the Availability Digest

www.availabilitydigest.com

Amazon's Cloud Downed by Fat Finger

May 2011

Amazon's Elastic Compute Cloud (EC2) is arguably today the leading service for deploying custom applications in a cloud environment. Amazon has gone to great lengths to ensure the availability of its cloud services. It has broken its cloud infrastructure into multiple regional, geographically-separated data centers; and within each region it provides independent Availability Zones. A customer can run a critical application in multiple Availability Zones within a region to ensure availability. If desired, an application can also have a redundant instance in other regions.

Yet just after midnight Pacific Daylight Savings Time on April 21, 2011, a maintenance error took down an entire Availability Zone in Amazon's U.S.-East Region, located in northern Virginia. The other three Availability Zones in the region were also affected, even though the Availability Zones were supposed to be independent.

Even worse, the result of this maintenance error was not to take the cloud down for a few minutes or even a few hours. It was not until late April 24th, four days later, that the U.S.-East Region was fully returned to service.

The EC2 Architecture

The EC2 regional data centers are truly independent of each other. The architecture of interest is that of the Availability Zones within a region. Each regional data center is configured with several Availability Zones. The U.S.-East Region data center, for instance, has four Availability Zones. The Availability Zones are intended to be completely independent of each other. Should one fail, the others continue uninterrupted. (As we shall see in the following description of Amazon's outage, this observation is flawed.)

As an option, Amazon supports customers who have critical applications running in one Availability Zone to be backed up by other instances of those applications running in different Availability Zones (albeit in the same region).

To understand how this failure occurred, it is necessary to understand the architecture of a region and its Availability Zones.

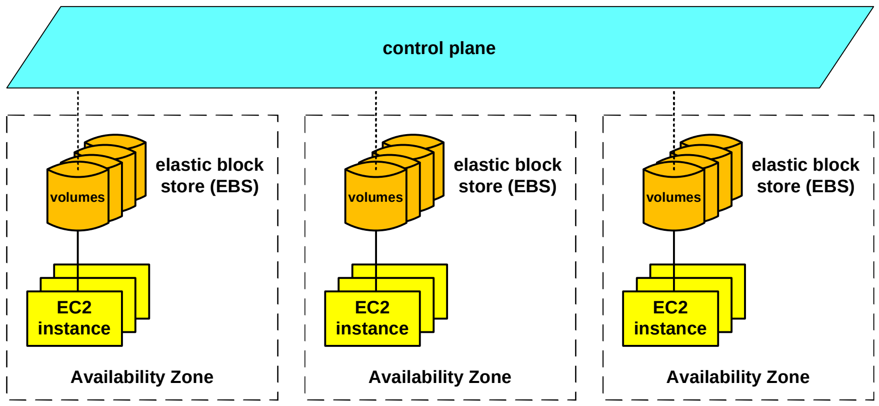
Elastic Block Store

The heart of each Availability Zone (AZ) is a clustered, highly redundant database called the *Elastic Block Store (EBS)*. EBS is a distributed, replicated block store optimized for consistency, availability, and low-latency read/write access.

EBS provides the data storage for the applications running in the AZ – the *EC2 instances*. An EBS cluster comprises a set of EBS nodes that store replicas of EBS volume data and that execute read/write requests issued by the EC2 instances. The contents of each volume are replicated to multiple backup EBS nodes. Fast failover provides recovery from a primary node failure in milliseconds.

A *control plane* provides services that coordinate user requests and that propagate user requests between EBS clusters in each of the AZs in the region.

The nodes within an EBS cluster are interconnected via two networks. The primary network is a high-bandwidth network that is used for normal communication between the EBS nodes within the cluster, with EC2 instances serviced by the cluster, and with the EBS control-plane services. The secondary network is a lower-capacity network used as a backup for the primary network should it fail and to provide overflow replication capacity if needed.



An Amazon EC2 Region

Remirroring

When a node loses connectivity to one of its replica nodes, it searches for a new node to which it can replicate its data. This is called *remirroring*. To do this, the primary node searches its EBS cluster for another node that has enough storage capacity to act as a replica. It typically takes only milliseconds to locate a node that can act as a new replica, and the transfer of volume data is then initiated to the new replica.

To ensure consistency, access to volumes that are being remirrored are blocked until the remirroring has been completed and a primary replica is identified. While this is happening, EC2 instances attempting to access this data are blocked – in EC2 terms, they are *stuck*.

