BUILDING FAULT-TOLERANT APPLICATIONS ON AWS

Group 2:
David King, Robert Lowstetter, William Hackney
WHAT IS FAULT-TOLERANCE?

“The ability for a system to remain in operation even if some of the components used to build the system fail”

Netflix describes Fault Tolerance as a “Requirement, Not a Feature”

It does not matter what time of day/night it is. The web site or application is expected to be available 24/7!

- Unavailability will frustrate customers
- Lost customer satisfaction, loyalty.
- Loss of $$$
AWS is not unique to being ideally suited for building fault-tolerant applications.

You could do this on almost any platform but, it would consume a lot of time and resources.

• THIS is what makes AWS unique in building Fault-Tolerant applications!
• You can build these Fault-Tolerant applications with little interaction and minimal investment.

• Remember our discussion of cloud elasticity?
  • Elasticity being the ability of the system to adapt to change
  • You want your systems resources to match its current use as much as possible
AMI = Amazon Machine Image
A template that holds a software configuration
Operating System, Applications, etc.
Gets applied to an instance type

- Instance Type = What hardware is in the VM?
  - You choose how much RAM and CPU meets the needs of your application.
  - Remember elasticity!
  - This is what made AWS unique in building Fault-Tolerant applications!
HOW DO AMIS HELP US?

Say we have an application that we have built on AWS

Customers are live with the application!
- If this were to fail, our users would become quite frustrated and likely begin to dislike our application.

We mitigate this by using AMIs!
- Say our application has failed and we do not know why.
  - We need to get this back up and running before we lose consumer loyalty!
- We can easily replace the application! All we need to do is launch another instance with the SAME AMI!
How Does This Work?

There are many tools at your disposal that you could use for replacing the failing application.

- Command Line
- AWS Management Console
- Auto Scaling Service (discussed later)
- Etc.

We could have an instance already running as a backup in case our server were to fail (Amazon Block Stores; more details later 😊)!

- All we would do is use an Elastic IP Address (described later) point to the failing instance and redirect the old instance!

Always Have a Backup!
This was just the first step of Fault-Tolerance!

- As you can see, this is very useful for keeping our application constantly running and keeping our consumers loyal!

The ability to easily bring up a new instance when one fails is critical for recovering from a failure!

Let’s get a firm understanding of the importance of Fault-Tolerance through an example I believe everyone will understand,
HOW IMPORTANT IS FAULT-TOLERANCE?

Look to the diagram to the right. This is how Netflix is typically used.

We would have a plethora of users that would request to access multiple services at a time.

What could cause a fault? There are so many possible ways that a dependency could go down.
INTERRUPTED STREAMING

New movie or popular show?
- Higher volume and traffic
- Fault Tolerance is necessary

Netflix mitigates fault via Elastic Load Balancing

This is just one example of how they implemented fault tolerance in AWS!
ELASTIC BLOCK STORES (AMAZON EBS)

EBS = Offline storage; backups!
Perfect for recovering from a fault
Because the new instance is a copy of the original you would lose essentially no data and no functionality!
Annual fail rate for EBS is .1% and .5% rather than 4% in a standard data scenario!
• This is due to being stored redundantly (permits data to recover from errors through reconstruction i.e. RAID)!

Stepping away from Netflix…
SNAPSHOTS AND ELASTIC IPS

Store data on EBS via snapshots, a copy of your EC2 instance at current time.

Re-associate IPs from the faulted instance to a new working instance.

This can be done via the API or through the Management Console.
HOUSE OF CARDS

2% of the entire Netflix population, 670,000 users BINGE watched all 13 episodes of House of Cards on its first weekend.

This is a 400% increase in traffic at the time over its first season

Paving the way for Internet TV

Utilizing Elastic Load Balancing, Netflix mitigates this extreme traffic.
FAILURES CAN BE USEFUL
YOUR APPLICATION AVAILABILITY INTERRUPTED

Sophisticated software systems are dependent on a number of components that are out of its control

e.g. operating system, firmware, and hardware
MOST SOFTWARE DEGRADES

1) Leak memory and/or resources
e.g. application frameworks, operating systems & device drivers

2) File systems fragment over time

3) Hardware physically degrades over time
   ~ particularly storage
REGULARLY MAINTAINED AND SERVICED

Traditional IT environment hardware maintenance and servicing has practical and financial limits.

These limits can constrain how efficient and effective servicing Traditional IT environments.
FAILURE FORCING RESOURCE REFRESH

AWS platform can be refreshed periodically with new server instances which reduces potential system degradation.

AWS server instances themselves become immaterial and even disposable.

