Amazon Web Services: Active/Active Server Configuration to Streamline System Failover

CS4991 Capstone Report, 2022

Rohith Jampani
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
rvj8uc@virginia.edu

Abstract
Capital One, a banking firm headquartered in McLean, VA, decided to optimize its cross-region server failover system by replacing its active-passive configured manual failover with an automated process to significantly reduce data loss, increase revenue, and protect its reputation. I upgraded their manual system to make operations faster and more automatic by creating an active-active server configuration that allows the failover process to maintain server uptime and reduce data loss with almost instantaneous cross-region database replication. By utilizing the mass computing power and various services of AWS (Amazon Web Services) I configured different AWS components such as DynamoDB, Route53, and ECS to interact with each other to create the failover system. For next steps, other Capital One teams will be downloading the Docker base image and attaching it to their projects so that any system will have a cross-regional automatic failover process in place.

1. Introduction

A common misconception among cloud customers is that cloud providers are unable to lose data. In 2019, AWS proved that misconception when one of their datacenters underwent a power loss causing the backup generators to fail, which resulted in physical servers going offline. The aftermath of the offline hardware included corrupted EC2 instances and EBS volumes which contained irretrievable customer data. Eventually, AWS restored most of the customer data, but a small percentage was still lost so this event showcased the commonly overlooked necessity for a backup system (Lee, 2019).

In fact, IT disasters such as the loss of a physical data center, server failures, or cyber-attacks are a common hinderance if a proper server failover system is not configured and cannot only disrupt the company's business, but also induce data loss, impact revenue, and damage its reputation (cStor, n.d.). To hedge against disaster scenarios, failover systems must be implemented where a backup server is configured to store the primary server's resources to maintain server uptime (cloudflare, n.d.). Minimal server downtime will result in high customer satisfaction and serve to reduce data loss.

2. Background

Prior to my arrival at Capital One, the team's failover system consisted of an active-passive configuration, where the primary server took all the DNS traffic when its resources were healthy while the secondary server waited on standby for a disaster scenario to affect the primary server. Once the scenario was detected, route 53 would map the secondary server's healthy resources to the new DNS traffic to

maintain server uptime. However, the active-passive configuration was incredibly slow because the failover program had to be manually started by a technician team once they received a message that the primary server did not satisfy the health check.

Owing to the system's specific configuration, rerouting the resources from the primary to the secondary server took up an RTO (Recovery Time Objective) of around an hour where RTO is defined as the acceptable time allowed for server down time before it is relaunched (Baginda, 2018). An hour of server downtime has serious implications ranging from detrimental damage to the company's reputation as well as revenue loss, especially if the servers contained transaction data.

3. Related Works

The AWS server failover problem and proposed disaster recovery solutions such as active-passive or active-active failover systems have existed since the release of AWS. However, how the failover system is implemented is what differentiates the specific configuration from other recovery solutions. Additionally, newer features have been added to the existing AWS services over the years which have contributed to more effective failover systems. For instance, IBM's Tivoli system for multi-platforms has an active-cold architecture which is a modified approach of the active-passive configuration. The Tivoli system provides high availability clustering, rapid server outage detection, and an immediate recovery of applications that fail during a disaster scenario (IBM, 2014).

Like the active-passive configuration my team used, the Tivoli system had no automation scripting but included other components which interacted with the applications. However, the application was required to start during failover which slowed down failover time. The main differences between the two systems are that the team's active-passive system ran the target application on both the active and passive servers while the active server took all the traffic, and the service did not have to be started during failover. This configuration results in more resources used by the passive server but less computational overhead since only one server is taking all the traffic (IBM, n.d.).

Similar to the active-active configuration I created during the internship, IBM's Oracle RAC is an active-active database cluster which distributes traffic between two or more active nodes. The servers share and update the same data to provide a common service for the applications hosted (IBM, n.d.). On the other hand, my active-active failover configuration was hosted on AWS to support the servers issued by Capital One while Oracle RAC can be figured on other cloud providers like soft Azure or Google Cloud.

4. System Design

The failover system required a unique re-design that provided more security, faster data retrieval, and less server downtime.

4.1 Review of System Architecture

I created a multi-site active-active configuration between the primary region (ex. us-east-1) and secondary region (ex. us-west-2), where both servers can utilize geolocated Route 53 records to distribute network traffic, database updates, and manage application requests through an elastic load balancer. Furthermore, the geolocation routing policy is crucial for active-active deployments because the policy determines what region receives the request based on the origin of the request. This strategy ensures that data can be stored

for certain consumers within a specific region.

To that effect, if us-east-1 goes offline, then us-west-2 already has traffic distributed to it. This strategy protects the data from server failure and with fast database replication, us-west-2 can maintain server uptime and minimize data redundancy (Izrailevsky, 2018). Moreover, the instant data replication offered by DynamoDB global tables allows the data stores across different regions to protect against common disasters such as data corruption or deletion because the data store backup can be restored to the last known healthy state of the database. Also, auto-scaling groups were setup for the application instances in each region which allowed for scalability based on how much traffic was sent to the region.

4.2 Platinum Customer Resiliency Requirements

The active-active automatic server failover system reached platinum customer resiliency, the highest tier of objectives that my team had planned to achieve. This platinum tier consisted of automating the entire failover process as well as reaching an RTO of 15 minutes instead of an hour server downtime with the previous active-passive failover configuration.

