What Can We Learn from Four Years of Data Center Hardware Failures?

Guosai Wang, Wei Xu
Institute for Interdisciplinary Information Sciences
Tsinghua University, Beijing, China
wgs14@mails.tsinghua.edu.cn, weixu@tsinghua.edu.cn

Lifei Zhang
Baidu, Inc., China
zhanglifei@baidu.com

***Abstract*—Hardware failures have a big impact on the dependability of large-scale data centers. We present studies on over 290,000 hardware failure reports collected over the past four years from dozens of data centers with hundreds of thousands of servers. We examine the dataset statistically to discover failure characteristics along the temporal, spatial, product line and component dimensions. We specifically focus on the correlations among different failures, including batch and repeating failures, as well as the human operators' response to the failures. We reconfirm or extend findings from previous studies. We also find many new failure and recovery patterns that are the undesirable by-product of the state-of-the-art data center hardware and software design.**

I. Introduction

To meet the ever-growing demand for Internet services, people build extremely large-scale data centers. The hardware reliability has always been a crucial factor in the overall dependability of these IT infrastructures. Hardware failures are common in large-scale systems [1, 2], and these failures may cause service level agreement (SLA) violations and severe loss of revenue [3].

It is important to understand the failure model, as it helps to strike the right balance among software stack complexity, hardware and operation cost, reducing the total cost of ownership (TCO) in data centers. In fact, researchers have studied hardware failures for decades, from Gray's study in 1986 [4] to Oppenheimer's in 2003 [2] to the recent research [5–19], failure patterns change over time as the computer system design evolves. Many recent studies focus on supercomputers instead of commodity data centers or a single component such as memory or hard drive [11, 16–21].

Today's data centers are different in many aspects. On the positive side, the hardware components are designed and manufactured to be more reliable. There are also better failure detection systems making hardware failures easier to notice and repair. Moreover, operators have accumulated more experience on how to maintain large-scale infrastructures. However, on the dark side, Internet companies become more cost-sensitive, and people adopt less-reliable, commodity or custom ordered hardware [1]. There is also more heterogeneity in both hardware components and workload, resulting in more complex failure models.

Most interestingly, in addition to the common belief that hardware unreliability shapes the software fault tolerance design, we believe it is also the other way around: years of improvement in software-based fault tolerance has indulged operators to care less about hardware dependability.

With all the changes, we believe it is now necessary to conduct a new study on failures in modern large-scale Internet data centers. In this paper, we present a comprehensive analysis of failures during the past four years from a major and successful Internet service company that operates dozens of data centers hosting hundreds of thousands of servers, serving hundreds of millions of users every day. Different from previous studies on a single supercomputer, our data centers host generations of heterogeneous hardware, both commodity and custom design, and support hundreds of different product lines. Also, we cover all hardware component classes as well as human operators' behavior in the study.

Specifically, we analyze all the hardware *failure operation tickets (FOTs)* collected from a centralized *failure management system (FMS)* that monitors most of the servers and records all component failures together with the operators' actions. We observe over $290,000$ such FOTs during the past four years. We draw statistically meaningful conclusions about these failures.

In the paper, we study the failures along different dimensions, including time, space, product lines owning the servers, operator's response, etc. We first explore the temporal and spatial distributions of failures in different components. Then we focus on correlated failures as people believe they affect software fault tolerance the most. Finally, we describe the operators' response to these failures.

To our best knowledge, this is the largest comprehensive hardware failure study focusing on commodity Internet data centers in a decade. We cover all major components as well as the human operator behaviors. During the study, we confirm or extend many counter-intuitive observations from previous studies, and observe many new patterns. For example,

1) Failures are not uniformly random at different time scales, and sometimes not even uniformly random at different spaces in a data center. We see many correlated failures. We even observe batch failures affecting thousands of servers in a few hours. These observations contradict software design assumptions of independent and uniformly random failures.

2) The time between failures (TBF), both at a data center scale and at an individual component scale, is hard to model with a well-known distribution. Different components exhibit highly distinctive failure patterns.

TABLE I. CATEGORIES OF FAILURE OPERATION TICKETS.

Failure trace	Handling decision	Percentage
D_fixing	Issue a repair order (RO)	70.3%
D_error	Not repair and set to decommission	28.0%
D_falsealarm	Mark as a false alarm	1.7%

3) Despite the fact that the traditional doctrine says that we should reduce mean time to recover (MTTR) to improve dependability, operators are reluctant to respond to hardware failures in many cases. Also, the operators spend less time debugging a failure to find the root cause, but are more likely to order a replacement for the component. They sometimes even leave the failures unhandled, if the server is out-of-warranty.

These observations are highly related to the new software design and workload in the data centers, as we will discuss in the paper. These failure patterns suggest not only better ways to develop new failure management systems but also calls for a new methodology for designing fault handling mechanisms.

The remainder of the paper is organized as follows: Section II introduces the dataset and our analysis methodology. Section III and IV discuss the temporal and spatial failure patterns for different components. Section V focus on the analysis of correlated failures. We highlight the operators' response to failures in Section VI. In Section VII we summarize our key findings and their implications to future dependability design. We review related studies in Section VIII and conclude in Section IX.

II. METHODOLOGY AND DATASETS

We conduct this study on a dataset containing all hardware *failure operation tickets (FOTs)* collected over the past four years from a major Internet service company that operates dozens of data centers with hundreds of thousands of servers. There are hundreds of thousands of FOTs in three categories, *D_fixing*, *D_error* and *D_falsealarm*. Table I shows definition and breakdown of each category. Operators do not repair failure in *D_error* mainly because the servers are out-of-warranty. The typical action to *D_fixing* is to issue a *repair order (RO)*. Note that a separate group of contractors handle the actual ROs, and our dataset does not contain the detail. As far as our operators' concern, issuing an RO closes an FOT.

We can see that over $1/4$ of the failures are in out-of-warranty hardware and thus are not handled at all. Operators leave the partially failed but operational servers in production and decommission the totally broken ones. We also notice that the false alarm rate is extremely low, showing the effectiveness (high precision) of hardware failure detection.

There are two sources for FOTs: programmatic failure detectors and human operators. Both sources enter FOTs into a central *failure management system (FMS)*. Figure 1 shows the simplified architecture of the FMS and the failure handling workflow.

FMS has agents on most of the hosts[1] that detect hardware component failures. A (logically) centralized server collects all FOTs from the agents for operators' review.

[1]There are still some old unmonitored servers, but the monitoring coverage has increased significantly during the four years.

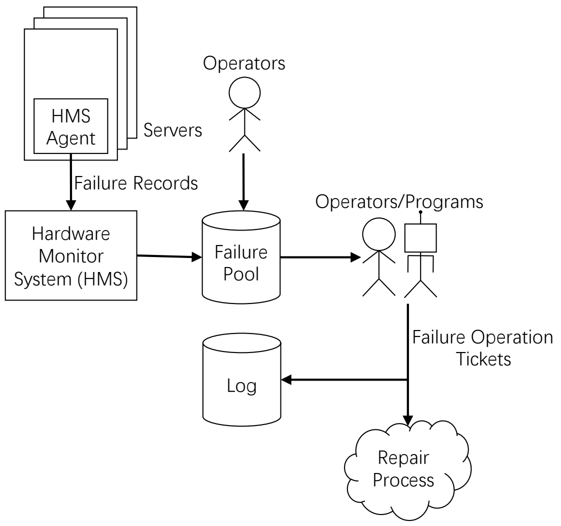


Fig. 1. Simplified failure management system (FMS) architecture.

FMS records over 70 types of failures covering nine component classes, including hard drive, SSD, RAID card, flash card, memory, motherboard, CPU, fan, and power supply. There is a special class, *miscellaneous*, covering all failures manually entered by human operators. These miscellaneous FOTs come with a natural language problem description, but most do not contain the root cause. They account for about 10% of the FOTs. FMS agents automatically detect the other 90% failures by two means: listening to syslogs and periodically polling device status and other metadata.

