Failures in Large Scale Systems: Long-term Measurement, Analysis, and Implications[*]

Saurabh Gupta
Intel Labs

Tirthak Patel
Northeastern University

Christian Engelmann
Oak Ridge National Laboratory

Devesh Tiwari
Northeastern University

ABSTRACT

Resilience is one of the key challenges in maintaining high efficiency of future extreme scale supercomputers. Researchers and system practitioners rely on field-data studies to understand reliability characteristics and plan for future HPC systems. In this work, we compare and contrast the reliability characteristics of multiple large-scale HPC production systems. Our study covers more than one billion compute node hours across five different systems over a period of 8 years. We confirm previous findings which continue to be valid, discover new findings, and discuss their implications.

ACM Reference format:
Saurabh Gupta, Tirthak Patel, Christian Engelmann, and Devesh Tiwari. 2017. Failures in Large Scale Systems: Long-term Measurement, Analysis, and Implications. In *Proceedings of SC17, Denver, CO, USA, November 12–17, 2017,* 13 pages.
DOI: 10.1145/3126908.3126937

1 INTRODUCTION

Maintaining high system reliability continues to be an essential aspect of large scale HPC computing facilities. Improved system reliability leads to more productive scientific computing and hence, faster scientific discovery. Unfortunately, improving and maintaining a high level of system reliability is quite challenging for multiple reasons. First, the number of components are increasing rapidly in large scale HPC systems to meet the compute power demands from the computational scientists, and hence, the likelihood of failures is also increasing [14]. Second, due to shrinking process technology, processors have become more susceptible to soft-errors, process variation related errors, and manufacturing defect [1, 10]. Third, as the complexity of the system grows, managing system reliability becomes more difficult [22, 26]. It becomes challenging to proportionally increase the number of studies that collect, analyze and share long-term infrastructural experiences providing comprehensive quantification of failures characteristics.

As an example, one of the highly cited and most comprehensive study that shared failure related characteristics of large HPC systems was published by academic researchers almost a decade ago [36]. Systems covered in that study spanned between 1996 and 2005. Since then, there have been a sequence of studies documenting the failure characteristics of large scale HPC systems [8, 11, 17, 24, 25, 29]. However, most of them either focus on particular component of the system such as disks/memory [3, 20, 27, 35, 37, 40], or they focus on one particular system for a short period of time [7, 11, 17, 24, 28]. Despite these efforts, the study by Schroeder et al. [36] continues to be the only work to analyze multiple HPC systems that spans over a long period of time. However, researchers and system practitioners rely on field-data studies to improve their understanding about the reliability characteristics of HPC systems and accordingly plan for better future systems.

To address this challenge, this work presents lessons learned from characterizing and analyzing multiple large scale production systems with very different system component composition and characteristics, spanning from 2008 to 2015. In particular, we analyze the system failure data from five different HPC systems deployed at the Oak Ridge National Laboratory (ORNL). These systems include Jaguar XT4, Jaguar XT5, Jaguar XK6, Titan, and Eos where the peak computational power alone varies from approximately 0.27 Petaflops to 27 Petaflops (almost 100x improvement). Overall, our study covers more than one billion compute node hours across five different systems across generations.

We share characterization, analysis, and lessons learned from deep investigation of multiple HPC systems. Our study reveals several interesting insights about large scale system reliability. Our work discusses new take-aways and confirm previous findings which continue to be valid. In particular, we investigate if the newer HPC systems have become dramatically less reliable as projected by technology roadmaps and other studies [10, 14, 26, 38]. We investigate how the failure characteristics change over time during the stable operational period of a HPC system. We quantify and characterize multiple metrics to capture the failure characteristics of a HPC system, and show how these can be used to discover interesting characteristics of different failure types and systems. We discuss the significance and implications of our findings and how they can be used by researchers, system acquisition teams, and system operators to potentially improve the overall efficiency of current HPC systems, and better provisioning and application software resiliency strategies for future HPC systems.

[*]This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the United States government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.
SC17, Denver, CO, USA
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM. 978-1-4503-5114-0/17/11…$15.00
DOI: 10.1145/3126908.3126937

Table 1: HPC Systems analyzed in this study.

System	Number of Nodes	Period
Jaguar XT4 (31328 cores, quad-core AMD Opteron processor per node, SeaStar2)	7,832	Jan'08-Mar'11
Jaguar XT5 (149504 cores, four dual-core AMD Opteron processor per node, SeaStar2+)	18,688	Jan'09-Dec'11
Jaguar XK6 (298,592 cores, 16-core Opteron-6274 processor per node, Gemini)	18,688	Jan'12-Oct'12
Eos XC 30 (23,553 cores, 2 sockets of 16-core Intel Xeon E5-2670 (with hyperthreading) per node, Aries)	736	Sep'13-Sep'15
Titan XK7 (560,640 cores, one 16-core Opteron-6274 and one K20x Nvidia GPU per node, Gemini)	18,688	May'13-Sep'15

Table 2: List of failure types observed on the systems in this study.

Failure Event	Type	Component Affected
Bad Page State	Software	–
Blade Heartbeat Fault	Hardware	Module
Core Hang	Hardware	Node/CPU
GPU Double Bit Error (DBE)	Hardware	GPU
HT Lockup	Hardware	CPU
Kernel Panic	Software	OS
L0 Heartbeat Fault	Hardware	Module
Lustre Bug (LBUG)	Software	File System
Lustre Server Failure	Software	File System
Machine Check Exception (MCE)	Hardware	CPU/Memory
Module Emergency Power Off (EPO)	Hardware	Module
Module Failed	Hardware	Module
Node Heartbeat Fault	Hardware	Module/Node
PCI Width Degrade	Hardware	GPU
RX message CRC error	Hardware	Interconnect
RX message header CRC error	Hardware	Interconnect
SCSI Error	Hardware	–
SeaStar Heartbeat Fault	Hardware	Interconnect
Seastar Lockup	Hardware	Interconnect
SXM Power Off	Hardware	GPU
VERTY Fault	Hardware	Module
Voltage Fault	Hardware	Module
WarnTemp Power Off	Hardware	CPU

2 BACKGROUND AND METHODOLOGY

This section first describes the high performance computing systems analyzed in this study. Then, we describe our data measurement and analysis methodology. The section concludes by describing the scope and potential limitations of this study.

Systems This study analyzes system failure data from five different production HPC systems. These systems are Jaguar XT4 supercomputer, Jaguar XT5 supercomputer, Jaguar XK6 supercomputer, Eos XC30 system, and Titan XK7 supercomputer (Table 1). Our study covers over a billion node-hours across five different production HPC systems.

Measurement The syslog data is collected continuously to extract record of failures, problems, and potential issues. This data is also used to report problems to vendors and build a knowledge base for system administration. The data measured and collected for this study using different instrumentation and logging methods, is from January 2008 to September 2015. In total, this is 11.26 system years or 1.22 Billion node hours of operational data. The data collected from these HPC systems is regularly parsed on software management workstations to alert the system operators of any faulty behavior and system failure events. Instrumentation points are already available in the hardware and system software which are turned on to collect critical information such that it does not cause significant interference with the operations of the machine and workloads running on the machine. For our analysis, we converted the data into a time series containing time-stamp, type of event, physical location of event, and other relevant information. This processing step is carefully designed, executed, and verified with sys-admin staff. Table 2 lists all the failure types across different systems. The table also shows the primary nature of the failure (software/hardware) and key component affected by the event.

Our data processing step revealed that some failure events are recorded multiple times in the system logs because of multiple locations reporting the side-effects of the actual failure event. We confirmed from the system operators, who have extensive experience in system monitoring and diagnostics, to carefully drop such events from our analysis to avoid side-effects on the validity of our findings. We only consider the actual parent events in our analysis and exclude these potential follow-up failure events that occur within a 300 second window after the actual failure. We chose a 300 second time period conservatively to avoid any bias in analyzing event correlation. Other works such as [29] have also adopted such filtering methods. We also drop failures from acceptance and early user tests (early part of the bath tub curve) to avoid skew in the analysis. We note that only statistically significant results are presented in this paper.

Scope and Limitations Despite the wealth of data analyzed in this study to gather insights into the systems and comparing several generation of systems, we recognize that, like any other work, it has its limitations and scope.

