VOID

2022
VOID REPORT

Get Ahead of the Incident Curve

IN PARTNERSHIP WITH
indeed jeli

Foreword

Around ten years ago I was one of the on-call managers at Netflix and I wrote some of the incident reviews that we published. We'd built a resilient system that continued to work in the presence of failures, and were pioneering Chaos Engineering along with regular game day exercises. However we still had major incidents, and they tended to be incidents that we'd never seen before, a series of unfortunate events conspiring together to find a weak spot or a bad assumption on our part. We were a learning organization, but there were very few relevant published incident reviews from others to help us accelerate our learning and harden our systems in advance.

The Verica Open Incident Database (VOID) fills this need, and is riding two trends. The first is that the amount of software that is performing safety and business critical automation is increasing rapidly, so it matters more. The second is that more organizations are publishing incidents and the "New View of Safety" ideas are spreading. Safety is the capability to absorb an incident, not the absence of failure, and incident reviews are blameless learning opportunities. Although it may seem counter intuitive, airlines that record the highest incident rate have the best safety record.

If you aren't recording and publishing incidents because you want to look good, then you are more likely to have a much bigger failure. This report raises some interesting questions, how can we measure near-misses, and can we find a better metric than Mean Time To Repair (MTTR) given the complex partial failure modes we see? I encourage everyone to publish more, include near misses in your incident reports, and to help everyone else build a safer world as a result.

Adrian Cockcroft
Partner, OrionX & Tech Advisor

Contents

Get Ahead of the Incident Curve

The **Verica Open Incident Database (VOID)** was created with a nod to the aviation industry's efforts to study and learn from both incidents and formal accidents. Our explicit goals are to raise awareness and increase understanding of software-based failures to make the internet a more resilient and safe place, akin to how sharing such information improved the aviation industry's safety record.

Aviation is one of the industries where failures have some of the highest stakes, here is one such example: On November 11, 2018, three pilots arrived in Portugal to **ferry an Air Astana jet back to Kazakhstan** after a round of mandated, heavy maintenance. Flying with them were three maintenance crew members who had performed weeks of complex, and at times, confusing work on the Embraer ERJ-190. Immediately after takeoff, the pilots knew they had problems. Still, the full scope of the issues only became apparent after a harrowing two hours during which they almost lost control of the plane numerous times. A copilot became sick and wasn't able to continue to help fly the plane, and another crew member was injured trying to look outside the plane while it veered wildly above the city of Lisbon. Even after determining that the issue was due to incorrectly installed aileron cables (which control a plane's roll) they barely managed to land safely on an emergency runway. Ultimately, the plane was so severely damaged that it was scrapped and never flown again.

After an extensive investigation, the conclusion of the Portuguese investigating body was eerily familiar to detailed software outage reports:

"In the end, Portuguese investigators would uncover a series of design flaws, poor decisions, and procedural errors which caused the plane to depart for Kazakhstan with its ailerons hooked up the wrong way around... The fact that this lack of knowledge was so widespread led investigators to conclude that the problem likely began on an organizational level."

As we said in the inaugural 2021 VOID report, software now runs "transportation, infrastructure, power grids, healthcare software and devices, voting systems, autonomous vehicles, and many critical societal functions. These systems are inherently (often by design) complex sociotechnical systems. They comprise code, machines, and the humans who maintain them, operating via their mental models of an environment shaped by multiple competing pressures, many of which aren't always readily apparent." These pressures, mental models, and interactions within complex sociotechnical systems only become more apparent when we seek to investigate and learn from their failures.

Executive Summary

This past year, the VOID grew from 2,000 to nearly 10,000 incident reports from close to 600 organizations. We rigorously collect the same metadata, along with a new value: severity. This enabled us to investigate whether there is a relationship between the reported length of the incident and the impact (or severity) of the incident. Prior to this report, that common assumption hasn't been evaluated with data.

VOID Metadata

- Organization name
- Organization industry
- Organization size (# of employees)
- Date of incident
- Date of report
- Report type (Status report, postmortem, media article, etc)
- Duration (in hours/minutes)
- Severity (None, Minor, Major, Critical)
- Technologies involved (DNS, Kubernetes, database, etc)
- Impact type (Full/partial outage, performance, increased errors, etc)
- Analysis format (RCA, etc)

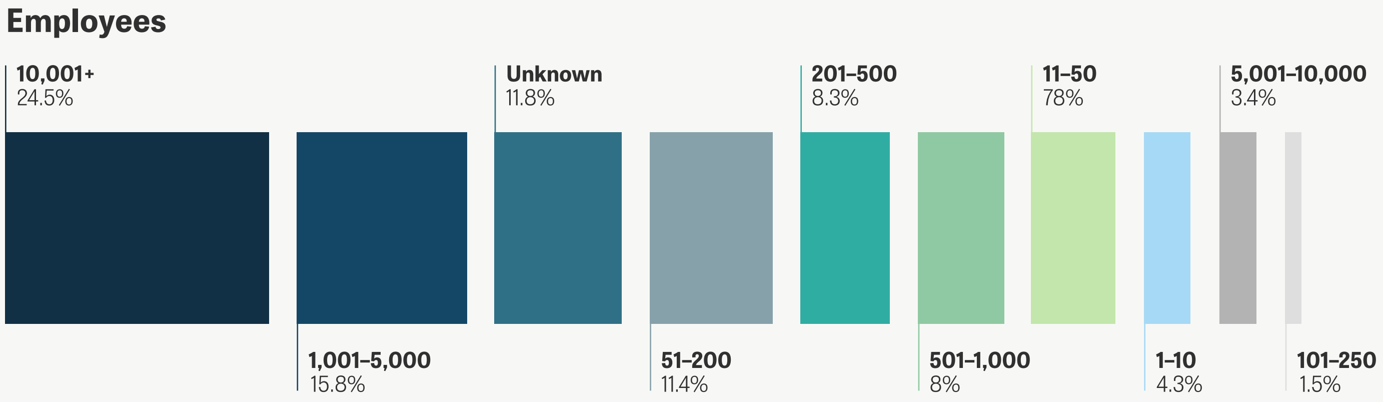
Additionally, the 400% growth in the number of incidents in the VOID provided the opportunity to both confirm patterns we first noted in 2021 (along with replicating other's research results), and perform new statistical analyses on key incident metadata.

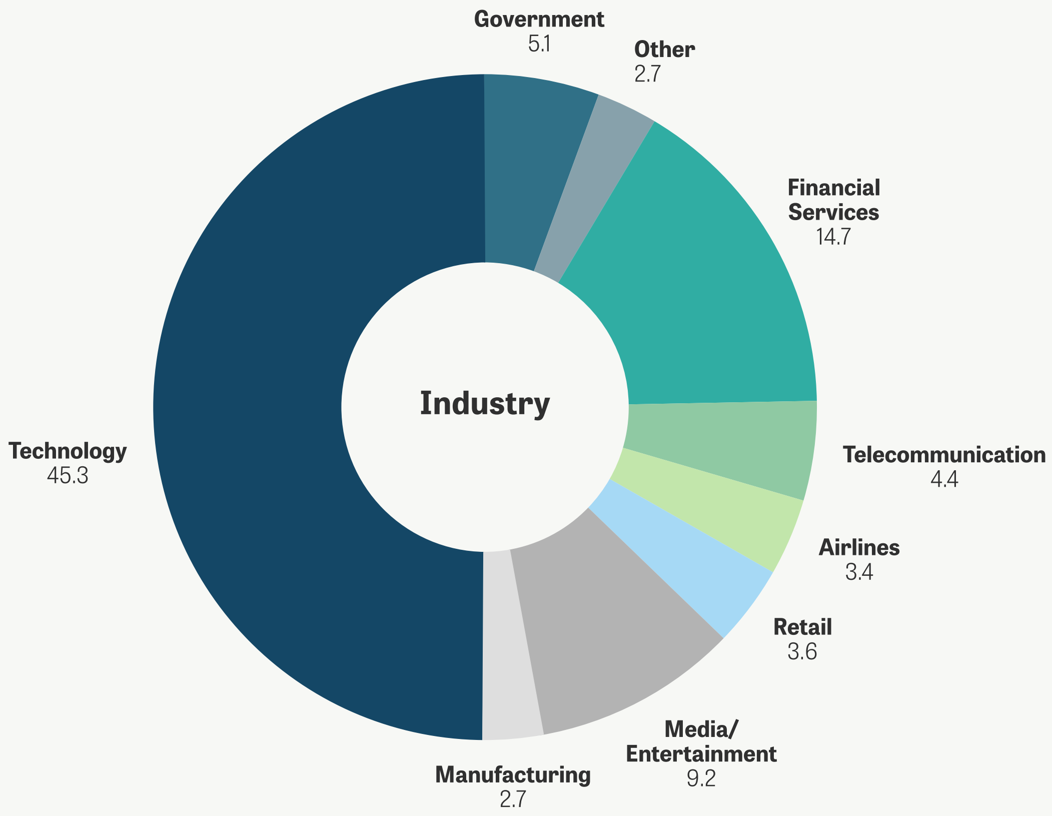
KEY FINDINGS:

- **No company is immune from incidents.** Incidents happen in organizations of all sizes, from startups to the Fortune 10. Software is mission-critical in every possible industry including banking, travel, agriculture, commerce, and more.
- **Length isn't as cut and dry as it appears: there are many insightful metrics to measure in an incident.** Duration of incidents conveys little meaning about the incidents themselves, in part because it can be very tricky to attribute when incidents start or stop.
- **SREs and others in similar roles should retire MTTR as a key metric.** This year's report confirms that MTTR isn't a viable metric for the reliability of complex software systems for a myriad of reasons, notably due to its underlying variance.
- **Common assumptions around incident duration and severity are debunked.** Analyzing thousands of incidents, we demonstrate that incident duration and severity are not related.
- **Organizations are moving away from short-sighted approaches like RCA.** Root Cause Analysis appears to be on the decline in orgs of all sizes, as they move toward more meaningful metrics and analysis.

New For 2022: Organizational Demographics

We have just under 600 organizations represented in the VOID. They vary by industry and size, with the majority representing the technology/software industry.[1]





[1] *These metadata were collected from LinkedIn using RapidAPI. Data collection, cleaning, and analysis were all done by William Ou, a PhD candidate at the University of British Columbia, during a summer internship with the VOID.*

Delving Deeper Into Duration

“The more deeply we study the nature of time, the better we understand that duration means invention, creation of forms, continuous elaboration of the absolutely new.”