The EBS Control Plane

A control plane provides common services to all EBS clusters in a region. It routes user requests to the appropriate EBS cluster even if that cluster is in a different AZ than the requesting EC2 instance. It also acts as the authority to ensure that there is one and only one primary replica for each volume at any one time. By interconnecting the AZs in a region, the control plane is the key to providing high availability and fault tolerance for the EC2 instances.

High Availability

An EC2 instance only needs to run in one AZ. However, it is then subject to the failure of that AZ (for instance, should the AZ's EBS cluster fail).

To guarantee availability even in the event of an AZ failure, a customer can elect to run a critical application in multiple AZs. Through control-plane services, the contents of an application's database running in one AZ is replicated to the EBS clusters in the backup AZs. Therefore, should an AZ fail, the EC2 instance can be rapidly failed over to one of its backup AZs.

A customer can run multiple instances of an application in multiple regions if it is desired to protect the application services from an entire regional data-center failure. However, Amazon does not provide data replication between regions. This is the responsibility of the application.

The Outage

This is a tale of multiple faults, one following the other to cascade an outage that should have been minutes into days. Following the failure, Amazon published a detailed account[1] of what happened (an unusual act of transparency for data centers of any kind). Its report is summarized below.

Human Error – The Fat Finger

The outage began with a normal maintenance activity. Shortly after midnight on April 21st, maintenance to upgrade the capacity of the primary network of one of the AZs in the eastern region was begun. The primary network is a redundant network. The first step was to shift all traffic off of one of the redundant routers so that its link could be upgraded.

Unfortunately, the traffic shift was executed incorrectly. Rather than shifting all traffic to one of the high-speed primary links, all traffic was instead rerouted to the much slower secondary network. The secondary network could not handle the traffic; consequently, the nodes in the cluster were effectively isolated from one another. So far as the nodes were concerned, their replicas were down.

The Remirroring Storm

The network was quickly restored by the maintenance personnel. At this point, the nodes started searching the cluster for other nodes that they could use to remirror their data. However, whatever free space was available in the cluster was quickly exhausted, leaving many nodes in a stuck state continuously searching the cluster for free space.

This *remirroring storm* exposed an unknown bug in the EBS software and caused a node to fail if it was trying to close a large number of replication requests. This caused more nodes to fail and increased the intensity of the remirroring storm.

The heavy control-plane traffic caused it to run out of threads. As a result, it could not service requests from other AZs and began to fail those requests.

The Recovery That Took Days

The recovery effort was intense for the Amazon staff:

April 21st

2:40 AM: Create Volume requests in the affected AZ were disabled to decrease the load on the control plane.

[1] Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US-East Region
http://aws.amazon.com/message/65648/

2:50 AM: Latencies and error rates for other EBS functions returned to normal.

5:30 AM: The latencies and error rates again increased for requested EBS functions due to control-plane overload. The control plane acts as the authority to negotiate the remirroring of a volume and designate a new primary volume. As more nodes failed due to the EBS bug described previously, the volume of control-plane negotiations increased. This caused a brownout of the control plane and affected EBS clusters across the region.

8:20 AM: The team began disabling all communication between the degraded EBS cluster and the control plane. Error rates and latencies returned to normal for all other AZs, but access to the affected AZ was now disabled.

11:30 AM: The affected cluster was still degrading as the remirroring storm continued. The team devised a way to prevent EBS servers in the degraded cluster from futilely contacting other servers in the cluster that did not have any space to share. The cluster stopped degrading further. Volumes started to remirror slowly, allowing stuck volumes to become unstuck.

At this time, another problem was resolved. EC2 instances in AZs other than the affected AZs were experiencing elevated error rates and latencies due to the control-plane overload. Making alarms more fine-grained allowed these conditions to be exposed and corrected more readily. By noontime, the other AZs had returned to normal operation.

At this point, the degraded AZ had stabilized; and the other AZs were working normally.

April 22nd

When a node fails, it cannot be reused until all replicas in which it was involved have been remirrored. But there was no space available for remirroring. Therefore, significant new storage had to be added. This required physically relocating excess server capacity from across the U.S.-East Region to the degraded AZ.

However, due to changes made to throttle control-plane negotiations to help calm the remirroring storm, it was difficult to install the new capacity in the cluster. The team had to carefully modify the negotiation throttles to allow negotiation with the new nodes without inundating the old nodes with requests that they could not handle.

12:30 PM: Most remirroring was completed. However, EC2 instances could not reach many of the newly functional nodes because their communication with the control plane was still blocked.

April 23rd

A separate instance of the control plane was built to service just the degraded AZ. In this way, the heavy traffic required to bring the nodes back online would not affect the other AZs.