Each FOT contains the following self-explanatory fields describing the failure: *id*, *host_id*, *hostname*, *host_idc*, *error_device*, *error_type*, *error_time*, *error_position*, *error_detail*. The FOTs in *D_fixing* and *D_falsealarm* also contain fields describing the operators' responses, including the action taken (usually an RO), the operator's user ID, and the timestamp *op_time* of the action.

We need to point out that there are many definitions of hardware failures other than a fatal stop. Even within the company among different product lines, there is no agreement whether to consider temporarily or partially misbehaving hardware, such as a disk drive with SMART (Self-Monitoring Analysis And Reporting Technology) alerts or occasional read/write exceptions, as failed. Specifically, in this paper, we consider every FOT in *D_fixing* or *D_error* as a failure.

A. FOT overview

In this section, we provide an overview of the FOTs in our datasets. Table II shows the breakdown of these FOTs by component classes, in both *D_fixing* and *D_error* (i.e. excluding false alarms). Not surprisingly, hard drive failures are most common in all data centers we investigate, accounting for about 82% of all failures. Other components, such as memory, power supplies, SSDs and RAID cards, only contribute to about 8% combined. Even the fraction is small, there are still at least thousands of failures for each component class, enough to be statistically meaningful.

TABLE II. FAILURE PERCENTAGE BREAKDOWN BY COMPONENT.

Device	Proportion
HDD	81.84 %
Miscellaneous	10.20 %
Memory	3.06 %
Power	1.74 %
RAID card	1.23 %
Flash card	0.67 %
Motherboard	0.57 %
SSD	0.31 %
Fan	0.19 %
HDD backboard	0.14 %
CPU	0.04 %

TABLE III. EXAMPLES OF FAILURE TYPES.

Failure type	Explanation
SMARTFail	Some HDD SMART value exceeds the predefined threshold.
RaidPdPreErr	The prediction error count exceeds the predefined threshold.
Missing	Some device file could not be detected.
NotReady	Some device file could not be accessed.
PendingLBA	Failures are detected on the sectors that are not accessed.
TooMany	Large number of failed sectors are detected on the HDD.
DStatus	IO requests are not handled by the HDD and are in D status.
BBTFail	The bad block table (BBT) could not be accessed.
HighMaxBbRate	The max bad block rate exceeds the predefined threshold.
RaidVdNoBBU -CacheErr	Abnormal cache setting due to BBU (Battery Backup Unit) is detected, which degrades the performance.
DIMMCE	Large number of correctable errors are detected.
DIMMUE	Uncorrectable errors are detected on the memory.

About 10.2% FOTs are manually submitted miscellaneous failures. These failures are complicated. In fact, operators do not leave any description in 44% of these failures and suspect about 25% to be hard drive related. They mark another 25% as "server crashes" without clear reasons. As we will see later, operators generate many of these FOTs during the deployment phase, when they try to debug problems manually.

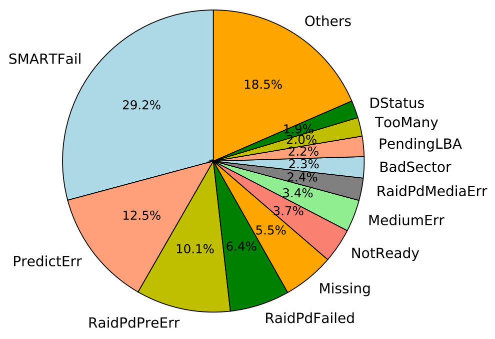
Our dataset contains many types of failures for each component class. Table III shows some examples. Figure 2 shows the percentage of each failure type for four representative component classes. Some failures are fatal (e.g. `NotReady` in a hard drive) while others warn about potential failures (e.g. `SMARTFail`).

B. Analytical methods

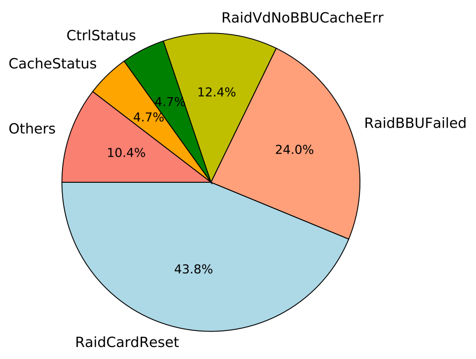
Visually, we characterize the temporal and spatial properties of hardware failures by plotting the probability density functions (PDF) or cumulative distribution functions (CDF).

Statistically, similar to previous work [5, 17], we conduct hypothesis tests to verify how well the observed distribution functions fit some well-known probability distributions, including uniform, exponential, Weibull gamma, and lognormal.

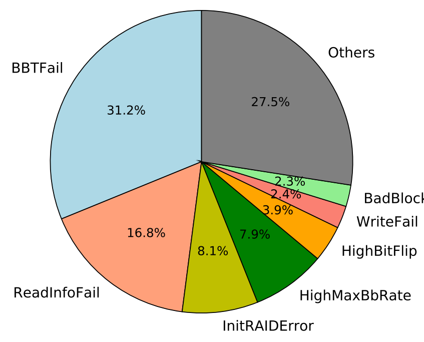
To see if a dataset fits a given distribution, we first estimate the parameters of the fitting distributions through *maximum likelihood estimation (MLE)* and then adopt *Pearson's chi-squared test*, a widely used hypothesis test on distributions of discrete random variables. For each null hypothesis we create on the distribution of a certain random variable, the test result is whether we can reject it at a certain significance level. The result implies whether the null hypothesis is a good characterization of the observations.



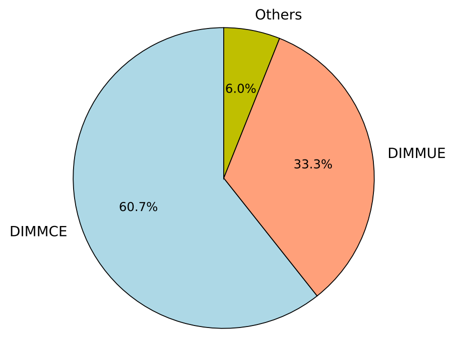
(a) HDD



(b) RAID card



(c) Flash card



(d) Memory

Fig. 2. Failure type breakdown of four example component classes.

III. Temporal Distribution of the Failures

The common belief is that components can fail at any time in a uniformly random way. In this section, we investigate the temporal distribution of the hardware failures. Focusing on the *error_time*, or the failure *detection* timestamp, in *D_fixing* and *D_error* (i.e. excluding false alarms), we analyze the number of failures in different time periods, the time between failures (TBF) for each component class and the failure probability during the life span of a single component.

A. Number of failures at different time periods.

Hypothesis 1. *The average number of component failures is uniformly random over different days of the week.*

Figure 3 shows the average number of failures during each day of the week. Due to limited space we only present the components with the most number of failures, and due to confidentiality concerns, we normalize the count to the total number of failures. It is obvious from the figure that the failure rates vary and are not uniformly distributed on each day in a week. More formally, a chi-square test can reject the hypothesis at 0.01 significance level for all component classes. Even if we exclude the weekends, a chi-square test still rejects the hypotheses at 0.02 significance level. The test indicates that failures do not occur uniformly randomly in each day of a week.

Hypothesis 2. *The average number of component failures is uniformly random during each hour of the day.*

Similarly, we calculate the number of failures during each hour in the day. Figure 4 shows eight component classes with the most number of failures. A similar chi-square test rejects the hypothesis at 0.01 significance for each class.

Possible Reasons. 1) The number of failures of some components are positively correlated with the workload. This is especially true for hard drive, memory and miscellaneous failures, as Figure 4 (a), (b) and (h) show. Such correlation between the failure rate and the workload is consistent with findings in [5, 22].

We want to emphasize that this correlation might not imply the causality that low workload reduces the probability of hardware failures. In fact, we believe that the higher utilization causes failures more likely to be detected asynchronously. For example, the agents detect hard drive and memory failures by monitoring specific log messages (e.g. *dmesg*) that are more likely to occur under heavy utilization.