– Henri Bergson

We continue to investigate duration data, commonly calculated as the end time minus the start time, as they are the underlying source of MTTR (Mean Time to Resolve), which we look at more closely in the next section. Duration is a form of what we call Gray Data, in that duration is:

- **High in variability, low in fidelity**
- **Fuzzy on both ends (start/stop)**
- **Sometimes automated, often not**
- **Sometimes updated, sometimes not**
- **A lagging indicator of what happened in the past in your system(s)**
- **Inherently subjective**

As an example of duration being negotiably updated, and therefore entirely subjective, consider this real scenario from a senior leader at a large enterprise technology company:

“A code change interacted with a latent defect and resulted in intermittent data corruption that was discovered one week later. The code change was reverted about a day after that. Over the following week, there was a bunch of work done to temporarily resolve the data corruption in downstream systems. This was necessary in order to meet a hard deadline.

Following this, the team worked for about 3 weeks to permanently address the data corruption in the original system, and managed to complete this before the next deadline. Did the incident end when the problematic code was reverted? Did the incident end when the temporary changes were made in time for the first deadline? Or did the incident end when the permanent changes were made in time for the second deadline? In this case, they considered the incident closed after the first deadline was reached successfully. A different team might have left it open until the permanent changes were completed.”

Another real incident seemed to defy the laws of physics, providing what was technically a *negative* incident duration:

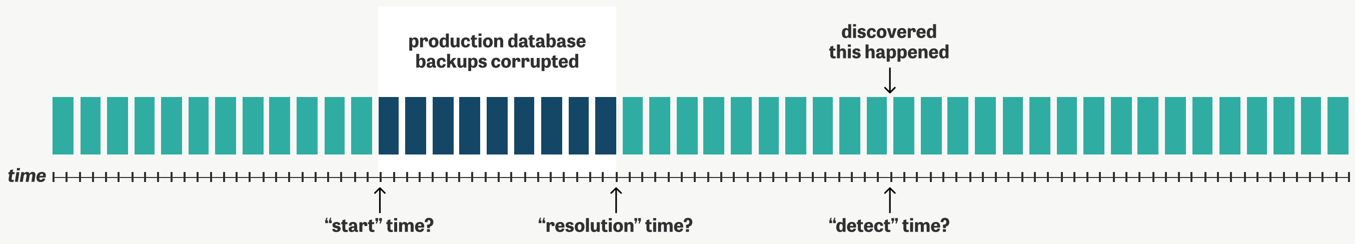


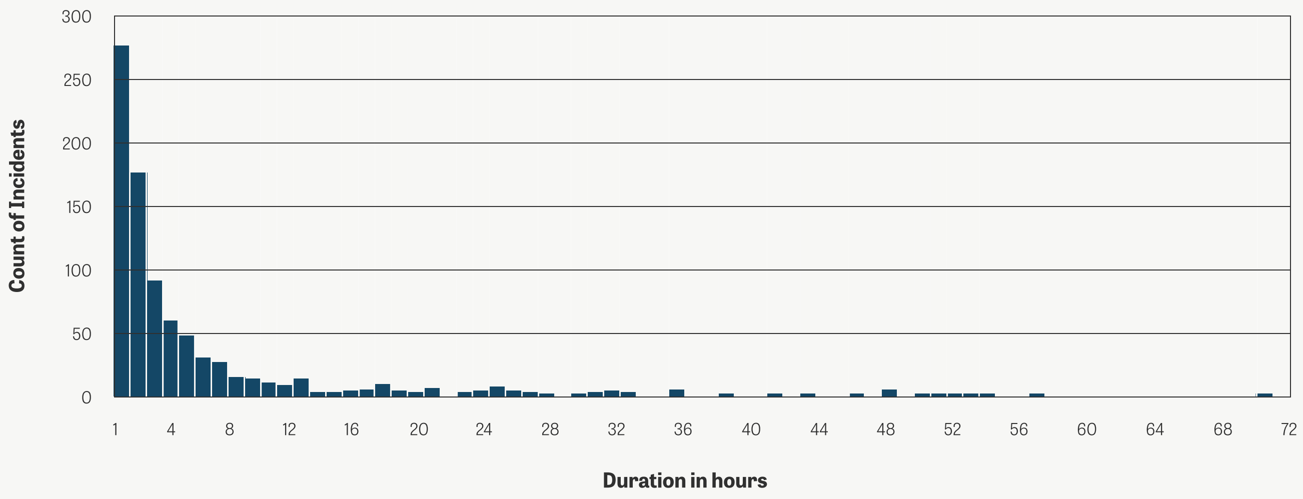
Figure 1: When "detect" time is actually after resolution time. (Source: John Allspaw)

In this incident, a bug prevented production database backups from being valid. An unrelated Chef change accidentally fixed the bug, and this period of bad backups wasn't detected until after it was fixed. One could imagine a scenario where the team decides to go back and update the start time to be when the database corruption started in order to have a more tidy (and calculable) TTR value.

Duration Distribution Remains Skewed

The underlying distribution of duration data gives a view of how varied these data sets can be. In the 2021 report, we noted that duration data were skewed, meaning the majority of incidents had durations under two hours, and the remaining incident durations fell off rather quickly.

Figure 2: Distribution of 2021 VOID duration data



The distribution of duration data (Figure 2) matters. Being skewed instead of normally distributed (in the familiar bell curve shape), means that central tendency measures like the mean aren't accurate representations of the underlying data. This is more closely examined in the next section on MTTR.

After adding 7,600+ additional incident reports, we found this skewed distribution of duration data hasn't changed. It was possible that the smaller set collected in 2021, had distributions that weren't consistent with other organizations or over specific timeframes, making comparisons with last year's data difficult and potentially invalidating some of the conclusions we reached. But this was not the case. Figure 3 shows the distribution of duration data from 12 companies to illustrate the consistency of this phenomenon.

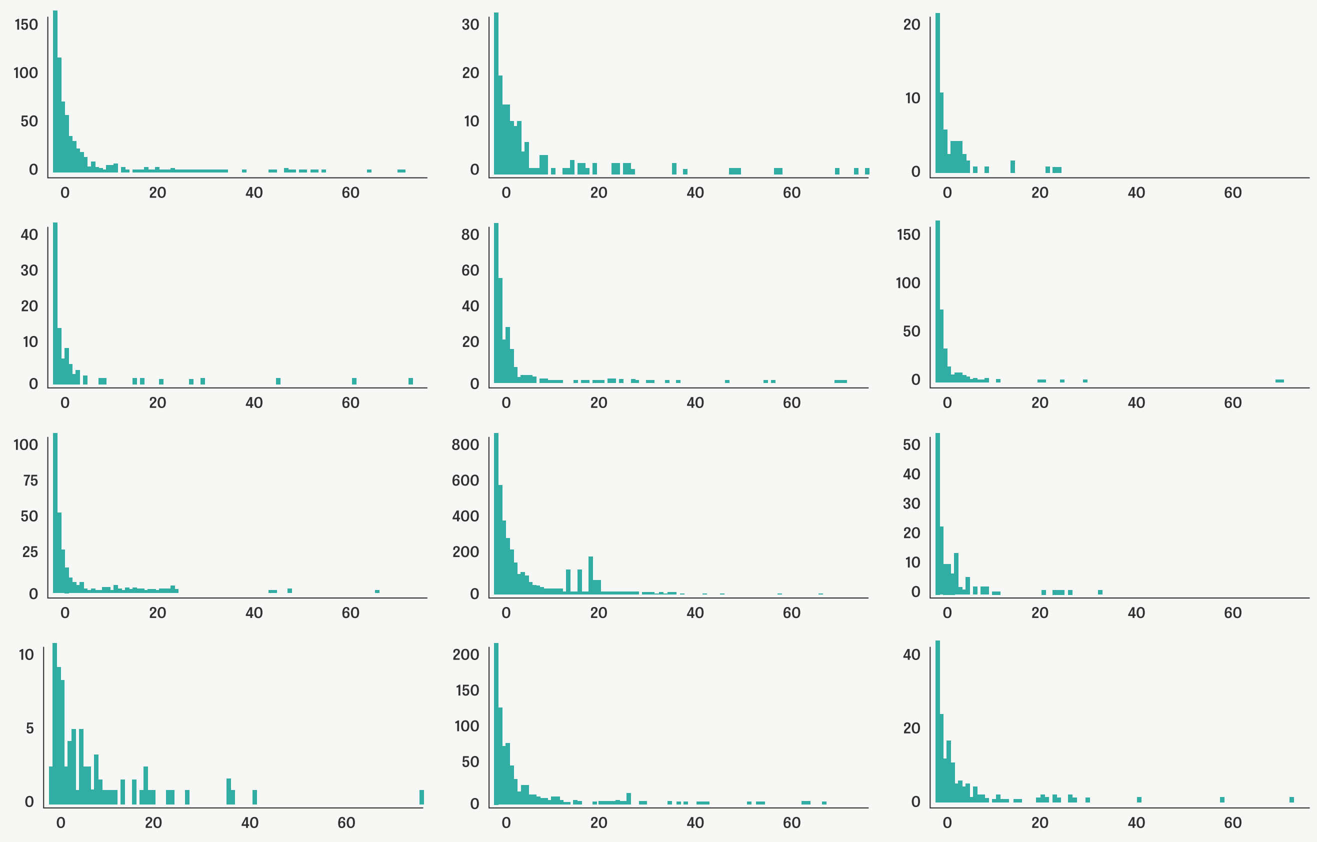


Figure 3: Distribution of duration data from 12 different companies' incidents from the VOID.

Distribution Fitting

While we don't plan to use transformed data in any analyses for this report, we did seek to understand whether incident duration fit a log normal distribution. An excellent place to explore the distribution is with histograms (as shown in Figure 3, above) paired with density plots for the distribution in question, paired CDF (Cumulative Distribution Function) charts, and Q-Q and P-P plots, which are descriptive statistics that provide goodness of fit visualizations.[2] A detailed description of these approaches is beyond the scope of this report. The key point is that if your data in question are a good fit for your chosen distribution, each chart should be visually similar.

Figure 4 shows these charts for a representative company's duration data, which were similar to all the other companies.

