Why Does the Cloud Stop Computing? Lessons from Hundreds of Service Outages

Haryadi S. Gunawi, Mingzhe Hao, and Riza O. Suminto
University of Chicago

Agung Laksono, Anang D. Satria, Jeffry Adityatama, and Kurnia J. Eliazar
Surya University

Abstract

We conducted a cloud outage study (COS) of 32 popular Internet services. We analyzed 1247 headline news and public post-mortem reports that detail 597 unplanned outages that occurred within a 7-year span from 2009 to 2015. We analyzed outage duration, root causes, impacts, and fix procedures. This study reveals the broader availability landscape of modern cloud services and provides answers to why outages still take place even with pervasive redundancies.

Categories and Subject Descriptors C.4 [*Computer Systems Organization*]: Performance of Systems: Reliability, Availability, Serviceability

1. Introduction

Cloud computing, "the practice of using a network of remote servers hosted on the Internet to store, manage, and process data, rather than a personal computer" [21], has fundamentally changed the way society performs daily businesses and social activities. Emails, text and video chats, picture and video sharing, blogs and news, are all backed by a large complex collection of Internet services, which we refer as "the Cloud".

As dependency on cloud computing increases, society demands high availability, an ideal 24/7 service uptime if possible. Yet, service outages are hard to escape from. It has become a new year's tradition that news websites report the worst cloud outages in the previous year [150, 151, 156]. Exacerbating the impact of an outage is the fact that many of today's cloud services are built on top of other services (*e.g.*, SaaS on IaaS). As a ramification, an outage can easily cripple down a large number of other services [19, 117, 118], hence a larger number of furious and frustrated users.

SoCC '16, October 05 - 07, 2016, Santa Clara, CA, USA.

ISBN 978-1-4503-4525-5/16/10. . . $15.00.
DOI: http://dx.doi.org/10.1145/2987550.2987583

Not only do outages hurt customers, they also cause financial and reputation damages. Minutes of service downtimes can create hundreds of thousands of dollar, if not multi-million, of loss in revenue [29, 36, 89]. Company's stock can plummet after an outage [111]. Sometimes, refunds must be given to customers as a form of apology [118]. As rivals always seek to capitalize an outage [2], millions of users can switch to another competitor, a company's worst nightmare [62].

There is a large body of work that analyzes the anatomy of large-scale failures (*e.g.*, root causes, impacts, time to recovery). Some work focus on *specific component failures* such as server machines [158], virtual machines [126], network components [134, 157], storage subsystems [132, 141], software bugs [137] and job failures [129, 133, 146]. Another set of work perform a *broader* failure analysis but only do so for *specific systems/services* such as HPC systems [131, 147, 153], IaaS clouds [125], data mining services [161], Internet portals and hosting services [149].

Although without a doubt the studies above are extremely valuable in providing deep insights into failures in large-scale systems, they do not necessarily paint *the larger landscape of hundreds of outages experienced by tens of popular services in recent years*. Existing work, as listed above, either performs a focused analysis of specific components (*e.g.*, storage, network) or a broader analysis of few cluster or service types (more detailed comparisons in Section 8). Therefore, there are many interesting questions left unanswered: How often and how long do outages typically happen and last across a wide range of Internet services? How many services do not reach 99% (or 99.9%) availability? Do outages happen more in mature or young services? What are the common root causes that plague a wide range of service deployments? What are the common lessons that can be gained from various outages?

Fortunately, such a broader study is feasible today, thanks to the era of providers' transparency. Unlike in the past where services were deployed in private and decentralized manners and studies of outages were only possible behind company walls, today, *public* cloud services are prevalent. An outage cannot be "silenced" with internal apologies and discounts. An outage, even a few minute long, can spark off various public reactions. Customers continuously monitor

Category	Service Names
CH: Chat	Blackberry Messenger, Google Hangouts, Skype, WeChat, WhatsApp
EC: E-Comm.	Amazon.com, Ebay
ML: Email	GMail, Hotmail, Yahoo Mail
GM: Game	PS Network, Xbox Live
PA: PaaS/IaaS	Amazon EBS, EC2, and RDS, Google Appengine, Microsoft Azure, Rackspace
SA: SaaS	Google Docs, Office365, Salesforce
SC: Social	Facebook, Google Plus, Instagram, Twitter
DT: Storage	Apple iCloud, Box, Dropbox, Google Drive, Microsoft SkyDrive
VD: Video	Netflix, Youtube

Table 1. List of cloud services. *The table lists the 32 Internet services we study. When presenting availability metrics, we anonymize service names with category abbreviation and an integer ID. For example, a service labeled CH2 represents one of the chat services (in* ***random*** *order).*

and time the outage, which in turn leads to *headline news*. Furthermore, as public cloud services are scrutinized more ever than before, cloud providers must provide transparency in the form of open, detailed, and accountable *post-mortem reports*. It is common that these reports are written in hundreds to thousands of words [16, 48, 63, 85]. These sources of information are the two untapped information we leverage uniquely in our work.

1.1 Cloud Outage Study (COS)

We conducted a cloud outage study of 32 popular cloud services including chat, e-commerce, email, game, IaaS, PaaS, SaaS, social, storage, and video services (Table 1). We collected and analyzed a total of *1247 headline news* and *public post-mortem reports* that detail *597 unplanned outages* that occurred within a *7-year span* from 2009 to 2015. Unlike other studies, our methodology in using public news as our dataset is relatively unique. In our work, a "service outage" implies an unavailability of full or partial features of the service that impacts all or a significant number of users in such a way that the outage is reported publicly.

From outage headline news and post-mortem reports, we manually extract "outage metadata" such as outage duration (downtime), root causes, impacts, and fix procedures. We further break them into 13 root causes, 6 impacts, and 8 fix procedures (Table 2). One valuable information in our dataset is the outage duration; 69% of outages are reported with downtime information. This allows us to quantify the overall availability of modern cloud services and analyze which root causes have longer impacts. We store the 1247 links and 3249 outage metadata tags in Cloud Outage Study database (CosDB), which we release publicly for others to use (downloadable from [1]).

This broad study also raises a perplexing question: *even with pervasive redundancies, why do outages still take place?* That is, as the principle of no single point of failure (No-SPoF) via redundancies has been preached extensively, redundant components are deployed pervasively in many levels of hardware and software stack. Yet, outages are still inevitable. Is there another "hidden" single point of failure? Studying hundreds of outages in tens of services reveal a common thread. We find that the No-SPOF principle is not merely about redundancies, but also about the *perfection* of failure recovery *chain*: complete failure detection, flawless failover code, and working backup components. Although this recovery chain sounds straightforward, we observe numerous outages caused by an imperfection in one of the steps. We find cases of missing or incorrect failure detection that do not activate failover mechanisms, buggy failover code that cannot transfer control to backup systems, and cascading bugs and coincidental multiple failures that cause backup systems to also fail.

In the following sections, we first present our methodology (§2), findings on availability (§3), and observations on the hidden sources of SPOF (§4). We then present in detail the individual outage root causes (§5), impacts and fix procedures (§6). Finally, we discuss the pros and cons of our methodology (§7) and our unique contributions compared to other related work (§8).

2. Methodology

Goals: Our main goal is to collect and analyze public reports of cloud outages, categorize them, and finally provide qualitative and quantitative analysis. Our second goal in to collect all of them in a single place, CosDB, so that other researchers in their respective areas can easily use the outage metadata in CosDB for motivational references or further statistical studies.

Dataset: We chose 32 popular cloud services as listed in Table 1. We picked services from diverse categories such as chat, e-commerce, email, game, PaaS/IaaS, SaaS, social media, data storage, and video sharing services. To search for outages of these services, we utilized search engines such as `google.com` and `bing.com` and typed the following query: "serviceName outage month year", for every month and year between January 2009 to December 2015 (a total of 7 years). We then read the first 30 search hits. At the end, we gathered 1247 unique links that describe 597 outages.

Outage Metadata (Tagging): For every outage, we read between 1 to 12 (2 on average) sources of information. This is necessary because outage information can be scattered (*e.g.*, some articles report the outage duration while the others only report the root cause). As there is no standardization of outage reports, we must manually extract the *outage metadata* including the outage duration (downtime), root causes, im-

Metadata	Sub-classification Tags
Root causes	BUGS, CONFIG, CROSS, HARDWARE, HUMAN, LOAD, NATDIS, NETWORK, POWER, STORAGE, SECURITY, SERVER, UPGRADE, and UNKNOWN
Impacts	FULLOUTAGE, OPFAIL, PERFORMANCE, LOSS, STALE, and SECURITY
Fixes	ADDRESOURCES, FIXHW, FIXSW, FIXCONFIG, RESTART, RESTOREDATA, ROLLBACKSW, NOTHING, and UNKNOWN
Downtime	Reported (in minutes/hours) or unknown
Type	Planned or unplanned outages
Scope	List of other services affected

Table 2. Outage metadata. *The table lists outage metadata that we manually extracted from public news and post-mortem reports. We use the term "fix procedures" as opposed to "recovery" to differentiate the built-in failure recovery mechanisms and the manual outage fix procedures.*

pacts, fix procedures, and nature of outage (planned vs. unplanned). Table 2 shows the outage metadata and the sub-classification tags. We performed this entire study in the last one year. For high tagging accuracy, each outage is reviewed multiple times by at least four authors of this paper.

We also note that root causes are sometimes described vaguely. For example, "due to a configuration problem" can imply software bugs corrupting the configuration or operators setting a wrong configuration. With this in mind, we do *not* speculate but rather only add tags based on concrete information; in this example, we only use CONFIG, but not BUGS or HUMAN.