This study is based on a post-hoc analysis, and therefore, it is not possible to answer 'what if' type questions that requires modifications to the system and observe the effect of such modification. We assessed that performing accurate and high-coverage root-cause analysis of failures is not possible given the complexity and granularity of data measurement, collection, and dynamic operational environment. Therefore, root-cause analysis is not the goal of this study. We also note that a large-scale computing facility has a dynamic environment which is subject to frequent system software updates. In this study, we do not attempt to analyze, model, or isolate the effect of those changes. Our study ensures that major disruptions caused by a particular maintenance phase or software updates doesn't skew our findings. As we draw conclusions from the measured data, we carefully consider and point out the methodological pitfalls or hidden factors that can affect the analysis, e.g., dropping failures from acceptance and early user tests (early part of the bath tub curve) to avoid skew in the analysis. We recognize that human-in-the-loop can also affect our analysis. We noticed that it especially impacts the repair time in certain cases. It was not possible to perform an accurate and robust analysis to account for this effect and its impact on the repair time. Hence, we decided to the exclude repair-time analysis.

Another limitation of our study may appear to be that all the systems studied here are Cray systems from ORNL. Note that there are various system components that are actually not Cray-specific such as various generations of AMD/Intel CPUs, Nvidia GPUs, and DIMM modules. The interconnect and maintenance styles are Cray specific. Knowledge of maintenance-and-testing style at ORNL has made the failure analysis more accurate. It is often not possible to obtain data of similar granularity and detail from other facilities to perform complete analysis with high level of confidence. Including analysis on partial data may not necessarily increase the confidence level of our findings and the methodology becomes inevitably less robust.

We also note that Oak Ridge Leadership Computing facility is a typical large-scale data center with standard operational practices in place. For example, the temperature and humidity sensors are placed throughout the data center room for monitoring. Temperature measurements are collected at the cage level too. Air-flow is monitored carefully to avoid hotspots. Four transformers are employed on-site with 480V/100A and 480V/20A circuits. Power-cooling is maintained vi standard chiller and cooling tower set-up, resulting in the overall PUE of 1.24. This study for several generation of Cray systems at ORNL provides a unique opportunity to understand the evolution of large scale HPC systems. While ORNL specific knowledge has made the failure analysis more accurate, we also recognize that findings can be sometimes specific to this facility or systems considered in this study.

We also discuss cases where similar observations were confirmed at other facilities. Overall, we believe that this work will enable others to learn from these lessons, apply new metrics, and verify/refute the findings presented here.

3 UNDERSTANDING AND ANALYZING SYSTEM FAILURE CHARACTERISTICS

The goal of this section is to understand and analyze the different characteristics of system failure events over a long period of time on five large-scale HPC production systems: Jaguar XT4, Jaguar XT5, Jaguar XK6, Eos XC30, and Titan XK7. We derive insights and lessons learned for researchers and practitioners by analyzing how failure characteristics change over time and are affected by different failure types.

Multiple studies have found and project that the newer HPC systems are typically expected to be much less reliable due to increasing complexity of the system design, decreasing device dependability, shrinking process technology [10, 14]. Therefore, we investigate if this is true for the data measured from the HPC systems considered in this study. Furthermore, multiple researchers have shown that the failure rates remain fairly stable during the stable operational period (typically referred to as the middle part of the bath tub curve) [11, 13]. Therefore, we investigate if the failure rate of the system changes during the stable period.

Table 3: Overall MTBF for different systems. Scale-Normalized MTBF is calculated assuming same system contains 18688 nodes.

System	MTBF (hours)	Scale-Norm. MTBF (hours)
Jaguar XT4	36.91	15.47
Jaguar XT5	22.67	22.67
Jaguar XK6	8.93	8.93
Eos	189.04	7.45
Titan	14.51	14.51

RQ1: *Are newer generations of HPC systems becoming less reliable?*
RQ2: *During the stable operational period, does the reliability of the system change significantly? If so, by how much?*

We use the mean time between failure (MTBF) as the first simple metric to study failure characteristic. Here, failure is defined as an hardware or system-software related error event that causes an application to crash (application related bugs that cause an application to fail are not counted toward MTBF calculation). Application MTBF caused due to system related problems has been a commonly used metric to represent how often an application is expected to experience a failure on average, due to system related errors. It is simple to measure and express. Therefore, we start our analysis by comparing the MTBF across systems in our study.

Table 3 shows the MTBF of each system and a scale-normalized MTBF metric. As shown in Table 1, all systems in our study are not of the same scale (in terms of number of nodes), therefore, simply comparing the MTBF is not a fair comparison across systems with different number of nodes, as a system with more number of nodes is more likely to experience higher number failures if everything else remains the same. Therefore, the scale-normalized MTBF metric is presented to compare MTBF as if all systems were deployed with the same number of compute nodes, as defined below. We also note that scale-normalized MTBF does not capture the non-linear increase in complexity of the system as the system size grows, or account for difference in number of components per node. This increased complexity and varying number of components per node may impact the MTBF. However, there is no robust methodology to capture this non-linear increase in complexity and its effect on MTBF. Therefore, we use scale-normalized MTBF to understand first-order trends.

$$\text{Scale-Normalized MTBF} = \frac{\text{MTBF}\times\text{Num of Nodes in the System}}{\text{Max Number of Nodes across all Systems}}$$

From Table 1, we note that Jaguar XT5 has the highest scaled MTBF, followed by the Jaguar XT4 and Titan XK7 systems. Jaguar XT4 and Jaguar XT5 are two consecutive generations of Cray systems that shared several design features. Similarly, Jaguar XK6 and Titan XK7 are also two consecutive generations of Cray systems. We found that it is possible that newer generation of systems may have higher scale-normalized MTBF than previous generation of systems. While one metric may not always capture the full reliability characteristics of a system as we discuss later, we observe the reliability doesn't necessarily decrease monotonically over different generations of the HPC systems, as projected by previous studies [14, 38]. Next, we show that comparison across systems based on scale-normalized MTBF averaged over the whole time may lead to incomplete and inaccurate characterization. Fig. 1 shows how the scale-normalized MTBF of the system changes over time. The plot shows the scale-normalized MTBF metric averaged over each quarter. We point out that we experimented with different granularities (e.g., week, month, quarter) and were able to ensure statistical significance for comparison with quarter granularity.

Fig. 1 quantitatively shows that the scale-normalized MTBF changes drastically even during the stable operational periods of the systems. For example, Jaguar XT5 systems shows approximately 4x change in scale-normalized MTBF and Titan system shows approximately 2.5x change in scale-normalized MTBF over time. We also observe that during some time periods newer generation of systems have higher scale-normalized MTBF, while during some time periods previous generation of systems are more reliable – indicating that there is not necessarily a monotonic trend at the system level as projected by technology trends and other studies [11, 13]. We also confirmed that these changes in MTBF are not due to software upgrades. As discussed later in detail, different failure types also exhibit this behavior. Also, as per our definition of MTBF, a system outage still counts as one failure and hence, does not account for variance in MTBF.

Summary Our field study shows that the reliability of HPC systems doesn't necessarily decrease monotonically over different generations of the HPC systems. Even during the stable operational period the MTBF may change by up to 4x, contrary to findings from prior studies showing that MTBF of HPC system during stable operational period doesn't vary significantly [11, 37, 39].

As optimal checkpointing intervals employed by applications depend on the MTBF [5, 15], this information should be exposed to the HPC users easily and systematically to reduce the impact of failures (i.e., wasted work). This finding can be used by other HPC centers to guide users to make better checkpointing decisions.

The variance in MTBF can be caused by the changing temporal and spatial characteristics of failure events. These characteristics and most frequent failure types are discussed later in the paper. Given the significant variance in system reliability, HPC system acquisition teams at HPC centers can use this finding by adding upper bound on the variance in MTBF as a key metric in the request for proposals and contracts. This will attempt to ensure that system manufacturers and integrators have additional responsibility and support available should the reliability drops below a certain threshold, as opposed to only system administrators trying to improve the user experience during such period. For example, if the MTBF drops dramatically below a certain threshold due to hardware issue, the spare parts required beyond the threshold could be cost-shared by the vendor. We also note that such ideas need to be implemented in ways where both parties always have rewards and risks associated with abnormal operational efficiency phases.