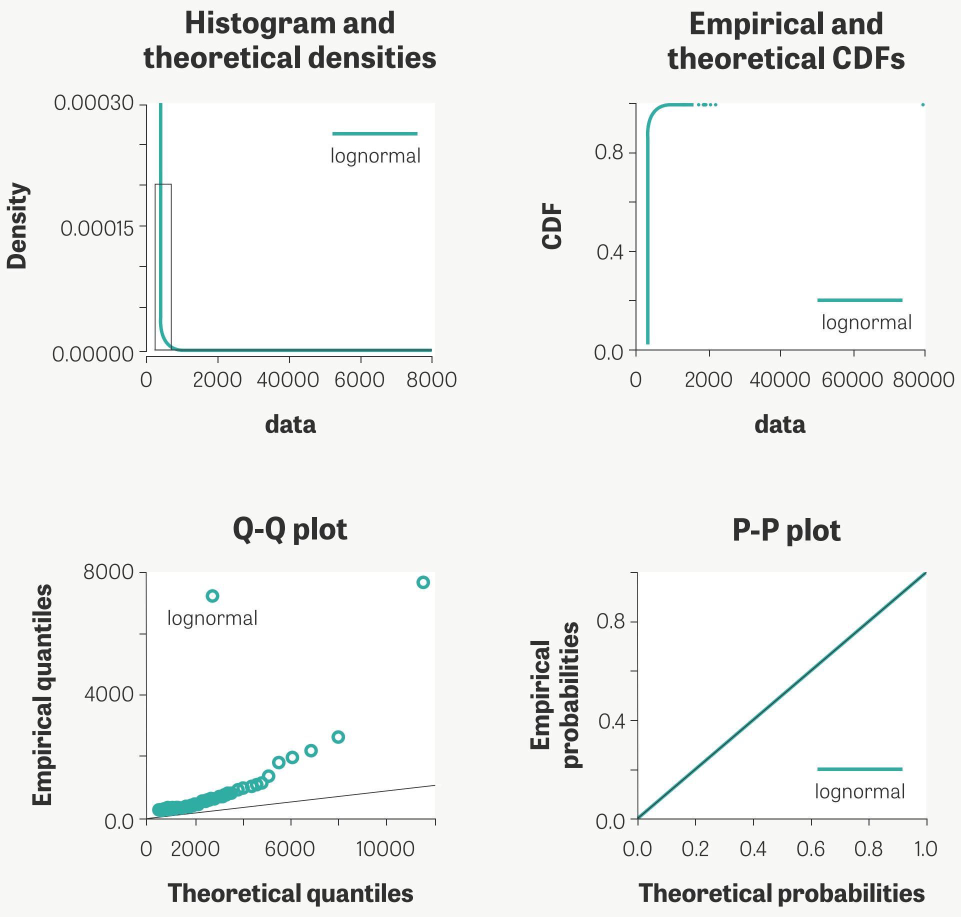


Figure 4. Histogram/density chart, empirical and theoretical CDFs, Q-Q plot and P-P plot for log transformed duration data from a single company.

All but the Q-Q plot appear to show a good fit between the duration data and a log normal distribution. Q-Q plots are more commonly used when evaluating skewed data as they are more sensitive to lack of fit in the tails, which can signal the presence of some process driving the extreme values. Based on the Q-Q plot, it is not clear that log normal is a good fit for incident duration data.

[2] https://towardsdatascience.com/explaining-probability-plots-9e5c5d304703

Log Transformation Distribution of Duration Data

As mentioned above, if a set of data are log normally distributed, then transforming them should result in a normal distribution of the transformed data. Figure 5 shows a sample of multiple companies' log transformed data distributions. While a few transformed histograms do appear to be visibility normally distributed, many are not.

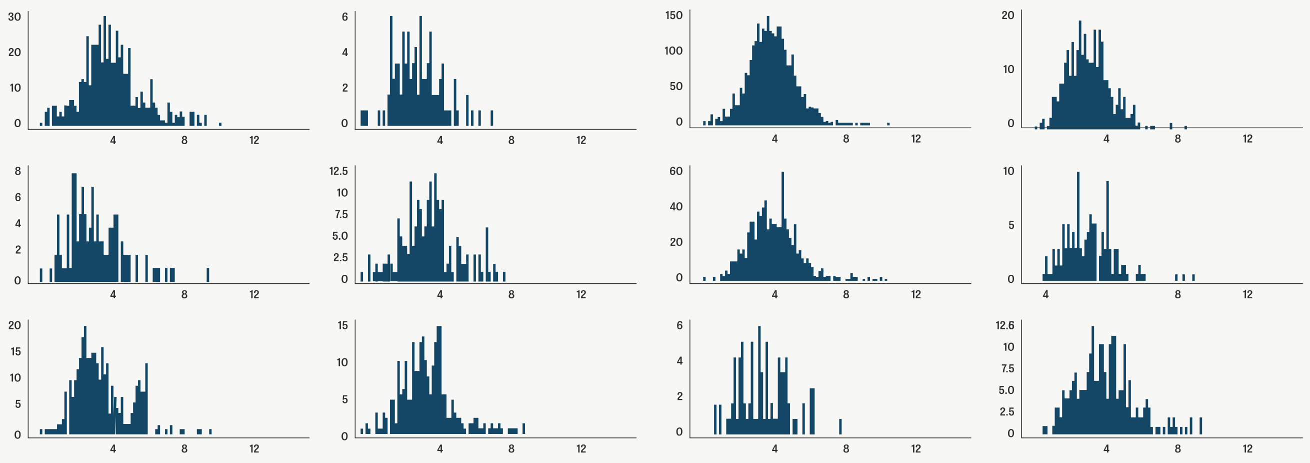


Figure 5. Distribution of log-transformed duration data from 12 companies in the VOID. Each chart represents a different company.

Additionally, preliminary fitting analyses of incident duration data to a log normal distribution do not indicate statistical significance for that distribution (nor for other distributions like Weibull or Gamma. However, the exact fit is less relevant to our needs here than what we already know: the long tail of distribution data merits further exploration to begin searching for patterns or clues, which we plan to undertake for next year's report.

We also have one final caveat regarding the prospect of transforming incident duration data. Even if you could effectively log-transform your data, any experiments or statistics related to those transformed data may not address the hypothesis of interest regarding the original data.[3] So if you're trying to eliminate the impact of outliers on metrics like MTTR, you're welcome to try, but it won't tell you if MTTR in its original context is increasing or decreasing.

In the next section we'll take a closer look at MTTR to illustrate its limitations.

[3] *Feng et al (2014). Log-transformation and its implications for data analysis. Shanghai Arch Psychiatry*

Moving on From MTTR

Last year, we focused on duration metadata from the reports in the VOID, as these data are the underlying input to Mean Time to Resolve (MTTR). MTTR originated in manufacturing organizations and was a measure of the average time required to repair a failed physical component or device. Such devices had simpler, predictable operations with wear and tear that lent themselves to reasonably standard and consistent estimates of MTTR. However, over time the use of MTTR has expanded to software systems, software companies view it as an indicator of system reliability and team agility/effectiveness.

We suspected that MTTR was not an appropriate metric for complex software systems, in part because of the distribution of duration data and because "failures" in such systems don't arrive uniformly over time. Each failure is inherently different, unlike issues with physical manufacturing devices. Operators of modern software systems regularly invest in improving the reliability of their systems, only to be caught off guard by unexpected and unusual failures. With the VOID data, we had a unique opportunity to use real incident data from the industry that showed that MTTR was not descriptive of system reliability.

In last year's report, we compared the distribution of the duration data in the VOID to what we saw in Štěpán Davidovič's **Incident Metrics in SRE: Critically Evaluating MTTR and Friends** report. We relied on inference based on the similarity of our data distributions to conclude that our data followed the same pattern Davidovič found in his Monte Carlo simulations (more on this below). He found that the large amount of variance in duration data rendered MTTR useless because changes were effectively impossible to detect. However, we did not have experimental validation of our data, so we first sought to replicate his results.

Experiment 1: Replication of Davidovič's MTTR Simulations

Methodology

For our first experiment, we sought to replicate Davidovič's approach, which collected incident data from three public companies along with internal incident data he had access to at Google. He ran Monte Carlo simulations on those data comparing original, unaltered incident durations with a set of incidents that had their durations intentionally reduced by 10%. His methodology was as follows:

Assume that the incidents follow the empirically observed distribution of the obtained data sets and evaluate what kinds of improvements you would see after a certain number of incidents—and with what confidence level.

1

Randomly draw two samples, with sizes N_1 and N_2 (where N1 = N2 to get a perfect 50/50 split), from the empirical distribution of incident durations.

2

Modify the incident durations in one of the populations, in this case, by shortening it by 10%.

3

Calculate MTTR for each of the groups, i.e., $MTTR_{modified}$ and $MTTR_{unmodified.}$

4

Take the difference, observed improvement = $MTTR_{unmodified} - MTTR_{modified.}$ (A negative difference means MTTR is worsening.

5

Repeat this process 100,000 times.

He selected duration data from a single year (2019), and limited the experiment to incidents that were greater than three minutes and less than three days (72 hours).[4]

For our experiment, the duration data came from 12 different companies, collected primarily from status pages. Like Davidovič, we limited our data to a single year, 2021, and filtered the durations to be between three minutes and three days. Table 1 shows the count of incidents within that time frame along with the calculated MTTR.

Company	Count	MTTR (hrs)
Company A	119	6
Company B	30	6
Company C	430	3
Company D	17	2
Company E	27	10
Company F	131	4
Company G	95	5
Company H	74	4
Company I	27	4
Company J	163	2
Company K	27	4

Table 1. Incident Count and MTTR for all 2021 data constrained between three minutes and three hours.

For each of the 12 companies, we followed Davidovič's approach (described above) and randomly assigned half of the filtered durations from that year into a control group (N_1) and assigned the other half into a test group (N_2). The test group received a 10% reduction across the board in the duration of each incident. We then ran 100,000 Monte Carlo simulations comparing control ($MTTR_{unmodified}$) to test ($MTTR_{modified}$), calculating the difference in MTTR between the two ($MTTR_{unmodified}$ - $MTTR_{modified}$). A negative value means that the MTTR got worse (e.g., longer), and a positive difference means the MTTR got better (e.g., shorter).[5]

[4] *There are potential issues with this decision, notably to remove the very end of the long tail of incident duration. We look at this more in later experiments.*
[5] *The R code for this analysis was written by William Ou, a PhD candidate at the University of British Columbia, during a summer internship with the VOID.*

Findings

Figure 6 recaps Davidovič's 2021 findings: even when reducing incident duration by 10% in the test group, the calculated MTTR *increased* between 20-40% of the time.

