Cassandra Consistency Levels

Let's explore how Cassandra manages data consistency.

We'll cover the following

What are Cassandra’s consistency levels?#

Cassandra’s consistency level is defined as the minimum number of Cassandra nodes that must fulfill a read or write operation before the operation can be considered successful. Cassandra allows us to specify different consistency levels for read and write operations. Also, Cassandra has tunable consistency, i.e., we can increase or decrease the consistency levels for each request.

There is always a tradeoff between consistency and performance. A higher consistency level means that more nodes need to respond to a read or write query, giving the user more assurance that the values present on each replica are the same.

Write consistency levels

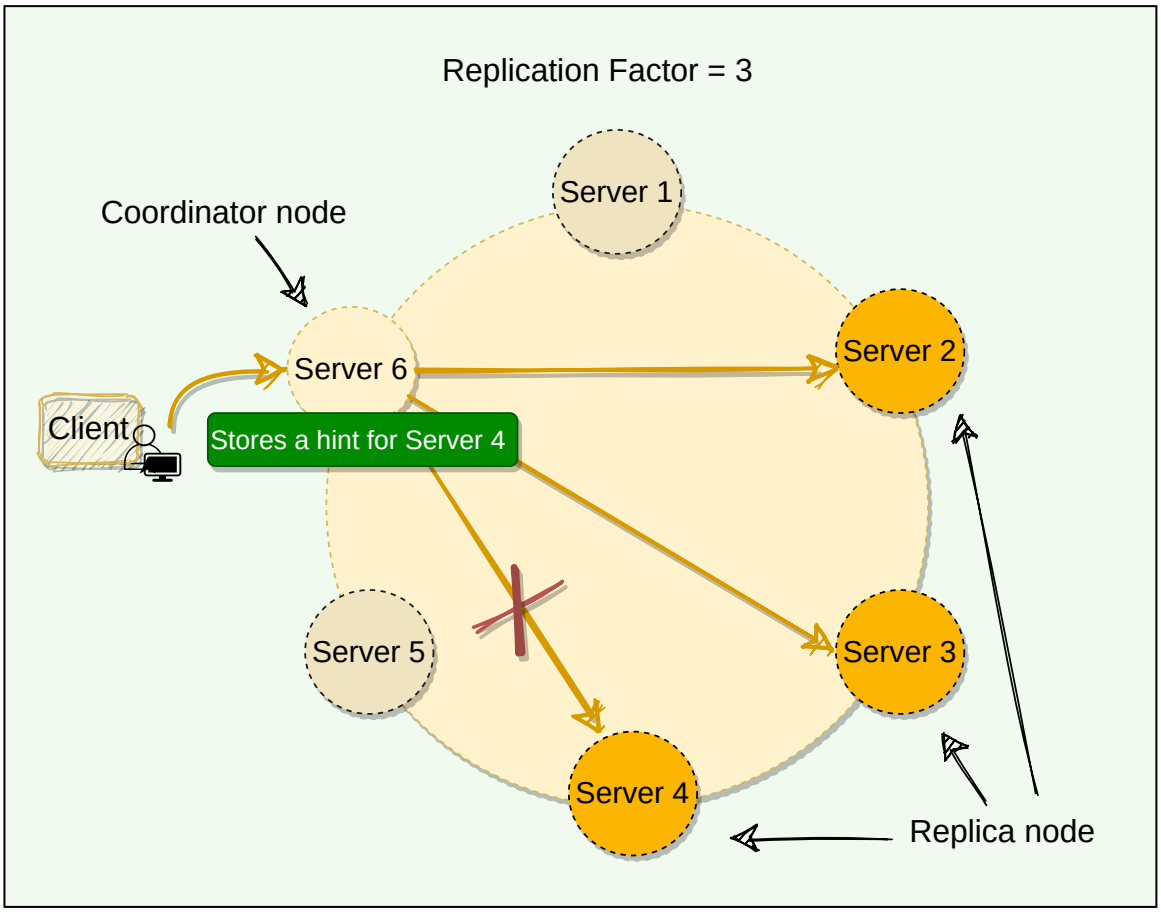
For write operations, the consistency level specifies how many replica nodes must respond for the write to be reported as successful to the client. The consistency level is specified per query by the client. Because Cassandra is eventually consistent, updates to other replica nodes may continue in the background. Here are different write consistency levels that Cassandra offers:

- **One** or **Two** or **Three**: The data must be written to at least the specified number of replica nodes before a write is considered successful.
- **Quorum**: The data must be written to at least a quorum (or majority) of replica nodes. Quorum is defined as $floor(RF/2 + 1)$, where RF represents the replication factor. For example, in a cluster with a replication factor of five, if three nodes return success, the write is considered successful.
- **All**: Ensures that the data is written to all replica nodes. This consistency level provides the highest consistency but lowest availability as writes will fail if any replica is down.
- **Local_Quorum**: Ensures that the data is written to a quorum of nodes in the same datacenter as the coordinator. It does not wait for the response from the other data-centers.
- **Each_Quorum**: Ensures that the data is written to a quorum of nodes in each datacenter.
- **Any**: The data must be written to at least one node. In the extreme case, when all replica nodes for the given partition key are down, the write can still succeed after a hinted handoff (discussed below) has been written. ‘Any’ consistency level provides the lowest latency and highest availability, however, it comes with the lowest consistency. If all replica nodes are down at write time, an ‘Any’ write is not readable until the replica nodes for that partition have recovered and the latest data is written on them.

How does Cassandra perform a write operation? For a write, the coordinator node contacts all replicas, as determined by the replication factor, and considers the write successful when a number of replicas equal to the consistency level acknowledge the write.

Hinted handoff#

Depending upon the consistency level, Cassandra can still serve write requests even when nodes are down. For example, if we have the replication factor of three and the client is writing with a quorum consistency level. This means that if one of the nodes is down, Cassandra can still write on the remaining two nodes to fulfill the consistency level, hence, making the write successful.



Hinted handoff

Now when the node which was down comes online again, how should we write data to it? Cassandra accomplishes this through hinted handoff.

When a node is down or does not respond to a write request, the coordinator node writes a hint in a text file on the local disk. This hint contains the data itself along with information about which node the data belongs to. When the coordinator node discovers from the Gossiper (will be discussed later) that a node for which it holds hints has recovered, it forwards the write requests for each hint to the target. Furthermore, each node every ten minutes checks to see if the failing node, for which it is holding any hints, has recovered.

With consistency level 'Any,' if all the replica nodes are down, the coordinator node will write the hints for all the nodes and report success to the client. However, this data will not reappear in any subsequent reads until one of the replica nodes comes back online, and the coordinator node successfully forwards the write requests to it. This is assuming that the coordinator node is up when the replica node comes back. This also means that we can lose our data if the coordinator node dies and never comes back. For this reason, we should avoid using the 'Any' consistency level.

If a node is offline for some time, the hints can build up considerably on other nodes. Now, when the failed node comes back online, other nodes tend to flood that node with write requests. This can cause issues on the node, as it is already trying to come back after a failure. To address this problem, Cassandra limits the storage of hints to a configurable time window. It is also possible to disable hinted handoff entirely.

Cassandra, by default, stores hints for three hours. After three hours, older hints will be removed, which means, if now the failed node recovers, it will have stale data. Cassandra can fix this stale data while serving a read request. Cassandra can issue a **Read Repair** when it sees stale data; we will go through this while discussing the read path.

One thing to remember: When the cluster cannot meet the consistency level specified by the client, Cassandra fails the write request and does not store a hint.

