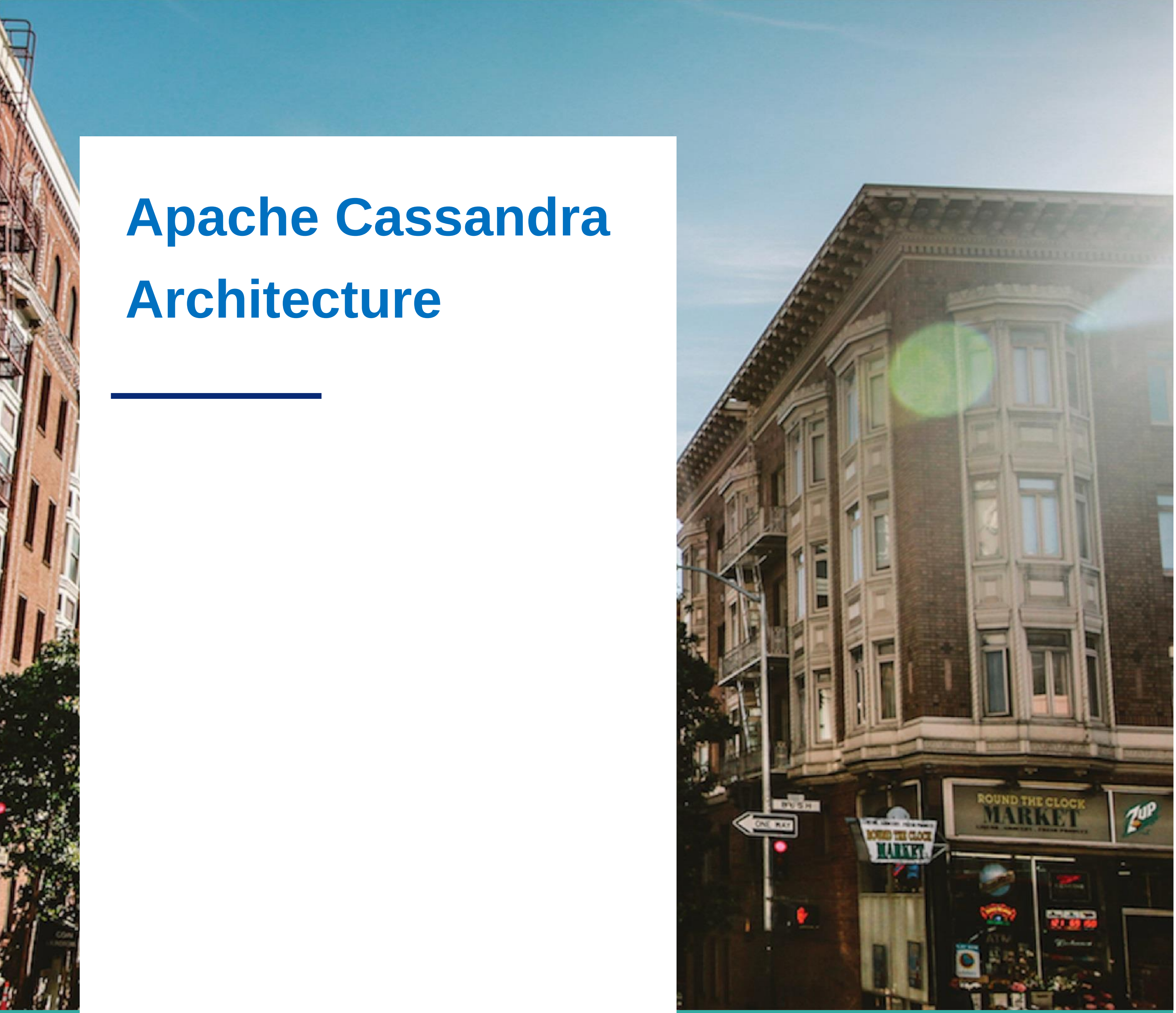
Apache Cassandra Architecture



In this pdf file I, Mohamed Metwalli, discuss the motivations for these changes and provide an architectural, and technical overview of DataStax Distribution of Apache Cassandra, and summarizes the paper found in this link :
https://www.datastax.com/resources/whitepaper/apache-cassandratm-architectur

RDBMS have proven difficult to scale to meet the challenges of modern large-scale applications such as capacity, availability, performance, and strong consistency. Here the role of NoSQL Databases appeared especially Cassandra that was developed by Facebook and open sourced in 2008.

