Analyzing a Five-year Failure Record of a Leadership-class Supercomputer

Elvis Rojas
National University of Costa Rica
Costa Rica Institute of Technology
erojas@una.ac.cr

Esteban Meneses
Costa Rica National High Technology Center
Costa Rica Institute of Technology
emeneses@cenat.ac.cr

Terry Jones, Don Maxwell
Oak Ridge National Laboratory
{trjones,maxwellde}@ornl.gov

***Abstract*—Extreme-scale computing systems are required to solve some of the grand challenges in science and technology. From astrophysics to molecular biology, supercomputers are an essential tool to accelerate scientific discovery. However, large computing systems are prone to failures due to their complexity. It is crucial to develop an understanding of how these systems fail to design reliable supercomputing platforms for the future. This paper examines a five-year failure and workload record of a leadership-class supercomputer. To the best of our knowledge, five years represents the vast majority of the lifespan of a supercomputer. This is the first time such analysis is performed on a top 10 modern supercomputer. We performed a failure categorization and found out that: i) most errors are GPU-related, with roughly 37% of them being double-bit errors on the cards; ii) failures are not evenly spread across the physical machine, with room temperature presumably playing a major role; and iii) software errors of the system bring down several nodes concurrently. Our failure rate analysis unveils that: i) the system consistently degrades, being at least twice as reliable at the beginning, compared to the end of the period; ii) Weibull distribution closely fits the mean-time-between-failure data; and iii) hardware and software errors show a markedly different pattern. Finally, we correlated failure and workload records to reveal that: i) failure and workload records are weakly correlated, except for certain types of failures when segmented by the hours of the day; ii) several categories of failures make jobs crash within the first minutes of execution; and iii) a significant fraction of failed jobs exhaust the requested time with a disregard of when the failure occurred during execution.**

***Index Terms*—Fault tolerance, resilience, failure analysis, high performance computing.**

I. Introduction

The computational demands for running scientific simulations and analyzing massive data repositories has been steadily growing. The need for ever-increasing processing power is present across many scientific disciplines and has resulted in groundbreaking discoveries from astrophysics to molecular biology. Often when experiments prove too costly or too dangerous, sophisticated supercomputers comprised of many heterogeneous computing elements are called upon to produce scientific simulations. These supercomputer systems, although successful, have two salient characteristics. First, they are very complex. They are composed of processors, memory modules, interconnection networks, disks, cooling devices, software stack, and others. It is exceedingly difficult to completely understand all possible interactions between parts of a highly integrated computer architecture. Second, the parts are faulty. Indeed, among the most powerful computers, the so-called *leadership-class* computers, simply getting 100% of the computer available for a single large scientific simulation is a significant challenge due to failures affecting some potentially small portions of the machine. In fact, it is urgent to tackle the reliability challenge as it represents one of the major hurdles to continue scaling big supercomputers [1], [2], [3].

We seek to improve the understanding of how different parts of today's large computing systems interact and fail to improve hardware and software designs for future resilient supercomputers. One approach to develop such an understanding is by analyzing component failures on a large machine. By inspecting the error logs and other abnormal events, it is possible to create a profile of the machine dynamics, improve root-cause analysis of failures, and identify areas with high potential for improvement in future systems resiliency.

We focus on understanding the hidden failure patterns of a leadership-class supercomputer in this paper. We start with a massive curated event database of the machine. The events in the database have been introduced by the system administrators and possibly reflect the best knowledge of the machine's internal operation. We distill the unique failures from the event database and provide several descriptive statistics of the system. We then analyze the failure rate at different levels. Finally, we aggregate the failure information to the workload records and study the interplay of both the usage and the failures of the system. Our analysis uses and correlates a five-year failure and workload record. To the best of our knowledge, this is the first time such analysis is performed on a top 12 modern supercomputer. Here are the highlights of this paper:

- A failure categorization for a five-year reliability record on a leadership-class supercomputer (§IV). We found that GPU-related errors dominate the record, with roughly

978-1-7281-4194-7/19/$31.00 ©2019 IEEE

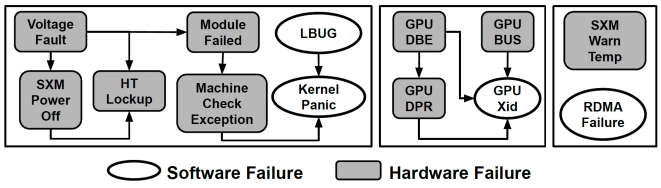


Fig. 1: Failure Propagation.

37% of them being double-bit errors. The physical distribution of nodes show some failure hotspots, with room temperature presumably playing a major role. Most failures only affect one single node, but software failures usually bring down several nodes concurrently.

- A close examination of failure rates using different scales (§V). We show how the system degrades, and highlight that at the beginning of the five-year period it was twice as reliable than at the end of the period. Weibull distribution best fits the mean-time-between-failure data.
- A study of the interplay between workload and failures on the system (§VI). Failure and workload records are weakly correlated, with the exception of software errors when analyzed by the hours of the day. Software errors typically make jobs crash within the first minutes of execution. A big portion of failed jobs complete their requested time, regardless of the failures that occurred during execution.