COSDB: The product of our data collection and classification is stored in COSDB (downloadable from [1]), a set of raw text files, data mining scripts and graph utilities. COSDB contains 597 outage descriptions, 1247 links, and 3249 outage metadata tags.

Terms and Definitions: In this paper, a *service outage* implies an *unplanned* unavailability of full or partial features of the service that affect all or a significant number of users in such as way that the outage is reported publicly. The "Impacts" row in Table 2 details the different types of outages. A full service outage is labeled as FULLOUTAGE. For partial service unavailability, we *exclude* failures of "non-essential" operations (*e.g.*, profile picture update, background image change), However, if the partial failures involve essential operations (*e.g.*, login, payment, search), we consider them an outage, labeled with OPFAIL. Data loss (LOSS) and staleness (STALE), and late deliveries (PERFORMANCE) that lead to loss of productivity are also considered an outage.

How long an outage lasts is represented by *downtime* (*e.g.*, in hours). We use *annual service uptime* (*e.g.*, 99%) based on the total downtime per year experienced by the service, per the outage definition above. This measurement could be different from other standards.

Presentations: When we present downtime/uptime metrics, we anonymize service names and use service labels (*e.g.*, CH2) as described in Table 1 (see "Disclaimers" below). To allow readers to easily use outages examples from this paper without downloading COSDB, we cite some interesting links (*e.g.*, [17]). As the reference section is longer than usual, we make it shorter by embedding the news/reports' hyperlinks behind the "(link)" text.

While typical failure datasets mainly contain quantitative failure metadata, our dataset uniquely contains *qualitative* outage descriptions, which we believe are valuable to summarize. Therefore, while many other work (§8) focus on statistical findings, the writing style of this paper focuses more on qualitatively summarizing why outages happen.[1]

Disclaimers: As availability is a sensitive matter to service providers, it is important for us to make several disclaimers. First, our study is not meant to discredit any service. Readers should take the high-level lessons but prevent themselves to compare service uptimes (*e.g.*, X is better than Y). For this reason, we anonymize service names (*e.g.*, CH2) when presenting numerical findings. Second, the more popular a service is, the more attention its outages will gather, hence more headlines. In fact, more popular services tend to be more transparent and provide detailed reports that we could learn from. For this reason, again, readers should not use this paper to claim a service is better than others. The pros and cons of our methodology will be presented in Section 7.

3. Availability

This section presents our quantitative analysis of cloud service availability. Specifically, we discuss annual outage count, service downtime/uptime, and correlation between outages and service maturity. We note that as our dataset is not complete (not all outages are reported publicly; §7), our findings below can be considered as "minimum" values.

3.1 Annual Outage Count

Figure 1a shows the annual outage count (number of *unplanned* outages per year) between 2009-2015 for every service in our study. As mentioned before (§2), we exclude planned maintenance and failures of non-essential operations. The thin (blue) line in Figure 2a plots the distribution

[1] We humbly do not attempt to provide suggestions to address every outage problem (although our research group addresses some part of the problems [130, 136, 140, 143–145, 155]), but rather we focus on summarizing the lessons which we hope can be valuable to the larger cloud community.

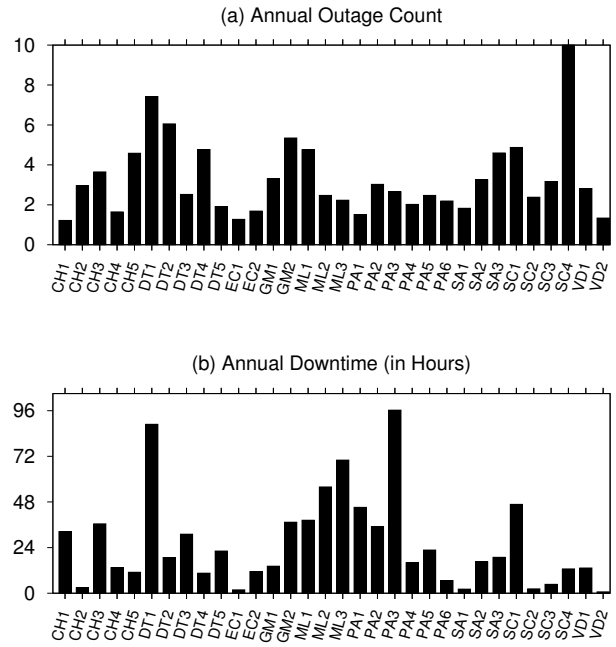

Figure 1. Annual outage count and downtime. *The top and bottom figures show the annual outage count and downtime (in hours) for every service in our study respectively. SC4 has 23 annual outage count, mostly come from the company's press releases (showing their high transparency).*

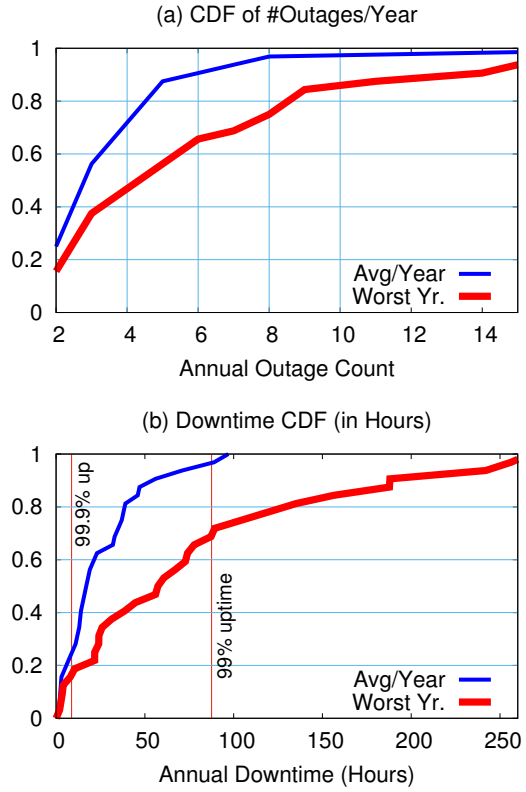

Figure 2. Outage count and downtime distribution. *The left and right figures show outage count and downtime distributions respectively. The thin blue line in the left and right figures plot the CDF of the annual datapoints in Figure 1a and 1b respectively. The thick red line shows the distribution from the services' worst years.*

of the average annual outage count across the 32 services. Almost 50% of the services in our study experienced at least 3 outages per year on average.

Since average numbers hide the "worst year" in which a service experienced a high number of outages, we also plot the outage count distribution from the *worst year* (between 2009-2015) of every service, shown by the thick (red) line in Figure 1a. In their worst years, almost half of the services experienced at least 4 outages. The distribution also has a long tail; 25% of services (y=0.75) suffered at least 8 outages in their worst years.

3.2 Annual Service Downtime/Uptime

69% of outages are reported with downtime information. Within this population, Figure 1b shows the average annual downtime (in hours) for every service. In Figure 2b, the thin (blue) line plots the distribution. The figure also plots two vertical lines that represent 99% (2-nine) and 99.9% (three-nine) service uptime (*i.e.*, not more than 88 and 8.8 hours of annual downtime respectively). On average across the six years, 2 services (6%) do not reach 99% uptime and 25 services (78%) do not reach 99.9% uptime. If we consider only the worst year from each service (the thick red line), 10 services (31%) do not reach two-nine uptime and 27 services (84%) do not reach three-nine uptime. These numbers can be considered the "minimum" as not all outages are reported in public reports and not all that are reported include downtime information. The Utopia of five-nine uptime (five minutes of annual downtime) is still far to reach.

3.3 Outage and Service Maturity

The next question we ask is: does service maturity help? That is, do outages happen less frequently in mature services? To answer this, Figure 3a shows the outage count in every year ('09-'15) bucketed by service age. For example, in 2014, there are 24 outages occurred in total from 9-year old services. With the same bucketing approach, Figure 3b shows the total downtime in hours. For example, in 2014, there are 267 hours of downtime collectively from 17-year old services. The outage numbers from young services are relatively small; we postulate that large mature services gain more public attention. Overall, the two tables show that outages can happen in any service regardless of its maturity. In fact, as a service becomes more mature and popular, it needs to handle more users and complexity tends to increase.

Total outage count from X-year old services in 20YY

	2	3	4	5	6	7	8	9	10	12	15	17	18
'15	1	8	5	5	2	2	26	3	7	8		3	2
'14	4	9	8	14	6		18	24	11	4	1	5	1
'13	7	16	5	8	1	5	25	5	18	5	14	2	4
'12	2	11	5	14	2		23	2	8	4	8	6	1
'11		6	3	11	2	1	16	2	5	3		6	2
'10			2	3	4		28	2	6	6	2	4	3
'09				4	1		60	1	5	4	1	3	2

X: Service Age

Total downtime from X-year old services in 20YY

	2	3	4	5	6	7	8	9	10	12	15	17	18
'15	1	50	12	12	18		20	3	6			1	6
'14	7	37	32	51	67		80	69	14	87	6	267	9
'13	21	270	20	34	0	89	39	1	40	32	72	121	188
'12	0	51	80	46	10		42	2	5	48	25	36	1
'11		25	74	216	135	3	143	1	203	24		55	3
'10			3	57	14		74	2	13	12	1	9	122
'09				156	6		32	1	247	32	0	20	9

X: Service Age

Figure 3. Outage vs. service maturity. *Each cell shows the total outage count (top figure) or total downtime in hours (bottom figure) collectively experienced by X-year old services in year 20YY as plotted in the x- and y-axis. Service release year is obtained from* `wikipedia.com`.

