International Journal of Computer Trends and Technology
ISSN: 2231 – 2803 / https://doi.org/10.14445/22312803/IJCTT-V69I4P101

Volume 69 Issue 4, 1-5, April 2021

Original Article

Caching in Amazon Web Services

Vandana Premkumar[1] , Vinil Bhandari[2]

Technology Architect[1], Director of Technology[2]
New York Stock Exchange, New York City, NY, United States of America.

Received Date: 23 February 2021
Revised Date: 25 March 2021
Accepted Date: 06 April 2021

Abstract *- Amazon Web Services(AWS) provides various solutions for small, medium, and enterprise-level organizations. These solutions cover a wide range of problems ranging from hosting a website, computing a complex AI algorithm, storing terabytes of data. Depending on the volume of data and the rate at which it is requested, different caching solutions may be needed. AWS provides a wide range of caching solutions at different stages of architecture; this article will focus on what caching is and different caching strategies. We will conclude the article by providing some of the industry-standard best practices for using caching in your architecture.*

Keywords *- Performance Improvement, Increased IOPS, Reduced Cost, Low Latency, Redis.*

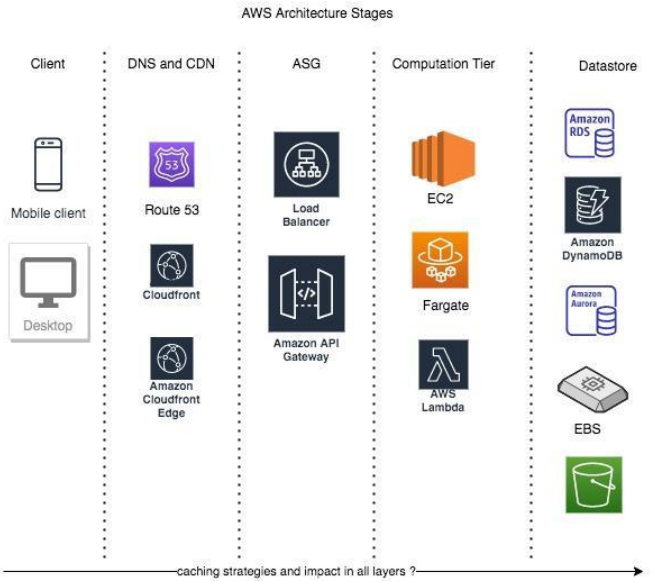
I. INTRODUCTION

Caching is a strategy used in software architecture as a means to store data in transient mode. In a client-server model, the server has computing services that typically take up most of the request time. Using caching techniques can elevate the time and cost savings by reducing the number of hits to backend computing services.

In order to go over different caching strategies, it is important that we look at different AWS services at each of the layers of architecture. We will then illustrate different caching used at these layers to improve the performance of the application.

II. AWS SERVICES OVERVIEW

AWS provides various services for a wide range of applications. Following is an illustration of these services in five-tier architecture.



A. Client

In a typical web, application clients can be a desktop browser or a wireless device like mobiles.

B. DNS and CDN

Domain Name Service (DNS), Content Delivery Network (CDN) is the first service that a client request would hit. Route 53 is AWS DNS service, and Cloudfront is AWS CDN service.

C. ASG

The request can then be optionally sent to a load balancing service for a highly scalable application. API Gateway helps in rate-limiting for an application programming interface. Auto scaling groups(ASG) enable the high availability of enterprise applications.

D. Computation Tier

Application computations are done in this layer. Some of the services are - Elastic Cloud Compute(EC2), serverless service like lambda, Container service like Elastic Compute Service, Fargate.

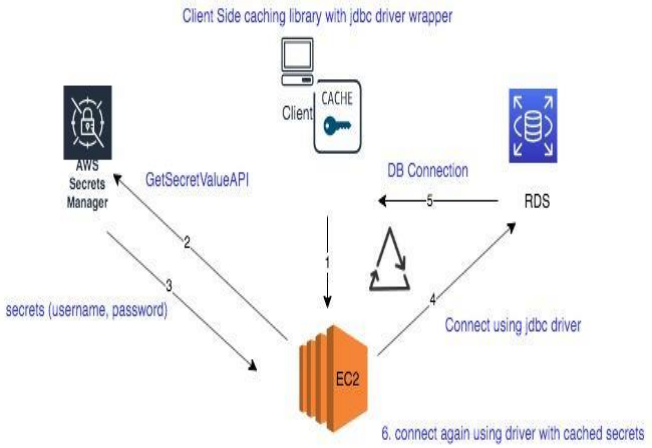
E. Datastore

AWS provides a wide range of storage services - relational databases like RDS, NoSQL databases like DynamoDB, Simple Storage Service (S3), Elastic Block Storage (EBS), and Instance Store.

