# Characterizing Temperature, Power, and Soft-Error Behaviors in Data Center Systems: Insights, Challenges, and Opportunities

Bin Nie*, Ji Xue*, Saurabh Gupta†, Christian Engelmann‡, Evgenia Smirni*, and Devesh Tiwari§
* College of William and Mary ({bnie, xuejimic, esmirni}@cs.wm.edu)
† Intel Labs (saurabg@gmail.com)
‡ Oak Ridge National Laboratory (engelmannc@ornl.gov)
§ Northeastern University (tiwari@northeastern.edu)

***Abstract*— GPUs have become part of the mainstream high performance computing facilities that increasingly require more computational power to simulate physical phenomena quickly and accurately. However, GPU nodes also consume significantly more power than traditional CPU nodes, and high power consumption introduces new system operation challenges, including increased temperature, power/cooling cost, and lower system reliability. This paper explores how power consumption and temperature characteristics affect reliability, provides insights into what are the implications of such understanding, and how to exploit these insights toward predicting GPU errors using neural networks.***

## I. Introduction

Graphics processing units (GPUs) while traditionally designed for gaming and graphics applications, are now intensively adopted to accelerate scientific applications that require high throughput and extensive parallel computational resources [17, 21, 39]. The powerful parallelism provided by general-purpose GPUs enables scientists to simulate physical phenomena more quickly and accurately, but at the expense of higher computational power. Moreover, the reliability of GPUs cannot be overlooked because most scientific applications are long-running, taking several hours or even days to complete. Consequently, unreliable GPUs will lead to significant loss in research time and increase overall cost.

An initial step of any reliability study is to develop a deep understanding on GPU errors in the field. Tiwari et al. [35] study the characteristics of various errors on the Titan supercomputer at Oak Ridge National Laboratory. They demonstrate that GPU errors correlate with temperature and show early quantitative evidences. El-Sayed et al. [9] investigate the impact of high temperature on system performance and failures and reveal the presence of a non-trivial relationship between temperature and hardware reliability. A host of prior works study the impact of temperature on device reliability, but they mainly focus on hard disk drives, solid state drives, and CPUs [4, 9, 15, 29, 32, 34].

Unlike previous work, we conduct an in-depth study on the GPU nodes in the Titan supercomputer to understand the interplay between temperature and GPU reliability, and more specifically GPU soft-errors. We also investigate how different levels of power consumption affect the data centers operations and reliability . We believe that a thorough understanding on the relationship between temperature, power consumption and GPU errors is key to improving the operational efficiency of data centers. Beyond characterizing the conditions that may lead to GPU errors, we also exploit the observed insights and implications for error prediction. This paper makes the following three contributions:

First, we analyze large amounts of measured system data to understand the characteristics of the temperature distribution on the Titan. In particular, we investigate how the GPU temperature distribution varies in time and space across the system. Furthermore, we compare GPUs with other components, such as CPU and DIMM, and study how their temperature distributions differ from one another over time. We also study how frequently Titan nodes become extremely hot and for how long they stay in such a hot state.

We discovered that the retention time histogram in the hot state and in the normal state varies significantly between CPUs and GPUs. We also found that GPUs switch in and out of the hot and cold states more frequently compared to CPUs and stay in these states for a shorter period of time. We observe that, surprisingly, the retention time in the hot state remains similar for cabinets from different temperature zones.

Second, we show that there exists an interconnection between temperature, power consumption, and GPU soft-errors. It is challenging to quantitatively exhibit and exploit this relationship. We point out that predicting future GPU soft errors based on past temperature and power is simply inconclusive as contrasting conclusions may be reached.

Third, we elaborate on how to exploit the above observations for GPU soft-error prediction. We propose a machine-learning-based technique that leverages observations of past system measurements to predict soft errors in GPUs. We show

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).

that temperature and power consumption are of almost equal importance in GPU soft-error prediction, and together with a host of other factors including resource utilization, node location, and application type, may determine whether an upcoming application execution on a set of nodes will result in soft GPU errors. We evaluate our technique under various scenarios to demonstrate its effectiveness and robustness.

This paper is organized as follows. Section II introduces background and data collection methodology. In Section III, we investigate the temperature behavior on the Titan. Then, we discuss the relationship between SBEs and temperature/power consumption in Section IV. With the discovered observations, we design a novel algorithm to address the imbalanced dataset challenge and neural-network-based solutions for SBE occurrence prediction, and evaluate our method in Section V. We discuss open problems and challenges in Section VI and present related work in Section VII. Section VIII concludes the paper.

## II. Background and Methodology

Oak Ridge National Laboratory holds America's fastest supercomputer – the Titan supercomputer, which provides powerful computational ability for scientific projects. This 4352 square feet supercomputer is organized as a $25 \times 8$ grid, resulting in a total of 200 cabinets. In each cabinet, there are 3 cages, each of which is composed of 8 slots. Each slot has 4 nodes and 2 high-speed interconnect Gemini routers (two nodes share one router). Every Titan compute node is equipped with an AMD Opteron 6274 16-core CPU (with 32GB DDR3 memory) and an NVIDIA Tesla K20X GPU (with 6 GB GDDR5 memory). There are 18,688 GPUs in the entire Titan supercomputer, which together with the same amounts of CPUs, provide a theoretical peak speed of 27 petaFLOPS. K20X GPU [1] is specially designed for general-purpose computation on high-performance computing system. It contains 14 streaming multiprocessors (SMs), sharing the 1.5MB L2 cache and 6GB GDDR5 memory. Every SM contains 192 CUDA cores, and is allocated with 64K 32-bit registers, 64KB of combined shared memory and L1 cache, 48KB read-only data cache. Major memory components (i.e., register files, cache, and DRAM) are protected by single-error-correction-double-error-detection (SECDED) error-correcting codes (ECC). Read-only data cache is parity-protected. There is no ECC protection for non-storage components, such as logic units, thread schedulers, instruction dispatch unit, and interconnection network.

### A. GPU Errors and Data Collection

The Titan supercomputer is extensively used by scientific applications from different domains. Everyday this large-scale system observes various failures and errors, including hardware failures and soft-errors. NVIDIA documents a list of GPU XID errors and their causes in [2]. In this paper, we focus on the most commonly observed error – single bit error (SBE). Its large amount makes SBE an appropriate candidate to perform statistically solid analysis on GPU reliability. Though SBEs are correctable, the ECC protection overhead is significant from the viewpoint of the entire system, in terms of both storage and memory bandwidth. Therefore, it is of great importance to study the characteristics of SBEs and seek opportunities to perhaps reduce ECC overhead.