Also, the platinum tier required for minimal RPO (Recovery Point Objective), the maximum acceptable amount of time since the last data recovery point (Baginda, 2018). This metric essentially determined what is considered an acceptable loss of data. The platinum tier required that RTO and RPO were at their absolute lowest during an active-active configured failover, but the active-active system was expected to be high-cost and very complex as the configuration had to manage active stacks in multiple regions. Furthermore, my team and I developed the failover system using two test regions (us-east-1 and us-west-2) but we included capabilities for the system to accommodate more than two regions to reach the platinum tier.

4.3 Key AWS Components

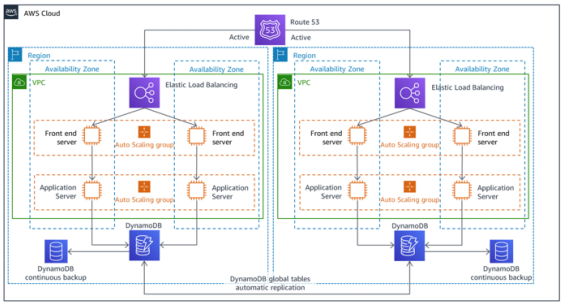

Figure 1: Diagram of AWS Failover System

The architecture in figure 1 displays how each AWS region was configured to host a highly available, multi-Availability Zone (AZ) workload stack. Specifically, as stated in the previous section, the system was built to test two regions which is why only two regions are shown in the diagram. The first key component of the system is traffic routing which was implemented using Route 53, AWS's reliable and scalable Domain Name System (DNS) and is shown at the top of figure 1. Essentially, Route 53 is used to determine which region receives a given request. During a failover situation, if the system determines that the workload stack in a region is unhealthy then the failover system will route traffic away to other healthy regions.

Also, the time to live (TTL) for the DNS records had to be low to ensure that the system met the platinum RTO standard of 15 minutes during failover. The second key component is shown at the bottom of figure 1 which is DynamoDB. This flexible NoSQL database service has a feature called global tables which can replicate a table to multiple regions simultaneously. The tables are also backed up because the system uses

DynamoDB continuous backup to ensure data is not lost during failover.

4.4 Challenges

The first challenge my team faced during system development was choosing the right routing policy since geolocation and latency routing had respective benefits in a multi-site active-active failover system. Latency routing allows Route 53 to automatically send requests to the region that provides the shortest time to receive and respond to the request. As a result, failover performance would be higher, but my team decided that the 15 minutes of RTO can still be reached with geolocation routing since that policy keeps data for certain users within the region, making it a much safer option.

The second challenge we faced was deciding between DynamoDB or Amazon Aurora as a database service. Like DynamoDB, Amazon Aurora has a feature called global database where a primary cluster would be deployed to a global write region and then read-only Aurora instances would be deployed to other regions. Requests travel through the AWS network instead of the internet which reduces latency. However, since we tested with a small data set of only two regions, DynamoDB's relational feature proved to be the better choice due to its less complex structure.

The final challenge we overcame was connecting the failover system on AWS with the docker base image. We did so with a Unix script that deployed both regions and deployed the failover process when a region was identified as unhealthy.

5. Results

By the end of the internship, the multi-site active-active failover system reached the platinum customer resiliency tier. In fact, the failover system allowed teams to automate their failover process in a deterministic way which provided data security and reduced the RTO by 75% to 15 minutes.

The system was well-received by the other Capital One teams because it provided relief from a long-standing problem of slow manual failover which most teams faced. Also, the failover system was an in-house project so it will only be used by Capital One teams and will not be outsourced to any other companies.

6. Conclusion

Capital One changed its manual active-passive server failover system to an automated active-active system in order to have less server downtime and better data security. By using multiple AWS services in tandem, my team and I were able to create a fast and reliable failover system which any team in the company can use by downloading and attaching the system's supporting docker image to their own project. Also, the system reached the platinum resiliency tier of objectives which means the failover process is as optimized as it can be because it ensures applications are always up and running for consumers and businesses.

7. Future Work

For next steps, Capital One will notify all the teams of this new automatic failover configuration that can be connected to their existing projects by downloading a docker image from the Capital One GitHub and attaching it to the project. On the technical side, the system cannot be improved in terms of performance because the failover configuration has already reached the platinum resiliency tier. However, teams can improve the system's security by creating specific security groups for the EC2 instances instead of using the default security groups given by AWS.

8. References

Y. P. Baginda, A. Affandi and I. Pratomo. Analysis of RTO and RPO of a service stored on Amazon Web Service (AWS) and Google Cloud Engine (GCE). Retrieved April 6, 2022 from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8534758&isnumber=8534734

Brandon.lee. 2021. Amazon AWS data loss shows cloud backups are crucial. (January 2021). Retrieved April 6, 2022 from https://www.virtualizationhowto.com/2019/09/amazon-aws-data-loss-shows-cloud-backups-are-crucial/

Using Amazon Web Services for Disaster Recovery - cStor. Retrieved April 6, 2022 from http://cstor.com/wp-content/uploads/2015/03/AWS_Disaster_Recovery.pdf

What is server failover? | failover meaning | cloudflare. Retrieved April 6, 2022 from https://www.cloudflare.com/learning/performance/what-is-server-failover/

IBM. Retrieved April 6, 2022 from https://www.ibm.com/common/ssi/cgi-bin/ssialias?infotype=an&subtype=ca&appname=gpateam&supplier=897&letternum=ENUS214-039

Y. Izrailevsky and C. Bell, "Cloud Reliability," in *IEEE Cloud Computing*, vol. 5, no. 3, pp. 39-44, May./Jun. 2018, doi: 10.1109/MCC.2018.032591615.https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8383675&isnumber=8383643