The design of Cassandra was influenced by both "Amazon's Dynamo", and "Google's Bigtable".

Cassandra was designed from the ground up for efficient distributed operation across multiple data centers and elastic scaling with exceptionally high performance and reliability.

Cassandra excels in the following areas:

1) Large-scale storage

2) Ease of management

3) Continuous availability

4) Write-intensive applications

5) Statistics and analytics

6) Geographic distribution

Cassandra has few prerequisites as It runs on almost any hardware or cloud platform.

Cassandra uses two internal protocols to manage data placement based on cluster topology: gossip and snitches.

Virtual nodes concept is common in Cassandra where each Cassandra node supports multiple token ranges distributed throughout the token ring after hashing. The generation and assignment of token ranges are handled automatically.

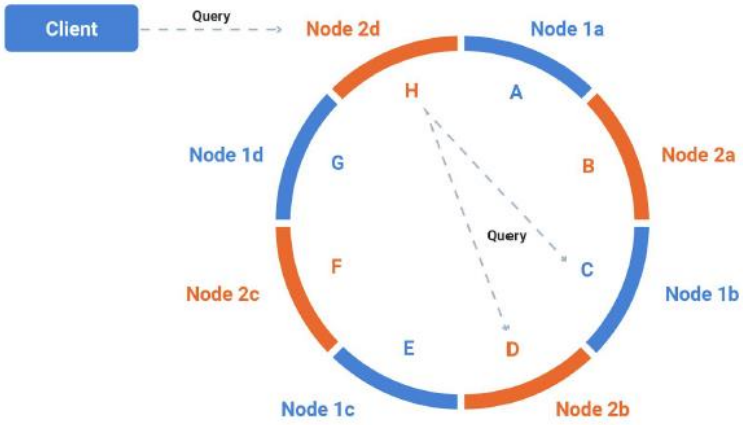


FIGURE 3 Clients connect to a coordinator node that passes queries to replicas

Users can set preferences related to data consistency using the *CONSISTENCY* command in CQL.

The Quorum Calculation Formula :

(sum of replication factors / 2) + 1 and rounding the result up to the nearest integer. So in a two-data center cluster,

where each data center has a replication factor of three (meaning six replicas) a quorum would be four nodes, meaning that two nodes could be down and the query could still succeed ([6 / 2] + 1 = 4).

When data is written to a node, it is first stored to the commit log so that the write can be recovered if the node fails.

Cassandra can complete a write as soon as data is logged to the commit log while other operations happen asynchronously. Also, performance scales directly with the number of nodes in the cluster. These make it impressive.

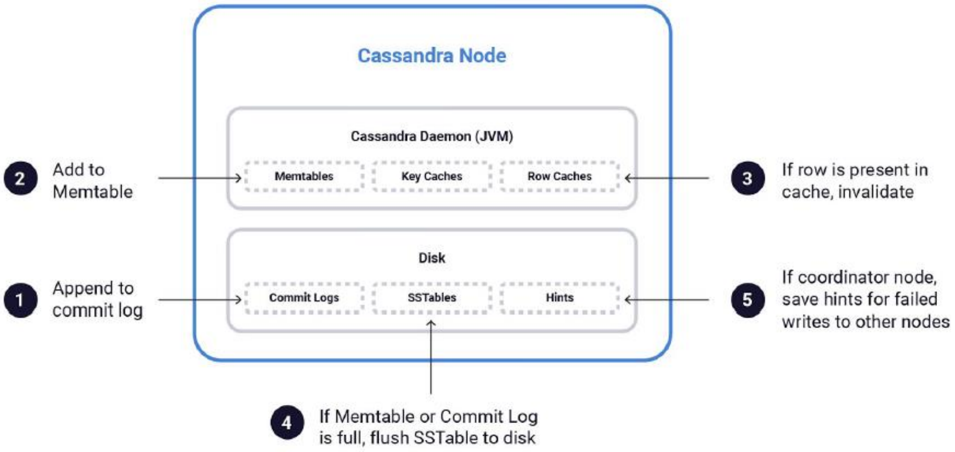

Cassandra Node
Cassandra Daemon (JVM)
Memtables
Key Caches
Row Caches
Disk
Commit Logs
SSTables
Hints
2 Add to Memtable
1 Append to commit log
3 If row is present in cache, invalidate
5 If coordinator node, save hints for failed writes to other nodes
4 If Memtable or Commit Log is full, flush SSTable to disk

FIGURE 5 The anatomy of a write operation in Cassandra

Cassandra behaves and stores data differently than a relational database. Cassandra stores data as a map of key-value pairs not in traditional rows and columns.

