the Availability Digest

www.availabilitydigest.com

Amazon Downed by Memory Leak

November 2012

On Monday, October 22, 2012, Amazon Web Services (AWS) suffered a major multi-hour outage in one of its Northeastern Availability Zones. The outage took down several customers, including Reddit, Pinterest, Foursquare, Minecraft, Heroku, GitHub, imgur, Pocket, HipChat, Coursers, Airbnb, and others.

The problem began in a noncritical program that had no significant role in the ongoing operations of AWS. It was a memory leak whose impact cascaded unnoticed over several hours to critical components and finally disabled most of an entire Availability Zone.

The series of faults that were invoked by a single innocuous memory leak are so complex that it first requires an understanding of the AWS architecture.

The AWS Architecture

We have described the AWS architecture in some detail in our earlier article, entitled "Amazon's Availability Zones."[1] We summarize that description here.

The AWS Cloud

Amazon Web Services (AWS) provides a suite of IT infrastructure resources that can be used by businesses to implement their applications. The processing, storage, and network capacity provided by the AWS services can expand and contract as the application's needs change. A customer pays only for what his applications use. Amazon's retail service runs in the Amazon cloud.

AWS Services

The primary AWS services include:

- Availability Zones – fault-isolated virtual data centers within a regional data center.
- Elastic Compute Cloud (EC2) – provides resizable compute, storage, and network capacity.
- Elastic IP Addresses (EIP) – static IP addresses that can be migrated between EC2 instances.
- Simple Storage Service (S3) – stores and retrieves data via a simple web interface (URLs).
- Elastic Block Store (EBS) – provides persistent storage for an EC2 instance.
- Relational Database Service (RDS) – scalable MySQL and Oracle database support.

[1] Amazon's Availability Zones, *Availability Digest*; November 2011. http://www.availabilitydigest.com/public_articles/0611/amazon_avilability_zones.pdf

- Elastic Load Balancing - distributes traffic across multiple EC2 instances that can be in different Availability Zones.

Below, we describe the AWS services that make up the Amazon cloud and how they interoperate to provide high availability of cloud-based applications.

Availability Zones (AZ)

For critical applications that just cannot go down, it is imperative to run instances of the application in two or more geographically separated data centers so that the application survives any destructive event. For this purpose, Amazon operates data centers in multiple regions throughout the world. Within each region are multiple fault-isolated Availability Zones. Amazon's Availability Zones are a major move towards achieving high availability in the cloud.

Availability Zones are distinct locations within a region that are engineered to be insulated from failures in other Availability Zones and that provide inexpensive, low latency connectivity to other Availability Zones in the same region. An Availability Zone (AZ) is in effect a separate data center in a region. It has its own power, cooling, and communication infrastructure that is separate from the other Availability Zones in its region. Therefore, a failure of one AZ will not affect another AZ except for a regional disaster.

An AZ is not necessarily in a different data center from the other AZs. It is only necessary that the data center be built to ensure the independence of the AZs.

A customer can run critical applications in two AZs, with one instance handling the production workload and the other standing by in order to take over should the production application instance fail.

Elastic Compute Cloud (EC2)

The Elastic Compute Cloud (EC2) is the primary infrastructure for the Amazon cloud, providing resizable compute facilities. The fundamental building block of the Elastic Compute Cloud is an *EC2 instance*. An EC2 instance is an application running in the AWS cloud as a virtual machine (VM).

To provide fault recovery, multiple EC2 instances of an application can be run in the same or different Availability Zones or regions.

Elastic IP Addresses (EIP)

An Elastic IP Address (EIP) is an IP address that can be assigned to an EC2 instance. However, unlike a static address, it can easily be migrated to another instance. When an EC2 instance is created, it is given an EIP address. All traffic routed to that EIP address then goes to the instance to which the EIP address is assigned. If the EIP address is reassigned to another EC2 instance, all further traffic will be routed to the new instance.

EIP addressing brings a significant availability advantage to EC2 in that it provides a simple mechanism to reroute traffic to a backup EC2 instance should the primary instance fail. All that needs to be done is to issue a command to point the EIP address to the new instance that should take over processing.