4. Where is the SPOF?

Before presenting the individual outage root causes (§5), we first provide an important answer to the most perplexing question in our study: *where is the (hidden) SPOF?* That is, to prevent outages from single points of failure, the concept of *redundancies* has been widely preached and pervasively deployed at various levels (*e.g.*, data replicas, power backups, geo-distributed zones). The fact that outages are still inevitable today implies that there are sources of SPOF that should be identified.

By studying hundreds of outages, we observe a common thread that addresses the question. We find that the No-SPOF principle is not merely about hardware redundancies, but also requires the *perfection* of failure recovery *chain*: complete failure/anomaly detection, flawless failover code, and working backup components. Each of these elements ideally must be flawless. Yet, many of the outages we study are rooted by some flaws within this chain as we elaborate below. (For simplicity, we use the word "failure" to represent the classical definitions of errors, faults, and failures [122]).

4.1 Incomplete Failure/Anomaly Detection

The first case is about incomplete failure/anomaly detection. That is, although the failover code that will activate the redundancies is *ready* to run, it will *sit idle* if the root failures are *not* detected or anticipated. We observe a handful of undetected failures such as memory leaks (that eventually crashed the entire system) [18], load spikes of authentication requests (that were not monitored, while spikes of read/write requests were monitored) [10], expiring certificates (that piled up and caused backlogs) [64], unforeseen "grey partial" hardware failures [18, 78], and corrupt configurations [55, 82].

While the cases above are caused by external events not detected by the systems software, there are cases where failures come from the software itself (*i.e.*, software is also a SPOF). If a software system does not detect its own misbehavior, the failures can cascade to other software layers. We find this issue prevalent in software upgrades, one of the most-common causes of outages (§5.1). The developers had tested the software upgrades, but when the updates were pushed to the full ecosystem, they caused new failures/anomalies that were undetected in the offline testing.

4.2 Failover that Fails

Let's suppose the failure is correctly detected. The next stage requires *flawless failover* to activate the redundancies properly. Unfortunately, there are cases of failover code that fails to mitigate configuration errors [69], activate the backup power generators [6, 15], recover to a backup network switch [20] or a backup system [101], and migrate data to another datacenter [40].

Furthermore, as discussed later in Section 5.5 (LOAD), recovery code might "work as it should be" but in certain situations it aggressively generates extra traffic that can create a positive feedback loop (*i.e.*, a *"recovery storm"*). We also observe an interesting *failover cold cache* problem; a datacenter failover caused a load spike to the backend database because of cold caches in the new datacenter destination [40]. Flawless failover/recovery code should anticipate all possible deployment scenarios.

4.3 Backups that also Fail

Presuming the failover code is flawless, the next question is: will the backups/redundancies work when they are needed? Based on the many cases of redundancy failures below, the answer is unfortunately "not always".

The first case is about simultaneous *multiple failures* of primary and backup components such as double failures of power [5, 17], network [52, 87], storage [95], and server components [104]. Moreover, not only can multiple failures originate from identical components, they can also be exhibited by different components (*i.e.*, multiple *diverse* failures). For example, unrelated failures of an external network failure and an intermittent server occurred together [76] and a network fiber cut and a separate storage failure coincided [43]. Although multiple points of failure ("MPOF") can be considered rare, the cases above prove that they can happen in reality and lead to fatal scenarios. Stress-testing failure recovery code with multiple diverse failures is crucial [136, 139, 142, 144].

Redundancies can also fail due to *cascading bugs*. That is, one bug simultaneously affects *many* or *all* of the redundancies, hence impossibility of failover. Cascading bugs happen because the *same software logic* runs in multiple redundant supposedly-independent nodes. For example, multiple data servers experienced the same type of memory leak as all of them experience communication issues with a dead server [18]; expiring certificates created connection errors in multiple storage servers [64]; the same memory allocation errors were present in both the primary and secondary network devices [99]; and the same timeout bug that cannot address slow responses caused a large number supernodes crashing [100]. More research is needed to ensure that independent nodes do not follow the same "crash path" given the same triggering condition.

4.4 Prolonged Downtime

After outages are resolved internally, cloud services should be aware of *post-outage request storms*. That is, after a service is back online after an outage, it must face a stampede of requests that had been waiting to re-connect, which can cause another downtime. For example, as a service connectivity was restored, an ensuing traffic spike caused several network elements to get overloaded [76]; in another case, a service that just returned to service went down again due to over-capacity from millions of users that re-connected [108].

5. Root Causes

We now present the result of our main categorization, outage root causes, as listed in Table 3. Before jumping to the discussion of every specific root cause, we first make general observations.

• **By count:** The "Cnt" column in Table 3 shows that 355 outages (out of the total 597) have UNKNOWN root causes. Among the outages with reported root causes, UPGRADE, NETWORK, and BUGS are three most popular root causes, followed by CONFIG and LOAD.

Many of the root causes in Table 3 are addressed in literature (*e.g.*, network failures [134], misconfiguration [121, 160], load spikes [127], storage failures [123, 154], human administrative mistakes [128]). UPGRADE, one of the largest problems in our study, requires more research attention (§5.1).

Another interesting matter is the fact that component failures such as NETWORK, STORAGE, SERVER, HARDWARE, and POWER failures should be anticipated with extra redundancies, but their failures still lead to outages. As discussed earlier (§4), the no-SPOF mantra is not merely about hardware redundancies but requires the perfection of the complete recovery chain.

• **By service:** The "#Sv" column in Table 3 shows that each root cause can plague many different services.

§	Root cause	#Sv	Cnt	%	Cnt '09-'15
	UNKNOWN	29	355	-	M.M.M.M.M.M.M
5.1	UPGRADE	18	54	16	7.4.M.5.M.4.7
5.2	NETWORK	21	52	15	4.4.6.8.M.8.5
5.3	BUGS	18	51	15	M.4.9.8.9.9.2
5.4	CONFIG	19	34	10	2.2.7.2.5.M.4
5.5	LOAD	18	31	9	2.5.5.5.4.8.2
5.6	CROSS	14	28	8	-.2.4.M.5.3.4
5.7	POWER	11	21	6	5.4.3.5.3.1.-
5.8	SECURITY	9	17	5	7.-.2.1.3.4.-
5.9	HUMAN	11	14	4	-.1.4.4.2.1.2
5.10	STORAGE	4	13	4	2.-.-.3.5.3.-
5.11	SERVER	6	11	3	-.3.-.2.2.4.-
5.12	NATDIS	5	9	3	1.1.3.2.1.1.-
5.11	HARDWARE	4	5	1	1.-.-.3.1.-.-

Table 3. Root cause statistics. *The table lists root causes sorted by count. "§" represents the section number (e.g., §5.1); "#Sv" the number of services that have suffered from the root cause; "Cnt" the number of occurrences; and "%" the count percentage among known root causes. The last column counts the occurrences in each year; "M" implies ≥10.*

• **By year:** We next ask whether there are root causes that perhaps had been solved and did not appear in latter years. The last column in Table 3 shows a seven-number string (*e.g.*, "M.4.9.8.9.9.2"; "M" implies ≥10) which represents the number of occurrences in each of the seven years between 2009-2015. We can conclude that every root cause can occur in large popular services almost in every year. As software continues to evolve, similar problems in the past might re-appear in new forms.

• **By duration:** Figure 4 shows the whisker plot of downtime (in hours) across the root causes, sorted based on the median value. The median values across all the root causes range from 1.5 to 24 hours. The minimum values span from 5 minutes to 1 hour. Almost all root causes (except HARDWARE which has a small population) have a maximum downtime of more than 50 hours. This suggests that the severity of a root cause varies depending on the specific situation. Natural disaster (NATDIS) and POWER failures tend to produce the longest downtimes because when they happen all resources are often affected, including the backup systems. HUMAN mistakes and BUGS can also lead to long downtimes as they can create cascading failures.

• **By cross-relation:** We emphasize that an outage can be tagged with more than one root causes, typically happens in a complex failure. For example, Azure 2014 outage was caused during an UPGRADE that also involved CONFIG changes across the entire infrastructure, that were mistakenly done by the HUMAN operator, which then exposed BUGS in their blob front-end servers [67]. As another example, an Amazon's outage was caused by a misconfiguration in the EBS layer (CONFIG) combined with mas-

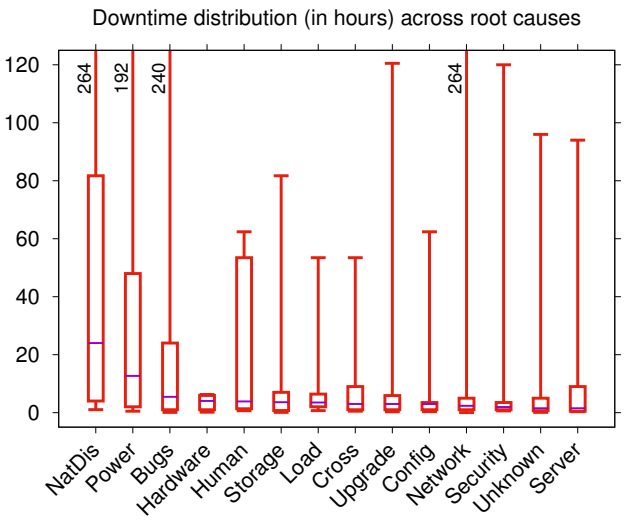

Figure 4. Downtimes across root causes. *The figure shows the whisker plot of downtime (in hours) for every root cause. The x-axis is sorted by the downtime median value.*

sive load from the re-mirroring storm from the EC2 layer (LOAD) [16]. Table 4 shows the top highly-correlated pairs of root causes, which will be discussed in their corresponding sections.