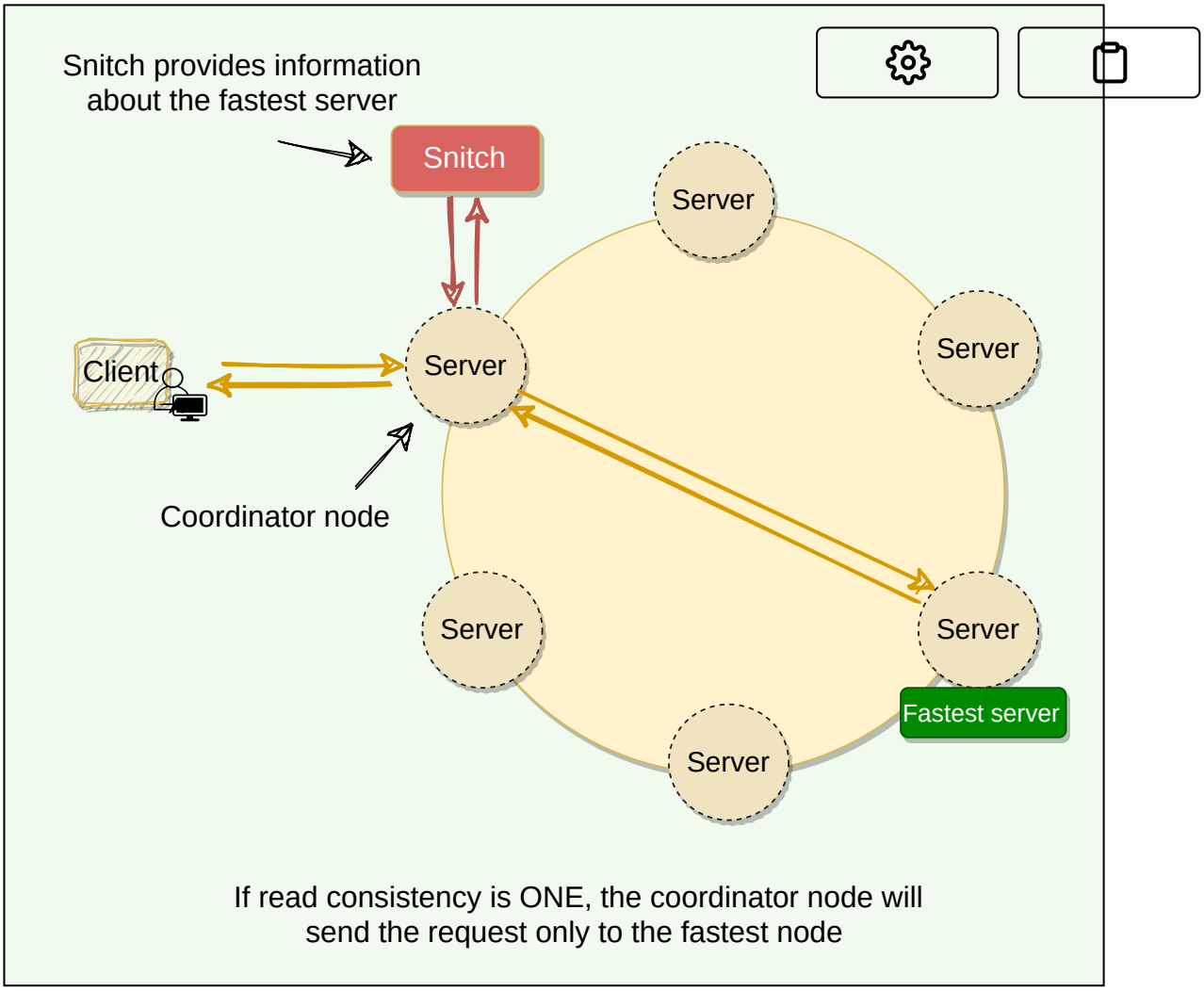
Read consistency levels#

The consistency level for read queries specifies how many replica nodes must respond to a read request before returning the data. For example, for a read request with a consistency level of quorum and replication factor of three, the coordinator waits for successful replies from at least two nodes.

Cassandra has the same consistency levels for read operations as that of write operations except for Each_Quorum (because it is very expensive).

To achieve strong consistency in Cassandra: $R + W > RF$ gives us strong consistency. In this equation, R, W, and RF are the read replica count, the write replica count, and the replication factor, respectively. All client reads will see the most recent write in this scenario, and we will have strong consistency.