II. System Description

For this paper, we studied data from the Titan supercomputer. Titan is a Cray XK7 system located at the Oak Ridge Leadership Computing Facility (OLCF). Titan features a hybrid architecture of GPUs and CPUs. Together, these processing elements combine to provide 17.59 petaFLOPs from a total of 18,688 nodes. Each node has 32 GB of main memory with an AMD 16-core Opteron CPU and an NVIDIA Tesla K20 GPU. In total, the system has 299,008 Opteron cores. At the time of writing this paper, Titan is at the ninth position in the world according to the Top500 list [4], but in 2012 it occupied the first position and it was one of the first supercomputers to use a hybrid architecture.

A. Failure Dataset

This work analyzes failure records from year 2014 to 2018. The system administrators of Titan maintain a failure database to keep track of all abnormal incidents on the machine. Every single incident is registered into a failure database [5]. The database is automatically populated by a program that uses a *simple event correlator* (SEC) [6] to apply correlation rules and insert records in the database. Using the correlation rules, the SEC examines individual output streams from each node and merges multiple reports of the same event into a single database entry. An entry in the failure database corresponds to an event that can be linked to a component for a specific job identifier. Therefore, the failure database contains a list of records with the following properties: `hostname` (Titan supercomputer for this study), `job_id` (the job number affected by this event), `fail_time`, `category` (either *hardware* or *software*), `reason` (either *system* or *user*), `description` (of the event), `text` (with more information on the location of the event), and `jf_id` (event identifier). Table I presents the number of events in the failure database for each year.

Year	2014	2015	2016	2017	2018
Events	161,209	258,376	443,967	451,966	1,347,994

TABLE I: Number of Events in the Failure Database.

There is a range of severity in the events of the failure database. Some events represent warnings while others catastrophic failures. Also, some events in the database may be related if a fault in one component has an effect on other parts of the system. Therefore, several (potentially many) events may represent a single failure.

Figure 1 presents the cause-effect dependencies in the events of the database [7]. These are *potential* dependencies, because some events can occur without the preceding events in the graph appearing. For instance, it is usual to have a *GPU DBE* error generating a *GPU DPR* error and then a *GPU Xid* error. In that case, the filtering analysis will only leave the original *GPU DBE* error. However, in some other instances *GPU Xid* errors are generated in isolation. The left part of Figure 1 shows general error propagation paths. A *Voltage Fault* may induce a *Module Failure*, a *SXM Power Off*, or an *HT Lockup*. However, errors like *HT Lockup* and *Kernel Panic* most often have other causes other than voltage problems. The center part of the figure shows the GPU-related errors, with *GPU DBE* (double-bit error) causing *GPU DPR* (dynamic page retirement). Other erros are *GPU BUS* (GPU off the bus) and *GPU Xid*, which is a general type of software error with several different causes. The right part of the figure has the isolated errors, which means they are not produced by other events and they can not produce other events.

B. Workload Dataset

Our analysis incorporates job scheduler records for the years 2014-2018. The MOAB job scheduling and management system is used on Titan. MOAB registers many events associated with job execution: launch, start, pause, finish, cancel, and more. Therefore, for each job it is possible to know the number of nodes requested, the submission time, the waiting time in the queue, the effective execution time, and practically any salient feature of the job. MOAB stores the job event data in a file for each day, using the MOAB data format [8]. There are 57 properties for each event in a MOAB log file. We use the word *workload* to refer to all executed jobs within a timeframe.

III. Methodology

Figure 2 shows the methodology we implemented for the data analysis in this paper. We divided the analysis into *processes*, which display interdependencies. The methodology was implemented through a series of Python scripts, with the purpose of automating and simplifying the manipulation and analysis of failure and workload data. However, we will only describe the functionality of the processes and not the specifics of the scripts or data format nuances. For those

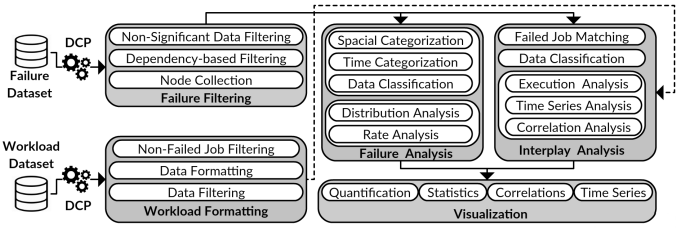


Fig. 2: Data Analysis Methodology.

details, we refer the reader to the open code repository at *https://github.com/emenesesrojas/slaf*.

The first process represents the *Data Cleaning and Preprocessing (DCP)* that enforced a consistent format in all input files, both from failure and workload sources. The next couple of processes consist of: i) selecting elements from the failure datasets (*Failure Filtering*), and ii) formatting elements from the workload dataset (*Workload Formatting*). These processes are necessary because datasets can not be directly used. The datasets generally contain a lot of redundant and unformatted data.

The *Failure Filtering* process starts by removing superfluous information in the dataset. We filter data considering two columns of the dataset, category and reason as explained in Section II. This process allows us to filter data by hardware or software failures and by type of error: memory errors or GPU errors in all its variations. This phase also filters out Heartbeat Fault entries. Such events are not considered failures, but system warnings. Then, redundant failure events are eliminated, i.e., those events that are dependent on other events with the same `job_id` number. We only retain events with no dependencies, because they are the root of the failure chain of events as suggested by Figure 1. Therefore, we avoid double counting failure events. The failure filtering process finishes by collecting the affected nodes in the failure chain of events, regardless of whether those events were removed.

