Cluster, Grid, Cloud Computing systems | Course by Andrey Shevel

Cassandra

A FREE AND OPEN-SOURCE DISTRIBUTED NOSQL DATABASE

Carl Kugblenu | ITMO University, St. Petersburg | 6th June 2017

Table of Contents

1. Introduction

Traditional relational databases (RDBMSs) were the primary data stores for business applications for 20 years. Then, as the first phase of the Web got under way, new databases (such as Oracle's MySQL) were introduced that had RDBMS roots, but different feature sets to handle the new data access patterns. Today, another change is required because applications must now scale to levels that were unimaginable just a few years ago.

1.1 NoSQL Databases

NoSQL database provides a mechanism for storage and retrieval of data which is modeled in means other than the tabular relations used in relational databases. Unlike traditional RDBMS's, NoSQL databases often trade how ACID (atomic, consistent, isolated, and durable) transactions are for improved performance. NoSQL databases are more scalable and provide excellent performance when compare to relational databases, and their data model resolves plenty of issues that the relational model isn't developed to deal with

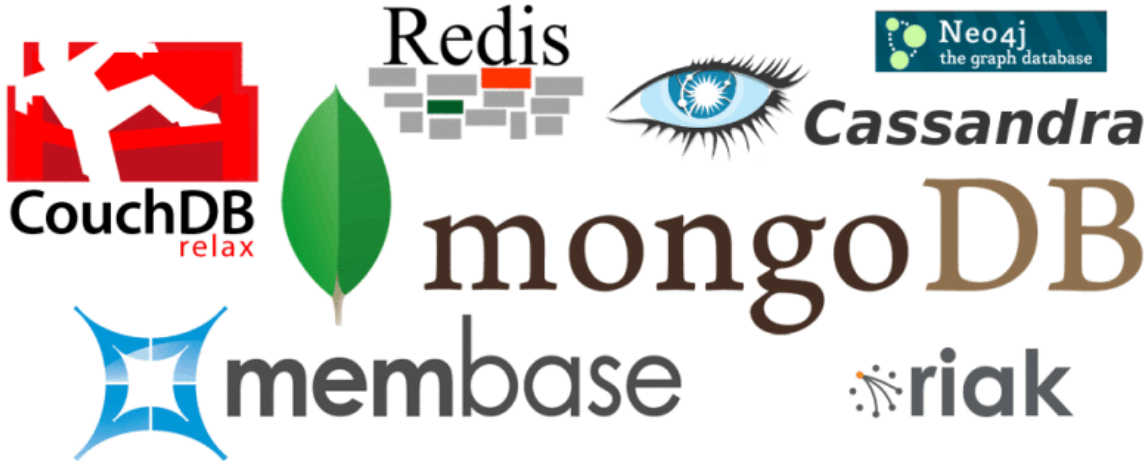


Figure 1. NoSQL databases

1.2 Cassandra

Apache Cassandra is a massively scalable NoSQL database. Cassandra's technical roots can be found at companies recognized for their ability to effectively manage big data – Google, Amazon, and Facebook – with Facebook open sourcing Cassandra to the Apache Foundation in 2009.

Used today by numerous modern businesses to manage their critical data infrastructure, Cassandra is known for being the solution technical professionals turn to when they need a NoSQL database that supplies high performance at massive scale, which never goes down. In particular, Cassandra addresses big data applications, which are exploding across nearly every industry.

Figure 2. Companies Using Apache Cassandra

2. Features

Cassandra has become so popular because of its outstanding technical features.

- **Elastic scalability**: Cassandra is highly scalable; it allows to add more hardware to accommodate more customers and more data as per requirement.

- **Always on architecture**: Cassandra has no single point of failure and it is continuously available for business-critical applications that cannot afford a failure

- **Performant**: Cassandra supports very high throughput. In particular the Cassandra write path allows for high performance, robust writes.

- **Flexible data storage:** Cassandra accommodates all possible data formats including: structured, semi-structured, and unstructured. It can dynamically accommodate changes to your data structures according to your need.

3. Architecture

The design goal of Cassandra is to handle big data workloads across multiple nodes without any single point of failure. Cassandra has peer-to-peer distributed system across its nodes, and data is distributed among all the nodes in a cluster.

- All the nodes in a cluster play the same role. Each node is independent and at the same time interconnected to other nodes.
- Each node in a cluster can accept read and write requests, regardless of where the data is actually located in the cluster.
- When a node goes down, read/write requests can be served from other nodes in the network.