Simple Storage System (S3)

The Simple Storage System (S3) provides storage services that can be used to store any amount of data. Data is stored in *buckets*. A bucket contains multiple data objects, typically files. A data object's size can be anywhere from one byte to 5 GB.

A bucket can be stored in any region. It is accessed over the Internet and is identified by a unique URL specified by the user. Objects within the bucket are specified by user defined keys.

Up to six replicas of an S3 bucket can be maintained across multiple AZs in different regions. S3 storage has high access latency since it is accessed over the Internet. Therefore, it is not suitable for applications that require high performance.

Elastic Block Store (EBS)

The Elastic Block Store (EBS) is persistent block-level storage. An EBS volume can be sized from 1 GB to 1 TB. It can be mounted by any EC2 instance (just one EC2 instance at a time) and can be used just as any block storage device. Typically, a database management system is run in an EC2 instance that uses EBS for its underlying storage.

EBS is designed to be highly available. A logical EBS volume is synchronously replicated to multiple EBS physical volumes in the same AZ. One of the physical volumes is the primary volume and is the one used by the EC2 instance that has it mounted. The other volumes are used only for redundancy to ensure that data is not lost due to a volume failure. If a volume should fail (primary or backup), EBS searches the physical volumes to find one with enough space to use as a replacement redundant volume. It then makes a copy of the logical volume onto the new physical volume. This is called *remirroring*. During remirroring, the logical volume is not accessible. Applications attempting to use this volume are said to be "stuck."

An EC2 instance can mount several logical EBS volumes. Backup EBS logical volumes can be created in other Availability Zones and kept synchronized with the primary volume using synchronous replication. In this way, if the primary AZ should fail, a new EC2 instance can be rapidly created in another AZ and can mount the backup volume. As soon as the Elastic IP address is redirected and remirroring is completed, the applications hosted by that EC2 instance are back up and running.

Relational Database Service (RDS)

The Relational Database Service (RDS) sets up, manages, and scales a relational database in the cloud. Both MySQL and Oracle are supported. RDS uses EBS as its underlying block store.

An RDS database can be deployed across multiple AZs either for backup or for read replicas. If used for backup, the backup databases are kept synchronized with synchronous replication. In this case, the backup databases cannot be used for reading.

RDS databases can also be deployed in multiple regions as read replicas. In this case, they are kept synchronized via asynchronous replication and will lag the primary database.

Elastic Load Balancing (ELB)

Elastic Load Balancing (ELB) distributes traffic across multiple EC2 instances that can be in multiple Availability Zones. It detects unhealthy EC2 instances and ceases to route traffic to them.

Intra/Inter Availability Zone Communications

The EC2 instances in a region communicate with their EBS storage arrays via the redundant EC2 Control Plane. All requests for creation, deletion, and read/write access between an EC2 instance and an EBS volume is carried by the Control Plane as shown in Figure 1.