This observation reveals the limitation of log-based failure detection - it does not detect failures in a component until it gets used. Also, detecting failures only when the workload is already heavy increases the performance impact of such failure. The failure management team is working on an active failure probing mechanism to solve the problem.

2) If failure reporting requires the human in the loop, the detection likely to happen during working days and regular working hours. This is true for most manually reported miscellaneous failures.

3) Some components tend to fail in large batches during a small period of time. For example, a large batch of failed RAID cards of the same model makes the distribution in Figure 3 (c) highly skewed, and we can also see many notable high spikes in almost all the plots in Figure 4. We discuss more about such batch failures in Section V-A.

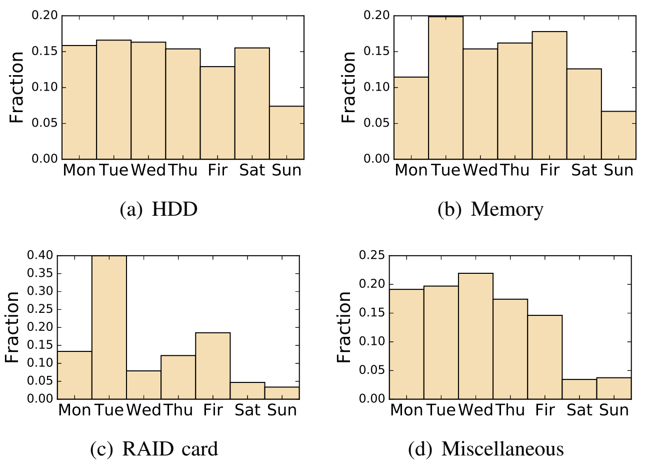


Fig. 3. The fraction of number of failures on each day of the week.

B. Time between failures (TBF)

In this section, we focus on the distribution of the time between failures (TBF) for each component class. The common belief is that the failure occurrences in a system follow a Poisson process, and thus people often model the TBF with an exponential distribution. However, previous studies show that the distribution of TBF for the hard drives or the HPC systems cannot be well characterized by exponential distribution [5, 17, 23]. We extend the result and show the TBF for each component class, as well as all components combined.

Hypothesis 3. *TBF of all components in the data centers follows an exponential distribution.*

We conduct the same chi-square test as previously described, and the chi-square test rejects the hypothesis at the 0.05 significance level. In fact, Figure 5 shows that none of the distributions including exponential, Weibull, gamma and lognormal fits the TBF data. The observation is different from some previous studies [5, 24–26], who report that the TBF of HPC and cloud (including hardware failures) can be well characterized by a Weibull distribution or a gamma distribution. We believe the disagreement is the result of the wide presence of batch failures, which makes the TBF distribution highly skewed, in the data centers we examine, which we discuss more below.

Hypothesis 4. *TBF of each individual component class follows an exponential distribution.*

We then break down the analysis to each component class. We also break down the failure by product lines. All the results are similar, that is, the hypotheses that the TBF follows exponential, Weibull, gamma or lognormal distributions can be rejected at the 0.05 significance level. We omit the figures here due to space limitation.

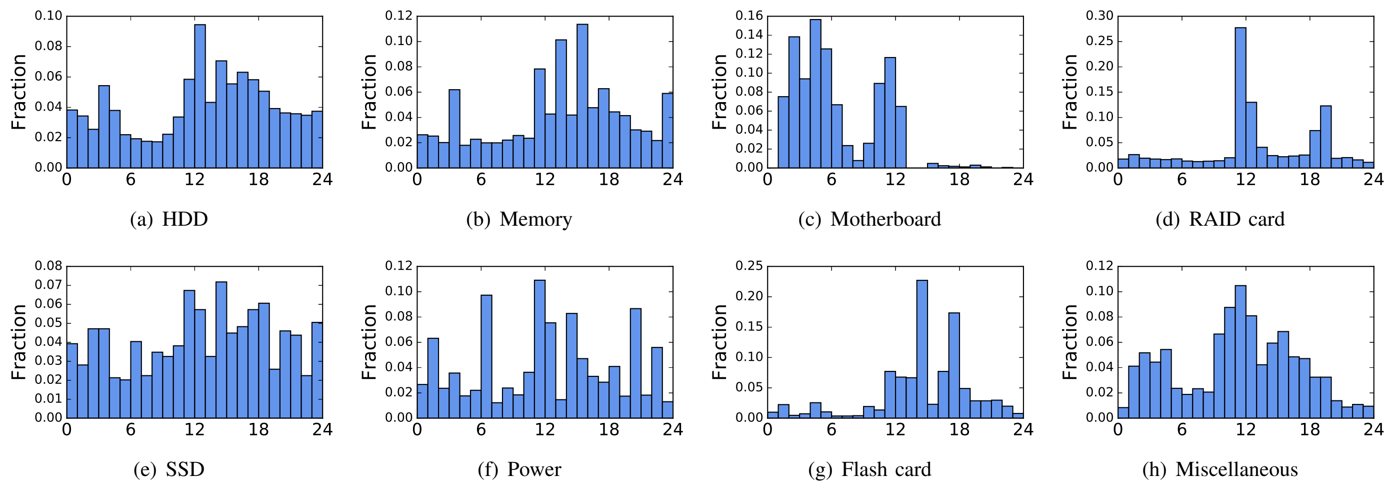


Fig. 4. The fraction of number of failures on each hour in a day. The values on the horizontal axis of each subfigure indicate the hours in a day.

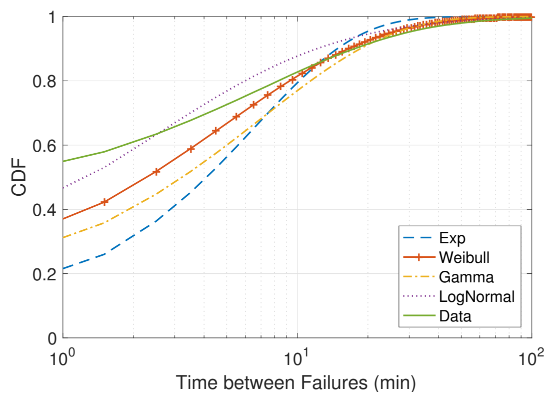


Fig. 5. CDF of TBF for all component failures, as well as some fitted distributions. The horizontal axis is on a logarithmic scale.

Possible Reasons. Two possible reasons cause the deviation of TBF from the exponential distribution.

1) As previous work has pointed out, the failure rates change over the lifetime of each component, or the entire system. This change affects the TBF over time in a way that the exponential distribution cannot capture [17]. We will take a closer look at the fact in Section III-C.

2) We observe lots of small values in TBFs, indicating that there are short time periods with many failure occurrences. The MTBF (mean time between failures) across all data centers we investigate (with hundreds of thousands of servers) is only 6.8 minutes, while the MTBF in different data centers varies between 32 minutes and 390 minutes. None of the distributions mentioned above capture these small TBF values, as Figure 5 shows. These small TBFs are related to the correlated failures especially batch failures we will discuss in Section V.

C. *Failure rate for a single component in its life cycle*

People commonly believe that the likelihood of a component failure is related to its life-in-service [5, 17]. Previous studies [17] show the probability of hard drive failure cannot be well characterized by a *bathtub curve* model [27], in which the failure rates are high both at the beginning ("infant mortality") and the end ("wear-out") of the lifecycle. In this section, we verify the claim on all the major component classes.

We consider the monthly *failure rate* (FR) for each component in its lifecycle[2]. We normalize all failure rates for confidentiality, and Figure 6 shows the change of failure rates of each component class during their first four years of service life. The main observations are as follows.

Infant mortalities. RAID cards obviously have high infant mortality rate (Figure 6 (f)). Of all the RAID cards that failed within fifty months of its service life, 47.4% of the failures happen in the first six months.