The *Workload Formatting* process performs crucial sub-processes. First, it removes redundant information. Not all entries in the workload dataset are job entries. Some entries are events originated by the system scheduler or events triggered by fatal exceptions on the system. This sub-process removes all entries not related to job submission. Second, this process consolidates information on submitted jobs. The following measurements are of particular interest for each job: waiting time, execution time, and number of requested nodes. Not every job submission case is successful. For this analysis, we filter the workload dataset to get only jobs with job events `JOBEND` or `JOBCANCEL`. That assures us the job was in execution.

The *Failure Analysis* process inspects the previously filtered data in search of meaningful patterns. We performed a failure categorization by reason or description, and by different time scales to provide a broad view of failure classification. We also studied the spatial distribution of failure according to the cabinet where failures occurred. Since our analysis contemplates several years, we also looked for hidden patterns in the time series. This process also deals with failure rate analysis. We characterize the empirical *Cumulative Distribution Function* (CDF) for the mean time between failures (MTBF) and how well that data is fit by traditional probability distributions: Weibull, lognormal, and exponential distributions. Also, we evaluate the *Kolmogorov-Smirnov Goodness of Fit Test* to determine how close the data fit statistical distributions.

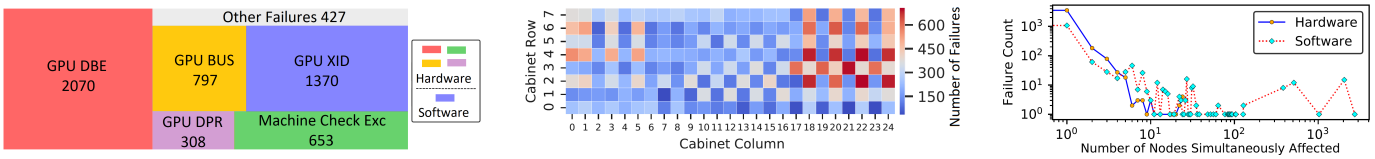
Additionally, we study the correlation between failures and workload (*Interplay Analysis* process). To evaluate such correlation, we implement the Pearson correlation index to determine if failures and workload are correlated in some way. Besides, to find possible correlations we extract and manipulate data from the filtering and formatting sub-process at different time scales.

Finally, the *Visualization* process uses the data to obtain an insightful visual display of results. We plot all necessary visualizations to show the correlations, categorizations, and results from our statistical analysis. This process also includes some transformation of data for the final projections.

IV. Failure Categorization

Each year, the SEC program inserts hundreds of thousands of records in the failure database (Table I). Each record can be classified using any of two variables: *category* (hardware or software) and *reason* (system or user). We wanted to focus only on system failures for this paper. Therefore, we discarded user failures. The total user failures removed in the five years were 816,826 (30% of total records for the 2014-2018 period). The share of system failures for each year is: 67.2% in 2014, 47.6% in 2015, 23.4% in 2016, 40.1% in 2017 and 98.8% in 2018. Nevertheless, although year 2016 has the lowest percentage of system failures, it is the year with the highest negative impact. It is essential to know the failure behavior throughout the five years. We started this study with the following research questions: what are the most common type of failures?, what physical locations are more likely to fail?, how many resources are affected by a failure?

Figure 3a shows failure categorization of the 2014-2018 period by hardware-software. Both categories have no comparable shares, 30.7% for software and 69.3% for hardware. Note that inside the hardware category there is a failure type (*GPU DBE*) with the highest percentage of all filtered failures with a 36.8% of total failures and 53% of the hardware category. This error can be caused by a variety of reasons such as voltage fluctuations and cosmic rays. Most of the GPU structures are protected with a *Single Error Correction Double Error Detection* (SECDED) ECC, but in this case it can only detect and it can not correct these errors. A *DBE* error does not necessarily mean that the execution terminates prematurely. But, when a *DBE* error is detected, the SECDED ECC always crashes the program. This is the case because a correct execution after a double-bit error detection cannot be guaranteed [9] [10]. The *GPU Xid* is a GPU error code from the NVIDIA driver indicating that a general GPU error occurred. The *Xid* errors can be an indicator of a hardware error, an NVIDIA software problem, or a user application



(a) Failures by Category and Description.

(b) Failures by Cabinet.

(c) Nodes Affected.

Fig. 3: Failure Categorization 2014-2018. Failures were categorized by their type in the dataset (3a), their spacial location (3b), and the number of hardware resources that were affected (3c).

problem [11]. Such a variety of causes explains the high percentage of *Xid* failures.

Figure 3a displays a section for *Other Failures* that includes both software and hardware failures. *Voltage Fault*, *SXM Power Off*, *SXM Warn Temp*, *HT Lockup* and *Module Failed* belong to the hardware category and *Kernel Panic*, *RDMA Failure* and *LBUG* belong to the software category.

Titan is composed of 200 cabinets, which are subdivided into cages (three cages per cabinet), blades (eight blades per cage) and nodes (four nodes per blade). Each node (CPU + GPU) has a unique ID for its identification. For example, the node with ID *"c24-1c2s7n1"* identifies the column 24($c24$), row 1(1), cage 2($c2$), blade 7($s7$) and the node 1($n1$). We carried out a spatial categorization based on the cabinet physical location. Figure 3b shows a heatmap with failure distribution across all cabinets. Note that the heatmap figure has 25 columns and 8 rows corresponding to Titan's physical distribution. We can see in Figure 3b the highest number of failed cabinets appear on the upper right corner and there exists an interleaved failure pattern in multiple rows. This is due to the fact a folded-torus cabling was used in Titan to avoid uneven length of cables in the *3-D torus Gemini Interconnection* [12]. For this reason, nodes within the same job will be allocated in this alternating manner [10]. This is interesting because it could be a signal of malfunction of some nodes that persists through time. In addition, the highest failure rate occurs on the right side of Figure 3b (the region between rows 1 to 7 and columns 17 to 24). According to the system administrator, across the aisle on that same side of the machine, there are other testbeds and clusters. Those machines might be pumping a considerable amount of hot air back into the room. Therefore, we presume temperature may play a role on increasing the failure rate of the components located on that side.

