovery time compared to software failures. This is because hardware components have multiple failure modes, and diagnosing each failure type takes significant time. On the other hand, software failures typically require restarting, patching the software, etc. which have lesser time spread. Second, we observe that failure types with a lower average time to recovery do not necessarily have a lower spread. Some failure types which might be relatively infrequent can have a high time to recovery and a higher spread too. This finding implies that as system operators and designers, we should not look to focus only on highly frequent failures, but instead assess their impact on the system too. Less frequent failure types with high recovery costs can affect the system more negatively. For example, on Tsubame-3, the

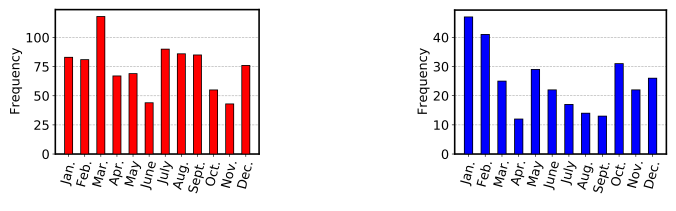


(a) Tsubame-2



(b) Tsubame-3

Fig. 11. Time to recovery distribution of Tsubame-2 and Tsubame-3 for different months.



(a) Tsubame-2 **(b) Tsubame-3**

Fig. 12. Distribution of failures based on month of occurrence.

"power board" category contributes to roughly 1% of the failures, however, its recovery can take up to 230 hours (i.e., ∼10 days). Similarly, on Tsubame-2, the "SSD" category is ∼4% of all failures, while recovering from some SSD failures requires ∼290 hours (i.e., 12 days). The longer recovery times highlight the need for appropriate spare provisioning of parts.

Finally, we inquire if the time to recovery has seasonal effects? That is, does the time to recovery become significantly worse during certain months (e.g., during the holiday season), or is increased when there is increased number of failures? To answer these questions, we plotted monthly time to recovery and number of failures (Fig. 11 and 12). We make two major observations. First, the time to recovery does not appear to have any clear seasonal impact. Although in the second half of the year, time to recovery seems to be higher – this is only true for Tsubame-2. For Tsubame-3, this trend is not true. In fact, there is significant variance in time to recovery during each month. We observed similar trends for different failure types as well, but results are not shown for brevity.

Second, one could hypothesize that the trends in time to recovery could be simply correlated with the failure density. That it, months with higher failure density are likely to see higher time to recovery. However, when Fig. 11 and 12 are analyzed together, we find that such a correlation does not exist. This is because, the cost of fixing each failure is different. Some failures may simply require rebooting and certain other failures require replacing the hardware. Hence, the cost of recovery is different and is not linear function of the number of failures. Also, these results highlight that the strategies to improve the time to recovery cannot be simply guided or triggered by seasonal impacts or failure density. Instead, lowering the time to recovery requires designing strategies that are specific to different types of failures and leveraging failure prediction to initiate recovery proactively where possible. Although not currently practiced, we believe design and deployment of such strategies would be operationally beneficial.

Summary. In summary, we need better strategies for reducing the time to recovery. These strategies need to be specific to each failure type and should be adaptive. Maintaining balance is the key. One can significantly reduce the MTTR by overly proactive measures such as keeping an excessive number of spare components on-site or more staff devoted to failure monitoring, but this comes at an increased operational cost. There is a need for innovative strategies that can initiate low-cost recovery actions based on failure prediction without much overhead.

IV. Related Work

In earlier sections, we discussed how our findings improve our current understanding compared to existing works. In this section, we discuss additional related work.

Failure Characterization and Analysis. Many of the prior works have focused on characterizing the fault tolerance and resiliency characteristics of data centers and supercomputers from the perspective of CPUs, GPUs, memory, interconnect network, and storage system [1], [8]–[15], [27]–[29]. For example, Gupta et al. [9] have characterized multiple HPC systems with different components, targeting their reliability. Most recently, Kumar et al. [7] analyzed the failures on multiple academic supercomputing clusters and used machine learning to predict resource usage.

Overview of Operational Practices. On the other hand, state-of-practice works have highlighted the high-level methods and approaches employed by large-scale systems to reduce failure rates and/or mitigate their effects [7], [30]–[33]. As an instance, faults can propagate and result in different failures across sub-systems. Pecchia et al. [30] propose a method to accurately classify different error entries in a failure log based on their causality relation to a known fault.