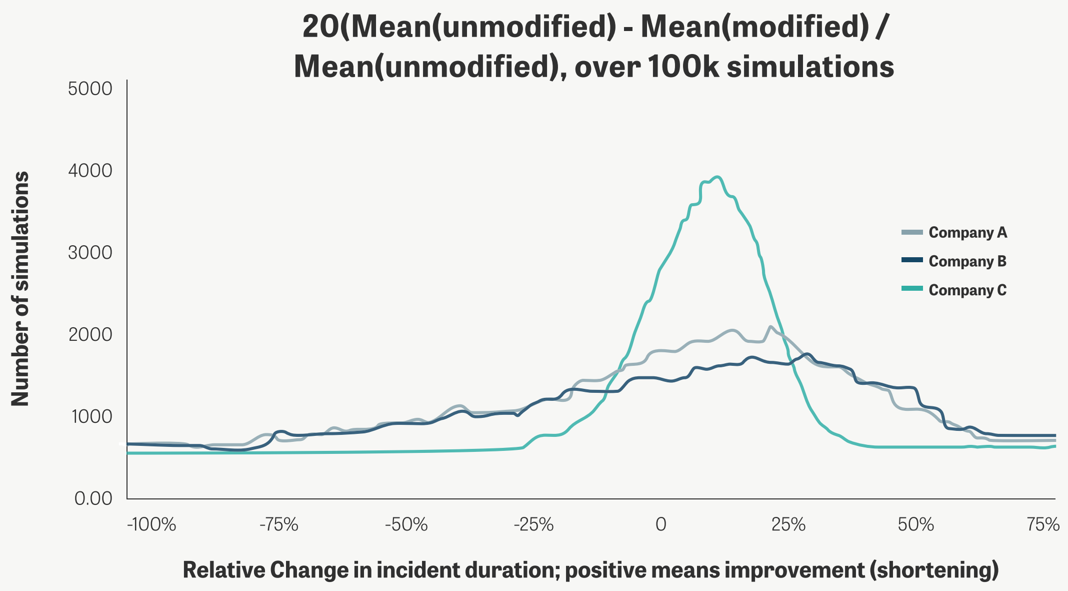


Figure 6. Davidovič's results from 2021 (credit: O'Reilly Media and Google)

Our results were very similar to Davidovič's (Figure 7, below). Reducing incident duration by 10% did not result in a reliable reduction in the calculated MTTR regardless of the sample size (e.g. total number of incidents). We explain this in more detail below.

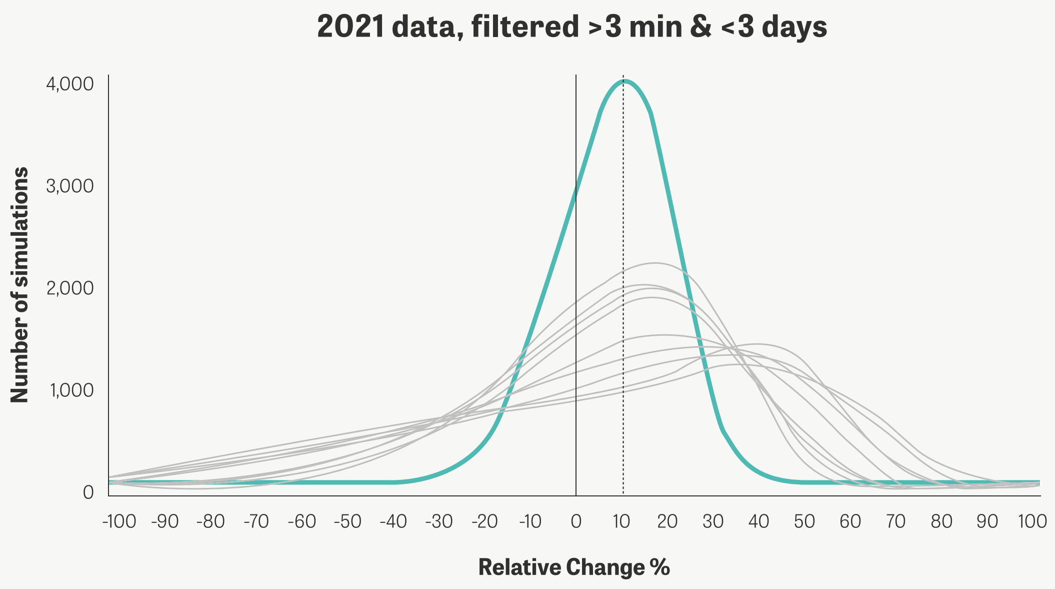


Figure 7. Distribution of simulated changes to VOID MTTR data given a 10% reduction in incident duration for the test group.

One Of These Curves Is Not Like The Other: Impact of Sample Size

You may notice that in Davidovič's results and ours, one company seemed to stand out, with a much taller, tighter curve around the 10% relative change mark. This tighter curve indicates that the MTTR for the treatment group wasn't as impacted by variance in the data, and more accurately reflected the 10% reduction in duration. In both experiments the company with the tall, tight curve had significantly more incidents than the other companies. This maverick curve also appears to be possibly normally distributed!

This led Davidovič to further explore the impact of sample size on confidence intervals of the simulated experimental data. A *confidence interval* is a numerical range of values likely to include a value from an experimental population within a certain degree of confidence (typically set at 90% or 95%). As the sample size increases, your ability to detect smaller changes should also increase (and perhaps become statistically significant).

Davidovič ran another series of simulations where he compared the means for three different sample sizes of incidents ranging from 10 to 100 to 1,000. He demonstrated that even with 1,000 simulated incidents, the 10% reduction in duration would still fall in the 90% confidence interval, meaning that the changed incident durations are not distinguishable from incidents with unchanged durations.

In less statistical terms, reducing incident duration by 10% does not guarantee a noticeable reduction in MTTR, regardless of the sample size. More incidents will theoretically help you get (slightly) better signal, but who wants MORE incidents?! You will still have a signal-to-noise problem with respect to MTTR.

Experiment 2: Monte Carlo Analysis Without Duration Filtering

We wanted to evaluate the impact of Davidovič's decision to eliminate certain durations, notably the extremely long tail ones that were greater than three days. In this case, we repeated the same methodology as above. However we took all incident duration data for the same set of companies from 2021 instead of only those greater than three minutes and less than three days.

Table 2 shows how the MTTR for the unfiltered data changed dramatically for at least a few of the companies, in some cases increasing the MTTR value by between 16–19 hours, even when the incident count barely changed.

	Data filtered		Outliers included		
Company	**Count**	**MTTR (hrs)**	**Count**	**MTTR (hrs)**	**Increase in MTTR**
Company A	119	6	129	22	16
Company B	30	6	32	12	6
Company C	430	3	439	4	1
Company D	17	2	18	21	19
Company E	27	10	27	10	0
Company F	131	4	138	15	11
Company G	95	5	96	7	2
Company H	74	4	74	4	0
Company I	27	4	28	4	0
Company J	163	2	163	2	0
Company K	27	4	28	13	9

Table 2: Comparing Incident count and MTTR when all duration data are included

For a couple of companies, the results were very similar to the original experiment, but in other cases, they were surprisingly different. For companies with at least one or more very long incidents (greater than 72 hours), the results of the simulations are not nearly as tidy as the previous experiment with filtered data. A 10% change in duration yields MTTR differences that look more like audio waveforms than predictable distributions (or, as in one company's results—Company K—a change in MTTR that pegs it at almost a 100% increase overall!).

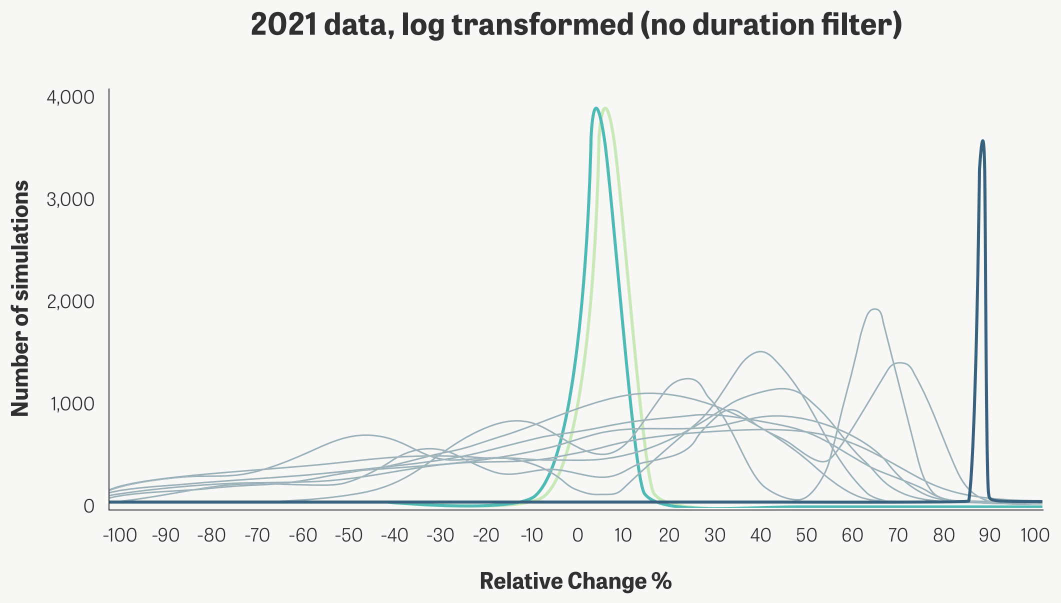


Figure 8. Monte Carlo simulations for 2021 log transformed data with no duration filter

The original methodology of filtering out incidents from the Monte Carlo simulations that are over three days effectively removes the biggest "outliers." However, these aren't really *outliers* in the strictly statistical sense, where it is reasonable to exclude them due to errors in measurement or other aspects of experimental data collection. These are incidents that did in fact happen! There is nothing in the data collection mechanism leading us to believe long or short incidents are any less or more valid than the others, or that their impact is worth dismissing. Excluding these seeming outliers from analyses makes your data look cleaner and easier to understand, but it also gives you an incomplete, and inaccurate, picture.