Towards this goal, we collect GPU error related information from February 2015 to June 2015 (more than 60 million node hours). We leverage the *nvidia-smi* utility to collect SBEs on every GPU node. This utility only provides snapshots of SBE counts before and after every executed application, which means that we cannot associate a timestamp with an individual SBE. Hence, we have to perform data analysis at the granularity of one "batch job" (also referred to as "job" or "batch"). We denote a batch job as a set of applications (also referred as "apruns") that are submitted by the same user. The SBE count is collected at the start and end of the batch job and the difference is used to represent the number of SBE experienced by this job. Our tracing framework is also able to identify the node location for SBEs. Besides error information, we collect GPU resource utilization data such as GPU core-hours, maximum memory consumption, and total memory consumption for every aprun. Temperature and power consumption information is collected every minute on every node in an out-of-band manner without instrumenting applications.

### B. Limitations and Scope

We also recognize the limitations and assumptions applicable in this study. First, we distinguish applications based on their binary names as we do not have the complete access to or knowledge about the users' code and their work flow execution practices. Secondly, the error collection is performed at the batch level. We conservatively assume that SBEs occur in all apruns of one SBE-affected batch. Finally, we also recognize that it is hard to accurately identify and model the effects of software stack changes or operational practices. Therefore, our data characterization and analysis does not explicitly model these effects.

## III. Temperature Behavior Characteristics

Before exploring the relationship between temperature, power consumption, and SBEs, we would like to develop a deep understanding of the temperature behavior on the Titan. Note that power consumption is highly correlated with temperature, i.e., the Spearman correlation coefficient is as high as $0.5$. Consequently, we primarily show analysis for temperature. Similar analysis (and conclusions) can also be applied to power consumption. There are two major questions we want to address for the temperature characteristics: (1) what are the GPU temperature characteristics at the node level, and how do these characteristics compare against other components in the system such as CPU and DIMM? (2) what are the GPU temperature characteristics at a coarser granularity such as at the cabinet level, and how do these characteristics differ from CPU and DIMM?

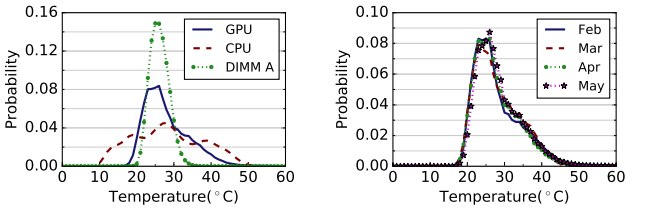



Fig. 1. (a) Histogram of temperature for GPUs, CPUs, and DIMMs. The average temperature of GPU, CPU and DIMM are $28.1°C$, $28.3°C$ and $26.1°C$, respectively. The standard deviation of GPU, CPU and DIMM are 6.1, 9.4 and 2.6, respectively. (b) Monthly Histogram of GPU temperature.

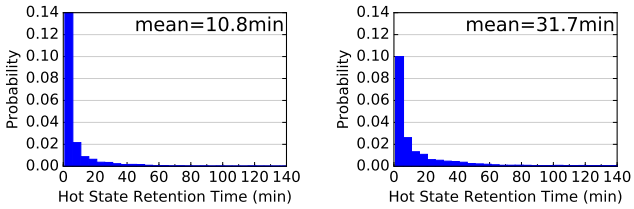



Fig. 2. *hot state*: retention time histogram for (a) GPU and (b) CPU. (Note that the long tail is truncated at 140min in both figures.)

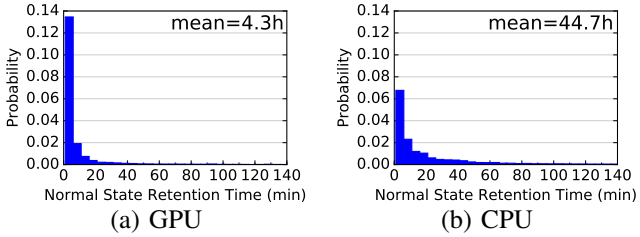



Fig. 3. *normal state*: retention time histogram for (a) GPU and (b) CPU. (Note that the long tail is truncated at 140min in both figures.)

We first show a general view of the temperature on the Titan supercomputer. Figure 1(a) presents the empirical pdf of temperature for different components (GPU, CPU, and DIMM) cumulated over the whole sampling period on the entire system. Each Titan node contains one CPU, one GPU, and four DIMMs. Due to space limitation, we only show the histogram of DIMM A, since all four DIMMs expose similar behavior. We observe that the GPU temperature histogram is fairly spread. While the mean is similar, the variance of the GPU temperature histogram is significantly different from the variance of the CPU and DIMM temperature histograms. This implies that all components attain a range of temperature values over time, and variance may vary from one component to another. Then, we show the monthly empirical pdf for temperature in Figure 1(b) and find that the temperature distribution remains steady over time. While the temperature distribution itself may not be used for SBE prediction, such high similarity in temperature distribution over time makes it amenable to learn and exploit it for other purposes by system administrators and facility operators.

Beyond the overall temperature distribution, we are also interested in how frequently Titan nodes become extremely hot and for how long they stay in such a hot state. Prior works [9, 27, 35] suggest that high temperature values are more likely to have high impact on errors. Here we choose *(mean+2std.)* as the hot state threshold. That is, a node enters a *hot state* if its temperature value exceeds *(mean+2std.)*; otherwise, it stays in the *normal state*. Note that the temperature threshold for GPUs (*Thr_GPU=40°*) is different from that for CPUs (*Thr_CPU=47°*) as the temperature distributions of the two components are different (see Figure 1). We define the continuous period during which a node stays in the hot state or normal state as the retention time. Note that we experiment with different temperature thresholds and find that observed trends and insights remain largely similar. Due to space constraints, we do not present these results. We also clarify that these thresholds do not have correlations with utilization levels, i.e., staying in normal state does not imply that the component is idle.

