# Summary of the October 22, 2012 AWS Service Event in the US-East Region

We'd like to share more about the service event that occurred on Monday, October 22nd in the US- East Region. We have now completed the analysis of the events that affected AWS customers, and we want to describe what happened, our understanding of how customers were affected, and what we are doing to prevent a similar issue from occurring in the future.

The Primary Event and the Impact to Amazon Elastic Block Store (EBS) and Amazon Elastic Compute Cloud (EC2)

At 10:00AM PDT Monday, a small number of Amazon Elastic Block Store (EBS) volumes in one of our five Availability Zones in the US-East Region began seeing degraded performance, and in some cases, became "stuck" (i.e. unable to process further I/O requests). The root cause of the problem was a latent bug in an operational data collection agent that runs on the EBS storage servers. Each EBS storage server has an agent that contacts a set of data collection servers and reports information that is used for fleet maintenance. The data collected with this system is important, but the collection is not time- sensitive and the system is designed to be tolerant of late or missing data. Last week, one of the data collection servers in the affected Availability Zone had a hardware failure and was replaced. As part of replacing that server, a DNS record was updated to remove the failed server and add the replacement server. While not noticed at the time, the DNS update did not successfully propagate to all of the internal DNS servers, and as a result, a fraction of the storage servers did not get the updated server address and continued to attempt to contact the failed data collection server. Because of the design of the data collection service (which is tolerant to missing data), this did not cause any immediate issues or set off any alarms. However, this inability to contact a data collection server triggered a latent memory leak bug in the reporting agent on the storage servers. Rather than gracefully deal with the failed connection, the reporting agent continued trying to contact the collection server in a way that slowly consumed system memory. While we monitor aggregate memory consumption on each EBS Server, our monitoring failed to alarm on this memory leak. EBS Servers generally make very dynamic use of all of their available memory for managing customer data, making it difficult to set accurate alarms on memory usage and free memory. By Monday morning, the rate of memory loss became quite high and consumed enough memory on the affected storage servers that they were unable to keep up with normal request handling processes.

The memory pressure on many of the EBS servers had reached a point where EBS servers began losing the ability to process customer requests and the number of stuck volumes increased quickly. This caused the system to begin to failover from the degraded servers to healthy servers. However, because many of the servers became memory-exhausted at the same time, the system was unable to find enough healthy servers to failover to, and more volumes became stuck. By approximately 11:00AM PDT, a large number of volumes in this Availability Zone were stuck. To remedy this, at 11:10AM PDT, the team made adjustments to reduce the failover rate. These adjustments removed load from the service, and by 11:35AM PDT, the system began automatically recovering many volumes. By 1:40PM PDT, about 60% of the affected volumes had recovered. The team continued to work to understand the issue and restore performance for the remaining volumes. The large surge in failover and recovery activity in the cluster made it difficult for the team to identify the root cause of the event. At 3:10PM PDT, the team identified the underlying issue and was able to begin restoring performance for the remaining volumes by freeing the excess memory consumed by the misbehaving collection agent. At this point, the system was able to recover most of the remaining stuck volumes; and by 4:15PM PDT, nearly all affected volumes were restored and performing normally.

We have deployed monitoring that will alarm if we see this specific memory leak again in any of our production EBS servers, and next week, we will begin deploying a fix for the memory leak issue. We are also modifying our system memory monitoring on the EBS storage servers to monitor and alarm on each process's memory consumption, and we will be deploying resource limits to prevent low priority processes from consuming excess resources on these hosts. We are also updating our internal DNS configuration to further ensure that DNS changes are propagated reliably, and as importantly, make sure that our monitoring and alarming surface issues more quickly should these changes not succeed. These actions will address the problems that triggered the event. In addition, we are evaluating how to change the EBS failover logic that led to the rapid deterioration early in this event. We believe we can make adjustments to reduce the impact of any similar correlated failure or degradation of EBS servers within an Availability Zone.

Impact on the EC2 and EBS APIs

The primary event only affected EBS volumes in a single Availability Zone, so those customers running with adequate capacity in other Availability Zones in the US East Region were able to tolerate the event with limited impact to their applications. However, many customers reported difficulty using the service APIs to manage their resources during this event. We have invested heavily in making our service APIs resilient to failure during events affecting a single Availability Zone. And, other than a few short periods, our monitoring showed what looked to be a healthy level of launch and create activity throughout the event. However, we've heard from customers that they struggled to use the APIs for several hours. We now understand that our API throttling during the event disproportionately impacted some customers and affected their ability to use the APIs.

We use throttling to protect our services from being overwhelmed by internal and external callers that intentionally or unintentionally put excess load on our services. A simple example of the kind of issue throttling protects against is a runaway application that naively retries a request as fast as possible when it fails to get a positive result. Our systems are scaled to handle these sorts of client errors, but during a large operational event, it is not uncommon for many users to inadvertently increase load on the system. So, while we always have a base level of throttling in place, the team enabled a more aggressive throttling policy during this event to try to assure that the system remained stable during the period where customers and the system were trying to recover. Unfortunately, the throttling policy that was put in place was too aggressive.