It is important to clarify that one failure can affect either only one node or multiple nodes. The number of affected nodes depends on the error type. For this reason, although in 2014 there were 565 failures (result of the failure filtering process), the same year reports $10,275$ affected nodes. For years 2015-2018, the number of failures were 649, 1824, 1291, and 1325, respectively. In the same range, the number of affected nodes were 1659, 2620, 1563, and 37,779, correspondingly. Due to the former premise, 70% of failed nodes correspond to 2018. It is important to note that in some cases, one failure could cause a thousand nodes being affected. Finally, it is important to remark that year 2018 is the year with most affected nodes and with the highest number of uniform failed nodes. This could be due to hardware problems that the system has been presenting since 2016.

Figure 3c shows the number of nodes affected by one failure. According to Figure 3c, more than 87% (4600 failures) of all failures only affect one node with 62% (2852 failures) of hardware failures and 38% (1748 failures) of software failures. Note that the number of failures affecting simultaneously more than one node decreases and it only represents 13% of all failures. This is important because we can see that in most cases the failures in Titan did not bring down multiple nodes.

If we analyze failed nodes by year, we find out that year 2016 shows the highest affectation with 35.5% of all failed nodes throughout 2014-2018. Years 2017 and 2018 have a share of 25.3% and 20.3%, respectively. It is important to point out that in almost all cases the highest value is the software category and only when one, two, three or four nodes are simultaneously affected is there an increase of hardware events. Another remarkable issue is that only 24 nodes were simultaneously affected by hardware problems. Nevertheless, by software problems the number of nodes simultaneously affected reached 2048 in 15 events. This is a indicator that software events were the main cause of bringing down multiple nodes.

V. Failure Rate Analysis

This section provides a characterization and modelling of failure rates in multiple dimensions. We set out with the following research questions: how are the different types of failures distributed in time?, what distribution better fits the data? This knowledge can be used to: i) address particular components during maintenance windows, ii) schedule jobs according to how critical they are, and iii) determine the way hardware and software errors differ. The most important metric to analyze the failure behavior on a system is the calculation of the *Mean Time Between Failures* (MTBF). In Figure 4 we can see that in the middle of the year 2015 the incidence of failures began to increase and in the year 2016 the failures reached the highest rate. After that, failure events decreased and in the year 2017 the failure rate was stable. Nevertheless, it was higher than the year 2014 and 2015. Finally, at the end of the year 2018 the failure rate began to increase again. This phenomenon has been described elsewhere [13], where the authors claim that the cause of failure increase in the

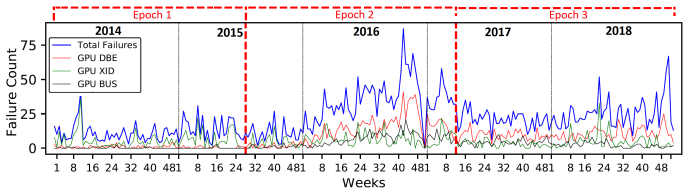
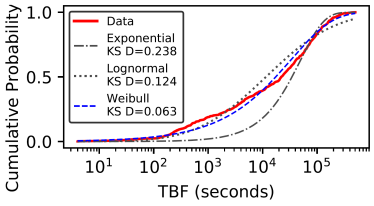


Fig. 4: Failure Time Series 2014-2018. It shows the total failure count and three of the main failure types.



(a) CDF of Epoch 1.

Epoch	Weib	LogN	Exp
1	0.06	0.12	0.24
2	0.02	0.05	0.13
3	0.02	0.09	0.08

(b) Kolmogorov Smirnov Test (KST).

Fig. 5: Cumulative Distribution Function (CDF).

year 2015 was attributed to the GPUs and the increase in the number of DBE failures was an indicator of a failing GPU device. After that, an effort was undertaken to fix the problem in July 2017, and 9500 GPU devices were replaced. Based on the former case and the description of the problem in the literature [13], we divided the failure time series in three epochs. We believe that the division by epochs (and not by years) provides a more precise analysis of the failure rate, because epochs exhibit distinctive failure patterns. The MTBF of each epoch in hours was 13.86 in epoch 1, 6.13 in epoch 2, and 7.16 in epoch 3. In general the failures appear distanced by a few minutes and the mean value decreases in more than 50% in epoch 2. After that, the mean value increases a little bit. Note that in epoch 2 the mean value decreases to just 6.13 hours. Failures increase from 1.73 per day in epoch 1 to 3.9 per day in epoch 2. An important reason behind the increase of failure frequency through the years is hardware degradation, especially degradation of GPU devices (see Section IV for more information).