Next natural question to ask is: what types of failures dominate the overall number of failures and what can we learn from their temporal characteristics?

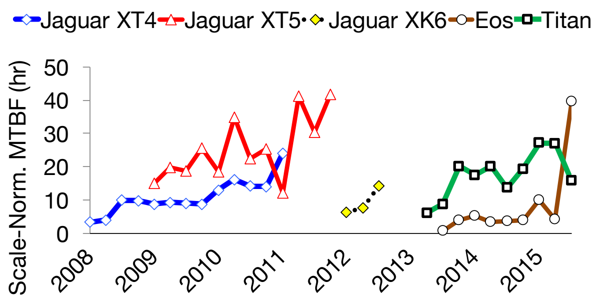


Figure 1: Scale-normalized MTBF of each system over time (averaged quarterly).

RQ3: *One may expect that few failure types dominate [11, 17], but how much do these dominant failures change over time and across systems?*

Fig. 2 shows the fraction of each failure type with respect to the total number of failures on the system.

First, we find that a small subset of failure types constitute a major fraction of total failures across many systems. Three failure types: Machine Check Exception (MCE), Kernel Panic, and Node Heartbeat fault combined together constitute more than 70% of the total failures on all three Jaguar systems (refer to Table 2 for failure type: hardware / software). On Titan, the GPU related errors (SXM power off and double bit errors) contribute roughly 30% of all failures. Significant fraction of hardware related errors, such as MCE and GPU related errors, emphasizes the importance of better provisioning and replication of CPU and GPU memory. For example, GPU related errors and MCEs constitute more than 60% of failures on Titan.

We note that major system software related bug such as Kernel Panic and Lustre Bugs contribute approximately 20%. We notice that SeaStar errors (interconnect related errors) interrupted applications on Jaguar XT5 system approximately 10% of the total failure instances. In comparison, interconnect related errors are less than 1% on new systems with Gemini and Aries interconnects.

We point out that we have carefully counted Node Heartbeat fault as system failure only when these events have resulted in application interruptions and not as false alarms. X86 CPUs log errors are detected by the CPU as MCEs. MCEs can be uncorrectable errors in the CPU caches, in main memory, in the front side bus or CPU interconnect, etc. Potential causes include cosmic radiation, voltage/power fluctuations, process variation, among other reasons.

It is natural to hypothesize that a dominant particular failure type could be a result of all failures of that particular type occurring in a short period of time. To refute this, Fig. 3 shows how the fraction of m failure types changes over time for Jaguar XT5 system (we observed similar trends for other systems). We show that dominant failure types occur throughout the period instead of being concentrated over shorter periods. We also note that their contribution toward number of failures changes over time.

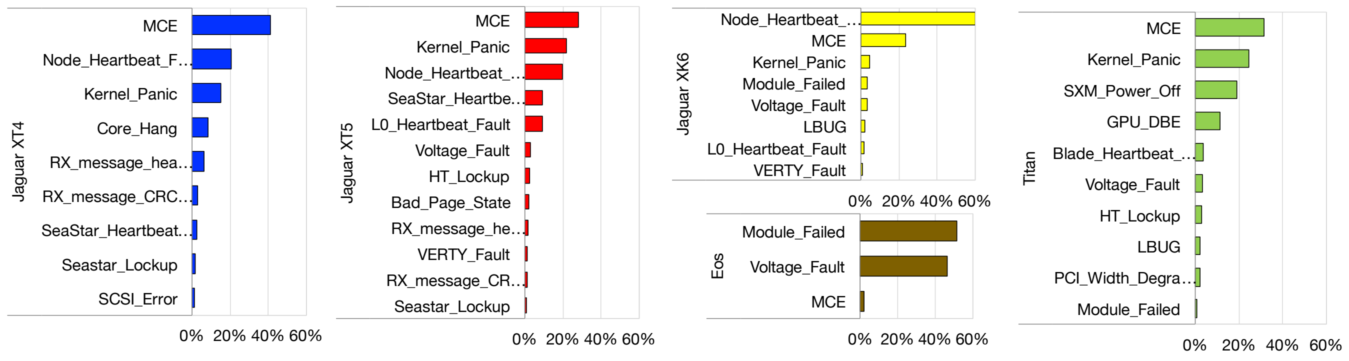


Figure 2: Contribution of different failure types for each system.

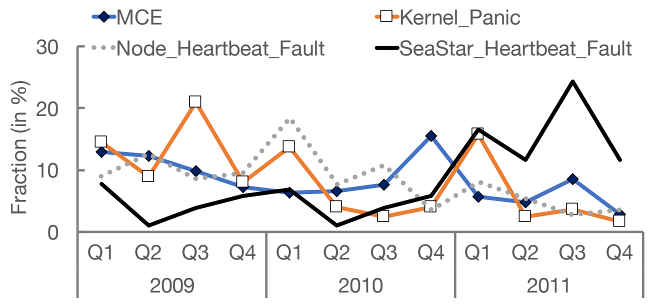


Figure 3: Contribution of key failure types Machine Check Exception(MCE), Kernel Panic, Node heartbeat fault, and Seastar heartbeat fault over time (averaged quarterly).

Summary Our study confirms that a few failure types constitute a major fraction of all failures.

However, unlike prior studies indicating that software failures are dominant failure types (e.g., file system and kernel related failures) [8, 11], we found that hardware related errors (e.g., uncorrectable memory errors) are equally or more dominant across systems over the whole period of time – implicating the importance of better provisioning and replication of CPU and GPU memory against such errors via compression, replication etc. [9, 23]. We also observed that resilience of Cray interconnect has improved significantly from older generation to newer generation.

Given the significant variance in MTBF among different failure types, HPC system acquisition teams should also consider adding MTBF bound for different failure types as a key metric in the request for proposals and contracts.

As noted by previous work [15, 16, 34], exponential distribution is not the best fitting parametric model for failure recurrence behavior; instead Weibull distribution has been shown to better fit the field data from real systems. Note that this has important implication on hazard rate of the system. Exponential distribution with mean λ has a constant hazard rate of $1/\lambda$. On the other hand, a Weibull distribution with parameters (λ, k) is given as Eq.1. Since this rate is not constant and decreases with the value of x (k is less than 1 for Weibull distribution), the failure rate is higher when the value of x is smaller. In other words, failures are more likely to re-occur close to failure events. Therefore, we propose to use k as the parameter to represent the degree of temporal recurrence behavior. The closer k is to 1 (i.e., closer to exponential distribution), the lower is the degree of recurrence. Moreover, k is complementary to using MTBF because MTBF and k can uniquely determine the parameters (λ, k) of the fitting Weibull distribution.

$$h(x;k,\lambda) = \frac{k}{\lambda}(\frac{x}{\lambda})^{k-1} \tag{1}$$

RQ4: *Prior studies have shown that system failures have temporal recurrence property [4, 15, 16, 28]. But, does this temporal recurrence property of system failures change across systems and different failure types?*

To the best of our knowledge, we are the first to investigate if the degree of temporal recurrence varies over time across different systems. Fig. 5 shows the temporal recurrence parameter for Jaguar XT4, Jaguar XT5, Jaguar XK6, and Titan XK7 systems at a quarterly granularity. Eos XC30 system is not included in this analysis because we found that the number of failure events were not enough to pass the test of statistical significance.

We observe that the degree of temporal recurrence changes significantly over time and across systems. For example, the temporal recurrence parameter for Jaguar XT4 varies between 0.58 and 0.99. For Jaguar XT5, it declines from 0.76 down to 0.66, and hence, show an increase in the degree of temporal recurrence. Interestingly and counter-intuitively, in the same period of time, the MTBF of Jaguar XT5 system increases significantly (Fig. 5). To test this trend further, we computed the correlation between the sequence of MTBFs and temporal recurrence parameter over different time period and across systems. We found the correlation coefficient for Jaguar XT4, Jaguar XT5, Jaguar XK6, and Titan are 0.81, 0.71, -0.97, and -0.03. This shows that a high likelihood of temporal recurrence doesn't necessarily mean lower MTBF. It varies significantly between these systems. MTBF and temporal recurrence parameter capture two different aspects of the system reliability and can not be used as a proxy for each other. As an implication, a highly reliable system (i.e., high MTBF) may still exhibit quite strong temporal recurrence of system failures (i.e., small temporal recurrence parameter).

