braze

The April 29 Braze Outage: What Happened, Why It Occurred, and How We're Responding

May 3, 2024

Jon Hyman
Braze Cofounder and CTO

On Monday, April 29, 2024, the Braze platform's US clusters experienced a near-total outage that persisted in whole or in part for nearly 11 hours, impacting customer access to our dashboard, as well as data processing and message sends. In the 13-year history of Braze, this is the first and only incident of this magnitude that we've ever had. Our relationship with customers is built on trust, and we've always taken tremendous pride in building resilient systems that have allowed our platform to remain available and performant, even under demanding workloads. During this outage, we failed to deliver on that promise, and we are deeply sorry for the impact this has had on each and every one of our customers.

To help you understand better what happened and why, I'm going to provide some important context around our architecture, describe the causes of the outage in detail, walk you through the changes we've already made, and outline what we will be working to implement in the near term and in the future.

Understanding the Braze platform's services architecture

Braze maintains separate sets of servers for each of our eight US clusters. The Braze US 01 through 07 clusters are hosted on Amazon Web Services (AWS), while the Braze US 08 cluster is hosted on Microsoft Azure. In each of these environments, Braze uses multiple availability zones and has redundancies for each part of our system. In the event of an availability zone failure, we routinely and transparently disable impaired availability zones from routing and processing. That makes it possible for us to reliably maintain service deliverability during a cloud service provider outage.

With few exceptions, each one of the US clusters has its own dedicated infrastructure and, except for a handful of components that are used across all environments in a given cloud region, these clusters are heavily isolated from each other, in order to reduce the risk that a disruption in one cluster will affect other clusters.

Due to the intensity of the Braze platform's real-time processing workload, since 2013 we have supplemented our use of AWS and Azure by partnering with Rackspace to host custom-configured hardware to run MongoDB databases. Our AWS and Azure environments are configured with AWS Direct Connect and Azure ExpressRoute private cloud connectors, linking the cloud environment to our internal network maintained in the Rackspace data centers over a dedicated fiber-optic cable. This encrypted connection, which is powered by network links that support multiple interfaces pushing over 100 gigabits of network traffic per second, provides highly performant access between our cloud and the Rackspace environments.

Our custom environment in Rackspace is home to hundreds of physical servers held in cabinets dedicated to Braze. This setup includes redundant power supplies and redundant top-of-rack switches. Top-of-rack switches are network devices that sit in a server rack and connect the devices and servers together in the same server rack. These switches are tested at least annually for failover without impact; last year's test was held successfully on August 3, 2023. In this custom environment, we maintain tens of thousands of virtualized containers for MongoDB databases. Similar to our cloud environments, we maintain high levels of physical isolation between containers on different US clusters to minimize disruption and reduce the risk that one container might affect other containers within our databases.

The cause of the April 29 outage and how we responded

The initial partial switch failure

On April 29 at 09:15 UTC, there was a partial failure of a primary Cisco Nexus network switch connected to our Rackspace server cabinets. The malfunctioning switch failed in a way that didn't automatically activate the secondary network switch via a failover, and instead kept the malfunctioning switch as primary.

Network switches route traffic by forwarding network packets—called "frames"—to their destination. To do this efficiently, these switches maintain an understanding of the network topology that connects devices together. It's common for networking routes to dynamically change in response to individual component failures or slowness. When these changes occur, switches and other network devices communicate with each other to maintain an accurate understanding of the network topology.

In large networks, more than one path will often exist between devices, which can cause routing "loops" (i.e. a condition where packets fail to stop at their intended destination, and instead loop back to where they originated). Loops can create "broadcast storms," where traffic ends up looping indefinitely, saturating network links and causing outages. To keep this from happening, switches are programmed to send messages that help identify redundant links. Those messages are called spanning tree bridge protocol data units (BPDUs) and are part of a network protocol called the Spanning Tree Protocol (STP). Switches then use this information to block traffic ports in order to prevent loops from occurring when they route traffic. If a physical link or switch fails, the STP helps to provide redundancy, since it can be used to identify other physical links that traffic can be sent over and promote the previously passive (i.e. blocked) link to active in order to maintain connections.

At the onset of the April 29 incident, when the switch started malfunctioning, it began continuously forwarding topology change notifications and BPDUs, flooding the network with those change notices. These packets propagated to distribution switches and other switches, causing a spanning tree switching loop that resulted in a complete network outage. While the STP is in place to reduce network loops by blocking network ports where looping is detected, during the April 29 incident the incorrectly forwarded spanning tree traffic from the malfunctioning switch caused other switches in the network to re-analyze their spanning tree topologies. Then, because those other switches detected that a high-priority BPDU packet was coming from the malfunctioning switch, the distribution switches began forwarding on previously blocked ports and resulted in a switching loop that overloaded the network and took it down.

The Rackspace remediation

As a result of this activity, Rackspace data center engineers received alerts at approximately 09:20 UTC regarding the unexpected network connectivity issues and began addressing the problem. Between 09:39 and 10:03 UTC, Rackspace data center engineers suspended the network switch, power-cycled the primary switch, and forced network interface cards (NIC) in the server cabinet to failover to the secondary switch. After the NIC failover, connectivity was restored to the physical servers in the Braze environment.

How Braze MongoDB databases function

The MongoDB databases used by Braze allow data to be stored across multiple database servers that are known as "shards." MongoDB provides a routing service called "mongoS," which processes queries from the Braze platform's services, determines the location of the relevant data in a sharded cluster, then routes a query to the appropriate MongoDB database. Braze has hundreds of distinct MongoDB databases, comprising over 15,000 "mongoD" processes, which are the programs that run the database and store data for a particular MongoDB shard.