Generalizability and Usability to Other Systems. The findings of this study will become increasingly relevant as newer supercomputers are employing multiple GPUs on the same node (e.g., Summit, Sierra, and Juwels) and host multi-generational HPC systems (e.g., NASA supercomputing center, TACC, and Ohio State HPC center). We found that similar to single-GPU-per-node systems, the non-uniform distribution of failures among racks is also present in multi-GPU-per-node systems and can become particularly challenging. Our spatial and temporal distribution insights could be used to design proactive mitigation strategies (such as spare-provisioning, checkpointing, and scheduling [34]–[41]).

V. Concluding Remarks

We performed the characterization of system failures over two generations of GPU-dominated HPC systems, with a new focus on the time it takes to recover from a failure. Some of our novel findings include that software and GPU failures are the most frequent out of all failure types, the failure rates vary for different GPUs, and that the recovery time is not only failure-dependent but also varies monthly.

Acknowledgment We are thankful to GSIC, Tokyo Institute of Technology for providing the dataset. We would like to thank Adwait Jog (our shepherd) and anonymous reviewers for their constructive feedback. This work is supported by NSF Award 1910601 and 1753840, and prepared by LLNL under Contract DE-AC52-07NA27344 (LLNL-CONF-820342).The views and opinions of the authors do not necessarily reflect those of the U.S. government or Lawrence Livermore National Security, LLC neither of whom nor any of their employees make any endorsements, express or implied warranties or representations or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of the information contained herein.

References

[1] K. Lee, M. B. Sullivan, S. K. S. Hari, T. Tsai, S. W. Keckler, and M. Erez, "On the trend of resilience for gpu-dense systems," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks–Supplemental Volume (DSN-S)*. IEEE, 2019, pp. 29–34.

[2] P. Rech, L. Carro, N. Wang, T. Tsai, S. K. S. Hari, and S. W. Keckler, "Measuring the radiation reliability of sram structures in gpus designed for hpc," in *IEEE 10th Workshop on Silicon Errors in Logic-System Effects (SELSE)*, 2014.

[3] A. Mahmoud, S. K. S. Hari, M. B. Sullivan, T. Tsai, and S. W. Keckler, "Optimizing software-directed instruction replication for gpu error detection," in *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2018, pp. 842–853.

[4] K. Lee, "Resilient heterogeneous systems with containment domains," Ph.D. dissertation, 2020.

[5] L. Yang, B. Nie, A. Jog, and E. Smirni, "Practical resilience analysis of gpgpu applications in the presence of single-and multi-bit faults," *IEEE Transactions on Computers*, 2020.

[6] B. Nie, A. Jog, and E. Smirni, "Characterizing accuracy-aware resilience of gpgpu applications," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. IEEE, 2020, pp. 111–120.

[7] R. Kumar, S. Jha, A. Mahgoub, R. Kalyanam, S. Harrell, X. C. Song, Z. Kalbarczyk, W. Kramer, R. Iyer, and S. Bagchi, "The mystery of the failing jobs: Insights from operational data from two university-wide computing systems," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2020, pp. 158–171.

[8] D. Tiwari, S. Gupta, J. Rogers, D. Maxwell, P. Rech, S. Vazhkudai, D. Oliveira, D. Londo, N. DeBardeleben, P. Navaux, L. Carro, and A. Bland, "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)*, 2015, pp. 331–342.

[9] 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.

[10] 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 *SC'15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2015, pp. 1–12.

[11] G. Ostrouchov, D. Maxwell, R. Ashraf, C. Engelmann, M. Shankar, and J. Rogers, "Gpu lifetimes on titan supercomputer: Survival analysis and reliability," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC) 2020*, 2020, pp. 15–20.

[12] B. Nie, J. Xue, S. Gupta, T. Patel, C. Engelmann, E. Smirni, and D. Tiwari, "Machine learning models for gpu error prediction in a large scale hpc system," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 95–106.

[13] 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.

[14] L. B. Gomez, F. Cappello, L. Carro, N. DeBardeleben, B. Fang, S. Gurumurthi, K. Pattabiraman, P. Rech, and M. S. Reorda, "Gpgpus: How to combine high computational power with high reliability," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–9.

[15] 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*. IEEE, 2014, pp. 610–621.

[16] Tokyo Institute of Technology. Tsubame hardware-software specification. [Online]. Available: https://www.gsic.titech.ac.jp/en/node/420

[17] N. Onodera and Y. Idomura, "Acceleration of wind simulation using locally mesh-refined lattice boltzmann method on gpu-rich supercomputers," in *Asian Conference on Supercomputing Frontiers*. Springer, 2018, pp. 128–145.

[18] A. Nukada, K. Sato, and S. Matsuoka, "Scalable multi-gpu 3-d fft for tsubame 2.0 supercomputer," in *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. IEEE, 2012, pp. 1–10.