We observe some infant mortalities in hard drives during the first three months, with 20.0% higher failure rate than that of the 4th to 9th month (Figure 6 (a)). We also see that the failure rates start to increase only six months after deployment, and rise significantly over the following couple of years. This is consistent with previous studies [17] but differs from the "bathtub curve", where the failure rates stay low and stable for much longer (e.g. at least a year).

Interestingly, we observe that the miscellaneous failure rates (Figure 6 (i)) are extremely high within the first month. After the first month, the failure rates become relatively stable. The reason is that most manual detection and debugging efforts happen only at deployment time. During normal operation, operators often respond to component failures with a simple replacement order, without much manual effort to debug. The "lazy" actions reduce the number of miscellaneous FOTs for older components.

Failures in motherboards, flash cards, fans, and power supplies (Figure 6 (c)(e)(g)(h)) are rare in the early years of servers' lifecycles. This effect may be related to the quality assurance process during manufacturing. Also, at the deployment

[2] The dataset reports the number of HDDs, SSDs, and CPUs on each server, and thus for these three categories, we know the numbers of properly-working components during each interval, which are used to compute the failure rates. For other components, we assume that the component count per server is similar, and use the number of servers as an estimation.

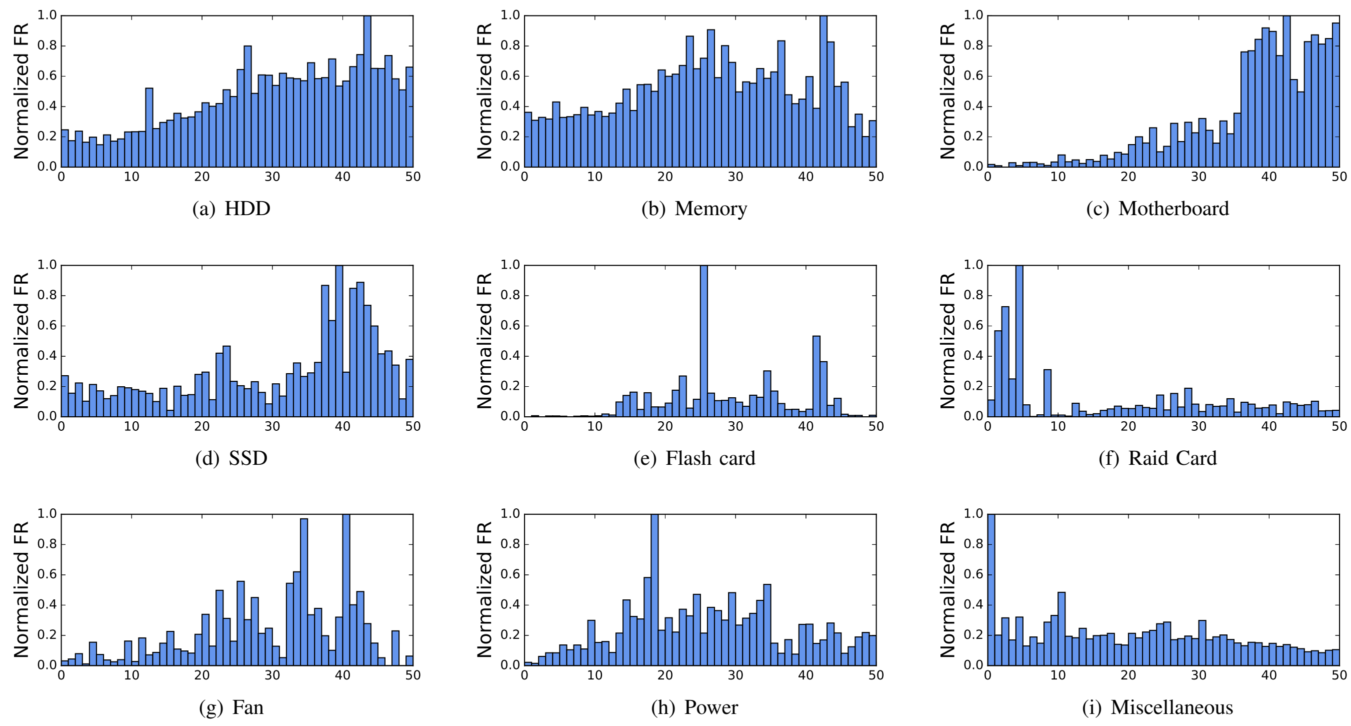


Fig. 6. Normalized monthly failure rates for each component class. The values on the horizontal axis of each subfigure indicate the time in production use (in months). Some components are omitted because the numbers of the samples are small.

and initial testing phase, operators may have already marked these failed components as miscellaneous.

Wear out. We observe an increase in failures in many component classes when they get older. The wear-out rate is different for these classes. For example, most (72.1%) of the motherboard failures occur three years after deployment.

Although only 1.4% of failures happen during the first 12 months for flash cards, failure rates rise fast after that, showing strong correlated wear-out phenomena. Memory failures show a similar pattern (Figure 6 (b)), though the failure rate is relatively stable during the first year, it gets higher starting between the 2nd and 4th year.

Mechanical components such as hard drives, fans, and power supplies (with fans too) (Figure 6 (a)(g)(h)) also exhibit strong, clear wear and tear pattern, as the failure rates are relatively small during the first year and gradually increases as servers get older.

D. Repeating failures and the effectiveness of repairs

We observe that a small number of server components fail repeatedly. We define *repeated failures* as cases when the problem marked "solved" (either by operators or an automatic reboot), but the same problem happens again afterward.

There are not many repeating failures. In fact, as operators "repair" a component usually by replacing the entire module, which is effective most of the time. Over 85% of the fixed components never repeat the same failure. We estimate that about 4.5% of all the servers that ever failed (thousands of servers) have suffered from repeating failures.

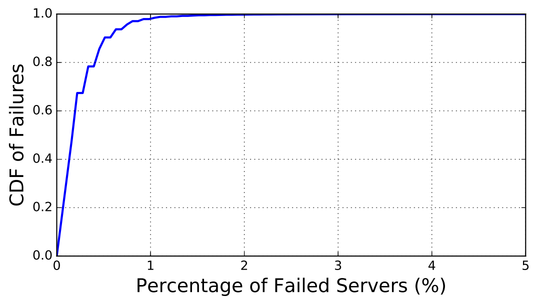


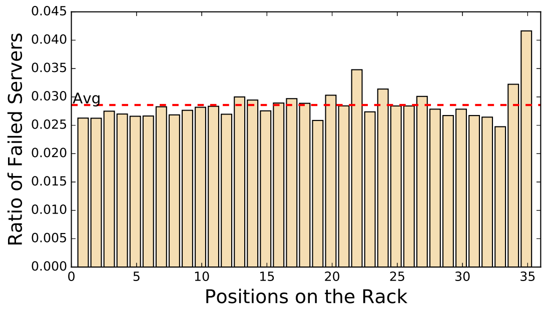
Fig. 7. CDF of the number of failures w.r.t. the percentage of the servers that ever failed.

However, surprisingly we observe that 2% of servers that ever failed contribute more than 99% of all failures. In other words, the failures are extremely non-uniformly distributed among the individual servers. Figure 7 shows the CDF of the number of failures with respect to the fraction of the servers that ever failed. This observation is consistent with findings in [7, 10, 22, 24].

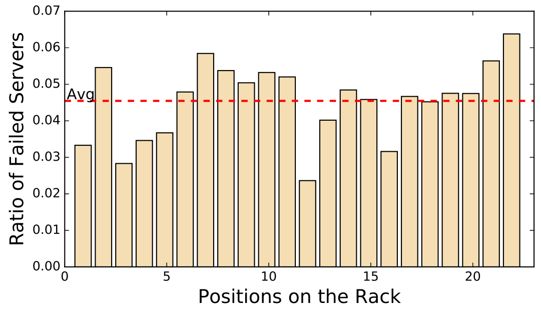
As an extreme example, we observe that one single server in a web service product line reports over 400 failures, either on RAID card or hard drives. The root cause is a BBU (Battery Backup Unit) failure causing the RAID card to be up-and-down. Each time, after an automatic recovery program reboots the server, the hard drive becomes online again, and thus the problem is marked "solved". However, it will fail again very soon. Without human intervention, the process has repeated for almost a year before someone finally realizes the root cause (the BBU) and solves the problem.