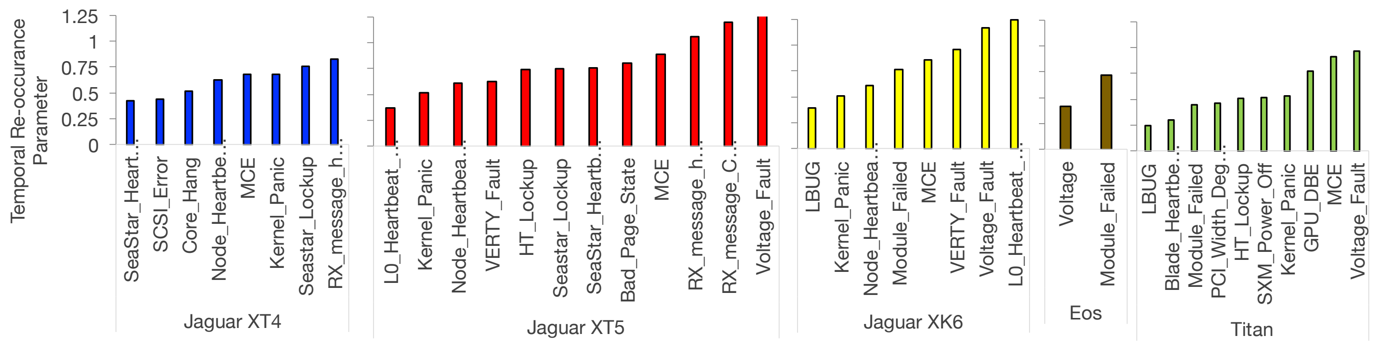


Figure 4: Temporal recurrence parameter per failure type for each system. Lower parameter value means higher temporal recurrence.

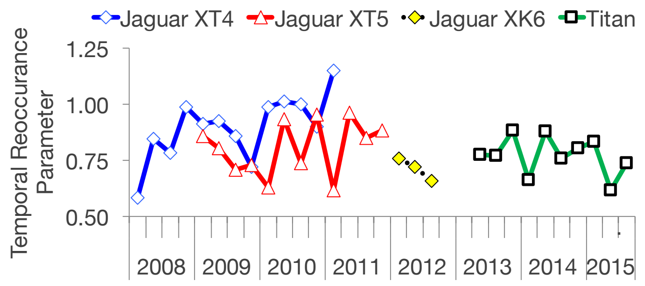


Figure 5: Temporal recurrence parameter of each system over time (averaged quarterly). Lower parameter value means higher temporal recurrence.

Continuing the similar line of investigation, we explore if the degree of temporal recurrence varies for different failure types across different systems. Fig. 4 shows the temporal recurrence parameter of each failure type across different systems. We point out that the MTBF of each failure type is different across different systems due to change in frequency of failures across systems. However, interestingly, we observe that the temporal recurrence parameter is similar for a given failure type across systems. For example, Kernel Panic and LBUG show low temporal recurrence parameter (approx. 0.40) across systems while Voltage Fault failure shows high temporal recurrence parameter (closer to 1) across systems except Eos. This findings indicate that temporal recurrence property is inherent to a particular type. Software related failures tend to exhibit higher temporal recurrence (and hence, lower temporal recurrence parameter). On the other hand, system power/cooling related failures are relatively less temporally correlated, resulting in relatively lower temporal recurrence (and hence, relatively higher temporal recurrence parameter). This observation can help in the planning for proactive action and repair mechanism for a given failure type. A system could also be proactively stress tested for some particular type of failures after the first occurrence of the failure – to avoid the impact of future failures on critical jobs.

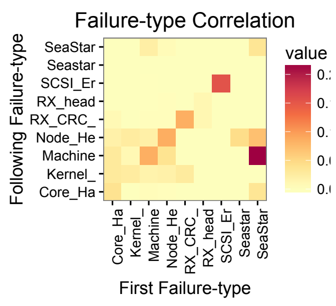
Next, we investigate the temporal recurrence characteristics *among* different failure types. In Fig. 6, temporal recurrence characteristics amongst different failure types is presented as heatmaps. The heatmaps show the probability of a failure type (say B) following another failure type (say A) within a time window. In other words, it is the conditional probability that given failure type A has just occurred, a failure of type B will occur within the time window (T). In Fig. 6, we show the results for 1 hour time window for Jaguar XT4, Jaguar XT5, Jaguar XK6, and Titan XK7. We found similar trends for larger time window sizes.

We observe that all failure types exhibit strong temporal recurrence relationship with themselves across all systems. In other words, a given failure is often followed by another failure of the same type within 1 hour across all systems. We note that this observation is not skewed since we have already discarded any superfluous reporting of same failure event in the failure logs by applying strong filtering methods (described in Section 2). Interestingly, we found a strong recurrence relationship between a Seastar Lockup being followed by Machine Check Exception (Fig. 6(a)). We investigated deeper and found that Seastar lockup may cause the memory corruption in the router that can also manifest as a MCE failure later depending on the memory access pattern and ECC scrubber frequency. Catching these failure correlations can be used by system designers and operators to deploy resilience related proactive measures for user jobs.

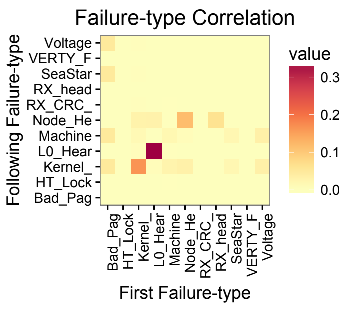
Summary We found that the temporal recurrence property varies significantly over time for a given system.

We are the first to show that the temporal recurrence property for different failure types are significantly different, but similar across systems. These observations can be used to their advantage by system operators to improve planning for proactive action and repair mechanism for a given failure type. We also showed how MTBF and temporal recurrence parameter captures two different aspects of the system reliability – any one alone is not sufficient. We also demonstrated and explained interesting temporal recurrence relationship between same failure types.

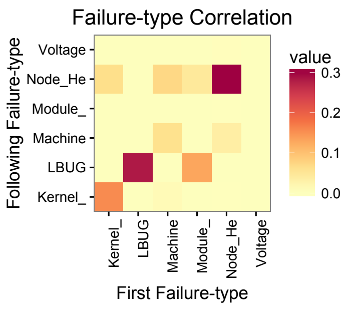
Next, we want to investigate temporal characteristics of failures. We explore the temporal characteristics of failures in more detail by asking the following specific questions:



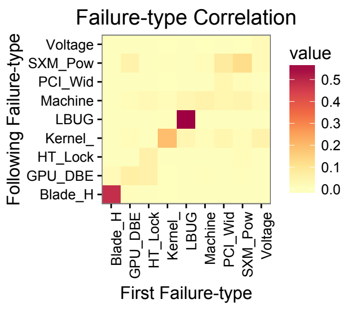
(a) Jaguar XT4



(b) Jaguar XT5



(c) Jaguar XK6



(d) Titan

Figure 6: Temporal recurrence relationship between failure types on different systems. The heatmaps show the results for one hour time window. SeaStar refers to SeaStar Heartbeat Faults and Seastar refers to Seastar Lockup events.

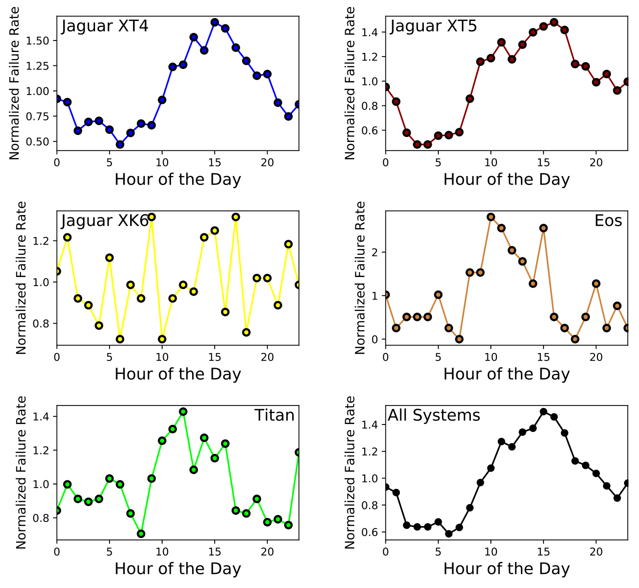


Figure 7: Normalized failure rate of all failures across systems at per-hour granularity (normalized to the average of failure frequency over 24 hours).