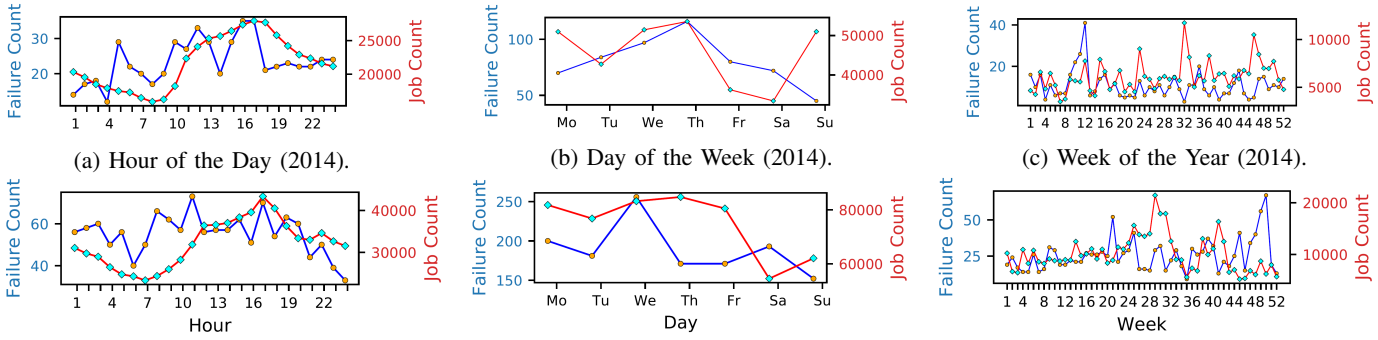
To better understand failure behavior, Figure 5 shows the MTBF *Cumulative Distribution Function* (CDF) of each epoch. We implemented three different types of distributions to find which one provides the best fit to the data failure frequency. Consistently, we see that in the three epochs the *Weibull* distribution fits the data better than the *Lognormal* and the *Exponential* distribution. The exponential distribution has the poorest fit. Also, we conducted the *Kolmogorov-Smirnov Goodness of Fit Test* (KST) to determine which distribution represents the cumulative distribution function of the MTBF better. The results of KST in each year (Table 5b) show us that the *Weibull* distribution fits the data best. We do not reject the null hypothesis because the computed D values of the *Weibull* distribution are the lowest and the p value for the three epochs is $p < 0.001$. The calculated *Weibull* shape parameter (k) was 0.57 in epoch 1, 0.75 in epoch 2, and 0.81 in epoch 3. That means, the defective items failed early and the failure rate decreased over time as the defective devices are removed out of the machine. The mean time of the *Weibull* distribution in each epoch was 14.77 in epoch 1, 6.01 in epoch 2, and 7.19 in epoch 3. We can see that the values of the model are similar to the values of the mean time data. These results are important for fault tolerance research and fault injection techniques that often assume that the failures come from an exponential distribution. This can lead to incorrect impact failure estimations, total execution times, and the optimal checkpoint intervals.

VI. Workload and Failure Interplay

Understanding the effect of the workload on the failure record is crucial to develop resilient software for HPC systems. We kicked off the interplay analysis with the following research questions: do failures and workload correlate?, which type of failures may be caused by users?, what is the impact of failures on the usage of the system? To determine other possible relationships between workload and failures in the system, we will present the data in different time series. Figure 6 shows two years of interplay between failures and workload. We chose to show only years 2014 and 2018 because they represent the two extremes in the time period and they have different interplay patterns. We devised a representation to display the full interplay details. We call this representation a *failure-workload correlation matrix* (FW matrix). An FW matrix has all years in the 2014-2018 period as rows and columns. The bottom triangular matrix (highlighted in blue) contains the correlation of failure time series for all pairs of years. The upper triangular matrix (highlighted in red) comprises the correlation of workload time series for all pairs of years. The main diagonal displays the failure and workload correlation of each year.

The first column of Figure 6 (6a, 6d) represents the interplay between failures and jobs per hour of the day. In Figure 6a we see a moderate similarity between failures and workload. This is confirmed with a correlation coefficient of 0.52. On the other hand, the year 2018 had the lowest correlation coefficient with 0.19. The second column of Figure 6 (6b, 6e) represents failure and workload rates by day of week. Figure 6b shows a strong similarity between workload and failures. But, its correlation coefficient is the lowest (0.21) of the five years. It is important to note that the year 2015 and 2016 have a correlation coefficient of 0.8 between the workload and failures. The last column of Figure 6 (6c, 6f) shows failures and workload per week of year. The correlation coefficient between failures and workload by week of year was clearly weak for all years (diagonal of Table 7c). This result suggests that by week of the year there is no correlation at all between failures and workload.

Figure 7 shows the interplay correlation coefficients in FW matrices. Looking only at the workload pattern in Figures 6a and 6d we get a correlation coefficient in hour of the day of more than 0.97 in all cases (upper triangular part of the FW matrix). Interestingly, this strong correlation in the workload shows us that the use of the system is very similar throughout the years. Also, matrix 7b shows us a strong correlation



(a) Hour of the Day (2014). (b) Day of the Week (2014). (c) Week of the Year (2014).

(d) Hour of the Day (2018). (e) Day of the Week (2018). (f) Week of the Year (2018).

Fig. 6: Interplay Between Failures and Workload for Years 2014 and 2018. Failures and jobs are shown per hour of the day (6a, 6d), day of the week (6b, 6e) and week of the year (6c, 6f).

Year	2014	2015	2016	2017	2018
2014	0.52	0.98	0.98	0.98	0.97
2015	0.53	0.81	0.98	0.98	0.97
2016	0.08	0.54	0.18	0.99	0.97
2017	-0.32	0.28	0.27	0.11	0.97
2018	0.14	0.39	0.25	0.51	0.19

(a) Hour of the Day.

