About hinted handoff writes

Hinted handoff is a Cassandra feature that optimizes the cluster [consistency](http://docs.datastax.com/en/archived/cassandra/2.0/cassandra/dml/dmlAboutDataConsistency.html) process and [anti-entropy](https://docs.datastax.com/en/glossary/doc/glossary/gloss_anti_entropy.html) when a replica-owning node is not available, due to network issues or other problems, to accept a replica from a successful write operation. Hinted handoff is not a process that guarantees successful write operations, except when a client application uses a consistency level of ANY. You [enable or disable hinted handoff](http://docs.datastax.com/en/archived/cassandra/2.0/cassandra/configuration/configCassandra_yaml_r.html#reference_ds_qfg_n1r_1k__hinted_handoff_enabled) in the cassandra.yaml file.

## **How hinted handoff works**

During a write operation, when hinted handoff is enabled and consistency can be met, the coordinator stores a hint about dead replicas in the local system.hints table under either of these conditions:

* A replica node for the row is known to be down ahead of time.
* A replica node does not respond to the write request.

When the cluster cannot meet the consistency level specified by the client, Cassandra does not store a hint.

A hint indicates that a write needs to be replayed to one or more unavailable nodes. The hint consists of:

* The location of the replica that is down
* Version metadata
* The actual data being written

By default, hints are saved for three hours after a replica fails because if the replica is down longer than that, it is likely permanently dead. You can configure this interval of time using the [max\_hint\_window\_in\_ms](http://docs.datastax.com/en/archived/cassandra/2.0/cassandra/configuration/configCassandra_yaml_r.html" \l "reference_ds_qfg_n1r_1k__max_hint_window_in_ms) property in the cassandra.yaml file. If the node recovers after the save time has elapsed, run a repair to re-replicate the data written during the down time.

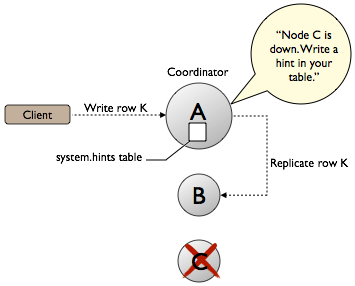
After a node discovers from [gossip](http://docs.datastax.com/en/archived/cassandra/2.0/cassandra/architecture/architectureGossipAbout_c.html) that a node for which it holds hints has recovered, the node sends the data row corresponding to each hint to the target. Additionally, the node checks every ten minutes for any hints for writes that timed out during an outage too brief for the failure detector to notice through gossip.

For example, in a cluster of two nodes, A and B, having a replication factor (RF) of 1, each row is stored on one node. Suppose node A is down while we write row K to it with consistency level of one. The write fails because reads always reflect the most recent write when:

W + R > replication factor

where W is the number of nodes to block for writes and R is the number of nodes to block for reads. Cassandra does not write a hint to B and call the write good because Cassandra cannot read the data at any consistency level until A comes back up and B forwards the data to A.

In a cluster of three nodes, A (the coordinator), B, and C, each row is stored on two nodes in a keyspace having a replication factor of 2. Suppose node C goes down. The client writes row K to node A. The coordinator, replicates row K to node B, and writes the hint for downed node C to node A.



Cassandra, configured with a consistency level of ONE, calls the write good because Cassandra can read the data on node B. When node C comes back up, node A reacts to the hint by forwarding the data to node C. For more information about how hinted handoff works, see ["Modern hinted handoff"](https://www.datastax.com/dev/blog/modern-hinted-handoff) by Jonathan Ellis.

## **Extreme write availability**

For applications that want Cassandra to accept writes even when all the normal replicas are down, when not even consistency level ONE can be satisfied, Cassandra provides consistency level ANY. ANY guarantees that the write is durable and will be readable after an appropriate replica target becomes available and receives the hint replay.

## **Performance**

By design, hinted handoff inherently forces Cassandra to continue performing the same number of writes even when the cluster is operating at reduced capacity. Pushing your cluster to maximum capacity with no allowance for failures is a bad idea.

Hinted handoff is designed to minimize the extra load on the cluster.

All hints for a given replica are stored under a single [partition key](https://docs.datastax.com/en/glossary/doc/glossary/gloss_partition_key.html), so replaying hints is a simple sequential read with minimal performance impact.

If a replica node is overloaded or unavailable, and the failure detector has not yet marked it down, then expect most or all writes to that node to fail after the timeout triggered by [write\_request\_timeout\_in\_ms,](http://docs.datastax.com/en/archived/cassandra/2.0/cassandra/configuration/configCassandra_yaml_r.html" \l "reference_ds_qfg_n1r_1k__write_request_timeout_in_ms)which defaults to 10 seconds. During that time, Cassandra writes the hint when the timeout is reached.

If this happens on many nodes at once this could become substantial memory pressure on the coordinator. So the coordinator tracks how many hints it is currently writing, and if this number gets too high it will temporarily refuse writes (withOverloadedException) whose replicas include the misbehaving nodes.

## **Removal of hints**

When removing a node from the cluster by decommissioning the node or by using the [nodetool removenode](http://docs.datastax.com/en/archived/cassandra/2.0/cassandra/tools/toolsRemoveNode.html" \o "Provides the status of current node removal, forces completion of pending removal, or removes the identified node.) command, Cassandra automatically removes hints targeting the node that no longer exists. Cassandra also removes hints for dropped tables.

## **Scheduling repair weekly**

At first glance, it may appear that hinted handoff lets you safely get away without needing repair. This is only true if you never have hardware failure. Hardware failure has the following ramifications:

* Loss of the historical data necessary to tell the rest of the cluster exactly what data is missing.
* Loss of hints-not-yet-replayed from requests that the failed node coordinated.

<http://docs.datastax.com/en/archived/cassandra/2.0/cassandra/dml/dml_about_hh_c.html>