We now discuss in detail the individual root causes in the same order listed in Table 3, which is sorted based on the number of occurrences ("Cnt").

5.1 Upgrade

UPGRADE label implies hardware upgrades or software updates typically done during maintenance events. Table 3 shows that UPGRADE is one of the largest root causes (16%). Although there is an ongoing progress on software update research (*e.g.*, Ksplice, kpatch, KGraft), we find that upgrade issues in the field involve a large software ecosystem (a software update causes anomalies in other software components). Such a large ecosystem is hard to reproduce in research environments. This finding calls both the industry and academia to rethink about upgrade-related research. Below we summarize a wide range of upgrade problems we find.

We observe that *a variety of component upgrades* can cause outages such as updates/upgrades of power substation (*e.g.*, failed power shift from local utility to a new substation [3, 83]), authentication mechanisms (*e.g.*, preventing user logins [24]), DNS infrastructure (*e.g.*, incorrect re-routing [33]), URL shortener update (a single point of gateway failure) [60], firmware (*e.g.*, suddenly causing overheat [65]), datacenter "environments" [77], SDN control cluster [85], database capacity (*e.g.*, causing instability in caching layer [97]), and miscellaneous software protocols as we expand below.

Root cause pairs		Count
BUGS	CONFIG	13
UPGRADE	NETWORK	13
LOAD	CONFIG	12
BUGS	LOAD	11
HUMAN	CONFIG	9
BUGS	HUMAN	8
NETWORK	CONFIG	7
NATDIS	POWER	7
UPGRADE	CONFIG	7
UPGRADE	BUGS	7
UPGRADE	HUMAN	5
UPGRADE	LOAD	5
BUGS	NETWORK	5

Table 4. Root cause pairs. *Some outages involve multiple root causes. The table lists the top highly-correlated pairs of root causes where the count is five or more.*

The UPGRADE & BUGS row in Table 4 points out that some failed upgrades were caused by *unexpected bugs in the new software*. For example, a new data geo-replication strategy caused a datacenter to be overloaded [44], a load balancer update incorrectly interpreted a portion of datacenters as unavailable [42], a feature update exposed a new memory management bug only evident under heavy usage [56], a storage software update temporarily lost users data [48], and new caching code caused missing tweets [107]. These cases are interesting to note because presumably the new software had been tested thoroughly in an offline environment, but upgrades pushed to the *full ecosystem* can be fragile (*e.g.*, "a code push which behaved differently in widespread use than it had during testing" [54]).

Not only in the new software, *bugs also appear in the upgrade scripts*. For example, a buggy upgrade script accidentally re-installed a number of live machines with active data [23] and internal prototypes were accidentally pushed publicly which forced the service to be taken offline [27]. These cases are also noteworthy because although human errors are reduced via automation (§5.9), errors are now shifted into the automation process.

The UPGRADE & NETWORK row in Table 4 suggests that outages due to *network upgrades* are quite common (*e.g.*, DNS and SDN controller updates [33, 85]). Failed upgrades can also lead to excessive load (UPGRADE & LOAD), which we discuss further in Section 5.5.

5.2 Networking Failure

NETWORK problems are responsible for 15% of service outages. They can originate from *broken hardware* such as dead core network switches [87]. In one intriguing case, a networking device exhibited an unforeseen "grey partial failure"; the operators had to perform a forensic investigation to understand how it failed [18].

To prevent single points of failure, network redundancies are employed, however *multiple networking failures* can occur. For example, core and secondary network switches simultaneously died [87] and redundant network paths failed at the same time [52].

The networking layer is also prone to *access misconfiguration* [66], *unsuccessful upgrades* (*e.g.*, of edge network [73], SDN control cluster [85], and some other networking equipments [34, 92]), and *software bugs* (*e.g.*, DNS issues [35], traffic control bugs [26, 46, 57], routing loops [84], corrupt port data from Open vSwitch [85], and memory allocation bugs in device firmware [99]). The availability a cloud service is also at the mercy of *external networks* outside the domain of the service [68, 76].

5.3 Bugs

BUGS label is used to tag reports that *explicitly* mention "bugs" or "software errors". With this, BUGS are responsible for 15% of outages with known root causes. Many other cases can be traced back to software bugs (*e.g.*, misconfiguration, distributed storage failures, security breaches), but again we do not label them with BUGS unless otherwise explicitly stated. Thus, there is a possibility that BUGS ratio is larger than 15%.

We observe *a wide range of outage-causing bugs* such as data races [16], buggy configuration scripts (§5.4), a leap-day bug [71], database bugs [32, 96] some of which can lead to staleness and inconsistency [112, 113], an operating system bug that deletes local files [58], and login-related bugs such as javascript login redirection [28], authentication denials [76], SSL certification errors [109, 110], and front-end server bugs [106].

Interestingly, although being addressed in literature for decades, *memory leaks still occur*. We observe a memory leak that was exposed only under heavy load [55], a memory overflow caused by backlogged threads [98], and a memory allocation error [99].

When component failures happen, recovery code is the last lifeline. However, *recovery code can be buggy*. For example, recovery code that cannot handle slow replies which led to crashes [100], a failover failure of configuration issues [69], recovery that generates positive feedback loop (§5.5).

Bugs in *traffic-related code can be dangerous* (BUGS & LOAD) as they can create an excessive load. For example, a network control bug caused an incorrect traffic shift [46] and a buggy geo-replication protocol caused a datacenter to be overloaded [44]. We further discuss load related issues later in Section 5.5.

As bugs are hard to eliminate, monitoring and error-checking code must be flawless in detecting anomalies and errors. In reality, *bugs also appear in monitoring systems and error-checking code*. For example, a monitoring bug failed to detect a specific case of memory leaks [18] and a DNS parser did not check malformed input string originated from a misconfiguration [74].

5.4 Misconfiguration

CONFIG problems cause 10% of service outages. Table 4 shows that misconfiguration is *not a single dimensional problem*. Section 5.5 for example already listed cases of misconfiguration that induce traffic overload (CONFIG & LOAD). Section 5.9 will discuss misconfiguration due to human mistakes (CONFIG & HUMAN).

Human is not the only one to blame; we find that configuration can be *corrupted by software bugs or failed upgrades* (CONFIG & BUGS/UPGRADE). For example, a software bug generated a corrupt configuration to live services that disabled them to handle normal load [55]; a failed upgrade to balance network traffic accidentally corrupted some configuration [74]; and node reboots after a rare failure corrupted global configuration files [82].

Another interesting finding is the problem of *ecosystem-dependent configuration*: a configuration change in one subsystem might need to be followed with changes in the other subsystems, otherwise the full ecosystem will have conflicting views of what is correct. As an example, a correct persistent change of a server configuration was interpreted invalid by all client nodes, which caused re-query stampede [31]. In reverse, a change in the ecosystem might require configuration updates. As an instance, a service increased its server and networking capacity, but the maximum load threshold in the network "safety-valve" mechanisms was not adjusted, causing conflicting views [66].

5.5 Traffic Load

Cloud services provision resources based on expected traffic load. In this context, unexpected traffic overload (LOAD) can easily lead to outages. In our study, load-related problems, which cover 9% of outages, originate from four sources: (1) user requests, (2) code upgrades, (3) misconfiguration, and (4) flawed recovery.

First, outages due to *load spikes of user requests* often happened on special days or events such as Christmas eve [72, 80], New Year eve [108], the President's inauguration day [102], World Cup's opening [103], company acquisitions (*e.g.*, WhatsApp's $19 billion acquisition by Facebook which led to a surge of sign-ups [116]), and captivating celebrity posts such as Justin Bieber's picture post which may have caused traffic overload from his millions of followers [61]. We also note that while main requests (*e.g.*, read/write) tend to be monitored and services can elastically scale accordingly, *load spikes of non-monitored requests* can be dangerous (*e.g.*, excessive database index file accesses [91] and spikes of authentication requests that caused extreme cryptographic consumption of resources [10]).

Second, increased load can also stem from *new code upgrades*. For example, a system upgrade that required account

migration caused a high volume of credential retry requests [39]; a new code that tries to keep data geographically close to its owner accidentally overloaded a datacenter [44]; and a code upgrade generated extra load to request routers which then told the rest of the system "stop sending us traffic" [50].

Third, *misconfiguration* can escalate traffic load. For example, a server configuration change triggered many client re-queries that overwhelmed the database servers [31]; a wrong traffic redirection caused a cycle of overloaded machines [45]; a miscalculation of memory usage caused the same normal traffic load unservable [43]; an authentication misconfiguration re-routed all authentication requests to a small set of servers [47]; and disabling CRON suddenly and unexpectedly made an unusual load spike [88].

Finally, as other work has noted, "the cure [can be] worse than the disease" [138]. That is, *flawed failure recovery* can introduce extreme load (a "recovery storm"). A prime example is the *positive feedback loop* problem. Here, flawed recovery causes extra traffic that congests existing resources, which then triggers more recovery traffic. For instance, to handle configuration errors, an automated recovery caused clients to make a flood of queries and overwhelmed the database cluster, which then caused clients to continue retrying [31]; storage nodes failed to find new space to re-mirror under-replicated volumes and kept performing "Create Volume" process [16]; a large number of VMs that were automatically rebooted at the same time overloaded the image storage and received timeout errors [67]; a load-balancing turned on an automatic DoS protection, shunting a portion of the high incoming traffic to a CAPTCHA, but then triggered more client retries, exacerbating the problem [38].