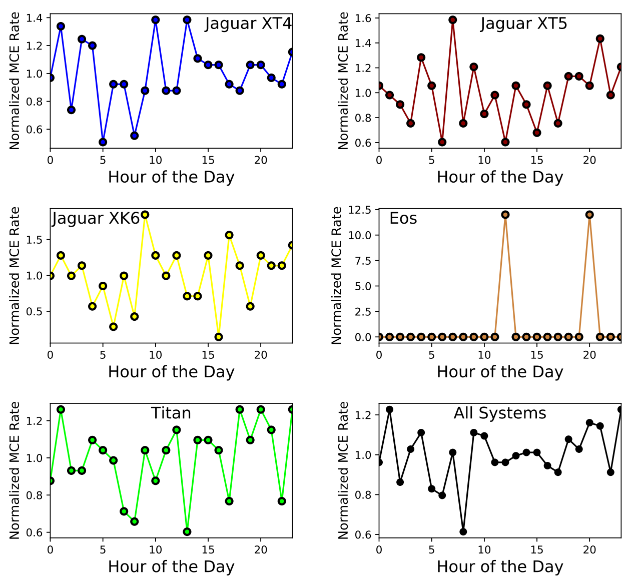


Figure 8: Normalized failure rate of Machine Check Exception failures across systems at per-hour granularity (normalized to the average of failure frequency over 24 hours).

RQ5: *How does the temporal characteristics of failures change over hour-of-the-day, day-of-the-week, and month-of-the-year?*

RQ6: *Do these characteristics change across systems and failure types?*

Fig. 7 shows the normalized failure rate of all failures across all systems at per-hour granularity. We found that the failure rate increases during afternoon hours, potentially because of increased utilization. However, the amount of variance is significant and depends heavily on the system. The normalized failure rate is approx. 20-40% higher than average during afternoon hours for most systems. However, interestingly the normalized failure rate is still quite high during the night and early morning hours (50% of the average rate), indicating that one can not assume very low failure rate during the night and early morning hours, as scientific users run high throughput batch jobs that can run overnight.

While the increased failure rate is expected during certain hours, we investigate if this trend holds true for individual failure types. Surprisingly, we discovered that hardware related failures do not necessarily follow the same trend of increase in failure rate during afternoon hours. To demonstrate this, we have chosen Machine Check Exception, a dominant hardware related error, as a proxy to demonstrate this trend; other hardware related errors show a similar trend. Fig. 8 shows MCE errors occur throughout all hours during a day. We note that Eos XC30 system is primarily a data analytics cluster and is not utilized at its peak capability all the time, unlike other systems. The users tend to schedule jobs on analytics

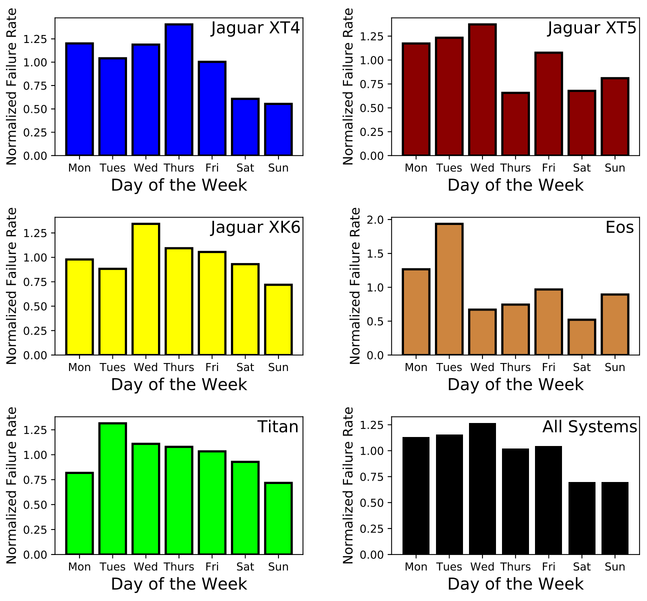


Figure 9: Normalized failure rate of all failures across systems at day-of-the-week granularity (normalized to the average of failure frequency over all seven days in the week).

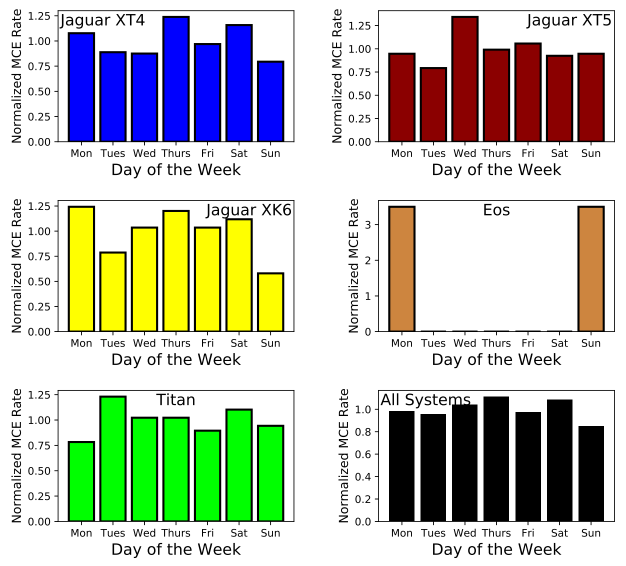


Figure 10: Normalized failure rate of Machine Check Exception failures across systems at day-of-the-week granularity (normalized to the average of failure frequency over all seven days in the week).

cluster in bursts; the bursty nature of MCE errors on Eos is an artifact of that utilization pattern.

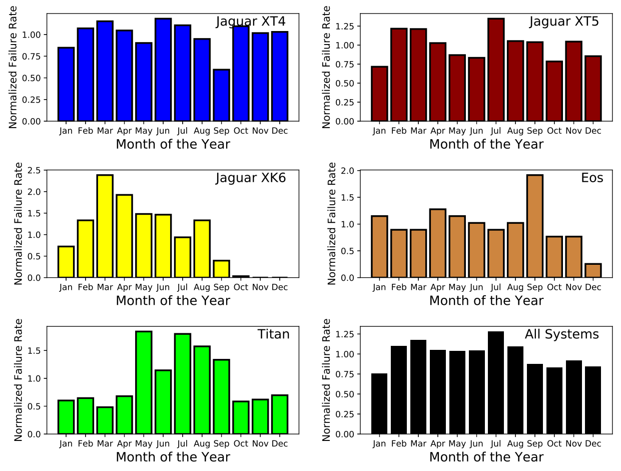


Figure 11: Normalized failure rate of all failures across systems at month granularity (normalized to the average of failure frequency over all twelve months in the year and length of the observation period).

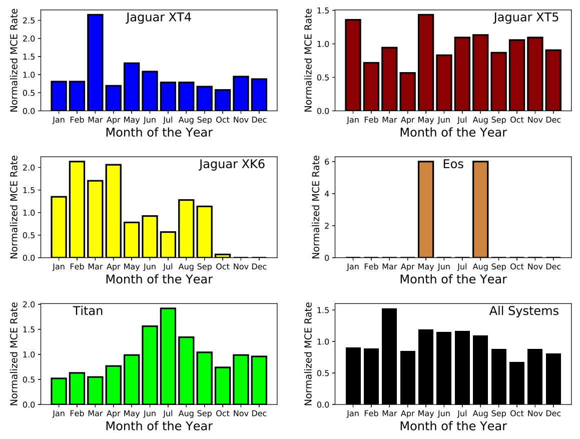


Figure 12: Normalized failure rate of Machine Check Exception failures across systems at month granularity (normalized to the average of failure frequency over all twelve months in the year and length of the observation period).

Next, Fig. 9 and 10 shows the normalized failure rate of all failures and machine check exceptions (uncorrectable hardware errors) across systems at day-of-the-week granularity. We found that the failure rate during the weekend can be 20-30% lower than weekdays, even for machine check exceptions (uncorrectable hardware errors). This could be better analyzed in conjunction with utilization data. Unavailability detailed and accurate resource utilization data limits us to explore the correlation between failures and utilization. It could be useful exercise for system operators to better understand and exploit such a possible correlation.