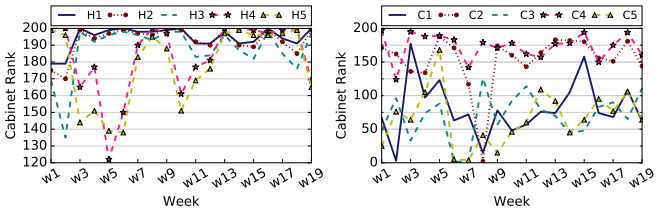
Figure 2(a) and (b) shows the retention time histogram for the hot state (for GPU and CPU components, respectively), while histograms for the normal state are presented in Figure 3(a) and (b). We make several interesting observations. First, the retention time histogram in the hot state and in the normal state varies significantly between CPUs and GPUs, implying a difference in their utilization patterns. Second, GPUs stay in the hot state for a shorter period compared to CPUs (i.e., when GPUs get relatively hotter, they are likely to remain in that state for a shorter period compared to CPUs). Recall that the absolute temperature thresholds for entering the hot state are different for GPUs and CPUs. At the same time, GPUs stay in normal state for shorter time too (i.e., when GPUs get relatively colder, they are likely to remain in that state for a shorter period as compared to CPUs). This indicates that GPUs switch in and out of the two states more frequently compared to CPUs and stay in these states for a shorter period of time.

Next, we look at the temperature distribution at the cabinet level (our previous analysis is at the node level). We investigate if the retention time for hot and normal states varies across cabinets with different relative hotness. To achieve this, we first rank all cabinets according to their cumulative temperature over every node. Then, we divide cabinets into three temperature zones based on their cumulative temperature values. We pick the 10 hottest cabinets (*HotCabs*), 10 coldest cabinets (*ColdCabs*), and 10 cabinets ranking in the middle (*MidCabs*), as representatives of the cabinets in each temperature zone. Table I shows the average retention time of the selected cabinets in each temperature zone for both hot and normal states (for GPU and CPU, respectively).

We observe that, surprisingly, the retention time in the hot state remains similar for cabinets from all three temperature zones. Notice that this is not an artifact of the cooling mechanism since it is not a reactive measure that kicks in after a threshold. In contrast, as expected, the normal state retention time of cabinets in *ColdCabs* is significantly greater than that of cabinets in *HotCabs*. In other words, cabinets in *HotCabs*

TABLE I
GPU AND CPU TEMPERATURE MEAN RETENTION TIME FOR CABINETS IN DIFFERENT TEMPERATURE ZONES.

| | GPU | | CPU | |
|---|---|---|---|---|
| | *hot state* | *normal state* | *hot state* | *normal state* |
| HotCabs | 10.9min | 2.7h | 34.6min | 22.0h |
| MidCabs | 11.0min | 4.2h | 31.6min | 56.0h |
| ColdCabs | 10.5min | 4.9h | 31.9min | 50.5h |




(a) Top 5 hottest cabinets (b) Top 5 coldest cabinets

Fig. 4. Weekly ranking for five hottest (a) and five coldest (b) cabinets.

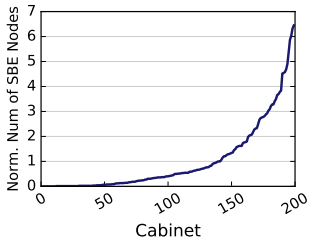



Fig. 5. CDF of normalized SBE node count at the cabinet level.

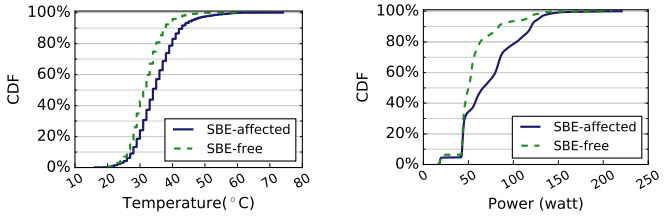



Fig. 6. CDF of temperature (a) and power (b) distribution of SBE offender nodes during SBE-affected period and SBE-free period.

enter a hot state more frequently, which holds for both GPUs and CPUs. While comparing GPUs and CPUs, we notice that the hot/normal state switch is more frequent for GPUs than for CPUs in all three temperature zones (consistent with Figures 2 and 3).

The preceding analysis is performed over the entire sampling period. It is also interesting to break down the time domain by looking into the dynamic nature of temperature distribution week by week. Towards this goal, we first rank all 200 cabinets according to their cumulative temperature over the entire sampling period. Note that a higher rank indicates a hotter cabinet, i.e., the hottest cabinet ranks 200 while the coldest one ranks 1. We present the weekly ranking over the entire period (19 weeks in total) for the 5 hottest and 5 coldest cabinets, see Figure 4. The 5 hottest and 5 coldest cabinets exhibit dramatic variation in their relative hotness ranking over the period. That is, the hot/cold ranking of cabinets changes significantly and frequently over time. We observe similar trends for other cabinets as well. This observation is particularly interesting because it suggests that although there are hotspot cabinets, these hotspots keep changing over time. Hence, to exploit the correlation between SBEs and temperature for SBE prediction, we must learn and capture this dynamic behavior accurately.

## IV. UNDERSTANDING SBE, TEMPERATURE, AND POWER

In Section III, we uncover several characteristics of the temperature on the Titan supercomputer. Here, we specifically look at the complex and obscure interplay between GPU soft-errors and temperature/power consumption. Our study reveals that different methodologies can lead to contrasting conclusions, sometimes even misleading ones.

We start from a coarse-grained overall viewpoint, by comparing the relationship between the number of SBE offender nodes and cumulative temperature over the whole sampling period at the cabinet level. Recall that the Titan is organized as a 25 by 8 grid of cabinets. For each cabinet, we count the number of SBE offender nodes, and then normalize the count by the average number. Figure 5 shows the normalized count of SBE offender nodes for every cabinet. Note that the cabinets are sorted by increasing count along the x-axis. The figure illustrates that SBE offenders are unevenly distributed across the system. In other words, some cabinets contain more SBE-affected nodes than others. Consequently, it is natural to ask whether the spatial distribution of temperature and power consumption correlates with the SBE count distribution. To this end, we collect aggregated temperature and power consumption over the whole sampling period for each cabinet, but we fail to observe any significant correlation with the SBE count distribution at the cabinet level. We caution that such a first order analysis may lead to premature conclusions since it only considers the cumulative value of temperature, power, and SBEs at a coarse granularity, i.e., at the cabinet level.

As a follow up, we investigate the issue at a finer granularity. We focus on the Titan compute nodes, more specifically SBE offender nodes only. We also incorporate the time dimension by dividing time in two parts: (1) time during which a SBE may have occurred, denoted as *SBE-affected period*, and (2) time during which no SBE occurs, denoted as *SBE-free period*. Figure 6 shows the cumulative distribution function of the temperature and power consumption on SBE offender nodes during these two periods.