Set expiry dates to refresh instances regularly to ensure that any leaks or degradation will not impact the application.
AWS AUTO SCALING

Rules can be defined for scaling EC2 capacity for launching or terminating server instances

1) When the number of functioning server instances is above or below certain number

2) Using Amazon CloudWatch for monitoring certain threshold of server instance fleet resource utilization

Add more instances in response to an increasing load; automatically terminate extra instances when no longer needed.
i.e. Create an CloudWatch alarm to occur whenever the CPUUtilization metric for the EC2 instances exceeds 90%.

When the alarm occurs, Auto Scaling launches and configures another instance to join the application tier.

The instance takes a couple of minutes to launch. During that time, the CloudWatch alarm could continue to fire, resulting in Auto Scaling launch another instance each time the alarm goes off.

AUTO SCALING COOLDOWN

With a cooldown period in place, Auto Scaling launches an instance and then suspends any scaling activities until a specific amount of time elapses.

The newly-launched instance has time to start handling application traffic.

After the cooldown period expires, scaling actions resume for the Auto Scaling group.

ELASTIC LOAD BALANCING

Sends any request to a DNS host name then distributes incoming traffic for the application across a pool of several EC2 instances.

Detects unhealthy instances within the pool of EC2 instances and automatically reroutes traffic to healthy instances.
ELASTIC LOAD BALANCING
COMBINING AUTO SCALING & ELASTIC LOAD BALANCING

Elastic Load Balancing uses a single DNS name for addressing.

Auto Scaling ensures there is always an right number of healthy EC2 instance to accept requests in conjunction with Elastic Load Balancing, each instance will handle a fraction of the incoming traffic.

Redundancy Pattern N+1 where N resources are sufficient to anticipated load.
Simultaneously running an application distributed geographically at distant Amazon Web Services datacenters achieves greater fault tolerance.

If a single datacenter fails the application is protected by geographically distant datacenters.
AWS GEOGRAPHIC REGIONS

5 regions:
1) US East (Northern Virginia)
2) US West (Northern California)
3) EU (Ireland)
4) Asia Pacific (Singapore)
5) Asia Pacific (Japan)

Amazon S3 (Simple Storage Service) US Standard encompasses datacenters throughout the United States.
AVAILABILITY ZONES (AZS)

Within each Region are Availability Zones that are engineered to be insulated from failures in other AZs and provide inexpensive, low latency network connectivity to other AZs in the same region.

The Amazon EC2 service level agreement commitment is 99.95% availability for each Amazon EC2 region.
Achieve High Availability by deploying your application and independent copy of each application stack that spans across multiple Availability Zones creating a multi-site solution.

Elastic Load Balancing will detect healthy & unhealthy EC2 instances in the same AZ or in multiple AZs then no longer route traffic to unhealthy EC2 instances and route traffic to remaining healthy EC2 instances.
FAULT TOLERANT BUILDING BLOCKS

Amazon Simple Queue Service (SQS)
Amazon Simple Storage Service (S3)
Amazon SimpleDB
Amazon Relational Database Service (RDS)
AMAZON SIMPLE QUEUE SERVICE (SQS)

Distributed message queuing system

EC2 Instance Based

URL-based message queues
- Any server process that understands HTTP can access the queue
- ACL-based security system

Four-day message retention

Auto Scaling of EC2/SQS

Visibility Timeout after message pull

Figure 8 - Amazon SQS System Architecture
AMAZON SIMPLE STORAGE SERVICE (S3)

Permanent storage service
Highly durable and fault tolerant
Stores objects redundantly on multiple devices across multiple facilities in a region
URL-based access to storage for web service (similar to SQS)
Provides versioning
AMAZON SIMPLEDB

Attribute decorated data storage

Objects retrieved via attributes (metadata) assigned when object was created

Can be used with or in place of MySQL or MS-SQL

Redundant storage

Accessible via URL’s with various APIs (JavaScript, etc.)

Download Amazon’s ScratchPad to begin learning how to use it
AMAZON RELATIONAL DATABASE SERVICE

Provides relational database services, such as MySQL, PostgreSQL, MS-SQL, Oracle, etc.

Snapshots similar to EDB instances

Multi-AZ ready instances-synchronous replica of DB instance is maintained in a different AWS zone
CONCLUSION

• Redundancy with load balancing and routing is key to fault tolerance
• Automatic and manual scaling/management via a myriad of methods is available to developers
• Amazon provides all levels of cloud computing (IaaS, PaaS, SaaS)
• Geography matters!
ANY QUESTIONS?

SlideShow References


GitHub Repo