Another set of example is where the recovery re-routes too much traffic from dead servers to some healthy servers, which in turn makes the healthy servers unresponsive. For instance, as a subset of servers were crashing due to a bug, the healthy servers cannot handle the extra re-routed traffic [100]; a bug in a network control software brought down some parts of network capacity and caused traffic shift that overloaded other servers [46]; and a global restart of traffic routers in one datacenter caused a widespread overload in other datacenters [37].

5.6 Cross-Service Dependencies

Cloud computing is essentially a stack of services (*e.g.*, SaaS on PaaS) wherein vulnerable dependencies linger. We use CROSS (8%) to label outages caused by disruptions from other services.

A service can be affected by other service outages such as ISP issues (*e.g.*, ISP's fiber cut and DNS problems [13, 68]), lower layer disruptions (*e.g.*, a PaaS outage caused a SaaS downtime [94]), or 3rd party failures (*e.g.*, a spam filtering bug [49], a 3rd-party database issue [90], and a URL shortener service disruption [105]).

The more popular a service is, the more eyes are on the service. Amazon Web Service, arguably the most popular destination for a variety of services, has garnered many headlines for causing ripple effects including to AirBnB, Bitbucket, Dropbox, Foursquare, Github, Heroku, IMDB, Instagram, Minecraft, Netflix, Pinterest, Pocket, Quora, Reddit, Tinder, and Vine [4, 7, 12, 14, 19, 22]. Microsoft Azure, another popular one, has also caused multiple downtimes to Xbox Live and also 52 other services [117, 118]. A Paypal downtime also caused Ebay, Etsy and many other merchants to halt sales [25].

Besides creating external impacts, a service outage, especially the storage service layer, can shut down other *internal* services as well. For example, Amazon EBS problem brought down EC2 [18], iCloud downtime impacted Apple applications [59], and Microsoft Azure outage rippled to Office 365 [75].

Finally, we highlight the vulnerability of *service collocation*. For economic and performance reasons, multiple services are often collocated in the same datacenter. As a ramification, a single problem can cause multiple service outages. For example, lighting strikes and power failures made multiple Amazon services go down concurrently [15–17]; a power failure at a Google datacenter caused Gmail, Google Search, Google Drive, and YouTube outages that collectively dropped internet traffic by 40% for 5 minutes [36, 51]; a failed load balancer upgrade simultaneously impacted Google Drive, Chat, Calendar, Play, and Chrome Sync [42]; similarly, a failed DNS patch caused downtimes of Office 365, Messenger, Outlook, and Xbox [70].

5.7 Power Outages

No power, no compute. POWER failures represent 6% of outages in our study. They can originate from natural disasters (*e.g.*, massive lightning storms [15, 17]), external human factors (*e.g.*, a vehicle crashed into utility poles [6]), and failed maintenance or upgrades of power utilities (*e.g.*, failed shift of power from a local utility to a new substation [3]; a short circuit occurred during a PDU testing phase rotation [83]).

The No-SPOF principle also applies to power sources. The main utility power provider can fail (*e.g.*, Amazon's power provider suffered a failure of a 110kV 10 megawatt transformer [15]), which should not cause major disruptions due to standard deployments of backup generators. However, a *second failure* in failing over to the backup generators is very much possible. For example, a PLC failed to synchronize the electrical phase between the generators [15]; backup generators also failed to operate due to lightning storm [17]; and redundant power paths failed simultaneously [5].

5.8 Security Attacks

Security-related issues are responsible for 5% of the outages in our study. Security attacks come in different forms including DDOS attacks (*e.g.*, 300 Gbps DDOS traffic to

Rackspace [86] and 1.2 Tbps DDOS traffic to Xbox Live and Playstation Network [81, 119], which made the hacker group gain 120,000 followers in two days), worm attacks (*e.g.*, "Mikeyy" worm on Twitter [114]), and botnet (*e.g.*, "Zerus" stealing bank account information from Amazon [8]). Security attacks coming from 3rd-party applications can be stopped by disabling the malicious applications (by some traffic pattern matching algorithm), however the algorithm can accidentally shut down well-behaving applications as well [30]. To prevent future security attacks, sometimes cloud services must shut down their services for applying security patches [9].

5.9 Human Errors

A running system involves code paths and "manual paths". The latter opens up the possibility of human errors, which in our study are responsible for 4% of the outages. In one way, this small rate sounds positive compared to the much higher rates reported 15 years ago [128]. On the other hand, more automation and safety checks are still needed.

More than half of human errors relate to upgrade and configuration issues (HUMAN & UPGRADE/CONFIG), for example, operators incorrectly executed traffic shift [16], did not follow upgrade procedures [67], failed to include certificate updates in new storage releases [64], and forgot to change a networking configuration after new capacities were added [66]. A single word error can be fatal; for example, an operator incorrectly enabled a configuration switch for "blob" storage front-ends, while the pre-production test was done for "table" storage front-ends [67].

Another vulnerable manual path is *post-failure* situations that require human involvement. For example, after some faulty storage nodes were taken to diagnostic and repaired, the operator put them back to the cluster without enabling the node protection, causing them to be accidentally reformatted [63]; on-call staffs were not trained enough to handle some specific type of power failures, causing a prolonged recovery [53].

5.10 Storage Failures

No data, no compute either. When storage nodes fail, compute nodes cannot progress [18, 41]. We mark an outage with STORAGE (4%) only if the report explicitly mentions failures from the "storage" layer, which could mean storage devices, the entire storage cluster, or file/database systems. We observe storage failures such as 75% error rates from a storage layer after a failed upgrade [40], corrupted database index [91], and failed primary database recovery causing 1-hour data loss [93].

To prevent multiple failures, three data replicas might suffice. However, there was a case where *two* out of three replicas are inaccessible due to *two independent failures* (a storage layer issue and a fiber cut) causing quorum writes to not reach a consensus [43].

5.11 Miscellaneous Server and Hardware Failures

When an outage report does not describe specific component failures, but mention "node/server", we label it with SERVER (3%). Some reports describe failures of specific server types that caused outages such as caching, LDAP, login, quota-check, scheduling, and search nodes. Similarly, we label a report with HARDWARE (1%) if it does not mention specific types of hardware failures. In most of these cases, the providers do not present the details.

5.12 External and Natural Disasters

Last but not least, external and natural disasters (NATDIS) such as lightning strikes [11, 15, 17], vehicle crashing into utility poles [6], government construction work cutting two optical cables [115], and similarly under-water fibre cable cut [120], cover 3% of service outages in our study.

6. Impacts and Fix Procedures

We breakdown the root-cause impacts into 6 categories (Table 2): full outages (59%), failures of essential operations (22%), performance glitches (14%), data loss (2%), data staleness/inconsistencies (1%), and security attacks/breaches (1%). Figure 5a shows the number of outages bucketed by root causes and implications.

Only 24% of outage descriptions reveal the fix procedures. We breakdown reported fix procedures into 8 categories: add additional resources (10%), fix hardware (22%), fix software (22%), fix misconfiguration (7%), restart affected components (4%), restore data (14%), rollback software (8%), and "nothing" due to cross-dependencies (12%). Figure 5b shows the number of outages bucketed by root causes and fix procedures. Due to space constraints, we unfortunately do not provide qualitative discussions on impacts and fix procedures.

7. Pros and Cons of COS Methods

We now describe the pros and cons of our methodology.

Pros: There are several advantages of our unique methodology. First, arguably headline news are *free from false positives*; they are published by either the providers themselves or other trusted news websites such as `cnn`, `datacenterknowledge`, `forbes`, `huffingtonpost`, `infoworld`, and `zdnet.com`. Many of the public news also include update reports from the engineers and operators of the corresponding services mentioned in the news. Second, as customers time high-profile outages, many of the reports (69%) include downtime information, which is valuable in measuring annual service uptime. Third, many of the reports contain detailed descriptions (*e.g.*, in hundreds to thousands of words [16, 48, 63, 85]) of how the problems affect many

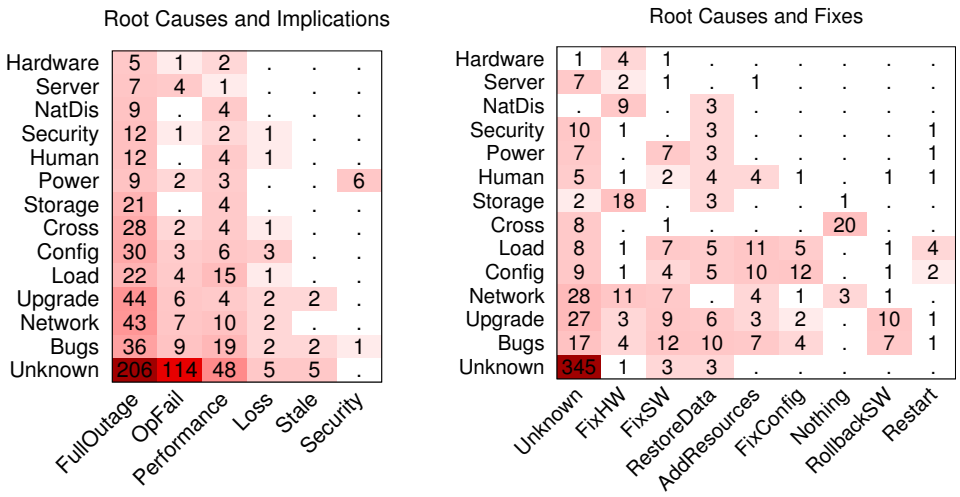

Figure 5. Root causes vs. (a) impacts and (b) fix procedures.

software layers within the services. This "multi-layer" report is typically unavailable in single-system error tickets.