We make a few important observations. First, Figure 6 (a) shows that the SBE offender nodes are relatively hotter during the SBE-affected period as compared to the SBE-free period. Similarly, the SBE offender nodes consume relatively higher power during the SBE-affected period than during the SBE-free period, see Figure 6(b). The observation is consistent with the calculated mean values shown in Table II. This indicates that SBEs are more likely to happen during time periods of elevated temperature. Note that high power consumption is

TABLE II
STATISTICS OF TEMPERATURE AND POWER ON SBE OFFENDERS.

| | Temperature (°C) | | Power (watt) | |
|---|---|---|---|---|
| | Mean | Std. | Mean | Std. |
| SBE-affected Time | 35.02 | 6.42 | 72.63 | 31.55 |
| SBE-free Time | 31.71 | 4.81 | 55.79 | 22.68 |

TABLE III
STATISTICS OF TEMPERATURE AND POWER ON NON-SBE OFFENDERS.

| | Temperature (°C) | | Power (watt) | |
|---|---|---|---|---|
| | Mean | Std. | Mean | Std. |
| SBE-affected Time | 34.30 | 6.76 | 68.22 | 33.09 |
| SBE-free Time | 30.44 | 5.21 | 48.17 | 23.79 |

likely to be a contributing factor toward increased temperature. But, due to varying cooling efficiency and workload characteristics, temperature elevation could be caused by other factors as well. In general, our measured data do not conclusively show that above a certain threshold of temperature/power consumption, SBEs definitely occur – making the SBE occurrence prediction non-trivial. Our results also show that temperature can be significantly high, sometimes even during the SBE-free period. In Table II, we also show standard deviation values. Temperature varies more dramatically during SBE-affected periods, possibly implicating that variance in temperature may be a contributing factor besides elevated temperature. A similar pattern can be also found for power consumption.

To summarize, we first show that one may prematurely conclude that SBE occurrence has almost no correlation with temperature or power consumption based on cumulative characteristics. However, when we go beyond simple cumulative behavior, we find evidence of correlation between SBE occurrence and temperature/power consumption. Our results indicate that SBE offender nodes typically consume more power and remain hotter during SBE-affected periods as opposed to SBE-free periods.

We caution that the effect of temperature or power consumption on SBEs is still not conclusive. The preceding analysis only considers SBE offender nodes – providing limited view of the whole system. For example, our previous analysis does not show that non-SBE offender nodes consistently attain lower temperature than SBE offender nodes during the SBE-affected period. So temperature and power consumption characteristics of non-SBE offender nodes should also be considered. Unfortunately, performing a meaningful and accurate data analysis on non-SBE offender nodes is challenging for multiple reasons. First, the number of non-SBE offender nodes is large ($\geq 17,000$ nodes) as compared to SBE-offender nodes ($\leq 700$ nodes). Second, the long observation period of this study induces difficulties in analyzing temperature and power consumption data in a meaningful and representative manner.

An intuitive solution to this problem is to randomly sample a subset of non-SBE offender nodes and perform comparisons with SBE-offender nodes. Unfortunately, this method leads to inaccurate conclusions. Random sampling of non-SBE offender nodes may include idle time at certain GPUs and hence, may likely result in lower average temperature and power consumption values. An alternative method is to sample only active GPUs at a given time. However, we found two issues that impede the practicality of this solution. First, current GPU resource utilization monitoring tools can not be used at runtime to monitor GPU utilization without imposing significant overhead on production systems. Second, sampled GPUs can execute workloads that finish at different times than the SBE-affected period on SBE offender nodes. To mitigate these challenges, we find that comparing against the other nodes in the same cage for a given SBE offender node result in consistent comparisons. The reasons are: (1) the nodes in the same cage are likely to be active at the same time due to the scheduling policy which is likely to pack one job in the same cage, (2) nodes in the same cage are likely to show similar variation in temperature due to power/cooling and spatial locality.

Table III shows the mean and standard deviation of the temperature and power consumption for non-SBE offenders in the same cage during SBE-affected and SBE-free period as observed on the SBE offender node (see Figure 6). We observe that even non-SBE offender nodes are relatively hotter during SBE-affected period compared to SBE-free period. Note that non-SBE offender nodes do not experience any SBE during an SBE-affected period or an SBE-free period. In addition, while non-SBE offender nodes are relatively hotter during the SBE-affected period, the SBE offender node is on average hotter than non-SBE offender nodes in the same cage. Similar observations can be drawn for power consumption.

The above observations imply that temperature and power consumption may have some effect on SBE occurrence, but, it is challenging to quantify the correlation due to monitoring limitations and interaction of other possible factors.

## V. PREDICTING SBES

Our characterization results reveal a relationship among temperature, power, and SBE occurrence, but not a clear one. It is unclear how to accurately predict SBE occurrence simply based on the statistical properties of temperature and power. In this section, we resort to neural networks to explore whether the time series of temperature, power, and other features can be used to predict SBE occurrence.

Artificial neural networks are inspired by biological neural networks and are composed of many interconnected neurons [3]. The weights associated with the neurons are used to approximate non-linear functions of the input features and are tuned during training. Training enables neural networks to capture the complex pattern between features and targets. Our purpose here is to use neural networks to explore hidden relationships among the selected features (e.g., temperature, power, utilization) and upcoming SBE occurrences.

First, we discuss how to select features that are potentially related to an SBE occurrence. The characterization analysis in the previous section shows that temperature and power are likely to be related to SBEs. Previous work has shown that spatial locality is another important feature for SBE

occurrence [12]. In addition, job log analytics indicate that different applications experience different rates of bit flips in hardware, possibly due to their data access pattern and interaction with hardware. Thus, application related information could provide potential features for SBE occurrence prediction, including aprun duration, memory utilization, and application type. Selected features are summarized as below:

- **Temperature:** We use the mean and standard deviation of temperature during an aprun as input features. To account for dynamic temperature behavior, the mean and standard deviation of the temperature *difference* between two consecutive minutes are also selected.
- **Power:** Similar to temperature, four metrics are selected for power: mean and standard deviation of consumed power during the aprun, and mean and standard deviation of the power difference between two consecutive power measurements.
- **Node location:** Row, column, and cage indices for each node are included (recall that the Titan is organized as a two-dimensional grid of cabinets, with each cabinet consisting of three cages).
- **Memory utilization:** GPU memory utilization for every node that the application is assigned to.
- **Application:** The aprun execution time and the application vector are also considered as features. The walltime of the aprun is the value normalized by the total number of nodes launched by this aprun, while the application vector represents which application is executed.