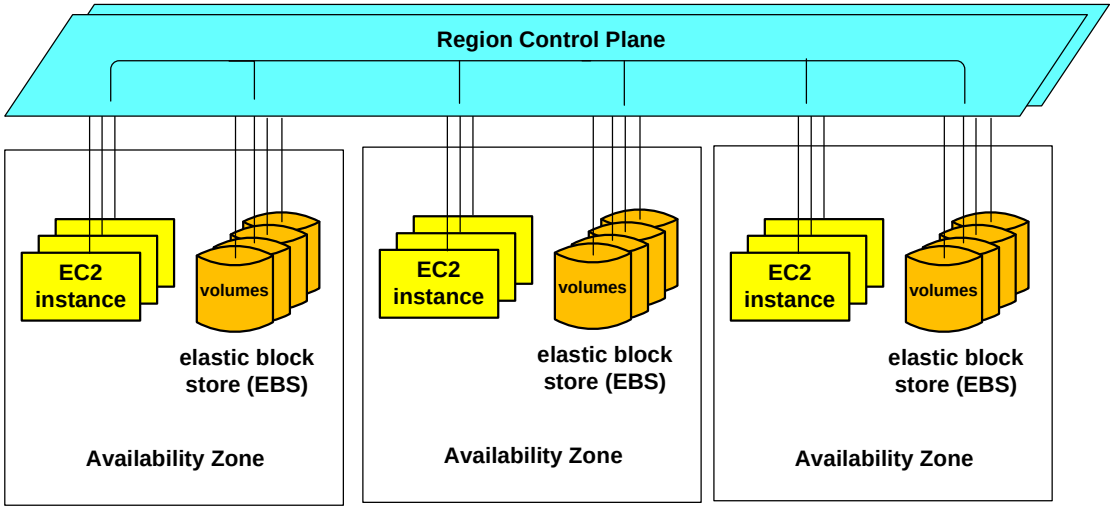

Figure 1: An Amazon EC2 Region

The AWS Outage

We can now turn our attention to the AWS outage of October 22. In typical (and wonderful) Amazon style, a detailed description of the outage, its symptoms, its causes, and its cures was subsequently published by Amazon.[2] The following description of events is taken from that description.

The Memory Leak

At 10 AM on Monday, October 22, 2012, AWS operations staff noted that a small number of EBS volumes in one Availability Zone in Amazon's Northeast region had become stuck and were unable to process further I/O requests. As designed, they began to fail over to healthy EBS servers in the Availability Zone. As time went on, more and more EBS volumes got stuck. Ultimately, there were not enough servers for the stuck EBS volumes to fail over to. EC2 instances were losing their EBS storage; and their applications failed, taking down a multitude of customers.

The team made adjustments to control the failover rate by throttling calls to AWS services, and the system began to slowly recover volumes. However, the large surge in failover/recovery activity made it difficult for the team to determine the root cause of the problem.

Finally, the problem was identified. It had been initiated several days previous to the outage when a data collection server failed. The AWS data collection servers collect information from the various systems in AWS to guide maintenance activities. Their operation is important but not time sensitive. They are designed to tolerate late and lost data.

The server was replaced, and a DNS record was modified to reroute traffic from the failed server to the new server. Let the bugs begin.

Bug Number 1 - The DNS change did not propagate to all of Amazon's internal DNS servers. The EBS servers use a data collection agent to send health information to the data collection servers. For those EBS servers that did not get the DNS update, the data collection agent continued to attempt to contact the failed data collection server, but to no avail. Because of the fault-tolerant design of the data collection servers and their agents, this did not cause an issue; and alarms were not generated.

[2] Amazon Web Services Message 680342, http://aws.amazon.com/message/680342.

Bug Number 2 – However, this action triggered a latent memory leak in the data collection agents on the EBS servers. The memory leak slowly consumed the servers' system memory until the servers could no longer function. At this point, they became stuck and attempted to fail over to healthy servers. Initially, these appeared to be normal failover events and caused no alarm on the part of the operations staff. However, as more and more EBS servers began to fail, the failover activity increased to the point that there were not enough healthy EBS servers to which to fail over. At this point, applications began to fail.

Bug Number 3 – The EBS servers monitor their use of memory and are configured to raise an alarm if a memory leak is detected. However, they make very dynamic use of memory; and it is hard to establish meaningful thresholds. As a result, the memory leaks were not identified; and no alarms were generated.

A little after 3 PM, the operations staff determined the cause of the problem and began restoring affected volumes by freeing the excess memory consumed by the misbehaving data collection agents. By 4:15 PM, most volumes were recovered and were functioning properly. The Amazon cloud's Availability Zone had been down for five hours.

However, the bugs did not end here.

The Impact on EC2 and EBS APIs

Many customers complained that, even though their applications were up and running, they couldn't manage them. AWS provides an extensive API (application programming interface) to configure and manage applications, and these APIs were executing painfully slowly. In some cases, their poor performance made them useless.

Before the memory-leak problem was understood, the operations staff moved to reduce the load on the system in a desperate attempt to allow the system more time to return to stability. They did this by throttling the rate at which APIs could be executed.

A level of throttling is implemented in AWS to prevent callers from overwhelming services. An example of such a caller is a runaway application that retries a failed request as fast as it possibly can.