[19] K. Lee, M. B. Sullivan, S. K. S. Hari, T. Tsai, S. W. Keckler, and M. Erez, "Gpu snapshot: checkpoint offloading for gpu-dense systems," in *Proceedings of the ACM International Conference on Supercomputing*, 2019, pp. 171–183.

[20] B. Pourghassemi and A. Chandramowlishwaran, "cudacr: An in-kernel application-level checkpoint/restart scheme for cuda-enabled gpus," in *2017 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2017, pp. 725–732.

[21] A. R. Anwer, G. Li, K. Pattabiraman, M. Sullivan, T. Tsai, and S. Hari, "Gpu-trident: efficient modeling of error propagation in gpu programs," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020, pp. 1–15.

[22] R. Garg, A. Mohan, M. Sullivan, and G. Cooperman, "Crum: Checkpoint-restart support for cuda's unified memory," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 302–313.

[23] R. Garg, G. Price, and G. Cooperman, "Mana for mpi: Mpi-agnostic network-agnostic transparent checkpointing," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, 2019, pp. 49–60.

[24] J. Wells, B. Bland, J. Nichols, J. Hack, F. Foertter, G. Hagen, T. Maier, M. Ashfaq, B. Messer, and S. Parete-Koon, "Announcing supercomputer summit," Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States), Tech. Rep., 2016.

[25] J. A. Kahle, J. Moreno, and D. Dreps, "2.1 summit and sierra: Designing ai/hpc supercomputers," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2019, pp. 42–43.

[26] D. Das, M. Schiewe, E. Brighton, M. Fuller, T. Cerny, M. Bures, K. Frajtak, D. Shin, and P. Tisnovsky, "Failure prediction by utilizing log analysis: A systematic mapping study," in *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, 2020, pp. 188–195.

[27] M. Kumar, S. Gupta, T. Patel, M. Wilder, W. Shi, S. Fu, C. Engelmann, and D. Tiwari, "Understanding and analyzing interconnect errors and network congestion on a large scale hpc system," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 107–114.

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

[29] S. Di, H. Guo, E. Pershey, M. Snir, and F. Cappello, "Characterizing and understanding hpc job failures over the 2k-day life of ibm bluegene/q system," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2019, pp. 473–484.

[30] A. Pecchia, D. Cotroneo, Z. Kalbarczyk, and R. K. Iyer, "Improving log-based field failure data analysis of multi-node computing systems," in *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*. IEEE, 2011, pp. 97–108.

[31] G. Li, K. Pattabiraman, S. K. S. Hari, M. Sullivan, and T. Tsai, "Modeling soft-error propagation in programs," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 27–38.

[32] E. Tremel, S. Jha, W. Song, D. Chu, and K. Birman, "Reliable, efficient recovery for complex services with replicated subsystems," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2020, pp. 172–183.

[33] V. Fratin, D. Oliveira, C. Lunardi, F. Santos, G. Rodrigues, and P. Rech, "Code-dependent and architecture-dependent reliability behaviors," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 13–26.

[34] D. Tiwari, S. Gupta, and S. S. Vazhkudai, "Lazy checkpointing: Exploiting temporal locality in failures to mitigate checkpointing overheads on extreme-scale systems," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 25–36.

[35] 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.

[36] L. Bautista-Gomez, A. Gainaru, S. Perarnau, D. Tiwari, S. Gupta, C. Engelmann, F. Cappello, and M. Snir, "Reducing waste in extreme scale systems through introspective analysis," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2016, pp. 212–221.

[37] Q. Liu, C. Jung, D. Lee, and D. Tiwari, "Compiler-directed lightweight checkpointing for fine-grained guaranteed soft error recovery," in *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2016, pp. 228–239.

[38] K. Tang, D. Tiwari, S. Gupta, P. Huang, Q. Lu, C. Engelmann, and X. He, "Power-capping aware checkpointing: On the interplay among power-capping, temperature, reliability, performance, and energy," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2016, pp. 311–322.

[39] R. Garg, T. Patel, G. Cooperman, and D. Tiwari, "Shiraz: Exploiting system reliability and application resilience characteristics to improve large scale system throughput," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 83–94.

[40] R. Basu Roy, T. Patel, R. Kettimuthu, P. Richa, A. Scovel, B. Allcock, and D. Tiwari, "Operating liquid-cooled large-scale systems: Long-term monitoring, reliability analysis, and efficiency measures," in *2021 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2021.

[41] L. Wan, F. Wang, S. Oral, D. Tiwari, S. S. Vazhkudai, and Q. Cao, "A practical approach to reconciling availability, performance, and capacity in provisioning extreme-scale storage systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2015, pp. 1–12.