Cassandra: Introduction

Let’s explore Cassandra and its use cases.

We'll cover the following

Goal#

Design a distributed and scalable system that can store a huge amount of structured data, which is indexed by a row key where each row can have an unbounded number of columns.

Background#

Cassandra is an open-source Apache project. It was originally developed at Facebook in 2007 for their inbox search feature. The Apache Cassandra architecture is designed to provide **scalability**, **availability**, and **reliability** to store large amounts of data. Cassandra combines the distributed nature of **Amazon’s Dynamo** which is a key-value store and the data model of **Google’s BigTable** which is a column-based data store. With Cassandra’s **decentralized architecture**, there is **no single point of failure** in a cluster, and its performance can scale linearly with the addition of nodes.

What is Cassandra?#

Cassandra is a **distributed**, **decentralized**, **scalable**, and **highly available** NoSQL database. In terms of CAP theorem (https://www.educative.io/collection/page/5668639101419520/5559029852536832/5998984290631680), Cassandra is typically classified as an AP (*i.e., available and partition tolerant*) system which means that availability and partition tolerance are generally considered more important than the consistency. Cassandra can be tuned with replication-factor and consistency levels to meet strong consistency requirements, but this comes with a performance cost. In other words, data can be highly available with low consistency guarantees, or it can be highly consistent with lower availability. Cassandra uses **peer-to-peer architecture**, with each node connected to all other nodes. Each Cassandra node performs all database operations and can serve client requests without the need for any leader node.

> **Disclaimer:** All of the following lessons are Cassandra version agnostic and try to explore the general design and architectural layout of different Cassandra components and operations.

Cassandra use cases#

By default, Cassandra is not a strongly consistent database (it is eventually consistent), hence, any application where consistency is not a concern can utilize Cassandra. Though Cassandra can support strong consistency, it comes with a performance impact. Cassandra is **optimized for high throughput** and **faster writes**, and can be used for collecting big data for performing real-time analysis. Here are some of its top use cases:

- **Storing key-value data with high availability** - Reddit and Digg use Cassandra as a persistent store for their data. Cassandra's ability to scale linearly without any downtime makes it very suitable for their growth needs.
- **Time series data model** - Due to its data model and log-structured storage engine, Cassandra benefits from high-performing write operations. This also makes Cassandra well suited for storing and analyzing sequentially captured metrics (i.e., measurements from sensors, application logs, etc.). Such usages take advantage of the fact that columns in a row are determined by the application, not a predefined schema. Each row in a table can contain a different number of columns, and there is no requirement for the column names to match.
- **Write-heavy applications** - Cassandra is especially suited for write-intensive applications such as time-series streaming services, sensor logs, and Internet of Things (IoT) applications.

← Back

Mock Interview: Dynamo

Next →

High-level Architecture

Mark as Completed

Report an Issue