Cons: There are several caveats in our methodology. First, our dataset is not complete. Not all outages are reported publicly (*e.g.*, small short outages). Second, our dataset is skewed. The more popular a service is, the more attention its outages will gather, hence more headlines. In fact, more popular services tend to be more transparent and provide detailed reports that we could learn from. Third, outage classifications are incomplete due to lack of information. For example, only 40% outage descriptions reveal root causes and only 24% reveal fix procedures.

Regardless of all the limitations above, we are aware of no work that employs a methodology such as COS. This method has successfully led us to interesting and unique qualitative findings (§8).

Other possible public-data methods: Note that the uniqueness of our study is the use of public data. One other public-data method we tried is using service dashboards (*e.g.*, `google.com/appsstatus`, `status.aws.amazon.com`, `status.azure.com`) or 3rd-party monitoring sites (*e.g.*, `downdetector.com`). These data sources are rich for statistical analysis, but for qualitative analysis, we faced several drawbacks. First, not all the sites allow us to easily dig the outage archives. Second, many of them do not describe the root causes in detail, which is not valuable for research perspective; those that do eventually point to the detailed postmortem reports which we used in our study. Third, 3rd-party monitoring sites typically provide much less information (*e.g.*, `downdetector.com`) than public news and reports and some became inactive (*e.g.*, `cloutage.org`). Finally, they mainly report when outages start but not the downtime (outage duration).

Other possible private-data methods: We did not use methods that depend on internal tickets only available behind company walls. As these methods are already common (§8), we seek a different approach albeit the new limitations. Plus, such private-data methods usually can only analyze one or few services; it is hard to convince multiple service providers to be open to this kind of study, while our goal is to broadly study outages across tens of services.

8. Related Work

We now describe the uniqueness of our study compared to other related work. Table 5 summarizes our qualitative and quantitative comparisons.

First, to the best our knowledge, our work is the first that leverages rich outage explanations from *headline news/reports* (labeled with ^{N} under the "DST" column in Table 5). Datasets in other work typically come from error messages^{E}, logs^{L}, e-mail threads^{M}, or tickets^{T}.

Second, to the best of our knowledge, our work analyzes the *largest* number of services (32) within the *multi-layer analysis* category (labeled "M" under the "Analysis Type (AT)" column in Table 5). This category analyzes all sorts of issues that cause service disruptions. Other work in this category generally focuses on a few number of services (1-3), as shown under the "#S" column. Some work analyze a large number of clusters/datacenters (*e.g.*, ^{u}5 to ^{u}20) typically from a single company/institution; hence, we tag them with ^{u} representing uniform type of systems. The work by Banerjee et al. [124] involves the 2nd largest number of services (11) as they analyzed public outage mailing lists. The rest of the "AT" column in Table 5 shows that many other work focus on *service subcomponents* such physical/virtual nodes (O), network (N), storage (S), hardware (H), jobs (J), and software bugs (B).

Related work	#S	#N	DST	#Yr	AT
COS	32	–	$1k^{N}$	6	M^{ds}
[124] Banerjee	11	–	$6k^{M}$	8	M
[125] Benson	1	–	$9k^{M}$	3.5	M^{d}
[131] El-Sayed	$^{u}20$	11k	$-^{T}$	9	M
[135] Gray	1	2k	$0.1k^{T}$	1.2	M
[149] Oppenheimer	3	3k	$0.6k^{T}$	1.3	M^{d}
[153] Schroeder	$^{u}20$	24k	$23k^{T}$	10	M
[161] Zhou	1	100k	$2k^{M}$	–	M^{ds}
[126] Birke	$^{u}5$	>10k	$3k^{T}$	1	O^{d}
[132] Ford	1	>10k	$-^{L}$	1	O^{d}
[133] Garraghan	1	12k	$-^{L}$	0.1	O^{d}
[158] Vishwanath	–	100k	$-^{T}$	1	O
[159] Yalagandula	3	230k	$-^{T}$	1.5	O^{d}
[134] Gill	6	1k	$46k^{L}$	1	N^{d}
[157] Turner	3	0.2k	$217k^{L}$	6	N^{d}
[141] Jiang	$^{u}4$	2m	$39k^{L}$	4	S
[154] Schroeder	1	100k	$100k^{T}$	0.5	S
[147] Liang	1	128k	$1.3m^{E}$	0.3	H
[148] Nightingale	2	1m	$-^{L}$	1	H
[129] Chen	1	12k	$26m^{L}$	0.1	J
[146] Li	1	1k	$0.2k^{T}$	0.1	J
[137] CBS	6	–	$3k^{T}$	4	B
[152] Sahoo	6	–	$0.3k^{T}$	>1	B

Table 5. COS vs. related work. *The six columns are: (1) Related work: either the project name or the last name of the first author; (2)* ***#S****: number of services/datacenters studied with* ^{u} *label if the datacenters are uniform; (3)* ***#N****: number of nodes/machines/devices involved in the study; (4)* ***DST****: the dataset* ***size*** *with different units (k:*10^3*; m:*10^6*) and dataset* ***type*** *(error messages*^{E}*, logs*^{L}*, e-mail threads*^{M}*, headline news*^{N}*, or tickets*^{T}*); (5)* ***#Yr****: the year range of the dataset; (6)* ***AT****: Analysis type such as Multi-layer, nOde (physical/virtual), Network, Storage, Job level, Hardware, and software Bugs analysis. Extra label* ^{ds} *is added if the work analyzes duration of service downtimes or* ^{d} *if only analyzes failure duration.*

Third, not all work report *outage duration* (downtime). Most work record the time when the failures happened and time between the failures, but not how long until the failures are mitigated [131, 147, 153]. Some work report failure duration information (^{d} under the "AT" column) but only with respect to *subcomponent* failures (*e.g.*, response time to user problems [125], repair time of physical/virtual machines [126, 133], periods of node ping absence [132, 159], time-to-recovery of network devices [134, 157]). However, subcomponent failure timespan does *not* necessarily translate to service downtime; normally, subcomponent failures are masked with some redundancies. Among the related work, we only found one work (Zhou et al. [161]) that reports service downtime (labeled with ^{ds} under "AT" column). However, this work only covers one service. In our work, Figures 1-4 testify that our dataset uniquely allows a wide range of analysis related to service downtime.

Finally, we would like to compare COS and our prior work, CBS [137]. CBS is a cloud bug study of more than 3000 bugs reported in repositories of six datacenter distributed systems (Hadoop, Cassandra, HBase, etc.). CBS only analyzes software bugs (B) in a single-layer fashion while COS is a multi-layer (M) analysis that studies high-level outages. In other words, while CBS is a "bottom-up" analysis, COS is more of a "top-down" analysis. As an example, while COS provides proofs of the occurrences of cascading bugs (§4), CBS provides many more detailed patterns of cascading bugs at the source code level (§4 in [137]). Hence, we feel CBS and COS complete each other.

In summary, while COS reiterates many important outage root causes that other studies have highlighted, COS brings new contributions such as our findings and observations of availability landscape of tens of popular cloud services (§3), hidden SPOF root causes including multiple points of failure (MPOF) and cascading bugs (§4), upgrade problems in relation to the full software ecosystem (§5.1), recovery storms (§5.5), and cross-service dependencies (§5.6). To the best of our knowledge, these aspects above are not significantly discussed in other studies we listed in Table 5.

9. Conclusion

Outages continue to happen. Although our dataset only covers up to 2015, high-profile outages in 2016 can easily be found on the web; the inaugural outage of 2016 occurred just within the first week of this new year [79].

A big challenge lies ahead: features and failures are racing with each other. As users are hungry for new advanced features, services are developed in a much rapid pace compared to the past. As a ramification, the complexity of cloud hardware and software ecosystem has outpaced existing testing, debugging, and verification tools. We hope our study can be valuable to cloud developers, operators, and users.

10. Acknowledgments

We thank the anonymous reviewers for their tremendous feedback and comments. This material was supported by funding from NSF (grant Nos. CCF-1336580, CNS-1350499, CNS-1526304, and CNS-1563956) as well as generous donations from Google, NetApp, Huawei, and EMC. We also would like to thank Yohanes Surya and Nina Sugiarto of Surya University for their support.

Any opinions, findings, and conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the NSF or other institutions.

Note: Hyperlinks are embedded behind "(link)".

References

[1] http://ucare.cs.uchicago.edu/projects/cbs/.

[2] After Amazon Outage, Rivals Seek to Capitalize (link), October 23, 2012.

[3] Amazon Addresses EC2 Power Outages (link), May 04, 2010.

[4] Amazon AWS outage knocks Amazon, Netflix, Tinder and IMDb in MEGA data collapse (link), September 20, 2015.

[5] Amazon: Brief Power Outage for Amazon Data Center (link), December 10, 2009.

[6] Amazon: Car Crash Triggers Amazon Power Outage (link), May 13, 2010.

[7] Amazon: DDoS attack rains down on Amazon cloud (link), October 04, 2009.

[8] Amazon EC2 cloud service hit by botnet, outage (link), December 11, 2009.

[9] Amazon: EC2 Maintenance Update (link), September 25, 2014.

[10] Amazon explains its S3 outage (link), February 15, 2008.

[11] Amazon: Lightning Strike Triggers Amazon EC2 Outage (link), June 11, 2009.

