Summary of the Amazon Kinesis Event in the Northern Virginia (US-EAST-1) Region

November, 25th 2020

We wanted to provide you with some additional information about the service disruption that occurred in the Northern Virginia (US-EAST-1) Region on November 25th, 2020.

Amazon Kinesis enables real-time processing of streaming data. In addition to its direct use by customers, Kinesis is used by several other AWS services. These services also saw impact during the event. The trigger, though not root cause, for the event was a relatively small addition of capacity that began to be added to the service at 2:44 AM PST, finishing at 3:47 AM PST. Kinesis has a large number of "back-end" cell-clusters that process streams. These are the workhorses in Kinesis, providing distribution, access, and scalability for stream processing. Streams are spread across the back-end through a sharding mechanism owned by a "front-end" fleet of servers. A back-end cluster owns many shards and provides a consistent scaling unit and fault-isolation. The front-end's job is small but important. It handles authentication, throttling, and request-routing to the correct stream-shards on the back-end clusters.

The capacity addition was being made to the front-end fleet. Each server in the front-end fleet maintains a cache of information, including membership details and shard ownership for the back-end clusters, called a shard-map. This information is obtained through calls to a microservice vending the membership information, retrieval of configuration information from DynamoDB, and continuous processing of messages from other Kinesis front-end servers. For the latter communication, each front-end server creates operating system threads for each of the other servers in the front-end fleet. Upon any addition of capacity, the servers that are already operating members of the fleet will learn of new servers joining and establish the appropriate threads. It takes up to an hour for any existing front-end fleet member to learn of new participants.

At 5:15 AM PST, the first alarms began firing for errors on putting and getting Kinesis records. Teams engaged and began reviewing logs. While the new capacity was a suspect, there were a number of errors that were unrelated to the new capacity and would likely persist even if the capacity were to be removed. Still, as a precaution, we began removing the new capacity while researching the other errors. The diagnosis work was slowed by the variety of errors observed. We were seeing errors in all aspects of the various calls being made by existing and new members of the front-end fleet, exacerbating our ability to separate side-effects from the root cause. At 7:51 AM PST, we had narrowed the root cause to a couple of candidates and determined that any of the most likely sources of the problem would require a full restart of the front-end fleet, which the Kinesis team knew would be a long and careful process. The resources within a front-end server that are used to populate the shard-map compete with the resources that are used to process incoming requests. So, bringing front-end servers back online too quickly would create contention between these two needs and result in very few resources being available to handle incoming requests, leading to increased errors and request latencies. As a result, these slow front-end servers could be deemed unhealthy and removed from the fleet, which in turn, would set back the recovery process. All of the candidate solutions involved changing every front-end server's configuration and restarting it. While the leading candidate (an issue that seemed to be creating memory pressure) looked promising, if we were wrong, we would double the recovery time as we would need to apply a second fix and restart again. To speed restart, in parallel with our investigation, we began adding a configuration to the front-end servers to obtain data directly from the authoritative metadata store rather than from front-end server neighbors during the bootstrap process.

At 9:39 AM PST, we were able to confirm a root cause, and it turned out this wasn't driven by memory pressure. Rather, the new capacity had caused all of the servers in the fleet to exceed the maximum number of threads allowed by an operating system configuration. As this limit was being exceeded, cache construction was failing to complete and front-end servers were ending up with useless shard-maps that left them unable to route requests to back-end clusters. We didn't want to increase the operating system limit without further testing, and as we had just completed the removal of the additional capacity that triggered the event, we determined that the thread count would no longer exceed the operating system limit and proceeded with the restart. We began bringing back the front-end servers with the first group of servers taking Kinesis traffic at 10:07 AM PST. The front-end fleet is composed of many thousands of servers, and for the reasons described earlier, we could only add servers at the rate of a few hundred per hour. We continued to slowly add traffic to the front-end fleet with the Kinesis error rate steadily dropping from noon onward. Kinesis fully returned to normal at 10:23 PM PST.