At 12:06PM PDT, the team implemented this aggressive API throttling policy to help assure stability of the system during the recovery. The team monitored the aggregate throttling rate as well as the overall activity (launches, volume creation, etc.) and did not at the time believe that customers were being substantially impacted. We now understand that this throttling policy, for a subset of our customers, was throttling a higher percentage of API calls than we realized during the event. The service APIs were still handling the vast majority of customer requests to launch and terminate instances and make other changes to their EC2 and EBS resources, but many customers experienced high levels of throttling on calls to describe their resources (e.g. DescribeInstances, DescribeImages, etc.). This made it difficult for these customers and their management applications to successfully use the service APIs during this period. It also affected users' ability to successfully manage their EC2 and EBS resources from the AWS Management Console. This throttling policy was in effect until 2:33PM PDT, after which we reduced the level of throttling considerably.

We have changed our operational procedures to not use this more aggressive throttling policy during any future event. We believe that our other throttling policies will provide us with the necessary service protection while avoiding the impact that customers saw during this event. We are also modifying our operational dashboard to add per-customer throttling monitoring (rather than just aggregate throttling rates) so that we have better visibility into the number of customers seeing heavy throttling. This will allow us to quickly understand the impact throttling is having on individual customers, regardless of what the overall throttling rate is, and make appropriate adjustments more quickly.

Throttling is a valuable tool for managing the health of our services, and we employ it regularly without significantly affecting customers' ability to use our services. While customers need to expect that they will encounter API throttling from time to time, we realize that the throttling policy we used for part of this event had a greater impact on many customers than we understood or intended. While this did not meaningfully affect users running high-availability applications architected to run across multiple Availability Zones with adequate running capacity to failover during Availability Zone disruptions, it did lead to several hours of significant API degradation for many of our customers. This inhibited these customers' ability to use the APIs to recover from this event, and in some cases, get normal work done. Therefore, AWS will be issuing a credit to any customer whose API calls were throttled by this aggressive throttling policy (i.e. any customer whose API access was throttled between 12:06PM PDT and 2:33PM PDT) for 100% of their EC2, EBS and ELB usage for three hours of their Monday usage (to cover the period the aggressive throttling policy was in place). Affected customers do not need to take any action; the credits will be automatically applied to their AWS account prior to their October 31 bill being calculated.

Impact on Amazon Relational Database Service (RDS)

This event also had an impact on the Amazon Relational Database Service ("RDS"). RDS uses EBS for database and log storage, and as a result, a portion of the RDS databases hosted in the affected Availability Zone became inaccessible. Throughout the course of the event, customers were able to create new RDS instances and access existing RDS instances in the unaffected Availability Zones in the region.

Amazon RDS provides two modes of operation: Single Availability Zone (Single-AZ), where a single database instance operates in one Availability Zone; and Multi Availability Zone (Multi-AZ), where two database instances are synchronously operated in two different Availability Zones. For Multi-AZ RDS, one of the two database instances is the "primary" and the other is a "standby." The primary handles all database requests and replicates to the standby. In the case where a primary fails, the standby is promoted to be the new primary and is available to handle database requests after integrity checks are completed.

Single-AZ database instances are exposed to disruptions in an Availability Zone. In this case, a Single-AZ database instance would have been affected if one of the EBS volumes it was relying on got stuck. During this event, a significant number of the Single-AZ databases in the affected zone became stuck as the EBS volumes used by them were affected by the primary EBS event described above. In the case of these Single-AZ databases, recovery depended on waiting for the underlying EBS volumes to have their performance restored. By 1:30PM PDT, a significant number of the impaired Single-AZ RDS instances were restored as the volumes they depended on became unstuck. By 3:30PM PDT, the majority of the affected database instances were restored, and by 6:35PM PDT, almost all of the affected Single-AZ RDS instances were restored.

During the course of the event, almost all of the Multi-AZ instances were promoted to their standby in a healthy Availability Zone, and were available to handle database requests after integrity checks were completed. However, a single digit percentage of Multi-AZ RDS instances in the affected Availability Zone did not failover automatically due to two different software bugs. The first group of RDS instances that did not failover as expected encountered an uncommon stuck I/O condition, which the automatic failover logic did not handle correctly. These instances required operator action and were fully restored by 11:30 AM PDT. We have developed a fix for this bug and are in the process of rolling it out. The second group of Multi-AZ instances did not failover automatically because the master database instances were disconnected from their standby for a brief time interval immediately before these master database instances' volumes became stuck. Normally these events are simultaneous. Between the period of time the masters were disconnected from their standbys and the point where volumes became stuck, the masters continued to process transactions without being able to replicate to their standbys. When these masters subsequently became stuck, the system blocked automatic failover to the out-of-date standbys. We have already been working on a fix for this issue which will allow the standby to be favored immediately when its master is in an impaired Availability Zone. Due to the subtle nature of the issues involved, we are still in the process of completing this fix and carefully testing it, but are on track to deploy it fully by December. Database instances affected by this condition were restored once the associated EBS volumes had performance restored. While we are disappointed with the impact to these Multi-AZ instances, we are confident that when we complete the deployment of these two bug fixes, the root cause of the Multi-AZ failures we observed during this event will be addressed. It is the top priority of the team to complete these fixes and get them deployed to the fleet.