[12] Amazon: Network Issues Cause Amazon Cloud Outage (link), September 13, 2013.

[13] Amazon 'Outage' Actually ISP Fiber Line Cut (link), January 06, 2014.

[14] Amazon: Power Outage Affects Amazon Customers (link), June 15, 2012.

[15] Amazon: Summary of the Amazon EC2, Amazon EBS, and Amazon RDS Service Event in the EU West Region (link), August 07, 2011.

[16] Amazon: Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US East Region (link), April 21, 2011.

[17] Amazon: Summary of the AWS Service Event in the US East Region (link), June 29, 2012.

[18] Amazon: Summary of the October 22, 2012 AWS Service Event in the US-East Region (link), October 22, 2012.

[19] Amazon Web Services suffers outage, takes down Vine, Instagram, others with it (link), August 25, 2013.

[20] BlackBerry outage blamed on 'extremely critical' network failure (link), October 10, 2011.

[21] Cloud computing. http://www.oxforddictionaries.com/us/definition/american_english/cloud-computing.

[22] Dropbox: Amazon Web Services suffers partial outage (link), June 14, 2012.

[23] Dropbox: Outage post-mortem (link), January 10, 2014.

[24] Dropbox: Yesterday's Authentication Bug (link), June 19, 2011.

[25] Ebay: Thursday's eBay and PayPal Outages Prove Costly to Sellers (link), October 29, 2015.

[26] Facebook: Facebook Website Experienced 20 Minutes Of Downtime, Mobile App And Website Were Still Working (link), November 30, 2012.

[27] Facebook Goes Down Amid Rollout of New Brand Pages (link), December 16, 2010.

[28] Facebook: In One Fell Swoop, Facebook Glitch Deep-Sixes the Web (link), February 07, 2013.

[29] Facebook Is Down On Web And Mobile (link), September 03, 2014.

[30] Facebook Malware Crackdown Caused App Outage (link), August 13, 2013.

[31] Facebook: More Details on Today's Outage (link), September 23, 2010.

[32] Facebook Outage Silences 150,000 Users (link), October 13, 2009.

[33] Facebook returns to full service, blames DNS update for downtime (link), December 11, 2012.

[34] Facebook says it's back up to 100 percent after maintenance 'issue' (link), February 03, 2014.

[35] Facebook suffers major outage, knocked offline (link), December 10, 2012.

[36] Google: A brief Google outage made total internet traffic drop by 40% (link), August 16, 2013.

[37] Google: About today's App Engine outage (link), October 26, 2012.

[38] Google App Engine Incident #15006 (link), March 05, 2015.

[39] Google App Engine Incident #15025 (link), December 07, 2015.

[40] Google App Engine Issues on Fri September 26th 2014 (link), September 26, 2014.

[41] Google App Engine issues on Thursday June 5th 2014 (link), June 05, 2014.

[42] Google Apps Incident Report Gmail Partial Outage December 10, 2012 (link), December 10, 2012.

[43] Google Apps Incident Report (link), March 17, 2014.

[44] Google blames Gmail outage on data centre collapse (link), February 24, 2009.

[45] Google Drive Down: Gmail Chat, GChat ,Hangouts, Google Docs Get 502 Error, Outage Suffered by Many (link), March 17, 2014.

[46] Google Drive hit by three outages this week - 1st Outage (link), March 18, 2013.

[47] Google: Gmail And Google Drive Are Experiencing Issues (link), April 17, 2013.

[48] Google: Gmail back soon for everyone (link), February 28, 2011.

[49] Google: Gmail hit by service disruption (link), May 08, 2013.

[50] Google: Gmails Down. Heres How to Check Its Status (link), September 01, 2009.

[51] Google goes dark for 2 minutes, kills 40% of world's net traffic (link), August 16, 2013.

[52] Google: More On Gmail's Delivery Delays (link), September 23, 2013.

[53] Google: Post-mortem for February 24th, 2010 outage (link), February 24, 2010.

[54] Google: The Dog Ate Our Homework, Google Drive Is Down (link), October 09, 2015.

[55] Google: Today's outage for several Google services (link), January 24, 2014.

[56] Google: What Happened to Google Docs on Wednesday (link), September 07, 2011.

[57] Google: Widespread Google outages rattle users (link), May 14, 2009.

[58] Icloud: Apple woes continue: Some reporting iCloud down across iOS, Mac, and web (link), September 30, 2014.

[59] iCloud had a big hiccup this morning, several services still experiencing problems (link), February 28, 2013.

[60] Instagram URL Shortener Issues: Instagr Outage Across Facebook, Twitter, Photo App Problems Due to Shortlink Glitch, Photos Not Appearing (link), March 21, 2013.

[61] Instagram Went Down, And It Might Have Been Justin Biebers Fault (link), March 03, 2014.

[62] Messaging app Telegram added 5m new users the day after WhatsApp outage (link), February 24, 2014.

[63] Microsoft: Details of the December 28th, 2012 Windows Azure Storage Disruption in US South (link), December 28, 2012.

[64] Microsoft: Details of the February 22nd 2013 Windows Azure Storage Disruption (link), February 22, 2013.

[65] Microsoft: Details of the Hotmail / Outlook.com outage on March 12th (link), March 12, 2013.

[66] Microsoft: Errant Safety Valve Caused Windows Azure Outage (link), July 26, 2012.

[67] Microsoft: Final Root Cause Analysis and Improvement Areas: Nov 18 Azure Storage Service Interruption (link), November 19, 2014.

[68] Microsoft: Major Microsoft Cloud Outage Blamed on DNS Failure (link), November 21, 2012.

[69] Microsoft Office 365, Azure portals offline for many users in Europe (link), December 03, 2015.

[70] Microsoft SkyDrive suffers outages (link), November 22, 2013.

[71] Microsoft: Summary of Windows Azure Service Disruption on Feb 29th, 2012 (link), February 29, 2012.

[72] Netflix: Amazon AWS Takes Down Netflix On Christmas Eve (link), December 24, 2012.

[73] Office 365: Feb 1 Office 365 Outage Incident Review released (link), February 01, 2013.

[74] Office 365, Google Docs go down again, could give pause to the cloud-wary (link), September 08, 2011.

[75] Office 365: Microsoft fixes Office 365 access after EU-wide outage (link), December 03, 2015.

[76] Office 365: Microsoft Offers Explanations for Lync and Exchange Service Outages (link), June 23, 2014.

[77] Office 365 outage Thursday night (link), August 29, 2013.

[78] Office 365: Update on recent customer issues (link), November 13, 2012.

[79] PSN Is Down In Inaugural Outage Of 2016 [Updates] (link), January 4, 2016.

[80] Psnetwork: PSN Suffers from "Connection Issue", Downtime Just Before Christmas, Error Code 80710092 (link), December 24, 2013.

[81] Psnetwork: Xbox Live And PlayStation Network Are Still Down, And The Hacker Gang Lizard Squad Is Loving It (link), December 25, 2014.

[82] Rackspace: 3 difficult days for Rackspace Cloud Load Balancers (link), May 20, 2013.

[83] Rackspace: Downtime At Rackspace Cloud (link), November 02, 2009.

[84] Rackspace: Network Issue Cited in Rackspace Outage (link), December 18, 2009.

[85] Rackspace: Network Virtualization Platform (NVP) Incident Reports (link), June 20, 2013.

[86] Rackspace Outage & Global DDoS Attack (link), March 27, 2013.

[87] Rackspace Outage Nov 12th (link), November 12, 2014.

[88] Rackspace: Performance Problems for Rackspace Cloud (link), January 14, 2010.

[89] RIM lost $54 million on four-day global BlackBerry outage (link), March 20, 2012.

[90] Salesforce CS10 instance disrupted (link), May 21, 2013.

[91] Salesforce disruption: NA2 and next day, CS1 (link), December 04, 2012.

[92] Salesforce Goes Down in North America and Europe (link), November 14, 2013.

[93] Salesforce: Isolated Data Loss at Salesforce (link), August 16, 2013.

[94] Salesforce: Major Outage for Salesforce.com (link), July 10, 2012.

[95] Salesforce: Major service disruption hits Salesforce.com (link), January 04, 2010.

[96] Salesforce NA11 and NA13 instances disrupted (link), May 26, 2013.

[97] Salesforce.com NA13 instance go down after an upgrade (link), March 19, 2013.

[98] salesforce.com NA8 outage (link), February 01, 2013.

[99] Salesforce.com Outage and Dashboards (link), January 06, 2009.

[100] Skype Holiday Present Down for a Day (link), December 22, 2010.

[101] Twitter: Extremely high volume of whales (link), January 20, 2010.

[102] Twitter Goes Down On Inauguration Day 2013 For Some Users (link), January 21, 2013.

[103] Twitter: Is the World Cup Bringing Down Twitter? (link), June 11, 2010.

[104] Twitter: Our apologies for todays outage. (link), July 26, 2012.

[105] Twitter outage caused by human error, domain briefly yanked (link), October 08, 2012.

[106] Twitter: sign in issue (link), December 28, 2014.

[107] Twitter: Site Availability Issues Due to Failed Enhancement of Our Timeline Cache (link), June 14, 2010.

[108] Twitter Site Back After 90-Minute Outage (link), December 31, 2011.

[109] Twitter: SSL Certificate Errors (link), July 27, 2010.

[110] Twitter: SSL Issue (link), July 13, 2010.

[111] Twitter Stock (TWTR) Dropped $1 During Site Outage (link), March 11, 2014.

[112] Twitter: Tracking down data inconsistencies (link), April 22, 2009.