After discussing the feature selection process, we provide details on the training data set for the neural network. We collect data for all apruns during the sampling period. For apruns executing on SBE offender nodes, we divide the node's busy time into two parts: (1) SBE-affected time, if the aprun sees at least one SBE; otherwise, (2) SBE-free time. Busy time is defined as the time when a given GPU node is not idle. By definition, for non-SBE nodes, the busy time is always SBE-free time. We use the first three and half months of our entire sampling data as the training data set, this encompasses about $70,000$ apruns (i.e., 6 million samples). Each sample is identified by $<aprun\_id, node\_id>$. For example, an aprun launching on $5$ nodes will produce $5$ samples. Note that the number of apruns per month are not the same across each month. We select the first three and half months to collect enough observation samples. Indeed, as shown later in Section V-B, the testing data set contains the samples in the following two weeks, which encompass 16,000 apruns, such that the testing data is about 23% of the size of the training data, which is around the rule-of-thumb ratio of the testing data set to the training data set [13].

### A. Challenge: Imbalanced Data Set

Our first effort is to use the raw samples as input to train the neural network. Unfortunately, both precision and recall for SBE occurrence is as low as 0.01, while the precision and recall for non-SBE occurrence prediction is as high as 0.95. **Precision** is defined as the ratio of correct predictions (true positives) to all predictions (true positives and false positives). **Recall** is the ratio of the number of correct predictions (true positives) to the true positives and false negatives. Clearly, the low precision and recall values for SBE occurrences imply that such a naive model is not useful as it mislabels all samples as non-SBE.

Looking into the training data set, we find that the raw training data set is extremely imbalanced: almost $98\%$ of all training data are non-SBE occurrence samples, which results in a highly biased model. Imbalanced data sets is a noteworthy difficulty to machine learning models [28] as the resulted models favor the majority class and almost ignore the minority class, which is exactly what we observe here. To mitigate the imbalanced data problem, there are two common solutions [28]: (1) over-sample the minority class or (2) under-sample the majority class. Over-sampling replicates samples by creating synthetic minority samples based on nearest neighbors [6]. Here, we opt for under-sampling of the majority class, since this allows us to work with real rather than synthetic data.

**Algorithm 1** Select representative samples for one aprun based on feature correlation.

```
1: procedure SIMILARITY_REDUCTION(S, ρ_thres)
2:     high_corr_samples ← hashtable(sid, {});
3:     for s_i in S do
4:         feature_i ← feature list of sample s_i;
5:         for s_j in S do
6:             feature_j ← feature list of sample s_j;
7:             corr ← pearson_corr(feature_i, feature_j);
8:             if corr ≥ ρ_thres then
9:                 high_corr_samples(s_i) ← s_j;
10:            end if
11:        end for
12:    end for
13:
14:    //sorted in descending order
15:    Sorted_S ← sort(size(high_corr_samples(sid)));
16:
17:    selected ← {};
18:    avail ← Sorted_S;
19:    for s_i in Sorted_S do
20:        if size(avail) ≠ 0 then
21:            selected ← s_i;
22:            avail.remove(s_i);
23:            for s_j in high_corr_samples(s_i) do
24:                avail.remove(s_j);
25:            end for
26:        end if
27:    end for
28:    return selected;
29: end procedure
```

One method for under-sampling is to *reduce similar samples* in the majority class. Here, we propose a customized under-sampling method, which is based on similarity comparison of the feature sets among different training samples from the same aprun in the majority class. Algorithm 1 shows how we select representative samples for one aprun. The key idea is that if two feature sets are highly correlated, we only select one of them for training. The algorithm inputs are: 1) the normalized features of all training samples for this aprun, denoted by $S$, and 2) a threshold $\rho_{thres}$, used to determine whether the Pearson correlation of the feature sets is strong enough. The larger the $\rho_{thres}$, the stronger the similarity between the samples. Sample thinning is based on $\rho_{thres}$ as

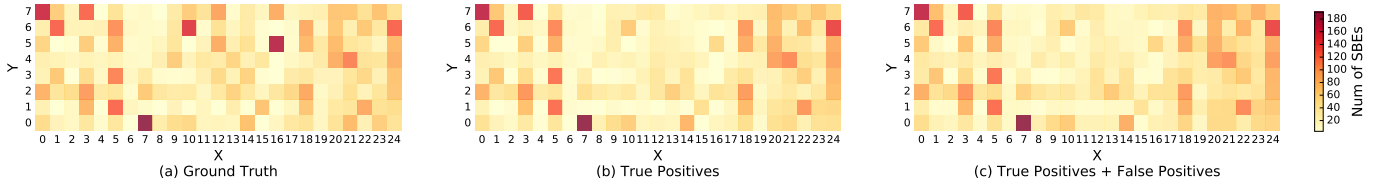



Fig. 7. SBE occurrence prediction at the cabinet level.

smaller values for $\rho_{thres}$ force more aggressive data reduction. We repeat the algorithm for every aprun in the training data set for the majority class. Several correlation threshold values for $\rho_{thres}$ can be used. With $\rho_{thres}$ less than 0.7, the data set reduction is too aggressive and too few representatives are left for the majority class when compared with the original minority class. This confirms our assumption that there are plenty of redundant training samples in the majority class. We select $\rho_{thres} = 0.9$ as the threshold value, this selection reduces the raw data satisfactory sufficiently.

Note that $\rho_{thres} = 0.9$ is a choice that achieves good reduction of the dataset but certainly not the only one that can be used. Experimentation shows that various $\rho_{thres}$ values close to 0.9 are also effective. Moreover, to avoid favoring some apruns (we certainly want to avoid an imbalanced data set for apruns), we guarantee that for each aprun we select at least 2 training samples. The above efforts result in a significant data set reduction to a total of 0.2 million samples of which 60% are non-SBE occurrences and 40% are SBEs.

### B. Evaluation with Oracle Data

For the testing data set, we choose two weeks after the training period (2015/5/16-2015/5/29), containing 16,000 apruns, i.e., 0.5 million samples, bringing the ratio of apruns of testing versus training to 23%. Some of the selected features are known prior to the aprun execution (i.e., node location and application information), while some are not (i.e., temperature/power and memory utilization). In this subsection, we assume that we know a priori future temperature, power, and utilization to test the neural network model. In the next subsection, we will discuss how to predict future data and compare with the results shown here.