Each database shard consists of one primary mongoD and at least two secondary mongoD processes, which provides high-availability in the event of a failure. Each mongoS server maintains a network connection to every mongoD that it can route queries to. These mongoD processes also connect to the primary and secondaries in its configuration; this is known as a replica set. If a mongoS can't connect to a given mongoD, it will mark that process as offline and schedule a retry. Similarly, if a mongoD can't connect to other mongoD members of its replica set within one second, it will timeout, mark those processes as offline, and schedule a retry.

The impact of the network loop on our MongoDB processes and how we responded

During the network loop, because our MongoDB processes couldn't connect normally, each one marked all of its associated processes as offline. That is, each mongoS process marked all of its associated mongoD processes as offline, and each mongoD process marked its related replica set members as offline. However, even after network connectivity was restored to our physical servers by Rackspace, neither the mongoS or mongoD processes re-established all connections to the other processes that had been marked offline.

At this point, in order to assist in our Rackspace database administrator (DBA) team's remediation efforts, Braze DevOps teams began rapidly debugging and testing out methods for restoring that connectivity. Our investigation determined that the only available option was to restart every one of the nearly 16,000 mongoD and 6,000+ mongoS processes in their virtualized containers. Given the sheer scale of this undertaking and the fact that automation did not exist to restart so many containers quickly, Rackspace engineers wrote new code during the incident to create on-the-fly automation capability.

Unfortunately, as the restart process ramped up, another issue arose in the Domain Name System (DNS) system. As mongoS and mongoD processes come online, they request information from DNS resolvers in a process that's known as a DNS lookup. With the continuous DNS lookups driven by the speed of the restarts, the DNS servers came under heavy load, leading to connectivity timeouts on the mongoS layers as they reconnected to the mongoD processes. To accommodate this growing load, our Rackspace team increased DNS CPU capacity, but was then forced to restart the mongoS tier a second time in order to create healthy connections from each mongoS to their associated mongoD processes.

Around 17:05 UTC, the restart process allowed some clusters to start to come back online. As database performance recovered and the clusters stabilized, Braze continued to ramp up data processing and messaging sending capacity; however, that effort was complicated by the massive backlog resulting from prior unavailability.

Given the scale and sophistication of our Rackspace database environment, which consists of hundreds of distinct databases, the length of the outage, and the resulting volume of our backlog (representing billions of messages and billions of ingested data points), the recovery process required in-the-moment adaptation of our normal recovery processes. This was necessary in order to ensure that database resources remained in balance with ongoing processing demands, including the introduction of a massive amount of overall processing capacity.

However, in doing so, Braze hit an AWS limit on the number of virtual CPUs we were able to provision, blocking us from adding any additional server capacity. The Braze team rapidly escalated this circumstance with our AWS team and received an increase, allowing our engineers to scale up our server processing capacity to a record level, which was more than 50% higher than our previous maximum capacity, which had been reached on **Black Friday 2023**.

By 20:30 UTC, all of our US clusters except US 01, US 03, and US 08 had finished processing their backlogs, and those remaining clusters were processing at or above peak rates from the day before the incident. Within a few hours, all clusters were caught back up to real-time processing and we declared the incident resolved.

How Braze communicates updates during incidents

Clear, frequent communication during incidents like this is essential. While technical issues in general are rare occurrences, and an issue of this magnitude was previously unheard of for Braze, we always do our best to communicate with affected parties with clarity and speed.

In situations like this, the guiding principles that inform our communication strategy are honesty, transparency, and trust. We know that we only have one opportunity to be honest and forthright about what is occurring, and that transparency about incidents—both in the moment and following their resolution—is essential to maintain trust. At the end of the day, we want our customers to be confident that Braze will always do everything in our power to explain, remediate, and prevent a recurrence of any incident.

During incidents, we use our **Status Page** to manage communications and provide regular updates on where things stand; I strongly encourage customers to sign up for updates from that page in order to stay up to speed. During the April 29 outage, we supplemented these frequent Status Page updates with an email from from my fellow cofounder and our CEO, Bill Magnuson. That message was sent to all Braze dashboard users and was provided to our account teams to share with other customer stakeholders as needed.

How Braze followed up the incident and future prevention plans

Prior to this incident, our experience had led us to believe that our network was resilient, due to our redundant network switches. However, the outage revealed that our network topology—which had supported us well for over a decade—was vulnerable to catastrophic failure. It is now clear that it must be enhanced going forward.

Since the incident, Braze and Rackspace have swapped out and replaced the faulty switch, and the Rackspace engineering team has completed a full audit of the networking infrastructure that the Braze services rely on. While hardware failures are incredibly difficult to predict, especially with in-warranty equipment, monitoring is in place to detect abnormalities and to ensure a rapid response. Both teams are collaborating now to make changes that will prevent future network loops, even in the event of malfunctioning equipment. Some networking changes will take time to implement, as we plan to make meaningful changes to our network topology to further improve protection against loops. We have also opened support tickets with both Cisco and MongoDB to better understand the failure scenarios we experienced, and to improve the ability for mongoD and mongoS processes to self-heal in the aftermath of network failures.

We have been in daily contact with Rackspace since the incident to take action on near-term changes and carry out a larger network enhancement, and will continue to do so until we have confidence that an improved, more resilient network design has been achieved.

I want to apologize again for this incident and for the impact that it had on your business and your relationship with your customers. The amount of time that Braze was unavailable is unacceptable. Our

engineering leadership and executive team are fully committed to making all the necessary changes needed to identify and resolve outage vulnerabilities as quickly and efficiently as possible, so that we can once again earn your trust as a reliable and performant partner.

— Jon