Finally, we attempt to understand the frequency distribution of failures over months in a calendar year. Fig. 11 and 12 show the normalized failure rate across systems at month-of-the-year granularity for all failures and MCE errors, respectively. We make two observations. First, the frequency of failures is not necessarily uniformly distributed. All systems tend to have lower failure rate during Oct-Jan time period by up to 20%, while March-May is relatively higher failure rate period.This observation can have significant implications about spare-provisioning and support/maintenance contracts from the vendor – which could be planned at the quarter granularity instead of year granularity and may result in cost-savings. Second, we noticed that MCE errors do not show significantly different behavior at month granularity. However, we do note that Titan has much higher failure rate during the Jun-Aug period because a specific project focused on a different deadline to produce results at scale during this period.

Summary We found that the failure rate increases during afternoon hours by upto 40%. However, this is not true for all failure types. For example, uncorrectable memory errors do not necessarily show increased failure rate during afternoon hours.

The failure rate during the weekend can be 20-30% lower than the weekdays in some systems. Our field data shows that the frequency of failures is not necessarily uniformly distributed over months. HPC systems tend to have lower failure rate during Oct-Jan time period by up to 20%, while March-May is relatively higher failure rate period. This observation can be beneficial to facility operators for devising better spare-provisioning strategies.

After investigating the temporal characteristics, we explore the spatial characteristics of failures. In particular, we ask the following questions:

RQ7: *What is the spatial distribution of failures across systems? How can one express this mathematically to enable its use in runtime systems and analytical tools that want to take advantage of spatial distribution of failures?*

RQ8: *Are spatial characteristics an artifact of temporal characteristics? How are they correlated with frequency of failures (MTBF)?*

Fig. 13 shows how failures are distributed among cabinets for different systems. We observe from the figure that the distribution of failures among cabinets is not uniform. To investigate deeper at finer compute granularity, we analyze the distribution of failures across the cages in each system. Fig. 14 shows the fraction of failures for each cage in the system sorted from highest to lowest for Jaguar XT5, Jaguar XK6, and Titan XK7. The figure shows that the failures are not uniformly distributed across cages either. Our results show that this is true across a number of systems of different sizes and component compositions..

To prove the statistical significance, we conducted Kolmogorov-Smirnov test (KS-test) to confirm that our observed empirical data

Table 4: Testing uniformity of neighborhood recurrence property: results of KS-test (D-statistic and critical D-value for 0.05 significance level).

System	D-statistic	Critical D-value	Null hypothesis
Jaguar XT4	0.5244	0.0455	Reject
Jaguar XT5	0.3955	0.0516	Reject
Jaguar XK6	0.4366	0.0516	Reject
Eos	0.1747	0.1408	Reject
Titan	0.3292	0.0364	Reject

is significantly different than randomly generated uniform distribution. The null hypothesis in KS-test is that the spatial distribution is taken from a uniform distribution. Table 4 shows the D-statistic and critical D-value for a 0.05 significance level. For each system, we find that the null hypothesis is rejected because D-statistic is higher than critical D-value. Therefore, for all the systems we study here, the observed distribution is significantly different than a uniform distribution. We also looked at distributions of failures on blade and node level, and observe that it also shows that the distribution is not randomly uniform.

This observation has significant implications for users and job schedulers. First, given this neighborhood recurrence property one may want to avoid scheduling critical jobs in the neighborhood of the location where last failure happened.

Second, users may use replication as a mechanism to tolerate failures. Replicated execution can decrease the perceived failure rate of the job. But, some job schedulers such Moab and scheduling techniques pack jobs in space to improve locality [41]. If replicated job is submitted under the same batch script, it is likely to experience higher failure rate due to neighborhood recurrence property. Users can submit replicas under different batch jobs in an attempt to avoid job scheduler's attempt to pack jobs closely in the same batch script. This method also increases the probability that these jobs may not start simultaneously, and may not be placed physically close by.

Next, we attempt to define neighborhood recurrence property more formally. Neighborhood recurrence property can be defined as the conditional probability that given a failure has occurred at a particular location, what is the probability a failure will reappear in the same locale in a given future window. Assume a time series of events F where each event has a timestamp (t) and location (θ). A time series can be expressed as:

$F_0(t = 0, \theta_0)$, $F_1(t_1, \theta_1)$, $F_2(t_2, \theta_2)$, .. $F_n(t_n, \theta_n)$, ...

F_0 is the current failure at $t = 0$ and F_n is a future failure at time $t = t_n$. Neighborhood recurrence property (NRP) is the conditional probability of the event, given current failure is at location A, a future failure within time T will be in the same locale. NRP is also a function of granularity Θ which decides if θ_0 and θ_n are in the same locale. Eq. 2 shows the neighborhood recurrence $NRP(T, \Theta)$ defined as the conditional probability. Since, the system has more than one unique location where failures can strike ($A_1, A_2, .., A_M$), $NRP(T, \Theta)$ can be expanded as shown in Eq. 3 by expressing it as a weighted sum of probability at those individual locations. Now, Bayes' Theorem allows us to simplify Eq. 3 into Eq. 4.

$$NRP(T, \Theta) = P(\theta_n = A \cap t_n - t_0 < T, \exists F_n \mid \theta_0 = A) \quad (2)$$

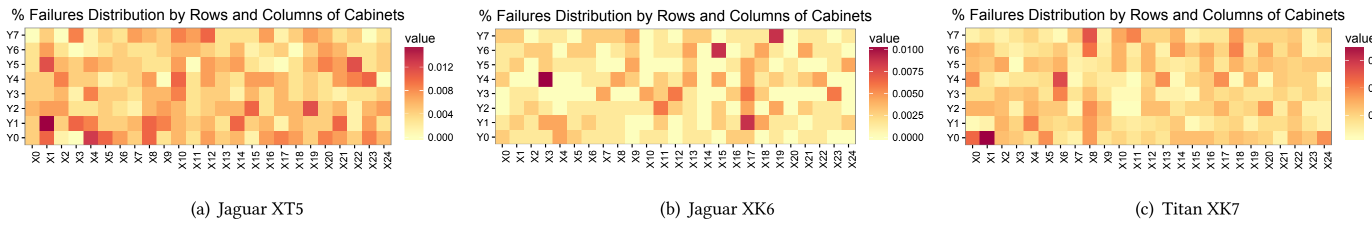


Figure 13: Non-uniform spatial distribution of system failures at the cabinet-level for different systems. Jaguar XT4 and Eos not plotted due to space restriction but show similar behavior.

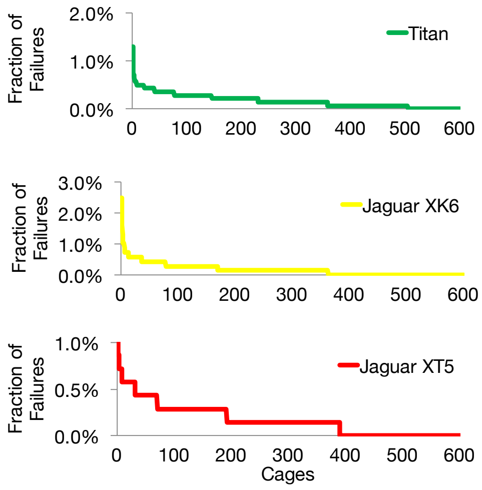


Figure 14: Non-uniform spatial distribution of system failures at the cage-level for different systems.

$$NRP(T, \Theta) = \sum_{i=1}^{M} P(\theta_n = A_i \cap t_n - t_0 < T, \exists F_n \mid \theta_0 = A_i) \times P(\theta_0 = A_i) \quad (3)$$

$$NRP(T, \Theta) = \sum_{i=1}^{M} P(\theta_n = A_i \cap t_n - t_0 < T, \exists F_n \cap \theta_0 = A_i) \quad (4)$$

As shown in above equations, the neighborhood recurrence effect is a function of future time window (T) and granularity of locale (Θ). In our analysis, we define the granularity of locale Θ with cabinet, cage, blade/module, and node. We calculate NRP at these granularities over different time windows.