For Kinesis, we have a number of learnings that we will be implementing immediately. In the very short term, we will be moving to larger CPU and memory servers, reducing the total number of servers and, hence, threads required by each server to communicate across the fleet. This will provide significant headroom in thread count used as the total threads each server must maintain is directly proportional to the number of servers in the fleet. Having fewer servers means that each server maintains fewer threads. We are adding fine-grained alarming for thread consumption in the service. We will also finish testing an increase in thread count limits in our operating system configuration, which we believe will give us significantly more threads per server and give us significant additional safety margin there as well. In addition, we are making a number of changes to radically improve the cold-start time for the front-end fleet. We are moving the front-end server cache to a dedicated fleet. We will also move a few large AWS services, like CloudWatch, to a separate, partitioned front-end fleet. In the medium term, we will greatly accelerate the cellularization of the front-end fleet to match what we've done with the back-end. Cellularization is an approach we use to isolate the effects of failure within a service, and to keep the components of the service (in this case, the shard-map cache) operating within a previously tested and operated range. This had been under way for the front-end fleet in Kinesis, but unfortunately the work is significant and had not yet been completed. In addition to allowing us to operate the front-end in a consistent and well-tested range of total threads consumed, cellularization will provide better protection against any future unknown scaling limit.

There were a number of services that use Kinesis that were impacted as well. Amazon Cognito uses Kinesis Data Streams to collect and analyze API access patterns. While this information is extremely useful for operating the Cognito service, this information streaming is designed to be best effort. Data is buffered locally, allowing the service to cope with latency or short periods of unavailability of the Kinesis Data Stream service. Unfortunately, the prolonged issue with Kinesis Data Streams triggered a latent bug in this buffering code that caused the Cognito webservers to begin to block on the backlogged Kinesis Data Stream buffers. As a result, Cognito customers experienced elevated API failures and increased latencies for Cognito User Pools and Identity Pools, which prevented external users from authenticating or obtaining temporary AWS credentials. In the early stages of the event, the Cognito team worked to mitigate the impact of the Kinesis errors by adding additional capacity and thereby increasing their capacity to buffer calls to Kinesis. While this initially reduced impact, by 7:01 AM PST errors rates increased significantly. The team was working in parallel on a change to Cognito to reduce the dependency on Kinesis. At 10:15 AM PST, deployment of this change began and error rates began falling. By 12:15 PM PST, error rates were significantly reduced, and by 2:18 PM PST Cognito was operating normally. To prevent a recurrence of this issue, we have modified the Cognito webservers so that they can sustain Kinesis API errors without exhausting their buffers that resulted in these user errors.

CloudWatch uses Kinesis Data Streams for the processing of metric and log data. Starting at 5:15 AM PST, CloudWatch experienced increased error rates and latencies for the PutMetricData and PutLogEvents APIs, and alarms transitioned to the INSUFFICIENT_DATA state. While some CloudWatch metrics continued to be processed throughout the event, the increased error rates and latencies prevented the vast majority of metrics from being successfully processed. At 5:47 PM PST, CloudWatch began to see early signs of recovery as Kinesis Data Stream's availability improved, and by 10:31 PM PST, CloudWatch metrics and alarms fully recovered. Delayed metrics and log data backfilling completed over the subsequent hours. While CloudWatch was experiencing these increased errors, both internal and external clients were unable to persist all metric data to the CloudWatch service. These errors will manifest as gaps in data in CloudWatch metrics. While CloudWatch currently relies on Kinesis for its complete metrics and logging capabilities, the CloudWatch team is making a change to persist 3-hours of metric data in the CloudWatch local metrics data store. This change will allow CloudWatch users, and services requiring CloudWatch metrics (including AutoScaling), to access these recent metrics directly from the CloudWatch local metrics data store. This change has been completed in the US-EAST-1 Region and will be deployed globally in the coming weeks.

