CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

DATACENTER OUTAGE STUDIES

Lecture B

AWS OUTAGE

Overview

- Occurred on April 21st, 2011
- AWS published a post-mortem analysis
  - http://aws.amazon.com/message/65648/
- Not our goal to say AWS is a bad infrastructure
  - In fact, quite the opposite – AWS treated customers very well
  - After the outage, AWS is still market leader
  - AWS fixed infrastructure to prevent recurrence
- During the outage
  - Several companies using AWS EC2 went down, e.g., Reddit, FourSquare
  - AWS dashboard showed problems with EC2, and other storage
  - Lasted 3.5 days (at least)
  - Led to some data loss

Background

- AWS Regions: Separate datacenters
  - E.g., us-east-1, us-west-1, etc.
  - Each region consists of availability zones
    - Can have automatic data replication across zones in a region (though not all customers do it)
- AWS Elastic Block Storage (**EBS**) – mountable storage "devices," accessible from EC2 instances
- 1 EBS volume runs inside an Availability Zone
  - Two networks: primary n/w used for EC2 and EBS control plane; secondary n/w used for overflow – has *lower capacity*
  - Control information replicated across zones (for availability)
- EBS volumes replicated for durability
    - Each volume has a primary replica
    - If out of sync or node failure, replicas programmed to do aggressive re-mirroring of data

Timeline of Outage

- *12:47 AM: Routine primary n/w capacity upgrade in an av. zone in US East Region*
- Traffic shifted off several primary n/w routers to other primary n/w routers
  - Critical Error: someone shifted traffic for one such router to a secondary n/w router
- => Several EBS volumes now had no/bad primary n/w
  - Primary n/w disconnected
  - Second n/w has low capacity and thus overwhelmed
  - Many primary replicas had no backup
- Team discovered critical error and rolled it back

(Is it over yet?)

Timeline of Outage (2)

- Team discovered critical error and rolled it back
  - Due to network partitioning, many primary replicas thought they had no backup: these automatically started re-mirroring aggressively
  - *All at once*: free n/w cap quickly used, replicas stuck in loop
  - Re-mirroring *storm*: 13% of EBS volumes
- N/w unavailable for control plane
  - Unable to serve “create volume” API requests for EBS
  - Control plane ops have long time-out; began to back up
  - When thread pool filled up, control plane started to reject create volume requests
- *2:40 AM: Team disabled all new “create volume” API requests*
- *2:50 AM: all error rates and latencies for EBS APIs start to recover*

(Is it over yet?)

Timeline of Outage (3)

- Two issues made things worse
  - Primaries searching for potential replicas did not back off
  - A race condition existed in EBS code that was only triggered by high request rates: activated now, caused more node failures
- *5:30 AM: Error rates and latencies increase again*
- Re-mirroring is negotiation b/w EC2 node, EBS node, and EBS control plane (to ensure 1 primary)
  - Due to race condition, EBS nodes started to fail
  - Rate of negotiations increased
  - Caused more node failures (via race), and rinse-n-repeat
  - “Brown-out” of EBS API functionalities
- *8:20 AM: Team starts disabling all communication b/w EBS cluster in affected av. zone and EBS control plane*
  - Av. zone still down, but control plane recovering slowly

Timeline of Outage (4)

- *11:30 am: Team figures out how to prevent EBS servers in av. zone from futile re-mirroring*
  - Affected av. zone slowly recovers
- Customers still continued to face high error rates for new EBS-backed EC2 instances until noon
  - Another new EBS control plane API had recently been launched (for attaching new EC2 instances to volumes)
  - Its error rates were being shadowed by new errors
- Noon: No more volumes getting stuck
- But 13% volumes still in stuck state

Timeline of Outage (5)

- Long tail of recovery
  - Read more on the post-mortem to find out how team addressed this
  - By noon April 24th, all but 1.04% of volumes had been restored
  - Eventually, 0.07% volumes could not be recovered, and were lost forever
- This outage also affected relational database service (RDS) that were single – av. zone.

General Lessons Learnt

Large outages/failures

- Often start from human error
- But balloon due to *cascading* sub-failures

Specific Lessons Learnt

Ways this outage could have been avoided:

- Audit n/w configuration change processes, create a step-by-step protocol for upgrades
- Higher capacity in secondary n/w
- Prevent re-mirroring storm: backing off rather than aggressively retry
- Fixing race condition
- Users who wrote code to take advantage of multiple av. zones within region not affected
- Better tools for communication, health (AWS Dashboard), service credit for customers (multi-day credit)