The complete picture for these incidents is inherently messy. Our results highlight how much the extreme variance in duration data can impact calculated changes in MTTR. In this case, Company K (the far-right curve near the 100% change mark in Figure 8) is likely to think its MTTR had actually gotten better *by much more than 10%*, even though that was not the case. They would probably think they have improved their processes and response in astounding ways. This may give them a sense of confidence that they are a very high-reliability organization that is doing things right. But, this perceived gain is the result of averaging across high-variance data.

“Incidents are untyped pointers to areas of interest in their system. The ‘untyped’ part is critical: the work of decoding what has happened is entirely left to the analyst.”

– Richard Cook

Alternatives To MTTR

At this point, we're frequently asked “What metric can replace MTTR?”. The answer is that we never should have used a single averaged number to try to measure or represent the reliability of complex sociotechnical systems. No matter what your (unreliable) MTTR might seem to indicate, you’d still need to investigate your incidents to understand what is truly happening with your systems. Qualitative *incident analysis* is the ideal path to finding appropriate replacement(s) for MTTR.

We’re not saying that you shouldn’t seek metrics that help you model the reliability of your systems. As Staff SRE Fred Hebert says in his post **Plato's Dashboards**, “metrics are absolutely necessary to compress complex phenomena into an easily legible value that can guide decision-making.” Metrics, by their nature, are a form of lossy compression. With MTTR, we’ve picked the metric first without the necessary context. Once you start analyzing your incidents, you’ll be able to identify metrics that reflect your systems and your organization’s needs.

Below, we provide a set of metrics (most of which come from incident analyses) to consider instead of MTTR.

SLOs and Customer Feedback

Service Level Objectives (SLOs) are commitments that a service provider makes to ensure they are serving users adequately (and investing in reliability when needed to meet those commitments). A full explanation is beyond the scope of this report. The general principle is that your system might be running at or above some level predetermined purely by technical standards (e.g. % CPU load) that your customers may not notice or care about. SLOs help align technical system metrics with business objectives, making them a more useful frame for “reliability.” **Chapter 4 of *Site Reliability Engineering*** provides a good overview and starting point while ***Implementing Service Level Objectives*** is an excellent deep dive into how to start developing and using SLOs.

It’s also important to note that SLOs aren’t an ideal replacement for MTTR either. Tracking SLO targets can share the following weaknesses with MTTR:

- They’re backward-looking only and don’t include information about known risks
- They don’t capture non-SLO-impacting near misses
- Incidents that push a system outside its SLO still happen with a high degree of randomness over time, so you still have potential signal-to-noise issues

Additionally, not every company or organization has a Site Reliability Engineering (SRE) team or the ability to develop SLOs.[6] Signals about external customer impact can come from other sources such as your Customer Support Center, social media, and (for a longer-term view of how they perceive your product or service’s reliability) customer surveys and polls.

[6] *There are of course, companies that provide SLO products as well, but it’s not in our purview to recommend one over another.*

Sociotechnical Incident Data

Your systems are *sociotechnical*, comprising code, machines, and the humans who build and maintain them.[7] Yet we tend to consistently collect only technical data to assess how they are doing. One rich source of sociotechnical data comes from the concept of "**Costs of Coordination**" as studied by **Dr. Laura Maguire**.[8] These type of data include:

The number of people involved hands-on in an incident?

Across how many unique teams?

Using which tools?

Via how many chat channels?

Were there concurrent incidents?

What was PR/Comms involvement?

Until you start collecting this kind of information, you won't know how your organization actually responds to incidents (as opposed to how you may *believe* it does).[9] This matters because Maguire found that "Hierarchical, role-based coordination structures can create workload bottlenecks that slow the response efforts and force responders to adopt strategies to reduce the costs of coordination." This increases the amount of time required to resolve the incident.

Collecting data about who was involved and their cognitive load—along with the tools and technical resources required—gives a more holistic picture of the resilience of your systems and teams. These kinds of data also provide a more accurate account of what your team does and their reactions. These are concrete activities that you can build better metrics on top of vs. counting things you hope won't happen.

[7] *Patrick Waterson, Michelle M. Robertson, Nancy J. Cooke, Laura Militello, Emilie Roth & Neville A. Stanton (2015). Defining the methodological challenges and opportunities for an effective science of sociotechnical systems and safety. Ergonomics.*
[8] *Maguire, Laura (2020). Controlling the Costs of Coordination in Large-scale Distributed Software Systems. Ohio State University.*
[9] *Hindsight Magazine, Issue 25 (2017).*
[10] *https://www.adaptivecapacitylabs.com/blog/2019/11/20/markers-of-progress-incident-analysis/*
[11] *https://doeslasvegas2022.sched.com/event/1BhFE/how-were-transforming-the-practice-of-learning-from-incidents-in-a-12000-person-organization*

Post-Incident Review Data

Another way to assess the effectiveness of incident analysis within/across your organization is to track the degree of participation, sharing, and dissemination of post-incident review information.[10] This can include:

- **Number of people reading write-ups**
- **Number of people voluntarily attending post-incident review meetings.**
- **Number of links to write-ups from:**
  - **Code comments & commit messages**
  - **Architecture diagrams**
  - **Other related incident write-ups**

While a nascent approach, tracking how and where incident reviews are shared and disseminated is cropping up in unexpected places! The office of the CIO at IBM—a 12,000-person organization with a vast portfolio of products and technologies—is now holding monthly CIO meetings to learn from their incidents, and they are tracking these very kinds of metrics to understand better how the practice of learning from incidents is growing and gaining traction within the organization.[11]

Near Misses

Another fledgling practice within the software industry is prioritizing learning from near misses and actual customer/user-impacting incidents. We know from the aviation industry that focusing on near misses can provide deeper understanding of gaps in knowledge, misaligned mental models, and other forms of organizational and technical "blind spots". Information about near misses can help teams invest in changes to help avoid similar, and more serious, incidents in the future.

However, deciding what constitutes a near miss is by no means straightforward. A few example scenarios:

- **System X is down, but users don't notice because system Y serves cached or generic content for the duration or the outage. Is this an incident?**
- **Your backups start failing but the team doesn't notice for a month, customers don't notice either. Is that an incident?**
- **GitLab's Consul TLS certs expired. The solution of updating certs and simultaneously restarting all their services and nodes turned out to be quite perilous, but there was no user impact or observable degradation to performance of their systems. Incident or near miss?**

As an industry, we have our work cut out for us to better characterize what constitutes a near miss. Ultimately this understanding will be relatively unique to each organization and their own business model, products, technical solutions, and organizational structures. What better time to start this work than now?

Companies that can track, analyze, adapt to and learn from near misses are studying both successes and failures. This provides a much more complete picture of how their systems function.

In that vein, we'll tackle another well-known, but not well-understood, belief about incident data in the next section.

Good Times, Bad Times: Duration and Severity Aren’t Related

"Severity levels are not objective measures of anything in practice, even if they're assumed to be so in theory. They are negotiable constructs that provide an illusion of control or understanding, or footholds for people as they attempt to cope with complexity."

– John Allspaw

Despite the overwhelming evidence that duration (and hence MTTR) isn't a useful metric, many teams remain tied to it for either political reasons or lack of sufficiently-appealing alternatives. The general feeling that duration must have some kind of value or meaning is still very strong amongst many people we've talked to while conducting and presenting this research. Some form of "But it must mean *something*, right?" is the most common refrain.

Couched with this assumption is the notion that duration is some form of indicator of "how bad" an incident is. This is a frustratingly vague notion but a difficult one to shake. The only real proxy the software industry has for this is *Severity*, which is typically assigned to an incident on a descending numeric scale from 4-1, with 4 being least severe ("no" or "low" impact) and 1 being most severe (i.e., "critical"). While a discrete scale like this lends clear, easy categorization, severity is also **plagued by many of the same fuzzy issues** we noted about duration in the 2021 VOID report.

SEVERITY IS:

- **Not necessarily implemented consistently across an organization or even within a single team**
- **In some cases, a proxy for "customer impact"**
- **In other cases, a proxy for "engineering effort required to fix" or "urgency"**
- **Sometimes automated, often not**
- **Sometimes updated over the course of an incident, sometimes not**
- **Subjectively assigned, for a variety of reasons, including**
    - **To draw attention to/get assistance for an incident**
    - **To trigger (or avoid triggering) a post-incident review**
    - **To garner management approval for desired funding, headcount, etc.**
    - **To invoke (or avoid invoking) SLA-based contractual requirements**

As we saw earlier, duration may vary both in why it is assigned, and whether it is updated or otherwise changed after the fact. Consider this example from an SRE regarding assignment of severity for a particular incident:

> **"At a previous job I had, Severity served at least two purposes: first signaling how many people should be jumping in to help and second deciding whether or not the impact counted against teams' availability metrics. So the practice of severity differed a little depending on who picked up the incident commander role.**

> **There was a general pattern where during the incident severity would rise and fall according to how much help was needed. For example, one started out as SEV1 because the impact was wide-spread...a remediation was found, but there was still a huge backlog of messages that needed to be consumed on the kafka topic... the status would drop to a SEV3 while a few people monitored the burndown of that backlog and most people got back to their other work; then closing the incident when the topic was back to normal levels."**

Was that a SEV1 incident or a SEV3 incident? A snarky response would be to take the average and call it a SEV2, but what matters in this example is that it illustrates the seeming clarity of severity, when the truth under the hood is much messier.

Methodology

To evaluate a potential correlation between Duration and Severity, we needed incidents with metadata for both. We found both in the form of status pages, which tend to either explicitly, or via their own metadata, reference the assigned severity for each incident. The subset of new VOID data containing both Duration and Severity data included 7,696 incidents from 10 companies that were also included in the above Monte Carlo simulations. (The other two companies in the Monte Carlo simulations did not provide Severity classifications for their incidents, and were not included in this analysis.)

Most people are generally familiar with correlation as it applies to our daily lives, e.g., whether two things vary together in a way that indicates a relationship between those two things. Correlation can be either positive, where the two variables either increase or decrease together, or negative, where an increase in one variable leads to a decrease in the other or vice versa.

One example of a positive correlation is the relationship between height and weight: taller people tend to weigh more. A negative correlation is when people who get less sleep tend to feel more tired. Things that don't vary together do not correlate (like how much you sleep and your shoe size).