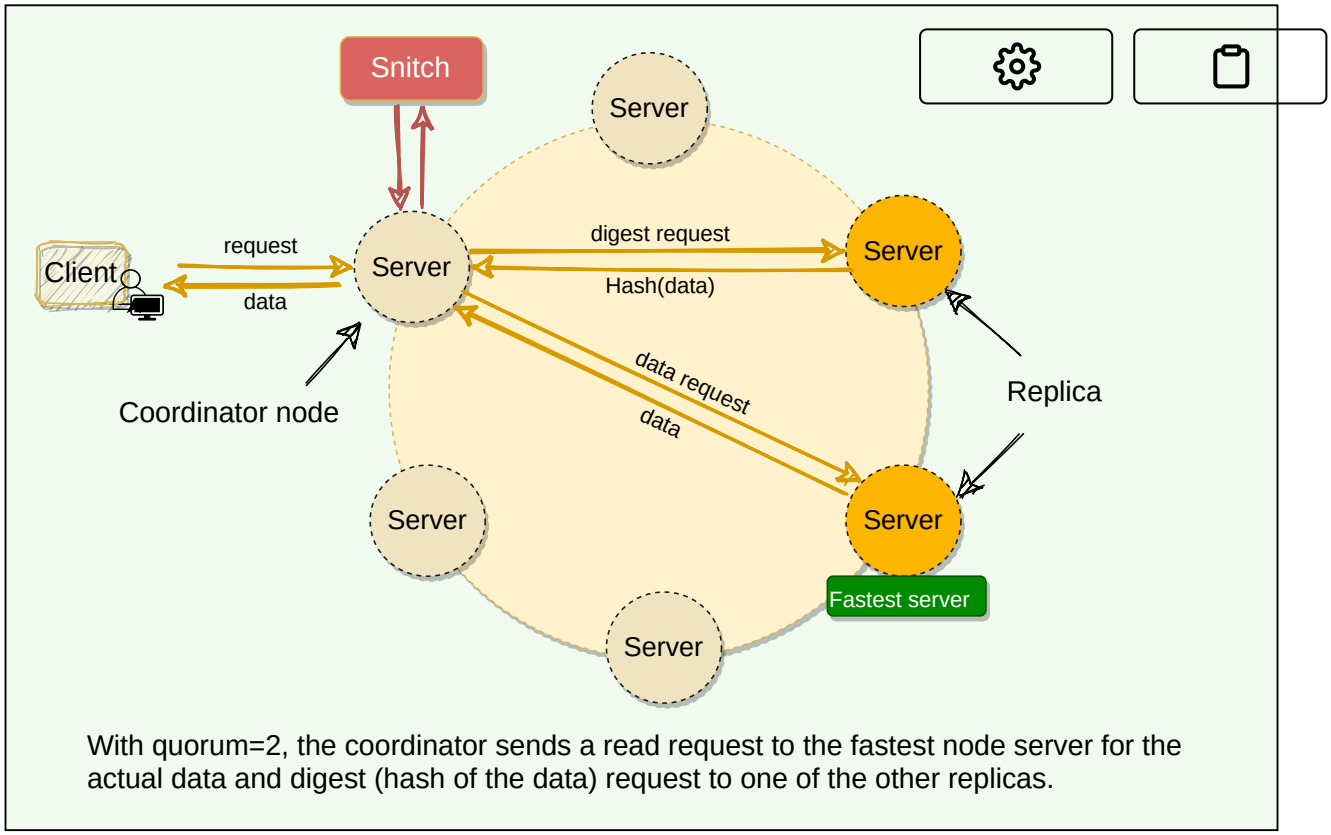
Snitch: The Snitch is an application that determines the proximity of nodes within the ring and also tells which nodes are faster. Cassandra nodes use this information to route read/write requests efficiently. We will discuss this in detail later.



Coordinator node forwards the read request to the fastest server

How does Cassandra perform a read operation? The coordinator always sends the read request to the fastest node. For example, for Quorum=2, the coordinator sends the request to the fastest node and the digest of the data from the second-fastest node. The digest is a checksum of the data and is used to save network bandwidth.

If the digest does not match, it means some replicas do not have the latest version of the data. In this case, the coordinator reads the data from all the replicas to determine the latest data. The coordinator then returns the latest data to the client and initiates a **read repair** request. The read repair operation pushes the newer version of data to nodes with the older version.



Read repair

While discussing Cassandra's write path, we saw that the nodes could become out of sync due to network issues, node failures, corrupted disks, etc. The read repair operation helps nodes to resync with the latest data. Read operation is used as an opportunity to repair inconsistent data across replicas. The latest write-timestamp is used as a marker for the correct version of data. The read repair operation is performed only in a portion of the total reads to avoid performance degradation. Read repairs are opportunistic operations and not a primary operation for anti-entropy.

Read Repair Chance: When the read consistency level is less than 'All,' Cassandra performs a read repair probabilistically. By default, Cassandra tries to read repair 10% of all requests with DC local read repair. In this case, Cassandra immediately sends a response when the consistency level is met and performs the read repair asynchronously in the background.

Snitch#

Snitch keeps track of the network topology of Cassandra nodes. It determines which data-centers and racks nodes belong to. Cassandra uses this information to route requests efficiently. Here are the two main functions of a snitch in Cassandra:

- Snitch determines the proximity of nodes within the ring and also monitors the read latencies to avoid reading from nodes that have slowed down. Each node in Cassandra uses this information to route requests efficiently.
- Cassandra's replication strategy uses the information provided by the Snitch to spread the replicas across the cluster intelligently. Cassandra will do its best by not having more than one replica on the same "rack".

To understand Snitch's role, let's take the example of Cassandra's read operation. Let's assume that the client is performing a read with a quorum consistency level, and the data is replicated on five nodes. To support maximum read speed, Cassandra selects a single replica to query for the full object and asks for the digest of the data from two additional nodes in order to ensure that the latest version of the data is returned. The Snitch helps to identify the fastest replica, and Cassandra asks this replica for the full object.

← Back

Next →

Replication

Gossiper

Mark as Completed

Report an Issue