You can take a look at detailed explanations of these services on AWS official website[1]

III. CLIENT LEVEL CACHING

If your application uses a secret manager to retrieve secrets such as API Keys and certification, it will be a good strategy to cache the secrets on the client. Each API call to Secret Manager will be charged a fee; by adding cache at the client level, the application will reduce the cost. Retrieving the secret from the cache will help improve performance and increase availability since the api call will not be impacted by any network-related issues. AWS provides open-source client-side caching for secret values. [2]



IV. ROUTE53 and CLOUDFRONT

DNS records are cached. By default, AWS does not have Time To Live(TTL) set. We must always specify TTL for all the records so that DNS records are cached for a specific length of time.

Cloudfront caching allows the requests to be served from edge locations that are closer to the client. This avoids multiple requests to the server hence reducing the latency and load on the server. Cloudfront Origin Shield can be used optionally as an additional layer to increase the cache hit ratio. Enabling the origin shield on AWS Region can help to improve network performance. Some of the real-world use cases where cloud front caching will be powerful

A. Borderfree

Borderfree is a leader in global e-commerce, which provides large retailers like Macys, J.Crew, Harrods with the technology that enables them to sell products to international customers.[3]

B. Brightcove

Provides just-in-time packaging for video live streaming service; caching at their CDN will help wider device reach and reduced latency.

C. Netflix

In some companies, multiple CDN's are needed to diversify their content based on the country. Netflix uses 4 CDNs [4] to increase the number of points of presence. Having caching for multiple CDNs will help limit the workloads on the server.

Multiple CDN Caching Serving Different Regions

V. LOAD BALANCING and API GATEWAY

AWS Load Balancers do not provide any caching services. API Gateway provides caching to help the performance of the Application Programming Interface (API).

APIs can help in your B2B strategies by providing and receiving different sources of data and services. APIs help Software as a Service(SaaS) companies to partner with much more traditional B2B companies.

Caching can be enabled at API Gateway by specifying its size. We can control the way the cache key is built and the values for TTL. If TTL is 0, caching is disabled. By default TTL value for API caching is 300 seconds. We can set TTL for up to 3600 seconds. Note that API caching is in the U.S.Health Insurance Portability and Accountability Act of 1996 (HIPAA) Eligible Service. The caching can be used for storing and transferring any Protected Health Information (PHI)[5]. API Gateway caching is charged at an hourly rate depending on cache size. In the case of WebSocket APIs, it is charged based on the number of requests sent and connection time.

Pricing Calculation Example with Caching Required. The following table gives the price range for the GB requirement of cache.

Table 1[6]

Cache Memory Size (GB)	Price Per Hour
0.5	$0.02
1.6	$0.038
6.1	$0.20
13.5	$0.25
28.4	$0.50
58.2	$1.00
118.0	$1.90
237.0	$3.80

If your API needs 5 GB of cache for its data, you can provision a 6.1 GB cache at $0.20/hr.[7]

$0.20 * 24 = $4.8/day

VI. COMPUTATION TIER

In this section, we will look at caching at the 3 most widely used computing services of AWS – Elastic Cloud Compute (EC2), Lambda, Elastic Container Service(ECS)