[113] Twitter: Update on site reliability progress (link), March 25, 2009.

[114] Twitter Worm: A Closer Look at What Happened (link), April 11, 2009.

[115] Wechat: China's WeChat Suffers Major Network Outage (link), July 22, 2013.

[116] WhatsApp Is Down, Confirms Server Issues (link), February 22, 2014.

[117] Xbox: Microsoft Azure and Xbox Live Services Experiencing Outages (link), November 19, 2014.

[118] Xbox: Microsoft To Refund Windows Azure Customers Hit By 12 Hour Outage That Disrupted Xbox Live (link), February 22, 2013.

[119] Xbox: The FBI Is Investigating Hacker Group 'Lizard Squad' Over Xbox Live And Playstation Network Attacks (link), Desember 28, 2014.

[120] Yahoo!: Yahoo Mail suffers yet another outage (link), November 20, 2014.

[121] Mona Attariyan and Jason Flinn. Automating Configuration Troubleshooting with Dynamic Information Flow Analysis. In *OSDI '10*.

[122] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, October 2004.

[123] Lakshmi N. Bairavasundaram, Garth R. Goodson, Shankar Pasupathy, and Jiri Schindler. An Analysis of Latent Sector Errors in Disk Drives. In *SIGMETRICS '07*.

[124] Ritwik Banerjee, Abbas Razaghpanah, Luis Chiang, Akassh Mishra, Vyas Sekar, Yejin Choi, and Phillipa Gill. Internet Outages, the Eyewitness Accounts: Analysis of the Outages Mailing List. In *Proceedings of Passive and Active Measurement Conference (PAM)*, 2015.

[125] Theophilus Benson, Sambit Sahu, Aditya Akella, and Anees Shaikh. A First Look at Problems in the Cloud. In *HotCloud '10*.

[126] Robert Birke, Ioana Giurgiu, Lydia Y. Chen, Dorothea Wiesmann, and Ton Engbersen. Failure Analysis of Virtual and Physical Machines: Patterns, Causes and Characteristics. In *DSN '14*.

[127] Peter Bodik, Armando Fox, Michael Franklin, Michael Jordan, and David Patterson. Characterizing, Modeling, and Generating Workload Spikes for Stateful Services. In *SoCC '10*.

[128] Aaron B. Brown and David A. Patterson. To Err is Human. In *EASY '01*.

[129] Xin Chen, Charng-Da Lu, and Karthik Pattabiraman. Failure Analysis of Jobs in Compute Clouds: A Google Cluster Case Study. In *IEEE 25th International Symposium on Software Reliability Engineering (ISSRE)*, 2014.

[130] Thanh Do, Mingzhe Hao, Tanakorn Leesatapornwongsa, Tiratat Patana-anake, and Haryadi S. Gunawi. Limplock: Understanding the Impact of Limpware on Scale-Out Cloud Systems. In *SoCC '13*.

[131] Nosayba El-Sayed and Bianca Schroeder. Reading between the lines of failure logs: Understanding how HPC systems fail. In *DSN '13*.

[132] Daniel Ford, Franis Labelle, Florentina I. Popovici, Murray Stokely, Van-Anh Truong, Luiz Barroso, Carrie Grimes, and Sean Quinlna. Availability in Globally Distributed Storage Systems. In *OSDI '10*.

[133] Peter Garraghan, Paul Townend, and Jie Xu. An Empirical Failure-Analysis of a Large-Scale Cloud Computing Environment. In *HASE '14*.

[134] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications. In *SIGCOMM '11*.

[135] Jim Gray. Why Do Computers Stop and What Can We Do About It? In *6th International Conference on Reliability and Distributed Databases*, June 1987.

[136] Haryadi S. Gunawi, Thanh Do, Pallavi Joshi, Peter Alvaro, Joseph M. Hellerstein, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, Koushik Sen, and Dhruba Borthakur. FATE and DESTINI: A Framework for Cloud Recovery Testing. In *NSDI '11*.

[137] Haryadi S. Gunawi, Mingzhe Hao, Tanakorn Leesatapornwongsa, Tiratat Patana-anake, Thanh Do, Jeffry Adityatama, Kurnia J. Eliazar, Agung Laksono, Jeffrey F. Lukman, Vincentius Martin, and Anang D. Satria. What Bugs Live in the Cloud? A Study of 3000+ Issues in Cloud Systems. In *SoCC '14*.

[138] Zhenyu Guo, Sean McDirmid, Mao Yang, Li Zhuang, Pu Zhang, Yingwei Luo, Tom Bergan, Madan Musuvathi, Zheng Zhang, and Lidong Zhou. Failure Recovery: When the Cure Is Worse Than the Disease. In *HotOS XIV*.

[139] James Hamilton. Inter Datacenter Replication and Geo-Redundancy. `http://perspectives.mvdirona.com/2010/05/inter-datacenter-replication-geo-redundancy/`, 2010.

[140] Mingzhe Hao, Gokul Soundararajan, Deepak Kenchammana-Hosekote, Andrew A. Chien, and Haryadi S. Gunawi. The Tail at Store: A Revelation from Millions of Hours of Disk and SSD Deployments. In *FAST '16*.

[141] Weihang Jiang, Chongfeng Hu, Yuanyuan Zhou, and Arkady Kanevsky. Are Disks the Dominant Contributor for Storage Failures? A Comprehensive Study of Storage Subsystem Failure Characteristics. In *FAST '08*.

[142] Pallavi Joshi, Haryadi S. Gunawi, and Koushik Sen. PREFAIL: A Programmable Tool for Multiple-Failure Injection. In *OOPSLA '11*.

[143] Tanakorn Leesatapornwongsa and Haryadi S. Gunawi. The Case for Drill-Ready Cloud Computing. In *SoCC '14*.

[144] Tanakorn Leesatapornwongsa, Mingzhe Hao, Pallavi Joshi, Jeffrey F. Lukman, and Haryadi S. Gunawi. SAMC: Semantic-Aware Model Checking for Fast Discovery of Deep Bugs in Cloud Systems. In *OSDI '14*.

[145] Tanakorn Leesatapornwongsa, Jeffrey F. Lukman, Shan Lu, and Haryadi S. Gunawi. TaxDC: A Taxonomy of Non-Deterministic Concurrency Bugs in Datacenter Distributed Systems. In *ASPLOS '16*.

[146] Sihan Li, Hucheng Zhou, Haoxiang Lin, Tian Xiao, Haibo Lin, Wei Lin, and Tao Xie. A Characteristic Study on Failures of Production Distributed Data-Parallel Programs. In *ICSE '13*.

[147] Yinglung Liang, Yanyong Zhang, Morris Jette, Anand Sivasubramaniam, and Ramendra Sahoo. BlueGene/L Failure Analysis and Prediction Models. In *DSN '06*.

[148] Edmund B. Nightingale, John R Douceur, and Vince Orgovan. Cycles, Cells and Platters: An empirical analysis of hardware failures on a million consumer PCs. In *EuroSys '11*.

[149] David L. Oppenheimer, Archana Ganapathi, and David A. Patterson. Why Do Internet Services Fail, What Can Be Done About It? . In *USITS '03*.

[150] JR Raphael. The worst cloud outages of 2013 (part 2). http://www.infoworld.com/article/2606921/cloud-computing/133109-The-worst-cloud-outages-of-2013- part-2.html, 2013.

[151] JR Raphael. The worst cloud outages of 2014 (so far). http://www.infoworld.com/article/2606209/cloud-computing/162288-The-worst-cloud-outages-of-2014-so-far.html, 2014.

[152] Swarup Kumar Sahoo, John Criswell, and Vikram Adve. An Empirical Study of Reported Bugs in Server Software with Implications for Automated Bug Diagnosis. In *ICSE '10*.

[153] Bianca Schroeder and Garth A. Gibson. A large-scale study of failures in high-performance computing systems. In *DSN '06*.

[154] Bianca Schroeder and Garth A. Gibson. Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you? In *FAST '07*.

[155] Riza O. Suminto, Agung Laksono, Anang D. Satria, Thanh Do, and Haryadi S. Gunawi. Towards Pre-Deployment Detection of Performance Failures in Cloud Distributed Systems. In *HotCloud '15*.

[156] Joseph Tsidulko. The 10 Biggest Cloud Outages Of 2015 (So Far). http://www.crn.com/slide-shows/cloud/300077635/the-10-biggest-cloud-outages-of-2015-so-far.htm, 2015.

[157] Daniel Turner, Kirill Levchenko, Alex C. Snoeren, and Stefan Savage. California fault lines: understanding the causes and impact of network failures. In *SIGCOMM '10*.

[158] Kashi Vishwanath and Nachi Nagappan. Characterizing Cloud Computing Hardware Reliability. In *SoCC '10*.

[159] Praveen Yalagandula, Suman Nath, Haifeng Yu, Phillip B. Gibbons, and Srinivasan Seshan. Beyond Availability: Towards a Deeper Understanding of Machine Failure Characteristics in Large Distributed Systems. In *WORLDS '04*.

[160] Zuoning Yin, Xiao Ma, Jing Zheng, Yuanyuan Zhou, Lakshmi N. Bairavasundaram, and Shankar Pasupathy. An Empirical Study on Configuration Errors in Commercial and Open Source Systems. In *SOSP '11*.

[161] Hucheng Zhou, Jian-Guang Lou, Hongyu Zhang, Haibo Lin, Haoxiang Lin, and Tingting Qin. An Empirical Study on Quality Issues of Production Big Data Platform. In *ICSE '15*.