11:30 AM: Processing began of the backlog of requests required to return the EBS nodes to service.

3:35 PM: Access to the control plane by the degraded EBS was enabled.

6:15 PM: Access to all but 2.2% of the EBS was restored.

April 24th

The recovery of the remaining 2.2% of volumes required restoring snapshots that had been taken early in the event as a precaution.

12:30 PM: All but 1.04% of the volumes had been restored. The team began forensics on the volumes that had suffered machine failure and that could not be backed up.

3:00 PM: The team began restoring the volumes that had no backups. Ultimately, all but 0.07% of the data was restored. The remaining data was irretrievably lost.

Relational Data Services Also Taken Down

Amazon’s Relational Data Services (RDS) facility allows customers to operate RDS instances either in a single AZ (single-AZ) or across multiple AZs (multi-AZ). Since RDS instances access multiple volumes, a single-AZ instance will be stuck if any one of its volumes becomes stuck. Thus, almost half of the single-AZ RDS instances running in the degraded AZ failed.

Multi-AZ synchronously replicates data in the primary volume to a volume in another AZ in the same region. If the primary volume fails, RDS automatically fails over to the replicated backup. Therefore, multi-AZ will survive an AZ failure. In this case, all but 2.5% of the affected multi-AZ RDS instances failed over. The unsuccessful failovers were due to a previously unknown bug in the failover mechanism.

Lessons Learned and Future Enhancements

This outage event was caused by a long string of root causes interacting with each other. Amazon, in its outage report, listed several actions that it will take to prevent such an occurrence in the future:

1. The trigger for this outage was a network configuration change. Maintenance procedures will be further automated. However, the system should have been able to tolerate this error.

2. Amazon will make a number of changes to prevent a remirroring storm:
    a. Excess storage capacity will be added to the EBS clusters.
    b. Retry logic will back off more aggressively when a large interruption occurs and will focus on reconnecting with previous replicas rather than futilely trying to find storage space on other nodes.

3. The control plane will be improved:
    a. Timeout logic will be improved to prevent thread exhaustion when an AZ cluster is taking too long to process requests.
    b. The capability to be more AZ-aware and to shed load intelligently when the control plane is overloaded will be added.
    c. EBS-related services will be moved out of the control plane into the EBS.

4. Multi-AZ deployments will be made easier to design and operate.

5. The recovery models for the various types of volume recovery will be automated.

6. Snapshots of stuck volumes will be allowed to more easily recover applications in other AZs.

7. Customer crisis communications will be improved.

Amazon will give a ten-day credit to all customers for their usage of EBS volumes, EC2 instances, and RDS instances in the affected AZ.

Post Script

Amazon's outage supports the caution that "the cloud can be dangerous to your health." True, the cloud brings with it numerous advantages, including being able to provision new applications quickly and reliably while paying only for what you use. Even with the spectacular cloud failures that are periodically reported in the press, it is likely that the cloud providers provide better availability, performance, and security than a data center that you might operate.

Still, if you have an application that is so critical that you can tolerate no downtime, you have to take precautions that may extend beyond the cloud. Using Amazon's Availability Zones for redundancy, especially across regions, is a good start. But you may want to have further control over your destiny.

A good case in point is Mashery, an Amazon EC2 user. Mashery helps engineer and monitor the APIs that tie a company's service to its customers. Mashery's customers include the New York Times, Netflix, and Best Buy.

Mashery made the assumption that everything in the cloud will fail, so it set up a failover site via an outside data-center service supplier, InterNAP.[2] Switchover from EC2 to the InterNAP instance of its application is accomplished simply by changing the URL routing in its Domain Name System (DNS) server.

When the EC2 outage occurred, Mashery's failover to its backup site functioned as expected. Its monitoring and reporting traffic was rerouted from Amazon's U.S.-East Region to its backup systems at InterNAP. Its services were down for about two minutes while failover was taking place.

Another example is that of Bizo, which provides web-based business-marketing services. Bizo uses two Availability Zones in the U.S.-East Region backed up by two Availability Zones in Amazon's second North American region in northern California. When the outage occurred, Bizo was able to restore service by DNS rerouting just as Mashery did. Failover took about eight minutes.

These are examples of backing up truly critical cloud applications either in another cloud as in Bizo's case or as in Mashery's case to a data center outside of the cloud. A technique similar to these should be considered mandatory if you decide to commit a highly critical application to the cloud.

[2] Amazon Cloud Outage Proves Importance of Failover Planning, *Information Week*; April 28, 2011.