Year	2014	2015	2016	2017	2018
2014	0.21	0.48	0.34	0.38	0.50
2015	0.64	0.80	0.87	0.96	0.96
2016	0.54	0.96	0.80	0.81	0.80
2017	0.54	0.81	0.76	0.67	0.96
2018	0.36	0.37	0.43	0.74	0.31

(b) Day of the Week.

Year	2014	2015	2016	2017	2018
2014	0.07	0.04	0.04	0.23	-0.13
2015	-0.12	0.06	0.05	0.32	0.02
2016	0.04	-0.25	0.19	0.06	-0.02
2017	0.14	0.29	-0.43	-0.2	0.31
2018	-0.11	0.02	0.22	-0.13	-0.09

(c) Week of the Year.

Fig. 7: Failures and Workload Correlations of the Period 2014-2018. It shows three correlation matrices in which the blue color represents the failure coefficients and the red color represents the workload coefficients.

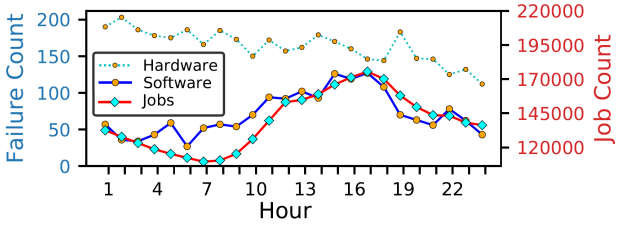


Fig. 8: Interplay Between Total Jobs and Hardware-Software Failures by Hour of the Day during 2014-2018.

between some years. The workload shows the best correlations between year 2015 and years 2016, 2017, and 2018. Also, the same year has strong failure correlation with years 2016 and 2017. Due to the former cases, we could conclude that there is a behavior pattern throughout the days of week and throughout the hours of the day, both of the workload and of the failures, among years. The matrix that represents the workload and failures by week of year (Figure 7c) has the poorest correlation coefficients.

Figure 8 shows five years of interplay between workload and hardware-software failures by hour of the day. Note that the software failures line visually coincides with the workload line, but with the hardware line there is no coincidence at all. The former premise shows us a possible correlation of the workload with the software failures. This hypothesis is reaffirmed with the strong *Pearson* correlation coefficient of 0.84 between the workload and the software failures. The correlation coefficient between the workload and the hardware failures is only 0.34. With these facts, we hypothesize that there is an incidence between workload and many of the software failure events that were generated in the system. Also, there is no apparent relationship between the workload and the hardware failures.

Figure 9 shows the time of execution of the year 2014 and 2018. It is essential to highlight that the five years show a similar pattern of execution time not only of failed jobs but also of the total quantity of jobs that were executed no matter if they failed or not. Only in the years 2014 and 2015 do the shortest execution times (range 0) provide more failed jobs. After that, failed jobs decreased keeping its number relatively low and constant until approximately a period of 20 minutes. Afterwards, there is a peak of failed jobs in the time range 50 to 99 minutes (50+) and in the last bar on the right which shows the number of all failed jobs with at least 10 hours of execution before failing. The time range 50+ has the highest number of failed jobs, because many of those failed jobs have a *wallclock time* limit of 60 or 90 minutes.

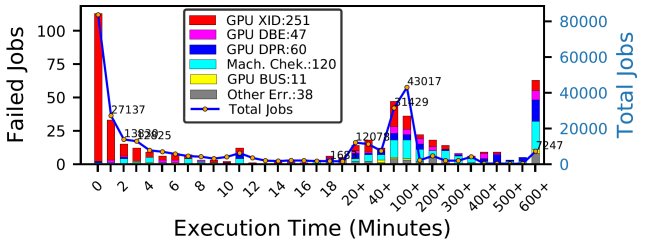
This provokes the idea that jobs that failed before two hours were not reported until the requested time ended. From 2016 to 2018, for example, Figure 9b registers less than 5 failed jobs in the execution range of 0 to 59 seconds (range 0), differing from years 2014 and 2015 (Figure 9a) that do not present failed jobs in that range. The root cause could be that in those years many of the jobs that failed did not do it within the first minute of execution and many failed jobs in that range were reported as user failures. Undoubtedly, the jobs with high execution times are not test or debug runs and are the most affected if resilience mechanisms are not incorporated.

The 0 range in years 2014 and 2015 presents a great number of failed jobs probably caused by the debug and test runs

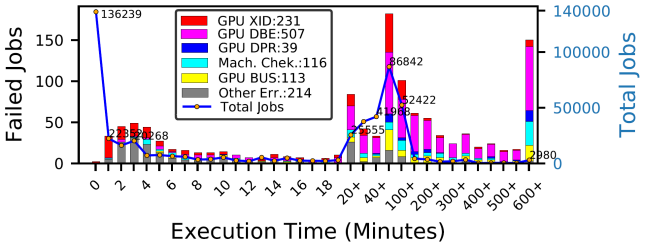
of the final user. Because they are debug and test runs, the execution does not take longer that one minute. Most of the jobs that fail in this range are type *GPU XID*. This type of failure corresponds to 98.3% in the year 2014 and 97.2% in the year 2015. According to NVIDIA's documentation [11], these types of errors can be an indicator that a general GPU error occurred (hardware problem, NVIDIA software problem, or a user application problem). In 2016, 2017, and 2018 range 0 had almost 0 failures.

