Summary: Cassandra

Here is a quick summary of Cassandra for you!

We'll cover the following

Summary#

1. Cassandra is a **distributed**, **decentralized**, **scalable**, and **highly available** NoSQL database.
2. Cassandra was designed with the understanding that software/hardware **failures can and do occur**.
3. Cassandra is a **peer-to-peer** distributed system, i.e., it does not have any leader or follower nodes. All nodes are equal, and there is no single point of failure.
4. Data, in Cassandra, is automatically distributed across nodes.
5. Data is replicated across the nodes for fault tolerance and redundancy.
6. Cassandra uses the **Consistent Hashing** algorithm to distribute the data among nodes in the cluster. Cassandra cluster has a ring-type architecture, where its nodes are logically distributed like a ring.
7. Cassandra utilizes the data model of Google's Bigtable, i.e., **SSTables** and **MemTables**.

8. Cassandra utilizes distributed features of Amazon's Dynamo, i.e., consistent hashing, partitioning, and replication.
9. Cassandra offers **Tunable consistency** for both read and write operations to adjust the tradeoff between availability and consistency of data.
10. Cassandra uses the **gossip protocol** for inter-node communication.

System design patterns#

Here is a summary of system design patterns used in Cassandra:

- **Consistent Hashing:** Cassandra uses Consistent Hashing to distribute its data across nodes.
- **Quorum:** To ensure data consistency, each Cassandra write operation can be configured to be successful only if the data has been written to at least a quorum of replica nodes.
- **Write-Ahead Log:** To ensure durability, whenever a node receives a write request, it immediately writes the data to a commit log which is a write-ahead log.
- **Segmented Log:** Cassandra uses the segmented log strategy to split its commit log into multiple smaller files instead of a single large file for easier operations. As we know, when a node receives a write operation, it immediately writes the data to a commit log. As the commit log grows and reaches its threshold in size, a new commit log is created. Hence, over time several commit logs could be present, each of which is called a segment. Commit log segments reduce the number of seeks needed to write to disk. Commit log segments are truncated when Cassandra has flushed corresponding data to SSTables. Commit log segments can be archived, deleted, or recycled once all its data has been flushed to SSTables.

- **Gossip protocol:** Cassandra uses gossip protocol that allows each node to keep track of state information about the other nodes in the cluster.
- **Generation number:** In Cassandra, each node keeps a generation number which is incremented whenever a node restarts. This generation number is included in gossip messages exchanged between nodes and is used to distinguish the node's current state from its state before a restart.
- **Phi Accrual Failure Detector:** Cassandra uses an adaptive failure detection mechanism as described by the Phi Accrual Failure Detector algorithm. This algorithm, instead of providing a binary output telling if the system is up or down, uses historical heartbeat information to output the suspicion level about a node. A higher suspicion level means there are high chances that the node is down.
- **Bloom filters:** In Cassandra, each SStable has a Bloom filter associated with it, which tells if a particular key is present in it or not.
- **Hinted Handoff:** Cassandra nodes use Hinted Handoff to remember the write operation for failing nodes.
- **Read Repair:** Cassandra uses 'Read Repair' to push the latest version of the data to nodes with the older versions.

Cassandra characteristics#

Here are a few reasons behind Cassandra's performance and popularity:

- **Distributed** means it can run on a large number of machines.
- **Decentralized** means there's no leader-follower paradigm. All nodes are identical and can perform all functions of Cassandra.

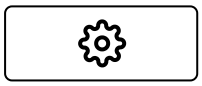
- **Scalable** means that Cassandra can be easily scaled horizontally by adding more nodes to the cluster without any performance impact.

  No manual intervention or rebalancing is required. Cassandra achieves linear scalability and proven fault-tolerance on commodity hardware.
- **Highly Available** means Cassandra is fault-tolerant, and the data remains available even if one or several nodes or data centers go down.
- **Fault-Tolerant and reliable,** as data is replicated to multiple nodes, fault-tolerance is pretty high.
- **Tunable consistency** means that it is possible to adjust the tradeoff between availability and consistency of data on Cassandra nodes, typically by configuring replication factor and consistency level settings.
- **Durable** means Cassandra stores data permanently.
- **Eventually Consistent** as Cassandra favors high availability at the cost of strong consistency.
- **Geographic distribution** means Cassandra supports geographical distribution and efficient data replication across multiple clouds and data centers.

References and further reading#

- Bigtable (https://research.google/pubs/pub27898/)
- Dynamo (http://www.read.seas.harvard.edu/~kohler/class/cs239-w08/decandia07dynamo.pdf)
- Datastax docs (https://docs.datastax.com/en/dse/6.8/dse-arch/datastax_enterprise/dbArch/archTOC.html)
- Tombstones common problems (https://opencredo.com/blogs/cassandra-tombstones-common-issues/)
- The Phi Accrual Failure Detector

- The Phi Accrual Failure Detector (http://citeseerx.ist.psu.edu/viewdoc/download?



doi=10.1.1.80.7427&rep=rep1&type=pdf)

- Cassandra introduction video (https://www.coursera.org/lecture/cloud-applications-part2/2-3-1-cassandra-introduction-olmpu)

← Back

Tombstones

Next →

Quiz: Cassandra

Mark as Completed

Report an Issue