Bug Number 4 - As it turned out, the team was too aggressive in its throttling restrictions, resulting in an inability to keep up with the normal API activity. There was no way to fine-tune throttling by customer, which may have allowed the team to give priority to critical applications. Rather, throttling could only be done on an aggregate basis across the entire AZ.

After 2:30 PM, the team significantly reduced the level of throttling and freed up API execution. This problem did not affect applications that were running across multiple AZs and that had failed over to their backup AZs.

The Impact on the Relational Database Service (RDS)

RDS uses EBS for its database and transaction log storage. RDS can be configured as a single-AZ database in which case it is running in only one AZ. Alternatively, it can be configured as a multi-AZ database in which a primary database is running in one AZ and its backup is running in another AZ. In this case, if the primary database fails, the standby database is promoted to primary and is available after its integrity checks are completed.

The single-AZ RDS databases became stuck if they depended upon a stuck EBS volume. However, multi-AZ databases were able to recover to their standby databases and continue operating – almost. A small portion of RDS multi-AZ RDS instances did not recover because of two additional software bugs.

Bug Number 5 – Some multi-AZ RDS databases encountered an uncommon stuck I/O condition and had to be failed over manually.

Bug Number 6 – Some primary databases had been disconnected from their backups just prior to the failure. Therefore, their backups were missing some data updates. The system blocked failover to the out-of-date standbys. The primary RDS databases could not be restored until their EBS volumes were restored.

Impact on Elastic Load Balancer (ELB)

The Elastic Load Balancer routes traffic to the customer EC2 instances. It uses Elastic IP addresses (EIPs) for routing within and across AZs, and it uses EBS for storing its configuration and monitoring information.

Because of the EBS problems, the ELBs became degraded, and the ELB service began executing recovery workflows to replace the degraded ELB instances. Many were recovered by 3:30 PM. But then:

Bug Number 7 – A bug in the ELB recovery workflow reared its head. The recovery service ran out of EIPs. This stalled the recovery workloads.

Bug Number 8 – To make matters worse, a bug in the traffic-shifting functionality in the ELB caused the ELB to not shift some traffic properly. The result was that some traffic continued to be routed to the affected AZ rather than to the backup AZs

The ELB problems were finally resolved by 9:50 PM, and AWS returned to normal. It had been hobbled for almost twelve hours.

Bug Fixes and Enhancements

Amazon immediately initiated efforts to correct the many deficiencies that it had found during this outage:

Bug 1: The DNS servers have been modified to ensure that changes are properly propagated.

Bug 2: The memory leak in the EBS server data collection agent has been plugged. Furthermore, limits on resource usage of low-priority processes have been implemented.

Bug 3: The monitoring of memory usage has been improved.

Bug 4: Throttling has been changed to be on a per-customer basis rather than on an overall aggregate basis.

Bug 5: The multi-AZ RDS stuck I/O condition has been fixed.

Bug 6: If the primary database is in an affected AZ, failover to the standby database is now allowed even if it is out of synchronization with the primary database.

Bug 7: The ELB recovery process has been modified to ensure that there are sufficient EIPs to support a recovery.

Bug 8: The traffic-shifting bug in the ELB has been fixed. Furthermore, the ELB has been modified to more quickly shift traffic; and customers can now control the routing of traffic to specific AZs.

Amazon has offered a wide range of credits to compensate customers for the problems caused by the AWS outage.

Summary

As can be seen from the Amazon AWS cloud, clouds are extremely complex. Bugs will always be lurking in their depths, but with experience they will be slowly flushed out over time and eradicated. Of course, new features will always add new bugs, so this will be a never-ending experience.

Amazon's approach to bug reporting is refreshing as it is very transparent. It helps customers to know exactly what happened and what is being done to correct the problem.

Nonetheless, the complexity of clouds and the propensity of bugs as evidenced in this outage emphasize the need to have an effective and tested business continuity plan to guide you when your critical cloud-based applications suddenly become unavailable.

Acknowledgement

Our thanks to our subscriber, Keith Evans, for pointing us to this incident.