If the causes of failed jobs in other time ranges are analyzed, it can be stated that a *GPU XID* remains as the main cause of failures. For example, in the 50-99 minutes (50+) these types of failures become 40% in the year 2014, 43% in the year 2015, 36% in the year 2016, 23% in the year 2017, and 25% in the year 2018. Finally, in the time range of 600 minutes plus (600+) the *GPU XID* decreases by 12%, 9%, 3%, 5%, and 5% in years 2014-2018, respectively. Nevertheless, in the same range *GPU DBE* failures increase by 11%, 20%, 37.5%, 44%, and 52% in years 2014-2018, correspondingly. Knowing the failure rates of *GPU XID* and *GPU DBE* may help the system administrators to focus on what hardware should be changed or fixed.

On the other hand, the highest execution time ranges and the highest number of failed jobs (i.e., 50+, 100+, and 600+) predominantly present failures of type *GPU DBE, GPU DPR, GPU BUS* and *Machine Check Exception*. The GPU errors are related to each other (Figure 1) and all, except *GPU BUS* failure, can be originated from *GPU DBE* failure. However, this type of failure should be very rare and according to the literature [10] there exists a probability that a higher temperature can lead to more *GPU DBE* errors. Additionally, some cards are more susceptible to this issue than others. In Figure 9b we observe that in most cases where there is a rise in the number of failed jobs, there also exists a rise in the number of the jobs that were executed. This indicates that the increase of failed jobs is correlated to the increase of the number of jobs in execution, which can affect the level of temperature of the systems. Also, with this information, the system administrators can change the scheduling policies to improve the system performance.



(a) Execution Time Distribution (2014).



(b) Execution Time Distribution (2018).

Fig. 9: Execution Time Distribution of Failed Jobs. Figures 9a and 9b show failed jobs (color bars) and total submitted jobs (solid line), respectively. Colors denote type of failure.

VII. Related Work

The analysis of failures on supercomputers in recent years has been an active research area. In [14], [15], [16] studies with multiple different HPC systems were carried out . Depending on the study, they analyzed the root cause of failures, failure rates, correlations, node reliability and statistical properties of time between failures, and the repair times. These three studies were made with datasets from Los Alamos National Laboratory (LANL). Perhaps, the biggest contrast between these studies and our study surrounds workload: despite the fact that these studies take into account many years of failure data, they do not have workload data for analysis. Other studies that included failure data from multiple supercomputer systems were [17] that analyzed the log failures of 10 public workload datasets from 8 HPC systems and [18] that analyzed the source of failures of Sunway BlueLight (multi-core CPUs) and Sunway TaihuLight (many-core CPUs) supercomputers.

Studies performed by [19], [20], [21], [22], [23] analyzed failure data from Blue Gene/L, Blue Gene/P and, Blue Waters supercomputers. In [19], [20], [21] failure data from Blue Gene/L was analyzed to perform statistical analysis, correlation analysis between failures and workload, failure categorization and implement filtering algorithms for the extraction and categorization of failure events. Failure data from Blue Gene/P and Blue Waters supercomputers were analyzed in [23] and [22], respectively. In [23], they analyzed both Reliability, Availability, and Serviceability (RAS) logs and job logs. In [22] performed statistical analysis with some distributions, analysis of system-wide outages, and characterization of the root causes of failures. Unlike [19] and [23], we use the failure and workload logs of five years; this allows us to perform comparative, time series and failure rate analysis of both failure and workload data.

Our analysis was developed with datasets from the Titan supercomputer, and it is important to recognize other failure studies for the same supercomputer. In [24], [10], [9] the authors analyzed Titan GPU errors from different perspectives. They included analysis of SBE, DBE and DPR errors, temporal and spatial characteristics of GPU failures, error frequency and the effect of soft-errors. Our study takes into account GPU errors too, but we did not analyze these failures deeply. In [25], [7], [26], they also used Titan failures. Not only did they analyze GPU failures, but also they analyzed failure events related to processor, memory, and system-user software. In [7], the authors performed a failure characterization with statistical analysis for an entire year, a quantitative analysis of the impact of failures on the workload of Titan, and they described a series of recommendations for understanding the

interplay between failures and workload on large supercomputers. Nevertheless, they only use a one-year dataset.

VIII. Conclusions and Future Work

Large supercomputers have provided the required processing power to push the envelope in many scientific fields. Larger systems address the challenges in areas as diverse as astrophysics and molecular dynamics. However, these extreme-scale systems integrate an immense amount of components that make them vulnerable to failures and hard to keep efficient. An understanding of the inner workings of the machine is pivotal to maintaining a HPC systems' productivity. Our analysis of failure and workload records of a leadership-class supercomputer shows that GPU-related components are highly vulnerable, to even double-bit errors. We found the Weibull distribution closely matches the MTBF data, outpowering other popular distributions. With the exception of software failures of the system, there is little correlation between failures and usage of the supercomputer. We provided support for the hypothesis that users are behind software system failures.

We plan on extending the current analysis by looking at user failures, a whole big category of events that were left out of the scope of this paper. In addition, we will analyze failure on particular components to better understand the reliability of a component by itself and create a model that predicts failure on a system that uses those components.

Acknowledgment

This research was partially supported by a machine allocation on Kabré supercomputer at the Costa Rica National High Technology Center.