A. Elastic Cloud Compute (EC2)

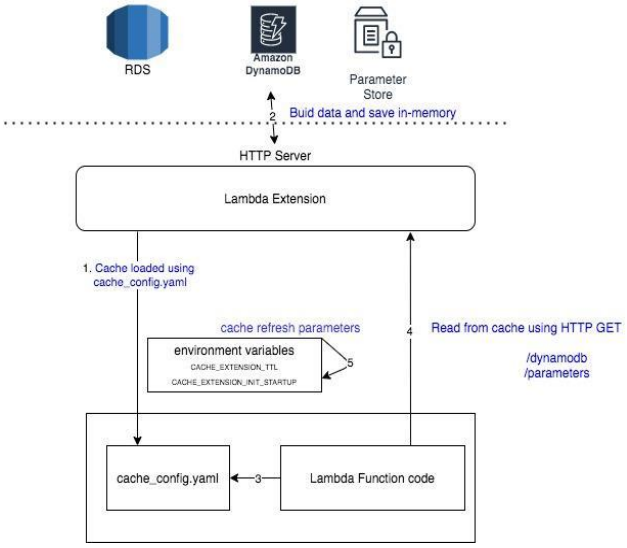
There are two kinds of storage in EC2 – Instance Store and Elastic Block storage. Instance store(ephemeral) is similar to RAM and is ideal for temporary storage. EBS is like a hard disk and is storing data in a more permanent fashion, which is not lost on instance reboot. EBS is a network-attached drive and is slower in performance when compared to instance stores. This is the reason why it is better to use an instance store for caching at the application level. The ephemeral datastore is used for storing temporary analytics calculation data such as Hadoop jobs [9].

B. Serverless Lambda

AWS Lambda Extensions was introduced on October 8 2020. [8] It is a wrapper that runs in the execution environment of lambda. Lambda extension has access to everything that lambda's function has – code, environment variables, /tmp folder of lambda. Lambda extensions provide multiple uses, one of which is caching.

An often-used use case for lambda is to connect to a database and serve data to API Gateway. For this lambda function, I would need to login and get the configuration details to access data from the database (RDS , Aurora, or DynamoDB). In order to do so, it will need to connect to the SSM Parameter store and load the parameters. [10]

Lambda extensions enable to cache of this information. Caching this information at initialization time will improve lambda function performance tremendously. Cache data is stored in memory and not in a file. Hence, there is no need for additional processes to maintain the lifecycle of the file. It is more secure to cache data in memory as data is not persisted on disk. Lambda function will need less code as it will communicate with lambda extension using HTTP to retrieve data. When using the lambda extension, the function will not need additional code to access the database, secrets manager, or local file system.



Cache extension is a Golang compiled binary, which can have a wrapper written in NodeJs, Python, Java, etc. YAML template is used to specify what data needs to be cached. Following cloudwatch[10] logs show the performance improvement made when using lambda extension. Cold start performance improved by 62% and following requests by 80%.[10]

C. Elastic Container Service (ECS)

In 2018 AWS provided a way to configure ECS to use a cached image to start the container instead of downloading from the registry. Previously, ECS needed to download large container images, which sometimes are stored outside of AWS every time a container is started. This caused performance to deteriorate.[11] Now, you can specify the following parameter in the *docker run* command to start the container from the cached image [12].

ECS_IMAGE_PULL_BEHAVIOR
Example values: *default* | *always* | *once* | *prefer-cached*
Default value: *default*

If *prefer-cached* is mentioned, the image is used from the cache. If there is no image in the cache, the image remotely pulled once. Note that automated image cleanup is disabled for the container so that cached images are not removed[12].

VII. DATASTORE

Disk-based databases can pose a lot of latency and scalability issues. For example, if your application is invoking queries that take a lot of time to execute, it can slow your requests tremendously. The response time to the client will include query processing time, query run time, and data retrieval speed. Scaling a database to have multiple replicas can add to the cost of the project when compared to in-memory cache[13]. Although relational databases provide great data model relationships, oftentimes, it is hard to retrieve the data view that the application needs, resulting in reduced performance.

Database cache can help in improving the performance of the data retrieval from the database. Three types of database caches are

A. Database Integrated Cache

Databases such as Aurora provide cache that is managed within the database engine. Cache updates based on the data updates in the database. The application tier does not control how the cache is updated. The cons of using database integrated cache are in regards to size and capabilities. It's allocated with limited memory and cannot be shared with multiple instances[13].

B. Local Cache

Local cache stores the data that is most frequently used by the application. This limits the database hits and hence saves network traffic, and time it takes to load data. The disadvantage is that this cache cannot be shared if your application is deployed in multiple EC2 instances under an ALB. For example, on an eCommerce website, it becomes convoluted not to share web session cart data. This challenge can be mitigated by adopting a remote cache. The remote cache is a separate instance that is used for caching at the application tier by multiple instances.