TABLE IV
PRECISION AND RECALL FOR THREE NEURAL NETWORKS.

| | Non-SBE | | SBE | |
|---|---|---|---|---|
| | Precision | Recall | Precision | Recall |
| All Features | 0.76 | 0.70 | 0.71 | 0.78 |
| No Power | 0.78 | 0.69 | 0.71 | 0.80 |
| No Temperature | 0.77 | 0.69 | 0.70 | 0.78 |

Table IV shows the precision and recall of non-SBE and SBE occurrence for the testing data set using *three* different neural networks: one with all features described in Section V, one with all features except power, and one with all features except temperature. All three models have similar prediction quality, while the one without power is slightly better than the rest two. Precision and recall are higher than 0.69 for all three

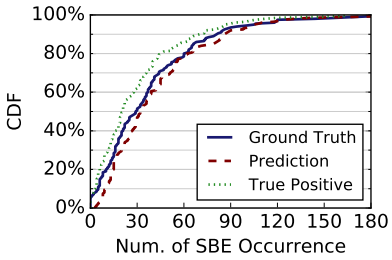


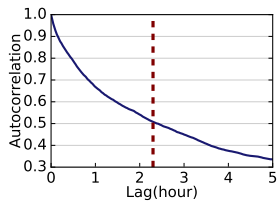

Fig. 8. Comparison between CDFs of ground truth, all prediction, and true positives for SBE occurrences at the cabinet level.

cases, suggesting that all models can identify most of SBE occurrences. Besides, we notice that the model without power and the one without temperature expose similar prediction ability. It is understandable since temperature and power consumption are highly correlated as stated previously. In the remaining of this section we focus on the neural network model that is trained without the power data. Note that we conducted experiments with all three neural networks and the results are indeed very close to each other and not reported here due to lack of space.

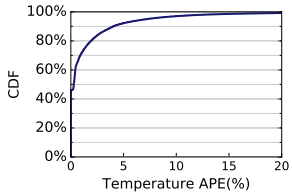
Precision and recall give an overview of the goodness of prediction. Figure 7 shows how many SBEs are predicted at the cabinet level throughout the Titan layout. Figure 7(a) corresponds to the ground truth, Figure 7(b) shows the raw predictions (true positives), and Figure 7(c) presents all predictions (true positives plus false positives) per cabinet. For most cabinets, prediction is quite close to ground truth with the exception of the middle upper part of Titan's layout.

To deliver a better overview of prediction, we compare the cumulative distribution plots of SBEs across the entire system to the ground truth, all predictions (true positives plus false positives), and true positives, see Figure 8. The three CDFs are close to each other, which further confirms that the neural network prediction is overall successful.

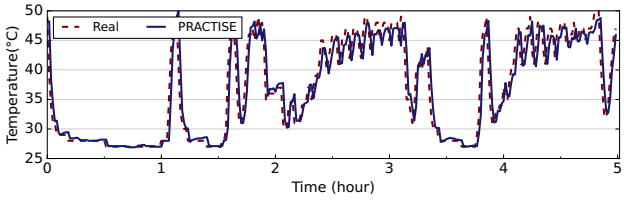
In addition, we observe that there are around 5% of cabinets where the neural network underestimates SBEs. These cabinets correspond to the ones in the upper middle part ($9 \leq X \leq 16$ and $5 \leq Y \leq 7$) of the cabinet layout, see Figure 7. To better understand why the neural network sometimes fails, we focus on two cabinets with underestimates and two cabinets with good predictions. For each SBE occurrence sample in the testing data set, we compute the correlation of feature sets, with every SBE sample and non-SBE sample in the training data set. We find that in the two cabinets with poor prediction 59% SBE occurrence samples in the testing data set have




(a) Autocorrelation of temp. in the training set.




(b) PRACTISE prediction accuracy for temperature.




(c) PRACTISE temperature prediction overtime.

Fig. 9. Autocorrelation and PRACTISE prediction for temperature.

similar features to non-SBE samples in the training data set. This number is dramatically low (only $5\%$) for the cabinets where the prediction is good. Essentially, it is not possible for the neural network to perform well for the cabinets with such close feature similarities. Perhaps more features for training are needed to increase prediction robustness to cover such situations.

### C. PRACTISE for Feature Prediction

In order to predict future SBE occurrence, we need to *predict the input features*. Location, memory utilization, and application related features are constant overtime, thus we use the average of recent observations as input. Temperature and power are not constant but rather fluctuate across time. To solve the challenge of temperature/power prediction, we leverage PRACTISE [40], which is a neural network prediction tool for time series data that is publicly available.

For PRACTISE to be successful, the time series needs to show temporal dependency. Autocorrelation is a mathematical representation of the degree of similarity in a time series and a lagged version of itself [18]. As such, it is ideal for discovering repeating patterns by quantifying the relationship between different points of a time series as a function of the time lag. The autocorrelation metric is in the range of [-1, 1]. Higher positive values indicate that the two points between the computed lag distance are "similar", i.e., have stronger correlation. Zero values suggest no periodicity. Negative values show that the two points that are lag elements apart are diametrically different. Figure 9(a) shows the autocorrelation of temperature for a random node in the training data set. The lag granularity is one minute. The vertical dashed line indicates the mean aprun duration, i.e., *2.3 hour*. The autocorrelation value of *lag=2.3h* for the temperature series is $0.5$ while autocorrelation values are much stronger for smaller lags. This implies the temperature series have strong temporal dependency.

Figure 9(c) shows the comparison between real values and PRACTISE-predicted temperature series of the node shown in Figure 9(a). The temperature prediction is very close to the actual values. Yet, this is just the prediction across a short time window. Figure 9(b) illustrates the CDF of the absolute prediction error (APE) for the temperature data for the entire prediction week. APE is the absolute difference between actual value and prediction value divided by the actual value.

$$APE = \frac{|Actual - Prediction|}{Actual} \quad (1)$$

The smaller the APE, the better the accuracy of prediction. Figure 9(b) shows that for more than $90\%$ of time, the APE is below $10\%$.

### D. SBE Prediction with PRACTISE

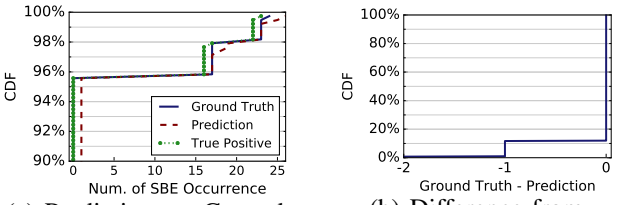
The above illustrates that PRACTISE can predict future temperature series accurately. As a next step, we apply the predicted temperature features to the neural network model, to test whether we can achieve good prediction of future SBE occurrences or not. All other features of the neural network model (node location, application) are known as well as duration and memory utilization (we use the average values from past runs of this application). Since we are interested in a fine granularity of prediction, i.e., on the specific node where the SBE may occur, we focus on a small set of cabinets. We choose 4 cabinets (384 nodes in total) in the upper left area (row 0 and 1 and column 6 and 7), which account for $10.4\%$ of the total number of SBE occurrences in the entire sampling period.