References

[1] M. Snir, R. W. Wisniewski, J. A. Abraham, S. V. Adve, S. Bagchi, P. Balaji, J. Belak, P. Bose, F. Cappello, B. Carlson, A. A. Chien, P. Coteus, N. A. Debardeleben, P. C. Diniz, C. Engelmann, M. Erez, S. Fazzari, A. Geist, R. Gupta, F. Johnson, S. Krishnamoorthy, S. Leyffer, D. Liberty, S. Mitra, T. Munson, R. Schreiber, J. Stearley, and E. V. Hensbergen, "Addressing failures in exascale computing," *Int. J. High Perform. Comput. Appl.*, vol. 28, no. 2, pp. 129–173, May 2014. [Online]. Available: http://dx.doi.org/10.1177/1094342014522573

[2] F. Cappello, G. Al, W. Gropp, S. Kale, B. Kramer, and M. Snir, "Toward exascale resilience: 2014 update," *Supercomput. Front. Innov.: Int. J.*, vol. 1, no. 1, pp. 5–28, Apr. 2014. [Online]. Available: http://dx.doi.org/10.14529/jsfi140101

[3] E. N. Elnozahy, R. Bianchini, T. El-Ghazawi, A. Fox, F. Godfrey, A. Hoisie, K. McKinley, R. Melhem, J. S. Plank, P. Ranganathan and J. Simons, "System resilience at extreme scale," Defense Advanced Research Project Agency (DARPA), Tech. Rep., 2008.

[4] Top500.org. (2018) Top500 supercomputing sites. https://www.top500.org/. Accessed: 2018-08-19.

[5] D. Maxwell, "Restoring the cpa to cnl," May 2008.

[6] R. Vaarandi. (2018) Sec - simple event correlator. https://simple-evcorr.github.io. Accessed: 2018-08-19.

[7] E. Meneses, X. Ni, T. Jones, and D. Maxwell, "Analyzing the interplay of failures and workload on a leadership-class supercomputer," May 2015.

[8] I. Adaptive Computing. (2019) Moab workload event format. http://www.adaptivecomputing.com/. Accessed: 2018-08-20.

[9] 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)*. Barcelona, Spain: IEEE Xplore Digital Library, March 2016, pp. 519–530.

[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*. Austin, TX, USA: IEEE Xplore Digital Library, Nov 2015, pp. 1–12.

[11] N. Corporation. (2018) Xid errors. https://docs.nvidia.com/deploy/xid-errors/. Accessed: 2018-08-19.

[12] M. Ezell, "Understanding the impact of interconnect failures on system operation," 2013.

[13] C. Zimmer, D. Maxwell, S. McNally, S. Atchley, and S. S. Vazhkudai, "Gpu age-aware scheduling to improve the reliability of leadership jobs on titan," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, ser. SC '18. Piscataway, NJ, USA: IEEE Press, 2018, pp. 7:1–7:11. [Online]. Available: http://dl.acm.org/citation.cfm?id=3291656.3291666

[14] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance computing systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 4, pp. 337–350, Oct 2010.

[15] N. El-Sayed and B. Schroeder, "Reading between the lines of failure logs: Understanding how hpc systems fail," in *2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. Budapest, Hungary: IEEE Xplore Digital Library, June 2013, pp. 1–12.

[16] B. Schroeder and G. A Gibson, "Understanding failures in petascale computers," *Journal of Physics: Conference Series*, vol. 78, 09 2007.

[17] Y. Yuan, Y. Wu, Q. Wang, G. Yang, and W. Zheng, "Job failures in high performance computing systems: A large-scale empirical study," *Comput. Math. Appl.*, vol. 63, no. 2, pp. 365–377, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.camwa.2011.07.040

[18] R.-T. Liu and Z.-N. Chen, "A large-scale study of failures on petascale supercomputers," *Journal of Computer Science and Technology*, vol. 33, no. 1, pp. 24–41, Jan 2018. [Online]. Available: https://doi.org/10.1007/s11390-018-1806-7

[19] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*. Edinburgh, UK: IEEE Xplore Digital Library, June 2007, pp. 575–584.

[20] Y. Liang, Y. Zhang, A. Sivasubramaniam, R. K. Sahoo, J. Moreira, and M. Gupta, "Filtering failure logs for a bluegene/l prototype," in *2005 International Conference on Dependable Systems and Networks (DSN'05)*. Yokohama,Japan, Japan: IEEE Xplore Digital Library, June 2005, pp. 476–485.

[21] N. Taerat, N. Naksinehaboon, C. Ch, J. Elliott, C. (box Leangsuksun, G. Ostrouchov, and S. L. Scott, ".1. using log information to perform statistical analysis on failures encountered by large-scale hpc deployments," 2008.

[22] C. D. 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*. Atlanta, GA, USA: IEEE Xplore Digital Library, June 2014, pp. 610–621.

[23] Z. Zheng, L. Yu, W. Tang, Z. Lan, R. Gupta, N. Desai, S. Coghlan, and D. Buettner, "Co-analysis of ras log and job log on blue gene/p," in *2011 IEEE International Parallel Distributed Processing Symposium*. Anchorage, AK, USA: IEEE Xplore Digital Library, May 2011, pp. 840–851.

[24] 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)*. Burlingame, CA, USA: IEEE Xplore Digital Library, Feb 2015, pp. 331–342.

[25] R. Ashraf and C. Engelmann, "Analyzing the impact of system reliability events on applications in the titan supercomputer," pp. 39–48, 11 2018.

[26] 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*, ser. SC '17. New York, NY, USA: ACM, 2017, pp. 44:1–44:12. [Online]. Available: http://doi.acm.org/10.1145/3126908.3126937