Fig. 15(a)-(d) compare the systems with respect to their neighborhood recurrence effect at each granularity. First, we observe that neighborhood recurrence effect is present in all systems at all computing granularities (node, blade, cage, cabinet). Jaguar XT4, Jaguar XK6 and Titan show similar neighborhood recurrence trends over time, while Eos and Jaguar XT5 show significantly different behavior.

Second, the relative neighborhood recurrence effect between these systems also changes as the time window is changed. For example, Eos has higher neighborhood recurrence effect for smaller time windows while Jaguar XT4 has higher neighborhood recurrence effect if the time window is larger than 64 hours.

We note that neighborhood recurrence effect is observed even for relatively smaller scale system (i.e., Eos) at each granularity (in Fig. 15(a)-(d)), indicating that MTBF (or scale-normalized MTBF) alone is not enough to capture the spatial characteristics of system failure events. There are two different phenomenon that are responsible for this behavior. First, the degree of temporal recurrence is significantly high, i.e., the temporal recurrence parameter is smaller than other three systems. This implies that the likelihood of subsequent failures after a failure event is higher, and therefore, we see significant portion of failures show up in a smaller time window after a failure, although the MTBF is quite large. Second, other than the temporal recurrence in failures, there is significant neighborhood recurrence in failures where subsequent failures occur in the vicinity of previous failure events.

We point out that temporal recurrence is a system level observation and it is not capturing the location dimension. On the other hand, neighborhood recurrence deals with the spatio-temporal behavior. Also, when comparing Jaguar XT5 with other systems, it shows similar degree of temporal recurrence property as others, but has much lower neighborhood recurrence effect. This further strengthens our argument that spatio-temporal behavior is not a manifestation of temporal recurrence alone, and instead it is a fundamental characteristic of system failures which should be used to describe the reliability characteristics of a system. We clarify that certain locations may experience higher failures temporarily (e.g., more failures in higher cages which are relatively hotter than lower cages), however, this alone doesn't encapsulate the neighborhood recurrence behavior. Even regions with relatively lower failure concentration exhibit neighborhood recurrence behavior.

To further support our argument, we computed the correlation coefficient between neighborhood recurrence effect (at the cage level) and MTBF for each quarter ($-0.68, -0.38, -0.79, \textit{and} -0.53$ for Jaguar XT4, Jaguar XT5, Jaguar XK6 and Titan system, respectively). Similarly, we computed the correlation coefficient between neighborhood recurrence effect (at the cage level) and temporal recurrence parameter for each quarter. The correlation coefficients are $-0.72, -0.45, 0.92, \textit{and} -0.30$ for Jaguar XT4, Jaguar XT5, Jaguar XK6 and Titan system, respectively. This indicates that neighborhood recurrence effect can be both positively and negatively correlated with temporal recurrence, depending upon the system. Presence of both types of correlation suggests that neighborhood recurrence effect is not subsumed by the temporal recurrence parameter

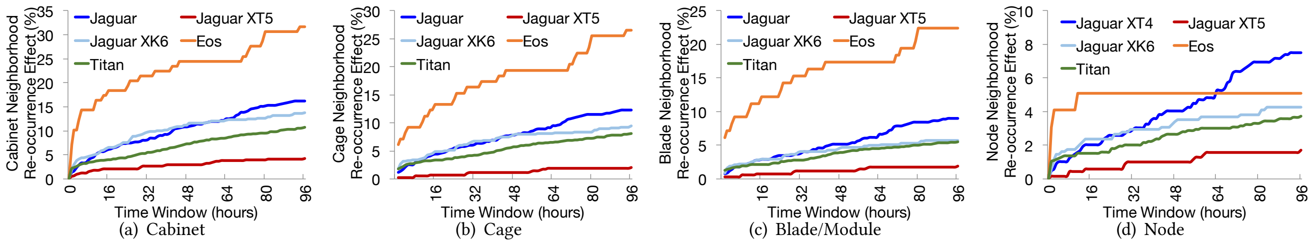


Figure 15: Neighborhood recurrence property at different granularity across systems for time window of up to 96 hours.

or MTBF metric; it is a separate property of a system that should be used if one desires to fully characterize the reliability of a system.

Summary We found that the spatial distribution of failures is not uniform at any compute granularity across systems. We discussed significant implications of this observation for users and job schedulers.
We showed how to capture neighborhood recurrence effect mathematically and demonstrated that neighborhood recurrence effect is not strongly correlated with MTBF or degree of temporal recurrence. Neighborhood recurrence effect should be used as a separate reliability characteristic of a system. It can not be subsumed by temporal characteristics such as MTBF or temporal recurrence. Interestingly, researchers at LLNL and Argonne National Laboratory have independently verified that our observations about spatial distribution of failures hold true in their systems as well [6, 30], which are not Cray systems and have very different system composition.

4 RELATED WORK AND CONCLUSION

Quantifying and characterizing the system failures is the first step for improving the reliability of HPC computing system. In order to characterize the failure event, works such as [35] find a fitting probability density function for the inter-arrival time of failures. Exponential, Weibull, Lognormal, and Gamma are some common distributions used to represent empirical data and the best fitting distribution is used to infer the system's behavior. For example, if the best fitting distribution is Exponential, then the system exhibits memoryless property, i.e. the failures are independent to each other; whereas, other distributions do not have this property.

Often the data is summarized by a single number (or a few numbers) instead of detailed mathematical functions. This leads to the average of these distributions, i.e., the mean of inter-arrival times or mean time between failures (MTBF) to be the one of the most common metric used in comparing systems. Some other metrics include reliability growth models to understand if the inter-arrival times are monotonically increasing [32]. Laplace test is applied to calculate Laplace factor that can be used to assess the failure intensity [32]. Gainaru et al. use autocorrelation metric to quantify the periodicity in system failure events [18]. Different peaks in autocorrelation represent some periodicity in the signal while a random signal will have a high correlation with itself (zero lag) and the correlation dies down with increasing lag [18].

On the other hand, in order to predict the failure events the analysis of the system failure events based on the event types and correlation among different event types is explored by previous works [8, 19]. The analysis and reliability metrics derived from the data are system specific and deal with understanding fault modes and their characteristics. The development of such analysis methodologies, tools and predictive models can lead to deeper insights about the underlying system behavior but it remains too complex to allow system administrators and users to compare reliability of two different systems. This work has introduced two new metrics to characterize temporal and spatial properties of failures that can be used by other failure analysis efforts [2, 6, 30].

Several studies, such as [8, 11, 17, 24, 25, 29, 31, 33, 34], focus on system failures characteristics to improve the reliability of HPC systems. For example, Liang et al. provided a thorough understanding of different component failures including disk, network, memory and processor for the Blue Gene/L system back in 2006 [24]. Oliner et al. investigated system RAS logs for multiple HPC systems including RedStorm and Thunderbird systems at the LANL and Sandia Lab back in 2007 [29].

Schroeder et al. have studied the system failures and its impact on multiple HPC systems at LANL [34]. System studied by Schroeder et al. spanned between 1996 and 2005, and the system with largest node count had 1024 nodes – in the same order as contemporary fastest supercomputers. However, there has been a lack of such large-scale reliability studies spanning across multiple systems, generations, and system types. Recent studies from UIUC and NCSA collaboration have attempted to address this gap, however they have only been limited on particular type of systems [7, 11, 12].

Some studies have also focused on studying the reliability of particular components such as DRAM, disks, and SSDs. For example, DRAM-focused efforts have shown manufacture specific insights and impact on DRAM reliability [20, 37, 40]. In contrast, this study covers multiple types of system failures and errors, and also investigates the impact of failure types, inter-arrival patterns and spatial correlation in system failures. We compare and contrast the reliability characteristics of multiple large-scale HPC production systems, and show how some of the characteristics can be used for future HPC system provisioning and overall system efficiency.

We note that due to the sensitive nature of the data, it can not be made publicly available as such. But, it is our continued effort and final goal to make parts of the data available for researchers in near future. Our study will enable others to learn from these lessons, apply new metrics, and test the findings presented here in their environment [6, 30]. We hope that these findings could be potentially applied to novel systems such as Minerva [21].

ACKNOWLEDGMENT

We are thankful to reviewers for their constructive feedback that has helped us improve the quality of this paper. This material is based upon work supported by Global Resilience Institute at Northeastern University, and by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, program manager Lucy Nowell, under contract number DE-AC05-00OR22725. Saurabh Gupta performed this work while employed at the Oak Ridge National Laboratory.