TABLE V
SBE OCCURRENCE PREDICTION: ORACLE VS. PRACTISE.

| | Non-SBE | | SBE | |
|---|---|---|---|---|
| | Precision | Recall | Precision | Recall |
| Oracle | 0.86 | 0.72 | 0.82 | 0.92 |
| PRACTISE | 0.88 | 0.62 | 0.82 | 0.95 |

Table V shows the precision and recall for SBE occurrence prediction using real values (i.e., if we know the future temperature features) and PRACTISE-predicted temperatures. We observe that it is effective to leverage PRACTISE-predicted temperature values for prediction. The similar precision values indicate that using PRACTISE is able to achieve the same level of correctness in prediction. While comparing recall values, the one with PRACTISE plugged-in is more conservative, reflected by the higher SBE recall and lower non-SBE recall.

Similarly to Section V-B (see Figure 8), we compare the CDFs of SBE predictions per-node: ground truth, all predictions, and true positive predictions, see Figure 10(a). We can barely distinguish the three lines from one another, indicating that the prediction is remarkably accurate. Figure 10(b), shows the CDFs of the difference between ground truth and all predictions. For less than $20\%$ of nodes, we over predict their SBE occurrences, but over-prediction is small (less than 2), especially comparing to the maximum number of SBE




(a) Prediction vs. Ground Truth

(b) Difference from Prediction

Fig. 10. Prediction for SBE occurrence at node level with PRACTISE.

occurrences per node, which is around 25. 90% of predictions are exactly accurate.

In sum, we have shown that it is possible to accurately predict future SBE occurrences on specific nodes. This could have multiple applications including tuning the ECC turn on/off period on selected nodes and for selected applications, resulting in significantly reducing memory space and memory bandwidth overheads for many applications.

## VI. Discussion

In this section, we discuss the applications of the proposed SBE occurrence prediction tool. Meanwhile, we will also demonstrate several open questions and challenges in this study and plans for future work.

### A. Application of SBE Prediction.

An intuitive application of SBE prediction could be dynamically turning on/off the ECC mechanism on certain nodes for certain applications based on the prediction result. However, one may argue that it is too risky to completely turn off the ECC protection, especially for long-running scientific applications, as the aftermath of even a small probability of false positives is much more severe than the overhead of wastefully turning on ECC for a large portion of true negatives across the entire system.

Fortunately, there are several opportunities for bypassing this risk. First, we can leverage the fact that not all hardware errors will be reflected in the application outputs, which means that some of the errors are *masked*. Several show this by evaluating the impact of soft-errors, especially single bit errors, on GPU architecture with various fault-injection models and frameworks [10, 14, 19, 42]. For example, Hari et al. [14] build a compiler-based error injection, SASSIFI, and show that on average 80% of the injected single bit errors are actually masked in the output and thus are not perceived by the end user. Moreover, in [38], the authors claim that even for those corrupted outputs, there are chances that the outputs are acceptable by the end users. Though this work is done in the CPU domain, it is reasonable to assume that similar opportunities exist for GPU-accelerated applications. Note that, this idea of not-accurate but acceptable output is consistent with the goal pursued by scientists in approximate computing, including domains such as bioinformatics [16, 23], performance analysis [37], data mining [24], and image recognition [22]. Consequently, for those applications that do not require very strict accuracy, we can dynamically decide whether to turn on or turn off ECC protection based on our prediction results. Clearly, we can always keep ECC on for those applications that need high-level of ECC protection. Therefore, by taking advantages of these opportunities, we are able to strike the balance between performance, overhead, and reliability.

### B. Open Problems and Challenges

There are still several interesting open problems and challenges that are worthy of more detailed discussions. First, in this paper, we perform feature selection based on the conclusions derived from the characterization section, as well as previous observations made by related works [12, 25, 35]. We demonstrate that the selected features all together are effective for SBE occurrence prediction. But it is also interesting to investigate which features or combinations of features play an irreplaceable role in prediction. Here, we leverage a neural network because of its powerful learning ability. Neural networks also require a lot of computational capability and sometimes are prone to overfitting. Comparing our neural network solution with other machine learning models, such as SVM and decision trees, needs to be explored. Finally, SBEs show apparent spatial and temporal locality, which can also be leveraged by the prediction model. We will investigate those aforementioned issues extensively in future work.

## VII. Related Work

Nowadays, large-scale supercomputers are extensively used by scientists to derive scientific insights. Consequently, it is of great importance to build a steady and reliable system. Many prior works have investigated and analyzed the impact of failures and errors on large-scale computer systems [15, 31, 34]. Oliner et al. [26] study raw system logs from 5 real deployed supercomputers, including Blue Gene/L, Red Storm, Thunderbird, Spirit and Liberty, and provide directions for future reliability researches. Schroeder et al. [30] analyze the statistics and root causes of several kinds of failures collected from two HPC systems. There are also works that point out pitfalls in error studies and uncover valuable insights especially for DRAM and HDD failures [4, 15, 29, 32, 34].

However, comparing to CPUs and disks, GPUs are recently largely deployed on large-scale systems, resulting in limited studies that look specifically into GPU errors in the field. Martino et al. [8] and Tiwari et al. [35] statistically analyze GPU failures and errors on the Blue Waters supercomputer at the University of Illinois and the Titan supercomputer at Oak Ridge National Laboratory, respectively. They uncover several previously unknown characteristics for various GPU errors, including spatial and temporal locality and their relationship to resource utilization. Nie et al. [25] take a specific look at the GPU soft-errors on the Titan supercomputer. We stress that no prior work has studied the complex impact of temperature/power consumption on GPU errors, nor proposed any predictive capabilities that leverage the observed characteristics.

Time series prediction tools (i.e., ARMA/ARIMA [5] and Holt-Winters exponential smoothing [11]) have been widely applied to quantify the impact of workload changes to application and/or system performance [33, 36, 41, 43]. Tran et al. [36] leverage ARIMA to improve block prefetching for scientific applications while Zhuang et al. [43] use ARIMA for effective user traffic prediction for capacity planning. Compared to traditional models, neural networks have shown to be efficient in capturing irregular patterns in data center resource usage [40], effective characterization of TCP/IP [7], and web server views [20]. In this work, we use neural networks to successfully predict the number of SBE occurrences at the node level and at the cabinet level in a large-scale HPC system. The use of neural networks is necessary for predicting SBE occurrences as the statistical analysis that has been used for prediction [12] is insufficient here. The proposed neural network combines a set of features that can be used as a whole for SBE prediction and shows that in addition to node location, utilization and workload type, temperature is also important for future SBE prediction.