TABLE IV. CHI-SQUARE TEST RESULTS FOR HYPOTHESIS 5.

p-value	Ratio
$p < 0.01$	10 out of 24
$0.01 \leq p < 0.05$	4 out of 24
$p \geq 0.05$	10 out of 24



(a) Data center A



(b) Data center B

Fig. 8. The failure ratio at each rack position. (a) In data center A, Hypothesis 5 cannot be rejected by a chi-square test at 0.05 significance. (b) In data center B, Hypothesis 5 can be rejected at 0.01 significance.

As a more interesting observation, multiple servers can repeat the failures synchronously, causing strong failure correlations. We provide an example in Section V-C.

IV. SPATIAL DISTRIBUTION OF THE FAILURES

The common belief is that all physical locations in the same data center are identical, without much impact on server failures. We find out, however, the spatial location sometimes does affect the failure rate of servers, especially the relative position (i.e. slot number) on the rack.

We study 24 production data centers in our dataset and count the average number of failures at each rack position. We filter out repeating failures to minimize their impact on the statistics. Moreover, as not every rack position has the same number of servers (e.g. operators often leave the top of position and bottom position of the racks empty), we normalize the failure rates to the total number of servers at each rack position. In this statistics, we count a server failure if any of its components fail.

Hypothesis 5. *The failure rate on each rack position is independent of the rack position.*

Interestingly, different data centers show different chi-square test results. Table IV summarizes the results. In general, at 0.05 significance level, we can not reject the hypothesis in 40% of the data centers while we can reject it in the other 60%.

Figure 8 shows the failure ratio in two example data centers. In data center B, we can reject the hypothesis with high confidence, while in data center A, we cannot.

Possible Reasons. One possible reason is the design of data center cooling and the physical structure of the racks. While the focus of the paper is not on the relation between the temperature and failure, we have an interesting observation. For both data center A and B, We can observe notable spikes at certain rack positions. Even though Hypothesis 5 cannot be rejected for data A, a further anomaly detection reveals that the FRs at rack position 22 and 35 are singularly high in data center A. Specifically, assume the failures occur on each rack position independently and uniformly randomly, according to central limit theorem, the FR on each rack position should follow a normal distribution with small variance as the number of failures gets large. We estimate the expectation μ and the variation σ^2 of the FR at each rack position and discover that the FRs of rack positions 22 and 35 in data center A lie out of the range $(\mu - 2\sigma, \mu + 2\sigma)$.

In fact, position 35 is close to the top of a rack. With the under-floor cooling design, it is the last position cooling air reaches. Position 22 is next to a rack-level power module in the custom rack design. Our motherboard temperature readings at these places are indeed several degrees higher than the average motherboard temperature in each rack. This higher temperature might result in higher failure rate at these two positions.

The data centers host multiple generations of servers partitioned to hundreds of product lines. Each data center has distinct building architecture too. With these many uncertain variables, we cannot provide a universal explanation of the uneven spatial distributions of failures. However, from the data centers we investigate, we find that in around 90% data centers built after 2014, Hypothesis 5 cannot be rejected at 0.02 significance level. That is, the hardware failures are more uniformly distributed on each rack position, probably because the new data centers have a better cooling design, making the inside environment more consistent across all rack positions.

V. CORRELATED FAILURES

Correlated failures are the least desirable, as hardware diagnostics and software fault tolerance usually assume independent failures. In this section, we take a closer look at correlated failures. We see two types of correlations.

- *Batch failures*, i.e. a large group of servers reporting the same failure at the same time;
- *Correlated component failures*, i.e. multiple components on a single server failing at the same time.

A. Batch failures

Different from the common belief that servers fail independently, we see cases where many servers fail in a batch, especially those servers with the same model, in the same

TABLE V. Batch failure frequency for each component class.

Device	$\mathbf{r}_{\mathbf{100}}(\%)$	$\mathbf{r}_{\mathbf{200}}(\%)$	$\mathbf{r}_{\mathbf{500}}(\%)$
HDD	55.4	22.5	2.5
Miscellaneous	3.7	1.3	0.1
Power	0.7	0.4	0
Memory	0.4	0.4	0.1
RAID card	0.4	0.2	0.1
Flash card	0.1	0.1	0
Fan	0.1	0	0
Motherboard	0	0	0
SSD	0	0	0
CPU	0	0	0

cluster and serve the same product line. Similar phenomenon was also observed by previous study [28].

Different operators and product lines have different definitions and tolerance of *batch failures*. In general, batch failures refer to a number of servers (above a threshold N) failing during a short period of time t. Both N and t are user-specific. A product tolerates a larger batch of failures with better software fault tolerance. Also, if the operators fix problems quickly, we can expect fewer batch failures.

We define a metric r_N to describe the relative frequency of batch failures. Let n_k be the number of failures of a component class on the k-th day ($k = 1 \dots D$, where D is the total number of days we examine). We informally define the batch failure frequency r_N as $r_N = (\sum_k I\{n_k \geq N\})/D$, where N is the threshold, and I is an indicator random variable. Intuitively, r_N is a normalized counter of how many days during the D days, in which more than N failures happen on the same day, and we normalize the count by the total time length D.

We calculate r_N for each type of components, and Table V shows the results with N = 100, 200 and 500. We find that batch hard drive failures are common. During 2.48% of the days (35 out of 1,411 days), we observe over 500 hard drive failures. We also observe batch failures in components such as memory, power supplies, RAID cards, flash cards and fans.

Examples of batch failures and possible reasons.

Here we present three batch failure cases associated with a single large product line owning tens of thousands servers in a single data center. These servers are incrementally deployed during the past three to four years, with five different generations. Most of the servers run batch data processing jobs (e.g. Hadoop jobs). We describe three batch failure cases in 2015.

Case 1: On Nov. 16th and 17th, there were thousands of servers, or 32% of all the servers of the product line, reporting hard drive `SMARTFail` failures, and 99% of these failures were detected between 21:00 on the 16th and 3:00 on the 17th. The operators ended up replacing about 28% of these hard drives and decommissioned the remaining 70%+ out-of-warranty drives. The reason of the batch failure is not clear yet.

Case 2: On Jun 4th, there were nearly 50 motherboards failing between 5:00 and 6:00, or between 16:00 and 17:00. The operators indicated that faulty SAS (Serial Attached SCSI) cards caused all these failures. They decided to decommission all these servers because they were out-of-warranty.

TABLE VI. Number of correlated component failures.

	HDD	SSD	Memory	Flash	RAID	Power	Fan	Motherboard
Misc.	349	2	18	2	4	6	-	6
Mother.	17	-	2	-	-	1	-	
Fan	1	-	-	-	-	7		
Power	3	-	-	-	-			
RAID	46	-	-	-				
Flash	22	-	-					
Memory	40	-						
SSD	15							

TABLE VII. Examples of correlated component failures.

Server ID	Partial FOTs
Server A	Fan fan_8 2016-01-22 06:35:35 Power psu_2 2016-01-22 06:36:51
Server B	Fan fan_3 2016-01-22 06:35:34 Power psu_2 2016-01-22 06:36:59

Batch failures in cases 1 and 2 may be related to unexpected homogeneity in these components and their operating environments. For example, components with the same model and same firmware version may contain the same design flaws or bugs that are triggered by the same condition. Also, as they are in the same data center, the long-term effects of the environment such as humidity, temperature, and vibration may be homogeneous, leading to simultaneous failures.

Case 3: On May 16th, nearly 100 servers experienced power failure between 1:00 and 13:00. This is a typical case as these servers used a single power distribution unit that caused the batch failure.