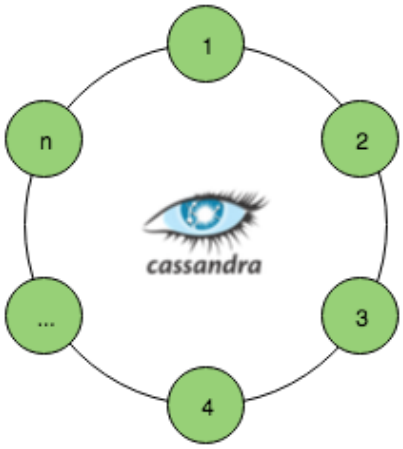


Figure 3. Cassandra Cluster Architecture

Cassandra uses Gossip communication protocol in which nodes periodically exchange state information about themselves and other nodes they know about. During a gossip exchange, older information is overwritten with the most current state for a particular node.

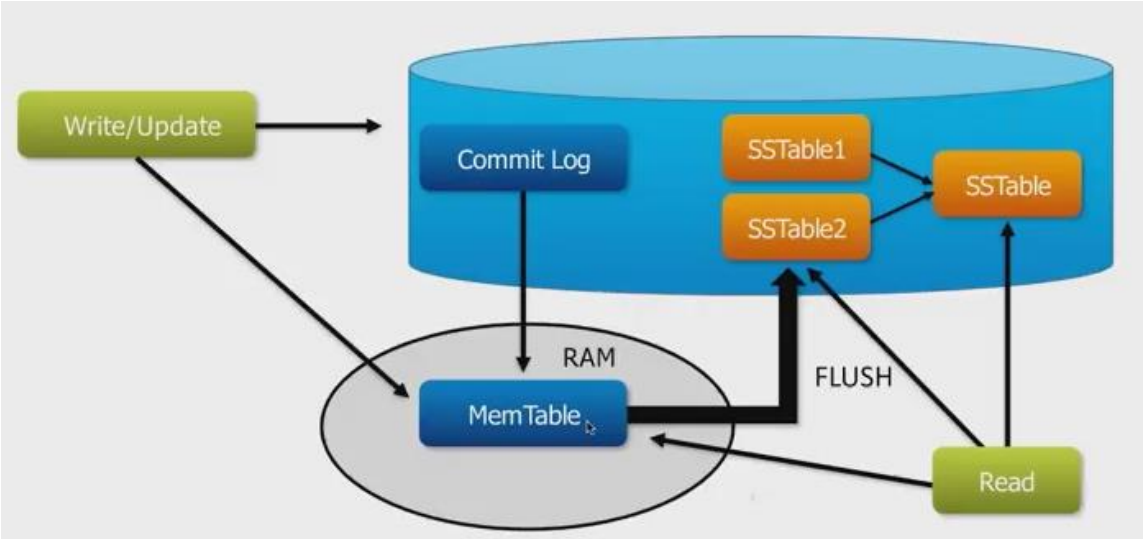
The key components of Cassandra are as follows:

- Node: It is the place where data is stored.
- Data center: It is a collection of related nodes.
- Cluster: A cluster is a component that contains one or more data centers.
- Commit log: The commit log is a crash-recovery mechanism in Cassandra. Every write operation is written to the commit log.

- Mem-table: A mem-table is a memory-resident data structure. After commit log, the data will be written to the mem-table. Sometimes, for a single-column family, there will be multiple mem-tables.

- SSTable: It is a disk file to which the data is flushed from the mem-table when its contents reach a threshold value.

- Bloom filter: These are nothing but quick, nondeterministic, algorithms for testing whether an element is a member of a set. It is a special kind of cache. Bloom filters are accessed after every query.



- *Figure 3. Cassandra Node Architecture*

4. Data Model

The data model of Cassandra which is based on a key value store is significantly different from what we normally see in an RDBMS. It introduces concepts of clusters, keyspaces and column families.

4.1 Cluster

Cassandra database is distributed over several machines that operate together. The outermost container is known as the Cluster. For failure handling, every node contains a replica, and in case of a failure, the replica takes charge. Cassandra arranges the nodes in a cluster, in a ring format, and assigns data to them.

4.2 Keyspace

Keyspace is the outermost container for data in Cassandra. The basic attributes of a Keyspace in Cassandra are:

- Replication factor: It is the number of machines in the cluster that will receive copies of the same data.

- Replica placement strategy: It is nothing but the strategy to place replicas in the ring. We have strategies such as simple strategy (rackaware strategy), old network topology strategy (rack-aware strategy), and network topology strategy (datacenter-shared strategy).

4.3 Column Family

Keyspace is a container for a list of one or more column families. A column family, in turn, is a container of a collection of rows. Each row contains ordered columns. Column families represent the structure of your data. Each keyspace has at least one and often many column families.