## VIII. Conclusion

In this paper, we reveal several interesting and useful insights obtained via studying the complex interplay between GPU soft-errors and temperature/power consumption. We analyze large amounts of measured system related data to understand the characteristics of temperature, power, and SBE distribution. Finally, we propose a machine learning based technique to exploit these insights for GPU soft-error prediction in an effective manner. Our technique also discovers several additional interesting findings that could not be easily derived otherwise. We evaluate our technique and demonstrate that it performs effectively under various scenarios.

## Acknowledgment

This material is based upon work supported by NSF grant CCF-1649087 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. The work is also partially supported by Northeastern University.

## References

[1] "Tesla K20X GPU accelerator," http://www.nvidia.com/content/pdf/kepler/tesla-k20x-bd-06397-001-v07.pdf.
[2] "XID errors," https://docs.nvidia.com/deploy/xid-errors/.
[3] N. K. Ahmed *et al.*, "An empirical comparison of machine learning models for time series forecasting," *Econometric Reviews*, 2010.
[4] L. N. Bairavasundaram, *Characteristics, impact, and tolerance of partial disk failures*. ProQuest, 2008.
[5] G. E. Box *et al.*, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
[6] N. V. Chawla *et al.*, "SMOTE: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, 2002.
[7] P. Cortez *et al.*, "Multi-scale internet traffic forecasting using neural networks and time series methods," *Expert Systems*, 2012.
[8] C. Di Martino *et al.*, "Lessons learned from the analysis of system failures at petascale: The case of blue waters," in *DSN*, 2014.
[9] N. El-Sayed *et al.*, "Temperature management in data centers: why some (might) like it hot," *SIGMETRICS*, 2012.
[10] B. Fang *et al.*, "Gpu-qin: A methodology for evaluating the error resilience of GPGPU applications," in *ISPASS*, 2014.
[11] P. Goodwin *et al.*, "The holt-winters approach to exponential smoothing: 50 years old and going strong," *Foresight*, 2010.
[12] S. Gupta *et al.*, "Understanding and exploiting spatial properties of system failures on extreme-scale HPC systems," in *DSN*, 2015.
[13] I. Guyon, "A scaling law for the validation-set training-set size ratio," *AT&T Bell Laboratories*, 1997.
[14] S. K. S. Hari *et al.*, "SASSIFI: Evaluating resilience of GPU applications," in *SELSE*, 2015.
[15] A. A. Hwang *et al.*, "Cosmic rays don't strike twice: understanding the nature of DRAM errors and the implications for system design," in *ACM SIGPLAN Notices*, 2012.
[16] R. K. Jena *et al.*, "Soft computing methodologies in bioinformatics," *European Journal of Scientific Research*, 2009.
[17] D. Kothe *et al.*, "Computational science requirements for leadership computing," *ORNL Technical Report*, 2007.
[18] L. M. Leemis *et al.*, *Discrete-event simulation: A first course*. Pearson Prentice Hall, 2006.
[19] G. Li *et al.*, "Understanding error propagation in GPGPU applications," in *SC*, 2016.
[20] J. Li *et al.*, "Forecasting web page views: Methods and observations," *Journal of Machine Learning Research*, 2008.
[21] C. L. Mendes *et al.*, "Deploying a large petascale system: The blue waters experience," *Procedia Computer Science*, 2014.
[22] J. Meng *et al.*, "Best-effort parallel execution framework for recognition and mining applications," in *IPDPS*, 2009.
[23] S. Mitra *et al.*, "Bioinformatics with soft computing," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 2006.
[24] S. Mitra *et al.*, "Data mining in soft computing framework: a survey," *IEEE Trans. Neural Networks*, 2002.
[25] B. Nie *et al.*, "A large-scale study of soft-errors on GPUs in the field," in *HPCA*, 2016.
[26] A. Oliner *et al.*, "What supercomputers say: A study of five system logs," in *DSN*, 2007.
[27] M. K. Patterson, "The effect of data center temperature on energy efficiency," in *ITHERM*, 2008.
[28] F. Provost, "Machine learning from imbalanced data sets 101," in *AAAI'2000 workshop on imbalanced data sets*, 2000.
[29] B. Schroeder *et al.*, "Disk failures in the real world: What does an MTTF of 1, 000, 000 hours mean to you?" in *FAST*, 2007.
[30] B. Schroeder *et al.*, "A large-scale study of failures in high-performance computing systems," *IEEE Trans. Dependable Sec. Comput.*, 2010.
[31] B. Schroeder *et al.*, "Flash reliability in production: The expected and the unexpected," in *FAST*, 2016.
[32] B. Schroeder *et al.*, "DRAM errors in the wild: A large-scale field study," in *ACM SIGMETRICS Performance Evaluation Review*, 2009.
[33] M. Shokouhi, "Detecting seasonal queries by time-series analysis," in *SIGIR*, 2011.
[34] V. Sridharan *et al.*, "Feng shui of supercomputer memory positional effects in DRAM and SRAM faults," in *SC*, 2013.
[35] D. Tiwari *et al.*, "Understanding GPU errors on large-scale HPC systems and the implications for system design and operation," in *HPCA*, 2015.
[36] N. Tran *et al.*, "Automatic ARIMA time series modeling for adaptive I/O prefetching," *IEEE Trans. Parallel Distrib. Syst.*, 2004.
[37] H. L. Truong *et al.*, "Soft computing approach to performance analysis of parallel and distributed programs," in *Euro-Par*, ser. Lecture Notes in Computer Science. Springer, 2005.
[38] R. Venkatagiri *et al.*, "Approxilyzer: Towards a systematic framework for instruction-level approximate computing and its application to hardware resiliency," in *MICRO*, 2016.
[39] J. S. Vetter *et al.*, "Keeneland: Bringing heterogeneous GPU computing to the computational science community," *Computing in Science and Engineering*, 2011.
[40] J. Xue *et al.*, "PRACTISE: Robust prediction of data center time series," in *CNSM*, 2015.
[41] J. Xue *et al.*, "Proactive management of systems via hybrid analytic techniques," in *ICCAC*, 2015.
[42] K. S. Yim *et al.*, "Hauberk: Lightweight silent data corruption error detector for GPGPU," in *IPDPS*, 2011.
[43] Z. Zhuang *et al.*, "Capacity planning and headroom analysis for taming database replication latency: experiences with linkedin internet traffic," in *ICPE*, 2015.