Not every column needs to be represented in each row. If a column value isn’t provided, Cassandra simply doesn’t store the key-value pair for that record (don't store null values).

If there is no consensus on a result when reading (consistency refers to the number of replicas that need to agree before a result is considered valid), Cassandra will internally run a "read repair" operation, forcing Cassandra to update pending changes lingering on replicas before returning a result to the client.

There is no single-instance bottleneck because any Cassandra node can act as a coordinator for a query.

Cassandra can automatically accommodate changes and rebalance tokens while the database continues to service requests.

Cassandra clusters can run in a single cloud or data center or optionally be distributed across multiple locations (data centers, cloud availability zones, cloud regions, or even cloud providers)

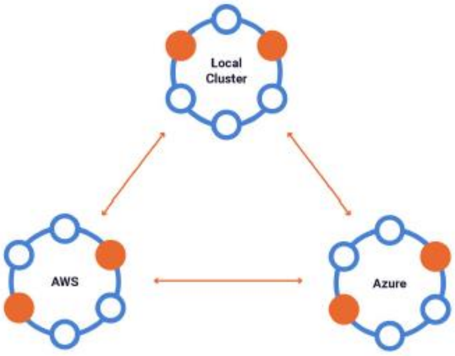
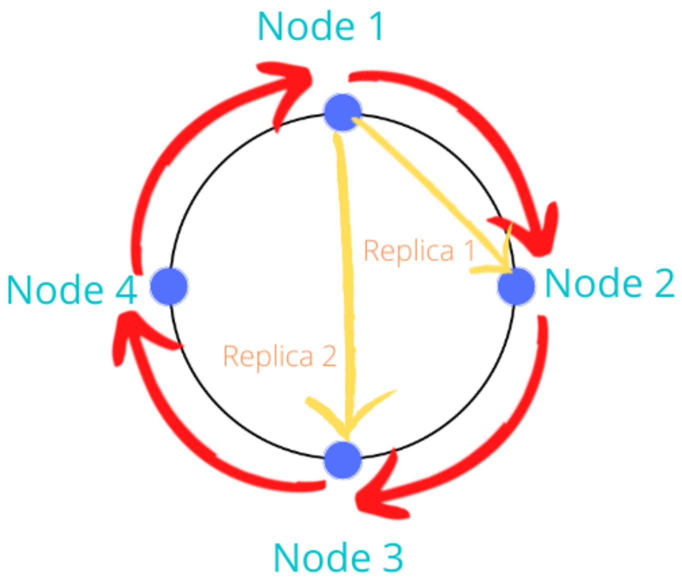


FIGURE 9 Cassandra cluster configured with two replicas per data center

The job of a “Snitch” is to inform each node about the relative proximity of other nodes. This information is used to determine which nodes to read from and write to, and how best to distribute replicas to maximize availability in case a node fails or a rack or data center becomes unreachable.

The “Gossip” protocol in Cassandra allows each node to keep track of the state of other nodes in the cluster.

With Cassandra’s “SimpleStrategy” used for replication within a single data center, replicas are simply placed on the next clockwise node around the Cassandra ring.



In addition to running Cassandra across local data centers or cloud machine instances, Cassandra can be deployed in a Docker container, providing even more deployment flexibility.

"DataStax" has provided the majority of the commits to Cassandra and is the main provider of open source Cassandra drivers for the most popular development languages.

The "DataStax Bulk Loader" supports a variety of file formats, handles parse errors and database insertion errors gracefully, and can load data up to 4x faster than the native CQL COPY command.

The "DataStax Kafka Connector" benefits from DataStax experience in driver development and delivers superior performance.

The advantage of Cassandra over the other NoSQL Databases :

- Cassandra uses a ring architecture which make all nodes in a cluster to be treated equally, and a majority of nodes can be used to achieve quorum.
- It has lack of a single point of failure, allowing for better fault tolerance compared to document stores such as MongoDB. And it has Automatic data balancing.
- Cassandra Query Language (CQL), closely resembles the traditional SQL syntax, so it can be easier for SQL users to understand which gives it some advantage over HBase database for example, as HBase uses the basic shell for queries.

- **Cassandra is Free, open-source, and downloadable at no extra cost. Setting it up is easy and free.**