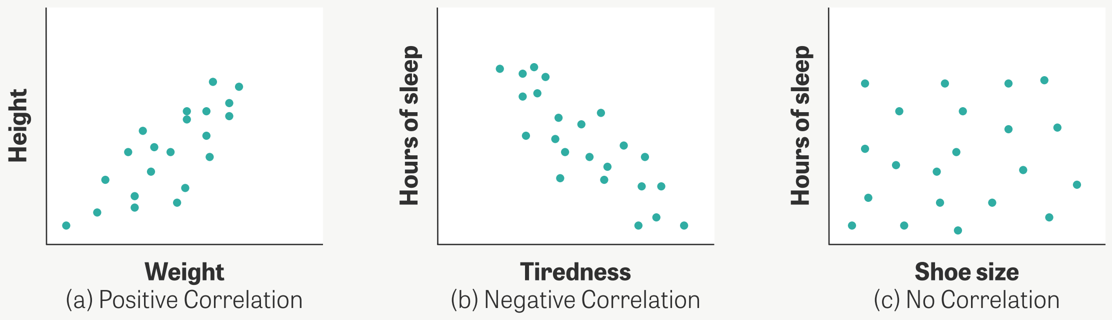


Figure 9. Examples of different types of correlations.

Where correlation gets a bit trickier is that it's not a yes/no answer. Correlation is evaluated by a correlation coefficient (aka the R value), which is a number that is always on a scale between -1 and 1, with larger numbers (in either direction) indicating the strength of the correlation. So an R value between -0.3 to 0.3 is relatively weak, whereas an R of ±0.7 or more indicates a very strong relationship between the two variables.

And last but not least: correlation is not causation. A strong correlation between two variables only denotes that they *do* vary together, not *why* they vary together. That requires further investigation, which partly depends on whether you really care about causation. If you merely want to predict one variable based on another variable, causation doesn't matter. However, when picking a product feature or implementing a policy to achieve a given outcome, if causation matters then you'd have to conduct further experiments that dig into the relationship in structured ways (**Process Tracing** being one such example of this type of approach). For now, we're just looking to see if a relationship exists in the first place, so we started with a basic correlational analysis.

Most correlation analyses use **Pearson's correlation** methodology. This methodology relies on a normal distribution of the data, and we've already shown in detail that duration data are not normally distributed. So instead we chose to use **Spearman's Rank correlation**, which assigns a rank to each value within each variable and then performs a standard (Pearson) correlation analysis of those ranked values. This is a better approach when dealing with skewed data for which outliers would impact a standard correlation analysis.[12] [13]

The null hypothesis for this experiment would be that there is no relationship between duration and severity. To reject that hypothesis, we'd need to find a strong correlation coefficient from the analysis. We set our significance level at a p value of < .05,[14] and ran the correlation analysis across various timeframes to assess the effect of varying sample sizes (due to constrained time windows) on the results.

Findings

The majority of the timeframes studied yielded no correlation between duration and severity. Figure 10 illustrates this, charting incidents for one such company during 2021–2022.

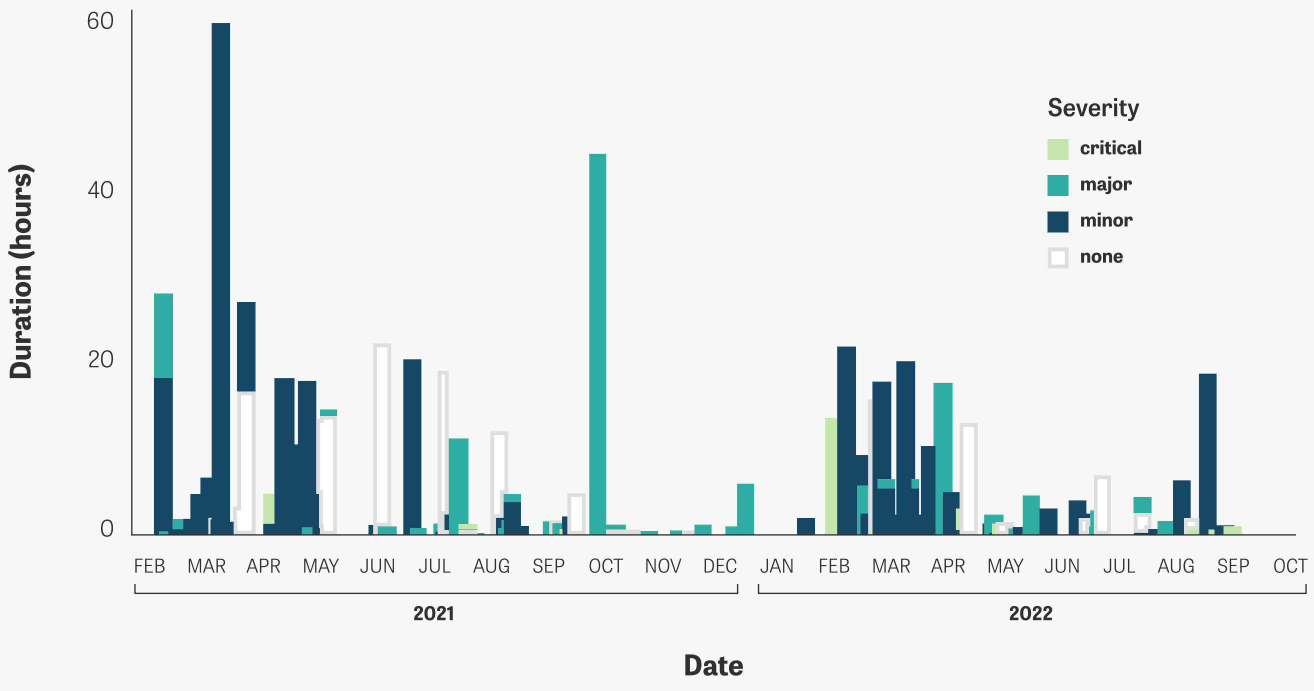


Figure 10. Incidents of varying duration (y axis) and severity (color) from Feb 2021–Sept 2022.

[12] *Daniel, W. (1990).* Applied Nonparametric Statistics. *PWS-Kent.*
[13] *Gregory W. Corder, Dale I. Foreman (2014).* Nonparametric Statistics: A Step-by-Step Approach (2nd Edition). *Wiley.*
[14] *The p value reflects the probability of the occurrence of a given event, in this case a large positive or correlation coefficient.*

Table 3 shows all the correlation coefficients and associated p values for each company across the timeframes.

Company		All Dates	2020–2022	2021–2022	2022	H1 2022	Q3 2022
Company A	R value	-0.01	0.00	0.01	-0.01	0.2	0.25
	P value	0.89	0.97	0.78	0.93	0.19	0.12
Company B	R value	**-0.18**	**-0.16**	**-0.17**	**0.16**	**-0.18**	-0.16
	P value	**<.00002**	**<.00002**	**<.00002**	**0.003**	**0.006**	0.07
Company C	R value	0.14	0.15	0.07	0.09	-0.25	**0.58**
	P value	0.13	0.24	0.65	0.65	0.38	**0.04**
Company D	R value	0.10	-0.11	-0.19	-0.21	-0.24	-0.1
	P value	0.14	0.25	0.08	0.12	0.19	0.63
Company E	R value	0.06	-0.01	0.01	0.04	-0.07	0.23
	P value	0.08	0.85	0.84	0.69	0.51	0.21
Company F	R value	**0.19**	**0.23**	**0.21**	0.21	0.19	0.32
	P value	**0.0005**	**0.001**	**0.009**	0.09	0.2	0.28
Company G	R value	**-0.17**	-0.11	-0.08	0.07	0.19	-0.42
	P value	**0.003**	0.20	0.41	0.68	0.09	0.12
Company H	R value	0.11	-0.14	-0.22	-0.22	-0.35	0.06
	P value	0.32	0.31	0.19	0.4	0.35	0.88
Company I	R value	0.06	-0.07	-0.11	-0.01	0.24	-0.35
	P value	0.23	0.16	0.08	0.9	0.08	0.05
Company J	R value	0.15	0.02	0.09	0.14	0.03	0.45
	P value	0.08	0.86	0.54	0.53	0.9	0.31

Table 3. Correlation between duration and severity for all included timeframes

One company in particular—Company C, which had the very tall, tight curve in the earlier Monte Carlo analyses—showed a consistent, but very minor, effect that was significant in all but one timeframe (Q3 2022). As you may recall, this company had vastly more incidents than any other company, in some cases as many as 8–10x more incidents than the others. It's possible that Company C's increased set of incidents does allow for detecting some form of relationship between incident duration and severity. For example, if they report on any and all types of incidents, from very minor performance issues up to full-blown outages, they might have more shorter-duration incidents which are minor in severity. However, the correlations were consistently very weak for Company C, coming in at around an R of -.0.17 (except for 2022, when it flipped positive, which further highlights how difficult it can be to extract meaning from these types of data). Given the consistently weak nature of any significant correlation we found between duration and severity, we chose not to dive deeper into understanding what might be driving these correlations.

This demonstrates that companies can have long or short incidents that are very minor, existentially critical, and nearly every combination in between. Not only can duration *not* tell a team how reliable or effective they are, but it also doesn't convey anything useful about the event's impact or the effort required to deal with the incident.

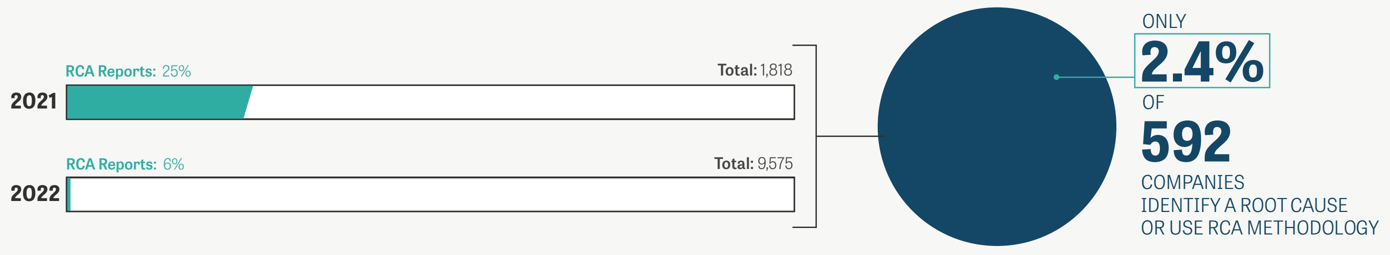
Root Cause: An Update

"Cause is not something you find. Cause is something you construct. How you construct it and from what evidence, depends on where you look, what you look for, who you talk to, what you have seen before, and likely on who you work for."