Case 3 represents a common design flaw of hidden single point of failure. The single dependency situation is common in data centers, especially for networking, power, and cooling.

Besides the cases mentioned above, we also observe batch failures caused by human operator mistakes. For example, a misoperation of the electricity provider caused a power outage on a PDU in a data center, resulting in hundreds of failed servers in August of 2016.

B. Correlated component failures

Correlated component failures rarely happen but often lead to confusing failure situations. We define *correlated component failures* as failures occurring on multiple components on the same server within a single day. These failures are unlikely to be coincidental: given the overall failure rate, we can calculate that the chance of two independent failures happening on the same server on the same day, which is less than 5%. In our dataset, these failures involve at most two components and experienced by only 0.49% of all servers that ever failed. In addition, 71.5% of these two-component failures have a miscellaneous failure report, indicating that some failures detected by the FMS are also noticed by the operators, who decide to report them immediately. Hard drive failures are related to nearly all the rest of two-component failures. Table VI shows the number of different correlated failure pairs.

Possible Reasons. We believe the primary reason for the component correlation is that one failure causes the other. For example, Table VII shows two cases of correlated failures of power and fan, both of which occurred on servers in the same

TABLE VIII. AN EAMPLE OF SYNCHRONOUSLY REPEATING FAILURES

Server C	Server D
SMARTFail sdh8 14-08-31 16:30:24	SMARTFail sdd4 14-08-31 16:26:15
SMARTFail sdh8 14-09-04 10:36:05	SMARTFail sdd4 14-09-04 10:30:13
SixthFixing sda1 14-09-05 13:42:06	SixthFixing sda1 14-09-05 13:42:09
SixthFixing sda1 14-09-05 17:34:02	SixthFixing sda1 14-09-05 17:34:03
SixthFixing sda1 14-09-09 17:12:05	SixthFixing sda1 14-09-09 17:12:04
SixthFixing sda1 14-09-16 16:32:10	SixthFixing sda1 14-09-16 16:32:21
PendingLBA sde5 15-07-21 12:38:34	

PSU (power supply unit) of the same data center on the same day. We believe that the power failure causes the fan problem.

Correlated component failures complicate failure detection and repair, especially automatic error handling. In the example above, we should not replace the fans even if they are reporting errors, while in other (common) cases, the operator should replace any alarming fans.

C. *Repeating synchronous failures*

Failures on some small groups of servers can appear highly correlated, as they repeat at the same time for many times. It is related to the repeating failure cases in Section III-D, mostly due to ineffective repair operations.

Table VIII shows an example. These two servers are almost identical: same product line, same model, same deployment time and located in adjacent racks, running the same distributed storage system. We see that their (repeating) disk failures occur almost synchronously for many times, which is obviously not a coincidence.

These synchronous failures may cause trouble for software fault tolerance, as most fail-over and recovery mechanisms do not expect such failure pattern. The best way to avoid these synchronous failures is to improve the repair effectiveness, making sure that the faulty components are either repaired or removed from production. However, as we will see in the next section, it is not yet the case.

VI. OPERATORS' RESPONSE TO FAILURES

Previous studies point out that shortening mean time to recover (MTTR) improves overall system dependability [1, 29]. An important component in MTTR is the operator's response, diagnosis and the initiation of following repairs. In this section, we analyze these response times.

We focus on the FOTs in *D_fixing* and *D_falsealarm* only and ignore those out-of-repair cases. We define the operators' *response time (RT)* for each FOT as $RT = op\_time - err\_time$, where err_time is the failure detection time, and op_time is the time when the operator closes the FOT (i.e. initiates an RO or marks it as not_fixing).

We notice that many factors, such as time, component class and product line all affect RT significantly, leading to high variations. The remainder of the section summarizes our key observations.

A. *RT is very high in general*

Figure 9 shows the CDF of RT across all FOTs. We can see that there are many extremely long responses. E.g., 10% of

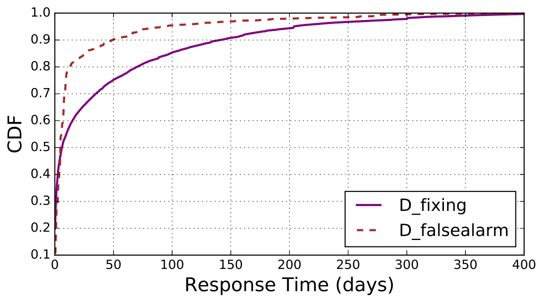


Fig. 9. The CDF of RT in *D_fixing* and *D_falsealarm*.

the FOTs has RTs of longer than 140 days and 2% even longer than 200 days. Surprisingly, operators do not actually abandon these FOTs, but eventually initiate an RO. These extremely long RTs significantly impact MTTR. The MTTR reaches 42.2 days and 19.1 days respectively for *D_fixing* and *D_falsealarm*, comparing to the median of repair time of only 6.1 days and 4.9 days. In contrast, MTTR that previous studies found in large-scale systems [5, 24] is much shorter.

Possible Reasons. We do not believe the prolonged RT is because the operators are incapable or too lazy to do the job. In fact, modern data center software and hardware design may cause the change.

Though the common belief is that hardware dependability determines the software redundancy design, we find that the other way around is also true. Thanks to the highly resilient software, operators know that even if the hardware has failed or is prone to imminent failure, it will not be a catastrophe. Thus the operators are less motivated to respond to the failures. For example, in some product lines, operators only periodically review the failure records in the failure pool (Figure 1) and process them in batches to save time. We will provide more details in the later part of this section.

Secondly, with early warnings and resilient hardware design, many hardware failures are no longer urgent. For example, SMART reports on hard drives warn about occasional and transient faults. The faults may be early warnings of fatal failures, but the failure is not imminent.

On the other hand, for those software systems without good fault handling, the repair operation can sometimes be costly and may bring extra down time. This is because to reduce hardware cost, current data center servers no longer support hot-swap on memory, CPUs or PCIe cards. Thus, it is likely that the operators have to shut down the entire server to replace the faulty component. Without proper software fault tolerance, the repair might require some planning such as manually migrating tasks out of the server, delaying the response.

B. *RT for each component class*

In this section, we want to see whether FOTs involving easy-to-repair component get quick responses.

Figure 10 shows the cumulative distribution of RT for each component class covering all FOTs in the dataset. We see that the median RTs of SSD and miscellaneous failures are the

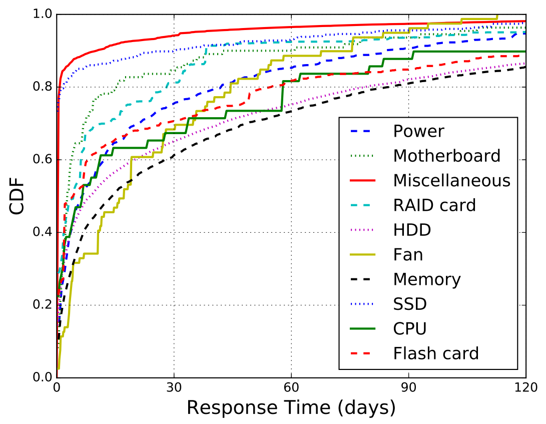


Fig. 10. The CDF of RT for each component class.

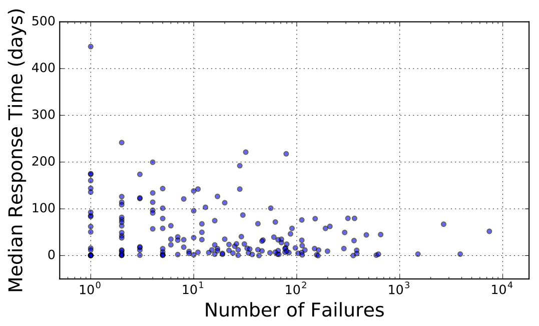


Fig. 11. The relation between the median RT to HDD failures and the number of HDD failures occurred on some randomly sampled product lines during the year 2015. Each point $P(x, y)$ indicates that the product line that corresponds to P encountered x HDD failures during the year 2015, and the median RT to the failures is y days. Note that the horizontal axis is on a logarithm scale.

shortest (several hours), while those of hard drives, fans, and memory are the longest (7 to 18 days).