C. Remote Caches

Redis and Memcached are used for remote cache. They are typically key/value NoSQL stores. They provide up to million requests per second per cache node.

AWS Elasticache for Redis provides high availability, which is a must-have feature for critical applications. Remote caches are ideal for distributed applications. There are many benefits of using Elasticache

- Elasticache provides a sub-millisecond latency to real-time power applications.
- Elasticache provides an ability to create user and user group-based access by setting up Role-Based Access Control (RBAC)[14].
- Elasticache supports Amazon's VPC, so you can isolate ip addresses for a more secure cache of data.
- It offers encryption in transit, at rest using AWS or customer-managed KMS. This allows Elasticache to be PCI compliant, HIPAA eligible, FedRAMP authorized.
- Elasticache is easily scalable. We can scale the redis cluster up to 500 nodes and 500 shards. It allows the addition of up to five read replicas across multiple availability zones[14].

VIII. CONCLUSION

Caching can be applied for a wide variety of use cases. We need to take precautions on what, where, and how the data of our application can be cached. It's important to check which data is cached at what point; for example, in a typical eCommerce website, product prices on the checkout page should not be cached since it needs to be authoritative, whereas prices in other pages can be cached since we can allow a minor difference in price when a shopper in viewing a product. It is also important to check that caching certain data is effective. For example, it is not worth caching data which is constantly changing. That being said, it might seem that only your most frequently accessed data is cached or data of an expensive operation is cached, however in practice, the in-memory cache is widely used and will improve your application performance immensely[15].

APPENDIX A

There are various ways of managing sessions. The distributed cache can be used for managing sessions. Sticky session (session affinity) enables load balancers to tie user sessions to a specific target. This feature helps to maintain stateful information to provide a good continuous experience for customers. Drawback is there may be a single point of failure due to this approach for certain users, when there is an outage in their target. During the scale-up scenario, traffic may be unevenly distributed[15].

APPENDIX B

There are different caching design patterns [15]

- Lazy caching
- Write-through time-to-live
- Evictions
- The thundering herd

REFERENCES

[1] Explore Our Solutions., Amazon Web Services, March (2021), http://www.amazon.com.

[2] Using the AWS-developed Open Source Client-side Caching Components., Amazon Web Services, March (2021), http://www.amazon.com.

[3] About Us., BorderFree, March (2021), http://www.borderfree.com

[4] Why Use Multi-CDN instead of single CDN., Globaldots, March 2021, https://www.globaldots.com/solutions/multi-cdn/

[5] Enabling API caching to enhance responsiveness, Amazon Web Services, March (2021), https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-caching.html

[6] Amazon API Gateway Pricing: A Comprehensive Guide, Jan (2021), learnaws, https://www.learnaws.org/2021/01/10/api-gateway-pricing/

[7] Amazon API Gateway pricing, March (2021), https://aws.amazon.com/api-gateway/pricing/

[8] Using AWS Lambda Extensions with Python, Dilex, Amazon Web Services, March (2021) https://www.dilex.net/data-blog/using-aws-lambda-extensions-with-python

[9] Backup AWS Ephemeral Storage of an EC2 Instance, n2ws, March (2021), https://n2ws.com/blog/aws-ec2-backup/ephemeral-storage-of-ec2-instance-part-1.

[10] Caching data and configuration settings with AWS Lambda extensions, Amazon Web Services, (2021) https://aws.amazon.com/blogs/compute/caching-data-and-configuration-settings-with-aws-lambda-extensions/

[11] Amazon ECS Adds Options to Speed Up Container Launch Times, Amazon Web Services, March 2021, https://aws.amazon.com/about-aws/whats-new/2018/05/amazon-ecs-adds-options-to-speed-up-container-launch-times/.

[12] Amazon ECS Container Agent Configuration, Amazon Web Services, (2021), https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-agent-config.html

[13] Database Caching, Amazon Web Services, March 2021 https://aws.amazon.com/caching/database-caching/.

[14] Amazon ElastiCache for Redis, Amazon Web Services, March 2021, https://aws.amazon.com/elasticache/redis/.

[15] Caching Best Practices, Amazon Web Services, March 2021, https://aws.amazon.com/caching/best-practices/.