– Sidney Dekker

We continue to monitor the total count of incidents and which companies use Root Cause Analysis (RCA) or identify some form of "root cause" for their incidents. We started this practice with last year's VOID report because an RCA-minded this approach can hinder an organization's ability to look beyond the so-called "single cause", and because language like "root cause" influences the way that the organization, and those external to it (notably the media), think about incidents.

An RCA-minded approach posits that an incident has a single, specific cause or trigger, without which the incident wouldn't have happened. Once that trigger is discovered, a solution flows from it, and the organization can take steps to ensure it never happens again. This sequence-of-events mindset (also known as the "Domino model") originates in Industrial Accident Theory[15] and is common in incident reports, both for software and other domains. However, as we'll show below, complex software failures are never simple. Their inherent complexity means they comprise numerous lurking latent failures, and the system is always operating in some form of degradation.[16] An event like a failure results from a specific combination of those latent factors that combine to create an unexpected outcome.



In the 2021 VOID report, we found that 25% of the 1,818 incident reports (454 total) either labeled their report as using Root Cause Analysis (RCA) or declared a given "root cause" for the incident (we'll refer to these as "RCA-tagged" for simplicity). This year, the overall percentage is significantly reduced to 6% due to the addition of over 7,600 new incident reports in 2022 for a total of 595 RCA-tagged incident reports from just 15 companies (just 2.4% of the overall 592 in the VOID).

We expect these numbers to continue shifting as we add more incident reports—they will remain a moving target until the VOID contains enough reports to be compre-hensive on this particular aspect of incident metadata.

One notable RCA-related difference from last year's VOID report to this year's report is worth discussing in more detail. In June of 2022, Microsoft Azure stopped reporting incidents according to an RCA-based format. We'll return to this in a moment, but first, a bit of background detail. While we don't have the full history of Azure incident reports due to Microsoft limiting how far back we could search when we first started the VOID, we can say the practice of evaluating Azure incidents through an RCA framework reaches back to at least last 2019.

These updates followed a consistent format:

- Summary: A brief (2–3 sentence) overview of the timeline and impact
- Root Cause: Typically 1–2 paragraphs detailing what was identified as being the "root cause"
- Mitigation: Actions taken to solve the problem(s)
- Next Steps: An apology for the incident and a set of action items/follow-up tasks

[15] *Heinrich, Herbert William (1931).* Industrial accident prevention: A scientific approach. *McGraw-Hill.*

[16] *Richard Cook (2002). How Complex Systems Fail.*

Consider this example from November, 2019:

RCA - Multiple Services - Downstream impact from Azure Front Door (Tracking ID HLMF-R88)

Summary of Impact: Between 00:56 and 03:40 UTC on 20 Nov 2019, multiple services across Microsoft including Azure, Microsoft 365 and Microsoft Power Platform leveraging the Azure Front Door (AFD) service experienced availability issues resulting from high request failure rates. During this event, some impacted services were able to divert traffic away from the AFD service to mitigate impact for them.

One of the impacted services was the Azure Status Page at **https://status.azure.com**. Engineering executed the failover plan to the secondary hosting location, but this resulted in a delay in status communication changes. Communications were successfully delivered via Azure Service Health, available within the Azure management portal.

Root Cause: Azure Front Door services provide network edge caching and web acceleration services to many of Microsoft's SaaS services, in addition to the optimization offering direct to Azure customers. A routine, periodic deployment was released through our validation pipeline that, when combined with specific traffic patterns, caused service-wide, intermittent HTTP request failures for all services utilizing the AFD service.

Investigation into the faulting behavior revealed that the combination of a sequenced code deployment, a configuration deployment and specific traffic patterns triggered a dormant code bug that instigated the platform to crash. These deployed changes were tested before being shipped to the broader cloud; however, the specific traffic pattern was not observed during test and pilot phases.

Azure Front Door deploys to over one hundred points of presence (PoPs) around the globe and deploys customer configuration globally to each of these PoPs, enabling customers to quickly make changes to their service. This is done to ensure customers are able to promptly remove regional components out of specification and update configuration for network security services to mitigate attacks. Through a staged deployment, these changes passed validation and service health-checks. Having passed these validations, propagation to global PoPs was quick, by design, to meet the aforementioned service objectives. After propagation, the fault triggering behavior was instigated only by specific traffic patterns, that occurred after the deployment had completed.

This resulted in impacted customers experiencing a high, but intermittent, rate of web request failures globally while accessing shared services across the Azure and Office platforms.

Mitigation: Global monitoring detected the issue and engaged engineers at 01:04 UTC. Engineers confirmed the multiple sources of the issue to be primarily triggered by the configuration deployment and identified a fix for the issue by 01:27 UTC. Engineers immediately initiated deployment rollback procedures to return the service to a healthy state; this rolled out quickly, progressively and completely to all global platforms by 02:40 UTC. Many of the Microsoft SaaS impacted services were able to initiate failover away from the AFD service, providing mitigation to customers while the underlying AFD mitigation was deployed.

Next Steps: We sincerely apologize for the impact to affected customers. We are continuously taking steps to improve the Microsoft Azure Platform and our processes to help ensure such incidents do not occur in the future. In this case, this includes (but is not limited to):

- Verify that the fix deployed globally to AFD, during mitigation, is a stable release and will remain in place until all internal reviews of this issue have been completed.
- Review all service change management processes and practices to help ensure appropriate deployment methods are used.
- Review the change validation process to identify components and implement changes, required to increase test traffic diversity, improving the scope of trigger and test code paths.
- Prioritize deployment of a component independent automated recovery process so impacted deployments, like that experienced during this incident, are automatically returned to the last-known-good (LKG) state at a component layer, quickly and without manual intervention, to help reduce time to mitigate and scope of impact.
- Investigate and remediate the delay experienced with publishing communications to the Azure Status Page during the impact window.

A closer read of the Root Cause section reveals not just a single root cause, but instead (emphasis ours): "...the *combination* of a sequenced code deployment, a configuration deployment, and specific traffic patterns triggered a dormant code bug that instigated the platform to crash. These deployed changes were tested before being shipped to the broader cloud; however, the specific traffic pattern was not observed during test and pilot phases."

This incident could not have happened without that unique combination of active and latent factors. This was a Contributing Factors Analysis masquerading as Root Cause Analysis! Why is this important? Because when discussing language in last year's VOID report we noted: "The structure of how an event is described can influence how people perceive and recall those events."

Flash forward to June of this year, and Microsoft Azure switched to publishing Post-Incident Reviews (PIRs). This new incident report format had a few surface-level changes and many other deeper changes. The general headings for the report switched over to

- What happened?
- What went wrong, and why?
- How did we respond?
- How are we making incidents like this less likely or less impactful?
- How can our customers and partners make incidents like this less impactful?

Notably, the language used here is more understandable and approachable. These are the kinds of questions that customers might ask. The last two PIR headings (ie. How are we making incidents like this less likely or less impactful? How can our customers and partners make incidents like this less impactful?) also nod to the fact that it's not necessarily possible to make a repeat incident less likely (though, of course, they will do their best to ensure that). These two sections also provide concrete information for customers/users to better understand the potential impact of an incident, and what they can do to try to reduce the impact on their end in the future.

As for the content of the sections, the difference is notable both in scope and detail, as can be seen in their first PIR published in June:

Post Incident Review (PIR) - Datacenter cooling event - East US 2 (Tracking ID NMB2-ND0)

What happened?

Between 02:41 and 14:30 UTC on 07 Jun 2022, a subset of customers experienced difficulties connecting to resources hosted in one particular Availability Zone (AZ) of the East US 2 region. This issue impacted a subset of storage and compute resources within one of the region's three Availability Zones. As a result, Azure services with dependencies on resources in this zone also experienced impact.

Since the vast majority of services that were impacted already support Availability Zones customers using always-available and/or zone-redundant services would have observed that this zone-specific incident did not affect the availability of their data and services. Five services (Application Insights, Log Analytics, Managed Identity Service, Media Services, and NetApp Files) experienced regional impact as a result of this zonal issue. These five services are already working towards enabling AZ support. Finally, while App Service instances configured to be zone-redundant would have stayed available, from the other AZs, control plane issues were observed regionally that may have prevented customers from performing service management operations during the impact window.

What went wrong, and why?

Microsoft experienced an unplanned power oscillation in one of our datacenters within one of our Availability Zones in the East US 2 region. Components of our redundant power system created unexpected electrical transients, which resulted in the Air Handling Units (AHUs) detecting a potential fault, and therefore shutting themselves down pending a manual reset.

The electrical transients were introduced by anomalous component behavior within Uninterruptible Power Supply (UPS) modules, and cascaded throughout the datacenter electrical distribution system including electrical power supply to the mechanical cooling plant. As a result of the AHU self-protective shutdown, cooling to the datacenter was interrupted. Although the electrical transients did not impact our compute, networking, or storage infrastructure – which did not lose power – the mechanical cooling plant shutdown led to an escalating thermal environment, which induced protective shutdown of a subset of this IT infrastructure prior to the restoration of cooling.

Thorough detailed analysis has resulted in an adjustment to the UPS gain settings, preventing any further oscillations. These oscillations are the power equivalent of having a microphone too close to an amplifier – just as setting the volume too high can trigger a self-sustained sound oscillation, power oscillations can occur when the gain of the UPS is too high. The normal process of adding load to the UPS units results in an increase in gain and, in this case, the gain went high enough to cause the oscillations to occur. Adjusting the control gain setting lower in the UPS returns them to stable operation for all load values, preventing disruptions to any other infrastructure such as the AHUs.

Subsets of equipment including network, storage, and compute infrastructure were automatically shut down, both to prevent damage to hardware and to protect data durability under abnormal temperatures. As a result, Azure resources and services with dependencies on these underlying resources experienced availability issues during the impact window. A significant factor of downstream service impact was that our storage infrastructure was amongst the hardware most affected by these automated power and thermal shutdowns. Eight storage scale units were significantly impacted – due to thermal shutdowns directly and/or loss of networking connectivity, itself due to thermal shutdowns of corresponding networking equipment. These scale units hosted Standard Storage including LRS/GRS redundant storage accounts, which in turn affected Virtual Machines (VMs) using Standard HDD disks backed by this storage, as well as other services and customers directly consuming blob/file and other storage APIs.

