CS 425 / ECE 428
Distributed Systems
Fall 2022

Indranil Gupta (Indy)

Lecture 28: Datacenter Disasters

Take a Guess!

Which of the following do you think the leading cause of datacenter outages?

1. Power outage
2. Over-heating
3. Human error
4. Fire
5. DOS attacks

Take a Guess!

Which of the following do you think the leading cause of datacenter outages?

1. Power outage
2. Over-heating
3. Human error (70%)
4. Fire
5. DOS attacks

Human Error Examples

- A system operator mistakenly deleted the $38 billion Alaska Permanent Fund database and then deleted its backup.
- A maintenance contractor’s mistake shut down the Oakland Air Traffic Control Center.
- A State of Virginia technician pulled the wrong controller and crashed a redundant SAN that already had suffered a controller failure.
- A technician with DBS Bank made an unauthorized repair on a redundant SAN and took down both sides.

Source:
http://www.availabilitydigest.com/public_articles/0704/data_center_outages-lessons.pdf

Human Error Examples (2)

- A system administrator closed all applications on one server in an active/active pair to upgrade it and then shut down the operating server.
- A test technician failed to disable a fire alarm actuator prior to testing the fire suppression system.
- Siren noise damaged several disks, including the virtual backup disks.
  - Shouting in the datacenter: https://www.youtube.com/watch?v=tDacjrSCeq4
- (hosting.com) Incorrect breaker operation sequence executed by servicing vendor caused a shutdown of UPS and offline time to websites of 1-5 hours
- Thirteen million German web sites went dark when an operator mistakenly uploaded an empty zone file.
- (And many more!)

Source:
http://www.availabilitydigest.com/public_articles/0704/data_center_outages-lessons.pdf

Why Study Outages?

- They’re fun! (Schadenfreude!)
- But really – so that we can learn lessons
- Learn more about the actual behavior of systems in the real world
- Design better systems in the future
- Not our goal to say some companies are worse than others
  - In fact, companies which suffer outages run better and more robust infrastructure afterwards!
  - “What doesn’t kill you, makes you stronger”
  - “It builds character” – Calvin’s Dad (in Calvin and Hobbes)
- We’ll see a few case studies of outages
  - And learn lessons from them

I. 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 (but 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 (Contd.)

- 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 (Contd.)

- 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 (Contd.)

- *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 (Contd.)

- 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)

II. Facebook Outage

Overview

- Outage occurred on 23rd September, 2010
- FB Unreachable for 2.5 hours (worst in past 4 years)
- Facebook published post-mortem
  - https://www.facebook.com/notes/facebook-engineering/more-details-on-todays-outage/431441338919
- Not our goal to say Facebook is a bad infrastructure
  - In fact, after the outage, Facebook still remained market leader in social networks
  - Facebook fixed infrastructure to prevent recurrence

Background

- Data stored in a persistent store, and cache
  - Persistent store = many servers
  - Cache = many servers running a distributed cache system
- Includes configuration data
- FB has automated system for verifying configuration values in the cache
  - and replace invalid values with updated values from the store

Timeline

- On Sep 23, FB made a change to the persistent copy of a configuration
  - Change was invalid
- All clients (FB cache servers) saw invalid value
  - All attempted to fix it
  - All queried cluster of databases
  - Databases overwhelmed quickly by 100K’s queries per second
- Team fixed the invalid configuration

(Is it over yet?)

Timeline (Contd.)

- When client received error from DB, it interpreted it as invalid and deleted cache entry
  - When DB failed to respond => client created more queries
  - No back off
  - Rinse-n-repeat
  - (Cascading failures)

Timeline (Contd.)

- FB's Solution
  - Turn off entire FB website
  - Stop all traffic to DB cluster
  - DB recovers
  - Slowly allow users back on: allowed clients to slowly update caches
  - Took until later in day for entire site to be back up

Lessons Learnt

- New configuration system design
- When cannot access resource
  - Don’t retry aggressively
  - But instead, back off
  - Each time a request fails, wait twice as long as last time
    - Called “Exponential backoff”
    - Used in networking protocols like 802.11 to avoid congestion

III. The Planet Outage

Overview

- Outage occurred on 31st May, 2008
- Source: http://www.availabilitydigest.com/public_articles/0309/planet_explosion.pdf

- The Planet – 4th largest web hosting company, supported 22K websites
  - 6 datacenters: Houston (2), Dallas (4)
- Took down 9K servers and 7.5K businesses

Timeline

- 5.55 pm: Explosion in H1 Houston DC
  - Short circuit in transformer set it on fire
  - Caused an explosion of battery-acid fumes from UPS backup
  - (Cascading failures)
  - Blew out 3 walls of first floor

Timeline (Contd.)

- No servers were damaged, but 9K servers brought down
- Fire department evacuated building
  - Directed that backup generators could not be turned on
  - Due to fire hazard, no staff allowed back in until 10 pm
- Reportedly, the Planet staff had to physically ship some critical servers to their other DCs (on pickups)
  - But limited by power and cooling at other DCs

Timeline (Contd.)

- 5 pm Jun 2: Power restored to second floor
- Jun 4: First floor servers were being restored one rack at a time

- All the while: The Planet provided frequent updates to customers (15 min to 1 hour)

Lessons Learnt

- Backup data & services across DCs, perhaps across different providers
  - "Business Continuity Plans"
  - Whose responsibility would this be?
  - Provider?
  - Customer? More difficult due to extra work and data lock-in across providers.
- May cost customers more
  - Like insurance premiums?

Wrap-up

Outages are Inevitable

- Outages are inevitable
- We’ve seen how AWS, Facebook, The Planet kept affected users updated throughout
  - Frequent updates
  - Coupons/discounts
  - Published post-mortems afterwards
  - All these bolster customer confidence
- Many companies run dashboards with real-time information
  - Google Apps status dashboard
  - AWS dashboard

Not all Companies …

Not all companies are as open as those discussed

- RIM Apr 2007 – day-long outage; no details
- Hostway Jul 2007 – informed customers that it would move its DC Miami → Tampa, and that outage would be 12 hours
    - Outage was 3-7 days

Overall Lessons Learnt

- Datacenter fault-tolerance akin to human ailments/medicine today
  - Most common illnesses (crash failures) addressed
  - But uncommon cases can be horrible (unexpected outages)
- *Testing* is important
  - American Eagle, during a disaster, discovered that they could not fail over to backup DC
- Failed upgrades common cause of outage
  - Need a fallback plan

Overall Lessons Learnt (2)

- Data availability and recovery
  - BCP, Disaster-tolerance
  - Cross-DC replication, either by provider or by customer
- Consistent Documentation
  - A Google AppEngine outage prolonged because ops did not know which version of docs to use for recovery
  - Google's fix: mark old documents explicitly as "deprecated"
- Outages always a cascading series of failures
  - Need more ways to break the chain and prevent outages

Overall Lessons Learnt (3)

- Other sources of outages
  - DOS-resistance
  - Internet outages
    - Under-sea cable cut, DNS failures, Government blocking Internet (mostly via DNS)
    - Solution: Alternate DNS services
- Many failures are unexpected
- But there are also planned outages (e.g., kernel upgrades)
  - Need to be planned well
  - Steps documented and followed
  - Fallback plans in place

Announcements

- MP4 due this Sunday, demos Monday.
- Final exam window: Dec 11 7pm to Dec 13 10pm
  - Fully online, via Gradescope. Individual work, no collabs
  - 4 hour window after you start for a 3 hour final exam.
  - Must submit exam by 10 pm

- Next lecture: Wrap up!