Customers affected by the Multi-AZ RDS issues did not get the availability they or we expected. If an application is Multi-AZ and has enough resources running to continue operating if one Availability Zone is lost, then that application component should remain available (with minimal service disruption). Accordingly, AWS will issue service credits to customers whose RDS Multi-AZ instances took longer than 20 minutes to fail over to their secondary copies, equal to 10 days of charges for those affected Multi-AZ instances. Affected customers do not need to take any action; the credits will be automatically applied to their AWS account prior to their October 31 bill being calculated.

Impact on Amazon Elastic Load Balancing (ELB)

This event also affected the Amazon Elastic Load Balancing (ELB) service. Each ELB load balancer uses one or more load balancer instances to route traffic to customers' EC2 instances. These ELB load balancer instances use EBS for storing configuration and monitoring information, and when the EBS volumes on these load balancer instances hung, some of the ELB load balancers became degraded and the ELB service began executing recovery workflows to either restore or replace the affected load balancer instances. Customers can use ELB with applications that the run in either single or multiple Availability Zones.

For customers using an ELB load balancer with an application running in a single Available Zone, ELB provisions load balancer instances in the Availability Zone in which the application is running (effectively creating a Single-AZ load balancer). During this event, a number of Single-AZ load balancers in the affected Availability Zone became impaired when some or all of the load balancer instances used by the load balancer became inaccessible due to the primary EBS issue. These affected load balancers recovered as soon as the ELB system was able to provision additional EBS volumes in the affected Availability Zone, or in some cases, when the EBS volumes on which particular load balancers relied, were restored. By 1:10PM PDT, the majority of affected Single-AZ load balancers had recovered, and by 3:30PM PDT, most of the remaining load balancers had also been recovered. Recovery of the last remaining load balancers was then slowed by an issue encountered by the ELB recovery workflows. ELB uses Elastic IP addresses (EIPs) to reliably route traffic to load balancer instances. EIPs are consumed as new load balancers are created and as existing load balancers are scaled. The increased demand for EIPs from the ELB recovery workflows (and the overall increase of customer activity during this period) caused ELB to consume all of the EIPs that were available to it. This stalled the recovery workflows and delayed recovery of the final affected load balancers. The team continued to manually recover the remaining impaired load balancers and was able to remediate the EIP shortage at 9:50PM PDT.

We are working on a number of improvements to shorten the recovery time of ELB for all customers. We will ensure that we have additional EIP capacity available to the ELB system at all times to allow full recovery of any Availability Zone issue. We are already in the process of making a few changes to reduce the interdependency between ELB and EBS to avoid correlated failure in future events and allow ELB recovery even when there are EBS issues within an Availability Zone. Finally, we are also in the process of a few additional improvements to our recovery workflows that will be released in the coming weeks that will further improve the recovery time of ELB load balancers during any similar event.

For customers using an ELB load balancer with an application running in multiple Availability Zones, ELB will provision load balancer instances in every Availability Zone in which the application is running. For these multiple Availability Zone applications, ELB can route traffic away from degraded Availability Zones to allow multiple Availability Zone applications to quickly recover. During this event, customers using ELB with applications running in multiple Availability Zones that included the affected Availability Zone may have experienced elevated error rates during the early parts of the primary event as load balancer instances or the EC2 instances running the customer's application were affected by the EBS issue. By 11:49AM PDT, the ELB service shifted customer traffic away from the impaired Availability Zone for most load balancers with multiple Availability Zone applications. This allowed applications behind these load balancers to serve traffic from their instances in other, unaffected Availability Zones. Unfortunately, a bug in the traffic shifting functionality incorrectly mapped a small number of the affected load balancers and therefore didn't shift traffic correctly. These load balancers continued to send a portion of the customer requests to the affected Availability Zone until the issue was identified and corrected at 12:45PM PDT. We have corrected the logic in the ELB traffic shifting functionality so this error will not occur in the future. We are also working to improve the sensitivity of the traffic shifting procedure so that traffic is more quickly failed away from a degraded Availability Zone in the future. Over time, we will also expose this traffic shifting functionality directly to ELB customers so that they have the ability to control the routing of their requests to the Availability Zones in which they run their applications. Finally, we will work on helping our customers understand and test the impact of this traffic shift so that they can be sure their applications can scale to handle the increased load caused by failing away from an Availability Zone.

Final Thoughts

We apologize for the inconvenience and trouble this caused for affected customers. We know how critical our services are to our customers' businesses, and will work hard (and expeditiously) to apply the learning from this event to our services. While we saw that some of the changes that we previously made helped us mitigate some of the impact, we also learned about new failure modes. We will spend many hours over the coming days and weeks improving our understanding of the event and further investing in the resiliency of our services.

Sincerely,
The AWS Team