The platform continuously monitors input/output transactions from the VMs to their corresponding storage. So even if the scale unit running a VM's underlying compute was operational, when transactions did not complete successfully within 120 seconds (inclusive of retries) the connectivity to its virtual disk is considered to be lost, and a temporary VM shutdown is initiated. Any workloads running on these impacted VMs, including first-party Azure services and third-party customer services, would have been impacted as their underlying hosts were either shut down by thermal triggers, or had their storage/networking impacted by the same.

How did we respond?

As soon as the AHUs shut themselves down as a result of the power disturbance, alerts notified our onsite datacenter operators. We deployed a team to investigate, who confirmed that the cooling units had shut themselves down pending manual intervention. Following our Standard Operating Procedure (SOP), the team attempted to perform manual resets on the AHUs, but these were not successful. Upon further investigation the onsite team identified that, due to the nature of this disturbance, recovering safely would require resetting the AHUs while running on backup power sources, to prevent the power oscillation pattern on the utility line from triggering a fault. This meant that two primary steps were required to recover–firstly, the impacted datacenter manually transferred from utility power to backup power sources, our onsite generators. By doing this, we changed the characteristics in the power lineup to obviate the creation of the oscillations. Secondly, the AHUs were then manually reset to recover them, which restored cooling to the datacenter.

Once temperatures returned to normal levels, some hardware including network switches needed to be manually power cycled to be brought back online. The network hardware and components serve different compute and storage resources for the scale units in this datacenter, including host instances for other applications and services. Onsite engineers then manually reviewed the status of various infrastructure components, to ensure that everything was working as intended.

Following the restoration of most storage network connectivity, recovery activities included diagnosing and remediating any host nodes that had entered an unhealthy state due to loss of network, and triaging any other hardware failures to ensure that all storage infrastructure could be brought back online. Even after all storage nodes returned to a healthy state, two storage scale units still exhibited slightly lower API availability compared to before this incident. It was determined that this was caused by a limited number of storage software roles being in an unhealthy state – those roles were restarted, which restored full API availability for those scale units.

Since our compute continuously monitors for Storage access, as storage/networking started recovering the compute VMs automatically started coming back up. This worked as expected in all cases expect on one scale unit, where the physical machines were shut down and did not recovery automatically. Since the VMs were originally down due to storage/networking issues, it was only detected once storage recovered, so we manually recycled the nodes to bring them back online. Upon investigation, an issue with the cluster power management unit prevented automatic recovery.

Two specific Azure services (ExpressRoute and Spatial Anchors) performed manual mitigations to fail customers over to use the other two Availability Zones within the region. Thus, while some impacted services recovered even earlier, full mitigation of this incident was declared at 14:30 UTC.

After cooling was restored and infrastructure was brought back online, our onsite teams opted to leave the datacenter running on backup power sources during additional investigations and testing, both focused on the UPS gain setting. In consultation with our critical environment hardware suppliers, we ran comprehensive testing to confirm the relevant gain settings based on the amount of load across the system. After these settings were deployed, we have since returned the datacenter back to our normal utility power feed.

How are we making incidents like this less likely or less impactful?

Already completed:
Updates to the gain setting, described above, have been deployed and the datacenter is back on utility in the impacted datacenter. We are confident that this has mitigated the risk of the power oscillation issue that was triggered.

Furthermore, our critical environment team has assessed systemic risk across all our datacenters globally, to ensure that none are at risk of the same situation. Of our 200+ Azure datacenters across 60+ regions, we identified only one other datacenter (beyond the impacted datacenter in East US 2) that had a similar power draw that could have potentially triggered a similar oscillation – this risk has since been mitigated with a similar configuration change.

Work in progress:
We have identified opportunities to improve our tooling and processes to flag anomalies more quickly, and are in the process of fine-tuning our alerting to inform onsite datacenter operators more comprehensively.

We are investigating why a subset of networking switches took longer than expected to recover. Although these were manually mitigated during the incident, we are exploring ways to optimize this recovery to ensure that customer workloads are brought online more quickly.

Similarly, we continue to diagnose a small subset of storage and compute nodes that remained in unhealthy states after restoration of networking, to streamline their recovery. This includes addressing a driver-related issue that prevented compute nodes in one scale unit from recovery automatically.

We are addressing some specific monitoring gaps including for compute nodes that have not been powered back on, specifically for scenarios in which they had been automatically shut down.

In the longer term:
We are developing a plan for fault injection testing relevant critical environment systems, in partnership with our industry partners, to be even more proactive in identifying and remediating potential risks.

We are exploring improved supplier diversity in the critical environment space, to minimize potential single points of failure within our hardware lineup.

We are investing in improved engineering tooling and processes that will accelerate the identification and remediation of unhealthy node states during incidents of this scale.

We have several workstreams in motion that will further improve storage node start-up times, learnings from this incident have validated the need to prioritize these optimizations.

Finally, we continue to invest in expanding how many Azure services support Availability Zones, so that customers can opt for automatic replication and/or architect their own resiliency across services: **https://docs.microsoft.com/azure/availability-zones/az-region**

How can our customers and partners make incidents like this less impactful?

Consider using Availability Zones (AZs) to run your services across physically separate locations within an Azure region. To help services be more resilient to datacenter-level failures like this one, each AZ provides independent power, networking, and cooling. Many Azure services support zonal, zone-redundant, and/or always-available configurations: **https://docs.microsoft.com/azure/availability-zones/az-overview**

Consider which are the right Storage redundancy options for your critical applications. Zone redundant storage (ZRS) remains available throughout a zone localized failure, like in this incident. Geo-redundant storage (GRS) enables account level failover in case the primary region endpoint becomes unavailable: **https://docs.microsoft.com/azure/storage/common/storage-redundancy**

Consider using Azure Chaos Studio to recreate the symptoms of this incident as part of a chaos experiment, to validate the resilience of your Azure applications. Our library of faults includes VM shutdown, network block, and AKS faults that can help to recreate some of the connection difficulties experienced during this outage – for example, by targeting all resources within a single Availability Zone: **https://docs.microsoft.com/azure/chaos-studio**

More generally, consider evaluating the reliability of each of your critical Azure applications using guidance from the Azure Well-Architected Framework and its interactive Well-Architected Review: **https://docs.microsoft.com/azure/architecture/framework/resiliency**

Finally, ensure that the right people in your organization will be notified about any future service issues - by configuring Azure Service Health alerts. These can trigger emails, SMS, push notifications, web-hooks, and more: **https://aka.ms/ash-alerts**

This was an unusual incident in that it dealt with data center power fluctuations instead of software issues, and yet the team still provided a thorough description of the nature of the problem: “The electrical transients were introduced by anomalous component behavior within Uninterruptible Power Supply (UPS) modules, and cascaded throughout the datacenter electrical distribution system including electrical power supply to the mechanical cooling plant. As a result of the AHU self-protective shutdown, cooling to the data center was interrupted. Although the electrical transients did not impact our compute, networking, or storage infrastructure – which did not lose power – the mechanical cooling plant shutdown led to an escalating thermal environment, which induced a protective shutdown of a subset of this IT infrastructure before the restoration of cooling.”

Their investigation surfaced numerous technical and organizational factors that made it more difficult to detect and/or remediate. Along with identifying a number of monitoring, alerting, supply chain, and communication changes they’ve either implemented or are investigating, this item was particularly interesting: “We have several work streams in motion that will further improve storage node start-up times, learnings from this incident have validated the need to prioritize these optimizations.” This line item quietly sends a strong message: **What you learn from incidents can help you make the case for backlog/improvements that may not have gotten prioritization in the past.**

We hope others follow the Azure team’s lead. As noted in last year’s VOID report, the language we use matters: it shapes how we think about failures and incidents. We plan to dig deeper into the language and cognitive strategies teams that use when investigating and describing their incidents.

Going Forward

After scrutinizing an entire year's worth of incidents one thing is crystal clear: Resilience saves time. Taking the time to learn how to better respond when something green turns red—learning from the people, the processes, and the systems—will make *the next incident* smoother. Because, yes, there will always be **the next incident**.

1. **Treat Incidents as Opportunities to Learn**
2. **Favor In-depth Analysis Over Shallow Metrics**
3. **Treat Humans as Solutions, Not Problems**
4. **Study What Goes Right Along With What Goes Wrong**

Incidents contain multitudes. They reveal contradictions, assumptions, and systemic pressures intermingled with successes, sources of resilience, and adaptive capacity. We continue to encourage organizations to adopt a New Way of thinking about incidents:

This needs to go beyond checking a few boxes on a To Do list and saying "It won't happen again". Companies that don't realize the benefit of supporting in-depth incident analysis will eventually fall behind their forward-thinking competitors.

As we've seen, incidents are inevitable in any organization. The key to success is turning these incidents into learning opportunities. By studying what goes right along with what goes wrong, you can create a process that not only prevents future incidents but also allows your team to learn and grow from the experience. At the VOID, we believe in using data-driven insights to help our community grow and learn.

If you're interested in learning more about joining the VOID community, including our quarterly learning labs and exclusive 1:1 access with industry leaders don't hesitate to reach out to us at **void@verica.io**.

About Verica

Verica uses the next step in the evolution of chaos engineering, Continuous Verification, to make systems more secure and less vulnerable to costly incidents. Verica Continuous Verification Platform provides out-of-the-box verifications that proactively uncover system weaknesses and security flaws before they disrupt business outcomes. All companies running complex systems experience failure, but as systems become more complex, Verica will be there to help maintain confidence in those systems. With Verica, you can trust that your software is working how it's meant to. Learn more at **www.verica.io**.

About The VOID

Now an industry standard yearly report, the VOID is the largest and most comprehensive of incident analysis to date, with nearly 10,000 incidents from just under 600 companies analyzed and scrutinized. This data comes from nearly 600 companies ranging from mega cap tech and Fortune 100s to startups. The mission of the VOID is to make public incident reports in a single database to generate open discussion about how to tackle software-based failures and outages. Anyone can **submit an incident** to the VOID or **become a member**.

Acknowledgements

Lead Researcher and Writer:
Courtney Nash

Research Assistant:
William Ou

Editorial:
Laura Franzese

Copy Editing:
Amy Franks

Design:
Lauren Garside

Reviewers and Feedback:
John Allspaw
Clint Byrum
Adrian Cockcroft
Fred Hebert
David Leigh
Laura Nolan
Casey Rosenthal
Amy Tobey
Chad Todd
Nick Travaglini
Thai Wood

Members and Partners:
Indeed
Jeli
Stanza Systems
Adaptive Capacity Labs
Auxon

© 2022 The VOID. All rights reserved.