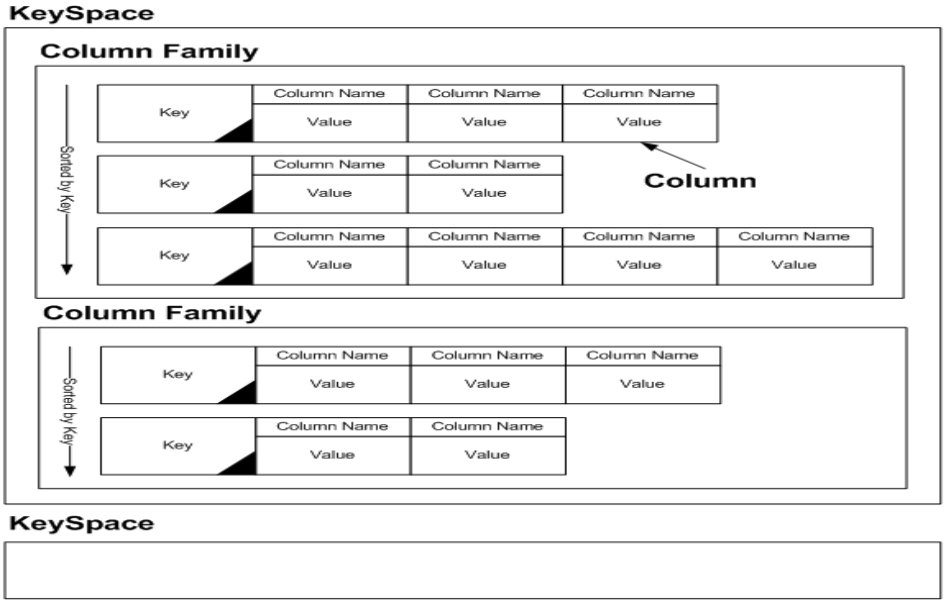


Figure 4. Cassandra Data Model

5. Advantages and Drawbacks

There are many different pros and cons to using Cassandra, many of which depend on the use case.

5.1 Advantages

- **Write Speed:** It is able to handle such a large volume of writes by first writing to an in-memory data structure, then to an append-only log. These data-structures are then "flushed" to a more permanent and read-optimized file at a later time.

- **Multi-DC Replication**: Out of the box, Cassandra comes with multi data center replication. This replication will copy the information to any number of instances of the Cassandra process. Additionally, you can create multiple "hot" data centers that also get a copy of the data. These can be used for geographical performance or for disaster recovery or both. The multi-datacenter setup is as simple as changing a single line in a configuration file and updating your schema. Multi-DC Replication is one of the top reasons people choose to use Cassandra.

- **Tunable Consistency**: When it comes to replicated data, you have to be able to decide what happens when an outage occurs in of one, or more, of your nodes. Cassandra allows you, on a query-by-query basis, to decide how to handle potential issues. There are many different ways to read/write your information. Cassandra allows you to choose the one that best fits your use-case and values.

5.2 Drawbacks

- **No Ad-Hoc Queries:** Beneath the covers, the Cassandra data storage layer is basically a key-value storage system. This means that you must "model" your data around the queries you want to surface, rather than around the structure of the data itself. This can lead to storing the data multiple times in different ways to be able to satisfy the requirements of your application.

- **No Aggregations:** Newer versions of Cassandra will have limited support for aggregations with a single partition. This is of very limited use. Because Cassandra is a key-value store, doing things like SUM, MIN, MAX, AVG and other aggregations are incredibly resource intensive if even possible to accomplish. If doing ad-hoc analysis is a requirement for your application then Cassandra may not be for you.

- **Unpredictable Performance:** Because Cassandra has many different asynchronous jobs and background tasks that are not scheduled by the user, the performance can be unpredictable. This means that you may see performance impacts that may not be related to a query, or volume of queries. This can make troubleshooting performance issues rather difficult.

- **JVM Based:** The Java Virtual Machine(JVM), while fast, is still a garbage collected language. This means that memory management is done by the language itself, not the application. If you get into very high volumes of information, be it in request volume or data size, the need to tune the JVM to fit the specific needs of

your implementation will be high. This means required expertise and knowledge of the language the database was written in.

6. Use Cases

The general ideal use case of Cassandra is when dealing with immutable data where updates and deletions are the exception rather than the rule. Below is a list of specific use cases where Cassandra has played a dominant role.

- Real-time, Big data workloads
- Time series data management
- High-velocity device data consumption and analysis
- Media streaming management (e.g., music, movies)
- Social media (i.e., unstructured data) input and analysis
- Online web retail (e.g., shopping carts, user transactions)
- Real-time data analytics
- Online gaming (e.g., real-time messaging)
- Software as a Service (SaaS) applications that utilize web services
- Online portals (e.g., healthcare provider/patient interactions)
- Most write-intensive systems

7. Additional Links

http://cassandra.apache.org/

https://www.datastax.com/2014/06/what-are-people-using-cassandra-for

https://opencredo.com/fulfilling-promise-apache-cassandra/

https://en.wikipedia.org/wiki/NoSQL

https://en.wikipedia.org/wiki/CAP_theorem

https://www.tutorialspoint.com/cassandra/index.htm

https://en.wikipedia.org/wiki/Apache_Cassandra