We find that there are both technical and business reasons for the vastly different RT for SSD and hard drives. Technically, the fault tolerance in hard drives (both hardware and software) are the most mature, making these failures less urgent. In comparison, the redundancy in SSDs is less due to higher SSD cost. Also, business-wise, only crucial and user-facing online service product lines afford SSDs, and they have more strict operation guidelines.

The short response time for miscellaneous failures is artificial. Miscellaneous failures are manually entered, and they happen mostly when the hardware get deployed. During the deployment period, operators streamline installation, testing and debugging of new servers, effectively reducing the RTs. Also fixing new servers do not need to take down jobs, another convenience factor for quick response.

C. *RT in different product lines*

Although the company centrally monitors all servers, it partitions the servers to different product lines. Each product line has its own operator team too. We want to see whether these product lines have similar RT.

For the same type of failure, e.g. hard drive failures, we find that different product lines can have a big difference in RT. The standard deviation of the RT is as high as 30.2 days across these product lines.

Again, the variation is highly correlated with the level of fault tolerance in the product. We see that product lines with better software fault tolerance tend to have longer RT on failures. For example, RT is often large for most product lines operating large-scale Hadoop clusters.

We suspect that operators delay some responses because they are too busy dealing with other failures, but our observations show little evidence. We observe that the median RT does not grow in proportionality with the number of failures. In fact, it is just the opposite: the top 1% product lines with most failures have a median RT of 47 days. Out of the product lines with fewer than 100 failures, 21% of them have a median RT exceeding 100 days. Figure 11 shows the median RT for all hard drive FOTs across some randomly sampled product lines during a 12-month period.

When interviewed, most of the operators believe that given their product design, the hardware failures no longer matter much – even the redundancy level is restored automatically in the modern distributed systems soon after a failure. This is especially true for product lines with large-scale Hadoop clusters. Also, operators want to batch up similar failures to improve efficiency.

VII. Discussions

During the study, we find that much old wisdom to improve system dependability still holds. For example,

1) Most failures are independent, but the software needs to handle occasional correlated or repeating failures on single or multiple components. Given that these correlated failures are more common than before, we need to improve software resilience accordingly.

2) Automatic hardware failure detection and handling can be very accurate, significantly reducing human labor. However, due to the existence of multi-component failures, we need to improve these systems to cover the new failure types.

3) In data center design, we need to avoid "bad spots" where the failure rate is higher, or at least avoid allocating all replicas of a service in these vulnerable zones.

In addition, we discuss in more detail about two new important findings and their implications. We also summarize the limitations of this trace-based study in this section.

A. *Is hardware reliability still relevant in data centers?*

The advancement of software redundancy and automatic failure handling significantly improve the end-to-end dependability for Internet services. MTTR for hardware components seems less important than before, given the observations in operators' RT for failures.

There is a quite big team in the company managing data center hardware. This team designed some software tools, algorithms and operation guidelines to improve hardware dependability. The monitoring system is quite effective, with very little false positives (Table I). They even designed a tool to predict component failures a couple of days early, hoping the operators to react before the failure actually happens. However, it looks like the operators are less motivated to handle failures than we expect.

It is surprising to see that operators ignore not only predictive warnings but also leave failed component unhandled for days or even months. Operators also choose to leave some partially failed out-of-warranty servers in the service too.

Delayed repair brings extra cost in many aspects. First, tolerating more failures lead to software complexity and development cost. Although open source solutions often offset this cost, they still come with high management overhead. Second, hardware failures reduce the overall *capacity* of the system.

Even worse, unhandled hardware failures add up and eventually appear to software as batch or synchronous failures, as Section V-C describes. Thus software developers have to design more aggressive methods for fault tolerance. This process is a downward slope towards the undesirable situation when people use unnecessarily complex software to handle hardware failure models that we can avoid easily at the hardware level.

We believe it is time to start considering failure handling as a joint optimization effort across software, hardware and operation layers, to lower the total cost of ownership of the data centers. In other words, data center dependability should evolve to the next level, where we not only focus on reliability metrics of a single component or a single layer but also consider how to achieve these metrics at low total cost.

B. Problems with the "stateless" failure handling system

In our FMS, the failure records present themselves as individual FOTs, including both diagnosis and operator responses. All closed FOTs become the archived log entries. This stateless design makes FMS simple and scalable. However, it becomes difficult to reveal the connections among different FOTs.

In fact, many FOTs are strongly connected – there are repeating or batch failures. The correlation information is lost in FMS, and thus operators have to treat each FOT independently. Of course, a single operator or a small operator team might learn from past failures, but given the annual operator turnover rate of over 50%, it is hard to accumulate actual operation experience and tools, except for a wiki-like documentation system that is difficult to maintain.

Thus, we believe it may be useful to build a data mining tool to discover the correlations among these failures. More importantly, we need to provide operators with related information about an FOT, such as the history of the component, the server, its environment, and the workload. This extra information can help operators reduce the number of repeating failures effectively. Considering the advances in data science techniques, we believe such system has become feasible.

C. Limitations of this study

Like all data-driven study, this work has its intrinsic limitations. First, although the dataset covers most of the failures, there are missing data points. Also, people incrementally rolled out FMS during the four years, and thus the actual coverage might vary over the four years and in different data centers.

Also, lacking matching workload and other detailed monitoring data, we can only describe the *statistical* observations and make educated *guesses* about the possible reasons. We made our best effort confirming our guesses with the operators, but there are still unconfirmed cases as the operators do not always remember the details.

Finally, the dataset comes from a single company, and may not represent the entire industry. However, given that the failures are from different generations of hardware, happen in dozens of distinct data centers (owned or leased, large or small), and are separately managed by hundreds of product lines running different workload, we believe they are good representations of state-of-the-art Internet data centers.

VIII. Related Work

Much previous work has studied characteristics of system failures extensively to understand the failure properties, which is crucial to highly reliable large-scale systems design. Most of these studies focus on high performance computers (HPCs) [5–15, 26]. They mainly focus on the composition of failure types, the temporal and spatial properties of failures, the statistical properties of time between failures and repair times, as well as correlated failures. However, as we have seen in the paper, the failure model in data centers is quite a difference from HPCs, due to the heterogeneity in hardware and workload.

Not as many recent studies focus on failures in commercial data centers. Vishwanath *et al.* [30] analyzed a 14-month slice in time of hardware repair log for over 100,000 servers. Ford *et al.* [28] analyzed the availability properties including correlated failure properties of Google storage clusters with tens of thousands of servers during a period of one year. Garraghan *et al.* [24] analyzed Google Cloud trace log consisting event logs of 12,500 servers over a period of 29 days. Birk *et al.* [25] analyzed both physical and virtual machine crash tickets from five commercial data centers consisting about 10K servers during a period of one year. Our work uses a much larger dataset with more component classes than the hard drive. We also observe more correlated failures and much longer MTTR in some of our product lines.

There are also many studies focusing on the failure characteristics on hardware component level, such as disks [16, 17, 27], memory [18–20], SSDs [31, 32], and GPUs [11, 21]. We use many similar analytic methods, such as hypothesis tests and various distribution assumptions from these work. We also borrow some metrics describing component failure properties, such as the metrics describing failure types, temporal distributions, TBF, and operator response time. In addition to using a very recent large-scale failure dataset, we focus on the data center as a whole, analyzing correlation among failures, batch failures, and repeating failures, making our study distinct from existing ones.

While much research analyzes the MTTR of failures in large-scale systems, we particularly focus on the operators' response component in MTTR and discovering the human factors of failure handling.

