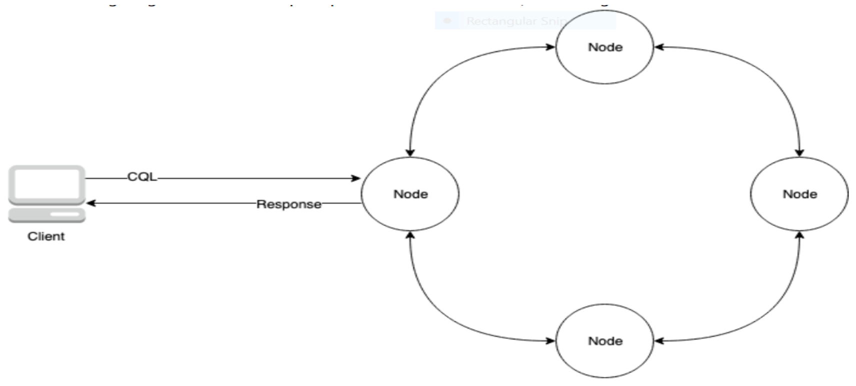
Amazon Keyspaces

Abhinav Angirekula

April 2021

1 Introduction

Like most of AWS's services, Amazon Keyspaces(for Apache Cassandra) is highly available, serverless, and scalable. More specifically, it is a managed database service that is compatible with Apache Cassandra. The purpose of the service is to allow developers to easily migrate, run, and scale Cassandra workloads in the Cloud without the need to deploy any infrastructure or install any software. Understanding Amazon Keyspaces requires understanding Apache Cassandra; it is a NoSQL database management system.



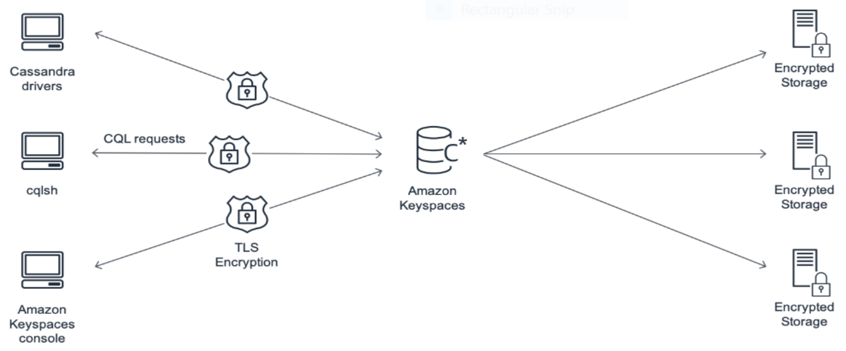
2 Apache Cassandra Basic Architecture

Cassandra is a system capable of handling massive amounts of data. In essense, the architecture is just a cluster of nodes that can accept read or write requests. All nodes can do this; there is no master node, as each node communicates equally.

Each node is the location where specific data resides within the cluster. The cluster itself is the complete set of data centers where all the data is stored, waiting to be processed. This architecture allows for readily available scalability: closely related nodes can be grouped together, nodes can be added, etc.

3 Basic Overview

Amazon Keyspaces is often preferred to vanilla Apache Cassandra; this is primarily due to the service's removal of administrative overhead, and this is primarily due to the differences in architecture existing between Amazon Keyspaces and Cassandra. Essentially, in vanilla Apache Cassandra, the developer deploys into a multi-node cluster. From there, the developer is entirely responsible for the management of these nodes. In production, a developer might be working with hundreds of nodes, each on a different physical computer, dealing with multiple different data centers. Managing all of this is no easy task.



Thus, Amazon Keyspaces endeavors to help lighten this burden developers face when working with traditional Apache Cassandra.

The service offers two distinct modes for reads and writes, the difference between the two modes being table throughput capacity:

1. On-demand Mode - paid for on demand. Developers are only required to pay for the reads and writes actually performed. Amazon Keyspaces will automatically adapt to the traffic of your application and there is no need to specify the table's throughput capacity beforehand, and as such, this mode is ideal for when the developer is dealing with a degree of unpredictability.

2. Provisioned Capacity Mode - However, if a developer has some grasp on how much traffic to expect, if there is a degree of predictability expected for the application's traffic, this mode is ideal. Here, you specify the table's throughput capacity and the number of reads and writes beforehand.

Keyspaces stores multiple copies of your data(three) in multiple availability zones. This is for both availability and security purposes.

3.1 Amazon Keyspaces Uses

While the main benefit of using Amazon Keyspaces over vanilla Apache Cassandra is simply due to the convenience of having your Cassandra workloads in the cloud, as in doing so, you relinquish the need to go through the tedious tasks of managing everything, there are other benefits to using Amazon Keyspaces over standard Apache Cassandra. One such benefit is the ability to use open-source technologies to build your applications; AWS provides a plethora of open-source Cassandra APIs and drivers(each compatible with a multitude of programming languages), allowing the developer to choose what works best for them.

4 Building Blocks of Amazon Keyspaces

4.1 Keyspaces

In Apache Cassandra, and by extension in Amazon Keyspaces, a single keyspace object is top-level database object that ”controls the replication for the object it contains at each datacenter in the cluster”. You can modify the replication factor of a keyspace, which changes the amount of replications of keyspace data stored within a cluster. With Amazon Keyspaces, developers are able to create and delete keyspaces, which are both types of data definition language(DDL) operations. This is done asynchronously.

4.2 Tables

In CQL, which is the language used by both Apache Cassandra and Amazon Keyspaces, data is stored in tables. Just like with keyspaces, Amazon Keyspaces allows developers to, asynchronously, perform DDL operations such as deleting and creating tables.

5 CRUD Operations

When working with tables in Amazon Keyspaces, developers are able to not only define tables(create and delete tables) through DDL operations, but they are also able to perform operations to modify tables. These operations are called CRUD operations:

1. Create - Inserting data into a table
2. Read - Reading data from a table
3. Update - Modify data in a table
4. Delete - Delete data from a table

6 Comparision

Currently, in terms of cloud-based Apache Cassandra compatible database services, Amazon Keyspaces is the popular one. In terms of functionality, there exists practically no differences, as they all provide the same general set of capabilities as they are all based off Apache Cassandra. Developers should choose which cloud provider to use based off their general familiarity with said provider, rather than based off differences in features between the services, as practically none exist. Microsoft provides their own alternative to Amazon Keyspaces, called Azure Managed Instance for Apache Cassandra; if you are more familiar with Azure than AWS, then by all means, use this service instead.

7 Demo

https://www.youtube.com/watch?v=zehVQzlSuEU

8 Works Cited

[1]https://docs.aws.amazon.com/keyspaces/
[2]https://www.bmc.com/blogs/apache-cassandra-introduction
[3]https://aws.amazon.com/keyspaces/
[4]https://docs.datastax.com/