Two services were also impacted as a result of the issues with CloudWatch metrics. First, reactive AutoScaling policies that rely on CloudWatch metrics experienced delays until CloudWatch metrics began to recover at 5:47 PM PST. And second, Lambda saw impact. Lambda function invocations currently require publishing metric data to CloudWatch as part of invocation. Lambda metric agents are designed to buffer metric data locally for a period of time if CloudWatch is unavailable. Starting at 6:15 AM PST, this buffering of metric data grew to the point that it caused memory contention on the underlying service hosts used for Lambda function invocations, resulting in increased error rates. At 10:36 AM PST, engineers took action to mitigate the memory contention, which resolved the increased error rates for function invocations.

CloudWatch Events and EventBridge experienced increased API errors and delays in event processing starting at 5:15 AM PST. As Kinesis availability improved, EventBridge began to deliver new events and slowly process the backlog of older events. Elastic Container Service (ECS) and Elastic Kubernetes Service (EKS) both make use of EventBridge to drive internal workflows used to manage customer clusters and tasks. This impacted provisioning of new clusters, delayed scaling of existing clusters, and impacted task de-provisioning. By 4:15 PM PST, the majority of these issues had been resolved.

Outside of the service issues, we experienced some delays in communicating service status to customers during the early part of this event. We have two ways of communicating during operational events – the Service Health Dashboard, which is our public dashboard to alert all customers of broad operational issues, and the Personal Health Dashboard, which we use to communicate directly with impacted customers. With an event such as this one, we typically post to the Service Health Dashboard. During the early part of this event, we were unable to update the Service Health Dashboard because the tool we use to post these updates itself uses Cognito, which was impacted by this event. We have a back-up means of updating the Service Health Dashboard that has minimal service dependencies. While this worked as expected, we encountered several delays during the earlier part of the event in posting to the Service Health Dashboard with this tool, as it is a more manual and less familiar tool for our support operators. To ensure customers were getting timely updates, the support team used the Personal Health Dashboard to notify impacted customers if they were impacted by the service issues. We also posted a global banner summary on the Service Health Dashboard to ensure customers had broad visibility into the event. During the remainder of event, we continued using a combination of the Service Health Dashboard, both with global banner summaries and service specific details, while also continuing to update impacted customers via Personal Health Dashboard. Going forward, we have changed our support training to ensure that our support engineers are regularly trained on the backup tool for posting to the Service Health Dashboard.

Finally, we want to apologize for the impact this event caused for our customers. While we are proud of our long track record of availability with Amazon Kinesis, we know how critical this service, and the other AWS services that were impacted, are to our customers, their applications and end users, and their businesses. We will do everything we can to learn from this event and use it to improve our availability even further.

Learn About AWS
What Is AWS?
What Is Cloud Computing?
AWS Inclusion, Diversity & Equity
What Is DevOps?
What Is a Container?
What Is a Data Lake?
AWS Cloud Security
What's New
Blogs
Press Releases

Resources for AWS
Getting Started
Training and Certification
AWS Solutions Portfolio
Architecture Center
Product and Technical FAQs
Analyst Reports
AWS Partner Network

Developers on AWS
Developer Center
SDKs & Tools
.NET on AWS
Python on AWS
Java on AWS
PHP on AWS
Javascript on AWS

Help
Contact Us
AWS Careers
File a Support Ticket
Knowledge Center
AWS Support Overview
Legal

Create an AWS Account

Amazon is an Equal Opportunity Employer: *Minority / Women / Disability / Veteran / Gender Identity / Sexual Orientation / Age.*

Language عربي | Bahasa Indonesia | Deutsch | English | Español | Français | Italiano | Português | Tiếng Việt | Türkçe | Русский | ไทย | 日本語 | 한국어 | 中文 (简体) | 中文 (繁體)

Privacy | Site Terms | Cookie Preferences | © 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.