IX. Conclusion

Hardware failure study is a decades-old topic. However, as both researchers and data center practitioners, we answer the following two questions: 1) how do failure patterns evolve, given the significant technology, economy and demand changes in the Internet service data centers recently? 2) given that we can get advanced software fault tolerance for free from open source community, is hardware reliability still relevant in data center operations?

We statistically analyze hundreds of thousands of failure tickets from dozens of production data centers, analyzing the failure patterns across time, space, component and product lines. We analyze all main components instead of a single component class, and we focus on correlations among different failures instead of individual failures. Our study reconfirms or extends many findings from previous work, and also observes many new patterns in failure models and operators' behavior.

Our work provides a fresh and deeper understanding of the failure patterns in modern data centers. The understanding not only helps to operate the data centers better but also calls for a joint effort in software and hardware fault tolerance mechanism, which minimizes the overall cost.

Acknowledgment

We would like to thank our colleagues at Baidu Inc. Yang Wang for providing the datasets, Wei Wang, Xingxing Liu, Kai Lang, and Qian Qiu for helpful comments on the failure analysis. We also thank the students of Tsinghua University Yong Xiang, Xin Liu, Hao Xue, Cheng Yang, Han Shen, Yiran Li, and Yi Li for helpful comments on the drafts of this paper.

This research is supported in part by the National Natural Science Foundation of China (NSFC) grant 61532001, Tsinghua Initiative Research Program Grant 20151080475, MOE Online Education Research Center (Quantong Fund) grant 2017ZD203, and gift funds from Huawei and Ant Financial.

References

[1] L. A. Barroso, J. Clidaras, and U. Hölzle, "The datacenter as a computer: An introduction to the design of warehouse-scale machines," *Synthesis lectures on computer architecture*, vol. 8, no. 3, pp. 1–154, 2013.

[2] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, "Why do Internet services fail, and what can be done about it?" in *USENIX symposium on internet technologies and systems*, vol. 67. Seattle, WA, 2003.

[3] D. A. Patterson *et al.*, "A simple way to estimate the cost of downtime." in *LISA*, vol. 2, 2002, pp. 185–188.

[4] J. Gray, "Why do computers stop and what can be done about it?" in *Symposium on reliability in distributed software and database systems*. Los Angeles, CA, USA, 1986, pp. 3–12.

[5] 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, 2010.

[6] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *International Conference on Dependable Systems and Networks (DSN'07)*. IEEE, 2007.

[7] N. El-Sayed and B. Schroeder, "Reading between the lines of failure logs: Understanding how HPC systems fail," in *International Conference on Dependable Systems and Networks (DSN'13)*. IEEE, 2013, pp. 1–12.

[8] S. Ghiasvand, F. M. Ciorba, R. Tsch, W. E. Nagel *et al.*, "Lessons learned from spatial and temporal correlation of node failures in high performance computers," in *PDP 2016*. IEEE, 2016.

[9] C. Di Martino, Z. Kalbarczyk, R. K. Iyer, F. Baccanico, J. Fullop, and W. Kramer, "Lessons learned from the analysis of system failures at petascale: The case of blue waters," in *International Conference on Dependable Systems and Networks (DSN'14)*. IEEE, 2014.

[10] Y. Liang, Y. Zhang, A. Sivasubramaniam, M. Jette, and R. Sahoo, "Bluegene/L failure analysis and prediction models," in *International Conference on Dependable Systems and Networks (DSN'06)*. IEEE, 2006, pp. 425–434.

[11] D. Tiwari, S. Gupta, J. Rogers, D. Maxwell, P. Rech, S. Vazhkudai, D. Oliveira, D. Londo, N. DeBardeleben, P. Navaux *et al.*, "Understanding GPU errors on large-scale HPC systems and the implications for system design and operation," in *HPCA 2015*. IEEE, 2015.

[12] Y. Liang, Y. Zhang, A. Sivasubramaniam, R. K. Sahoo, J. Moreira, and M. Gupta, "Filtering failure logs for a BlueGene/L prototype," in *International Conference on Dependable Systems and Networks (DSN'05)*.

[13] A. Pecchia, D. Cotroneo, Z. Kalbarczyk, and R. K. Iyer, "Improving log-based field failure data analysis of multi-node computing systems," in *International Conference on Dependable Systems and Networks (DSN'11)*. IEEE, 2011, pp. 97–108.

[14] S. Gupta, D. Tiwari, C. Jantzi, J. Rogers, and D. Maxwell, "Understanding and exploiting spatial properties of system failures on extreme-scale HPC systems," in *International Conference on Dependable Systems and Networks (DSN'15)*. IEEE, 2015.

[15] B. Schroeder and G. A. Gibson, "Understanding failures in petascale computers," in *Journal of Physics: Conference Series*, vol. 78, no. 1. IOP Publishing, 2007, p. 012022.

[16] E. Pinheiro, W.-D. Weber, and L. A. Barroso, "Failure trends in a large disk drive population." in *FAST*, 2007.

[17] B. Schroeder and G. A. Gibson, "Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?" in *FAST*, 2007.

[18] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM errors in the wild: a large-scale field study," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 1. ACM, 2009, pp. 193–204.

[19] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, "Revisiting memory errors in large-scale production data centers: Analysis and modeling of new trends from the field," in *International Conference on Dependable Systems and Networks (DSN'15)*. IEEE, 2015.

[20] L. Bautista-Gomez, F. Zyulkyarov, O. Unsal, and S. Mcintosh-Smith, "Unprotected computing: a large-scale study of DRAM raw error rate on a supercomputer," in *International Conference for High PERFORMANCE Computing, Networking, Storage and Analysis*, 2016.

[21] 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 *HPCA 2016*. IEEE, 2016.

[22] R. K. Sahoo, M. S. Squillante, A. Sivasubramaniam, and Y. Zhang, "Failure data analysis of a large-scale heterogeneous server environment," in *International Conference on Dependable Systems and Networks (DSN'04)*. IEEE, 2004.

[23] T. N. Minh and G. Pierre, "Failure analysis and modeling in large multi-site infrastructures," in *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 2013, pp. 127–140.

[24] P. Garraghan, P. Townend, and J. Xu, "An empirical failure-analysis of a large-scale cloud computing environment," in *HASE 2015*. IEEE, 2014.

[25] R. Birke, I. Giurgiu, L. Y. Chen, D. Wiesmann, and T. Engbersen, "Failure analysis of virtual and physical machines: patterns, causes and characteristics," in *International Conference on Dependable Systems and Networks (DSN'14)*. IEEE, 2014.

[26] D. Tiwari, S. Gupta, and S. S. Vazhkudai, "Lazy checkpointing: Exploiting temporal locality in failures to mitigate checkpointing overheads on extreme-scale systems," in *International Conference on Dependable Systems and Networks (DSN'14)*. IEEE, 2014.

[27] J. Yang and F. B. Sun, "A comprehensive review of hard-disk drive reliability," in *Reliability and Maintainability Symposium, 1999. Proceedings*.

[28] D. Ford, F. Labelle, F. I. Popovici, M. Stokely, V. A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," in *Usenix Symposium on Operating Systems Design and Implementation, OSDI 2010*.

[29] A. Fox, "Toward recovery-oriented computing," in *VLDB 2002*. VLDB Endowment, 2002.

[30] K. V. Vishwanath and N. Nagappan, "Characterizing cloud computing hardware reliability." in *ACM Symposium on Cloud Computing*, 2010.

[31] J. Meza, Q. Wu, S. Kumar, and O. Mutlu, "A large-scale study of flash memory failures in the field," *ACM Sigmetrics Performance Evaluation Review*, vol. 43, no. 1, pp. 177–190, 2015.

[32] I. Narayanan, D. Wang, M. Jeon, B. Sharma, L. Caulfield, A. Sivasubramaniam, B. Cutler, J. Liu, B. Khessib, and K. Vaid, "SSD failures in datacenters: What, when and why?" *ACM Sigmetrics Performance Evaluation Review*, vol. 44, no. 1, pp. 407–408, 2016.