REFERENCES

[1] ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems, Peter Kogge, Editor and Study Lead, 2008.

[2] The HMDR Project: Holistic, Measurement-Driven Resilience. http://portal.nersc.gov/project/m888/resilience/.

[3] Lakshmi Narayanan Bairavasundaram. 2008. *Characteristics, impact, and tolerance of partial disk failures.* ProQuest.

[4] Leonardo Bautista-Gomez, Ana Gainaru, Swann Perarnau, Devesh Tiwari, Saurabh Gupta, Christian Engelmann, Franck Cappello, and Marc Snir. 2016. Reducing waste in extreme scale systems through introspective analysis. In *Parallel and Distributed Processing Symposium, 2016 IEEE International.* IEEE, 212–221.

[5] John T Daly. 2006. A higher order estimate of the optimum checkpoint interval for restart dumps. *Future Generation Computer Systems* 22, 3 (2006), 303–312.

[6] Sheng Di, Rinku Gupta, Marc Snir, Eric Pershey, and Franck Cappello. 2017. LogAider: A tool for mining potential correlations of HPC log events. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.* IEEE Press, 442–451.

[7] Catello Di Martino, William Kramer, Zbigniew Kalbarczyk, and Ravishankar Iyer. 2015. Measuring and Understanding Extreme-Scale Application Resilience: A Field Study of 5,000,000 HPC Application Runs. In *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on.* IEEE, 25–36.

[8] Nosayba El-Sayed and Bianca Schroeder. 2013. Reading between the lines of failure logs: Understanding how HPC systems fail, DSN. (2013).

[9] James Elliott, Kishor Kharbas, David Fiala, Frank Mueller, Kurt Ferreira, and Christian Engelmann. 2012. Combining partial redundancy and checkpointing for HPC. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on.* IEEE, 615–626.

[10] Cappello et al. 2009. Toward exascale resilience. *The International Journal of High Performance Computing Applications* 23, 4 (2009), 374–388.

[11] Di Martino et al. 2014. Lessons Learned From the Analysis of System Failures at Petascale: The Case of Blue Waters. *44th international Conference on Dependable Systems and Networks* (2014).

[12] Gupta et al. 2015. Understanding and Exploiting Spatial Properties of System Failures on Extreme-Scale HPC Systems. *International Conference on Dependable Systems and Networks (DSN)* (2015).

[13] Jones et al. 2012. Application monitoring and checkpointing in HPC: looking towards exascale systems. In *Proceedings of the 50th Annual Southeast Regional Conference.* ACM, 262–267.

[14] Snir et al. 2014. Addressing failures in exascale computing. *International Journal of High Performance Computing Applications* (2014), 1094342014522573.

[15] Tiwari et al. 2014. Lazy checkpointing: Exploiting temporal locality in failures to mitigate checkpointing overheads on extreme-scale systems. In *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on.* IEEE, 25–36.

[16] Tiwari et al. 2015. Reliability Lessons Learned From GPU Experience With The Titan Supercomputer at Oak Ridge Leadership Computing Facility. *Proceedings of SC15: International Conference for High Performance Computing, Networking, Storage and Analysis* (2015).

[17] Tiwari et al. 2015. Understanding GPU errors on large-scale HPC systems and the implications for system design and operation. In *High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on.* IEEE, 331–342.

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

[19] Ana Gainaru, Franck Cappello, Marc Snir, and William Kramer. 2012. Fault prediction under the microscope: A closer look into HPC systems. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis.* IEEE Computer Society Press, 77.

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

[21] Patricia Kovatch, Anthony Costa, Zachary Giles, Eugene Fluder, Hyung Min Cho, and Svetlana Mazurkova. 2015. Big omics data experience. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* ACM, 39.

[22] Patricia Kovatch, Matthew Ezell, and Ryan Braby. 2011. The Malthusian Catastrophe is Upon Us! Are the Largest HPC Machines Ever Up?. In *European Conference on Parallel Processing.* Springer, 211–220.

[23] Scott Levy, Kurt B Ferreira, and Patrick G Bridges. 2016. Improving application resilience to memory errors with lightweight compression. In *High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for.* IEEE, 323–334.

[24] Yinglung Liang, Yanyong Zhang, Morris Jette, Anand Sivasubramaniam, and Ramendra Sahoo. 2006. BlueGene/L failure analysis and prediction models. In *Dependable Systems and Networks, 2006. DSN 2006. International Conference on.* IEEE, 425–434.

[25] Yinglung Liang, Yanyong Zhang, Anand Sivasubramaniam, Ramendra K Sahoo, Jose Moreira, and Manish Gupta. 2005. Filtering failure logs for a bluegene/l prototype. In *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on.* IEEE, 476–485.

[26] Robert Lucas. 2014. Top Ten Exascale Research Challenges. In *DOE ASCAC Subcommittee Report.*

[27] Justin Meza et al. 2015. A Large-Scale Study of Flash Memory Errors in the Field. *ACM SIGMETRICS Performance Evaluation Review* (2015).

[28] Bin Nie, Devesh Tiwari, Saurabh Gupta, Evgenia Smirni, and James H Rogers. 2016. A large-scale study of soft-errors on gpus in the field. In *High Performance Computer Architecture (HPCA), 2016 IEEE International Symposium on.* IEEE, 519–530.

[29] Adam Oliner and Jon Stearley. 2007. What supercomputers say: A study of five system logs. In *Dependable Systems and Networks, 2007. DSN'07. 37th Annual IEEE/IFIP International Conference on.* IEEE, 575–584.

[30] Ayush Patwari, Ignacio Laguna, Martin Schulz, and Saurabh Bagchi. 2017. Understanding the Spatial Characteristics of DRAM Errors in HPC Clusters. (2017).

[31] Antonio Pecchia, Domenico Cotroneo, Zbigniew Kalbarczyk, and Ravishankar K Iyer. 2011. Improving log-based field failure data analysis of multi-node computing systems. In *Dependable Systems & Networks (DSN), 2011 IEEE/IFIP 41st International Conference on.* IEEE, 97–108.

[32] Brian Randell, Jean-Claude Laprie, Hermann Kopetz, and Bev Littlewood. 2013. *Predictably dependable computing systems.* Springer Science & Business Media.

[33] Ramendra K Sahoo, Mark S Squillante, A Sivasubramaniam, and Yanyong Zhang. 2004. Failure data analysis of a large-scale heterogeneous server environment. In *Dependable Systems and Networks, 2004 International Conference on.* IEEE, 772–781.

[34] B Schroeder and Garth Gibson. 2010. A large-scale study of failures in high-performance computing systems. *Dependable and Secure Computing, IEEE Transactions on* 7, 4 (2010), 337–350.

[35] Bianca Schroeder and Garth A Gibson. 2007. Disk failures in the real world: What does an MTTF of 1, 000, 000 hours mean to you?. In *FAST*, Vol. 7. 1–16.

[36] Bianca Schroeder and Garth A Gibson. 2007. Understanding failures in petascale computers. In *Journal of Physics: Conference Series*, Vol. 78. IOP Publishing, 012022.

[37] Bianca Schroeder, Eduardo Pinheiro, and Wolf-Dietrich Weber. 2009. DRAM errors in the wild: a large-scale field study. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 37. ACM, 193–204.

[38] John Shalf, Sudip Dosanjh, and John Morrison. 2011. Exascale computing technology challenges. In *High Performance Computing for Computational Science–VECPAR 2010.* Springer, 1–25.

[39] Vilas Sridharan and Dean Liberty. 2012. A study of DRAM failures in the field. In *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for.* IEEE, 1–11.

[40] Vilas Sridharan, Jon Stearley, Nathan DeBardeleben, Sean Blanchard, and Sudhanva Gurumurthi. 2013. Feng shui of supercomputer memory: positional effects in DRAM and SRAM faults. In *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis.* ACM, 22.

[41] Christopher Zimmer, Saurabh Gupta, Scott Atchley, Sudharshan S Vazhkudai, and Carl Albing. 2016. A multi-faceted approach to job placement for improved performance on extreme-scale systems. In *High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for.* IEEE, 1015–1025.