« Performance improvements in Cassandra 1.2 The data dictionary in Cassandra 1.2 »

Modern hinted handoff

BY JONATHAN ELLIS - DECEMBER 11, 2012 | 9 COMMENTS

Hinted Handoff is an optional part of writes in Cassandra, enabled by default, with two purposes:

1. Hinted handoff allows Cassandra to offer full write availability when consistency is not required.
2. Hinted handoff dramatically improves response consistency after temporary outages such as network failures.

This post applies to Cassandra 1.0 and later versions.

How it works

When a write is performed and a replica node for the row is either known to be down ahead of time, or does not respond to the write request, the coordinator will store a hint locally, in the `system.hints` table. This hint is basically a wrapper around the mutation indicating that it needs to be replayed to the unavailable node(s).

Once a node discovers via gossip that a node for which it holds hints has recovered, it will send the data row corresponding to each hint to the target. Additionally, it will check every ten minutes to see if there are any hints for writes that timed out during an outage too brief for the failure dectector to notice via gossip.

Hinted Handoff and ConsistencyLevel

A hinted write does not count towards ConsistencyLevel requirements of ONE, QUORUM, or ALL. If insufficient replica targets are alive to sastisfy a requested ConsistencyLevel, UnavailableException will be thrown with or without Hinted Handoff. (This is an important difference from Dynamo's replication model; Cassandra does *not* default to sloppy quorum. But, see "Extreme write availability" below.)

To see why, let's look at a simple cluster of two nodes, A and B, and 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 ConsistencyLevel.ONE. We must fail the write: recall that the ConsistencyLevel contract is that "reads always reflect the most recent write when W + R > RF, where W is the number of nodes to block for on write, and R the number to block for on reads."

If we wrote a hint to B and call the write good because it is written "somewhere," the contract would be violated because there is no way to read the data at any ConsistencyLevel until A comes back up and B forwards the data to him.

Extreme write availability

For applications that want Cassandra to accept writes even when all the normal replicas are down (so even ConsistencyLevel.ONE cannot be satisfied), Cassandra provides ConsistencyLevel.ANY.

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. So pushing your cluster to maximum capacity with no allowance for failures ia a bad idea. That said, Cassandra's 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, so replaying hints is a simple sequential read with minimal performance impact.

But if a replica node is overloaded or unavailable, and the failure detector has not yet marked it down, then we can expect most or all writes to that node to fail after `write_request_timeout_in_ms`, which defaults to 10s. During that time, we have to keep a hint callback alive on the coordinator, waiting to write 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 (with `UnavailableException`) whose replicas include the misbehaving nodes.

Operations

When removing a node from the cluster (with decommission or removetoken), Cassandra automatically removes hints targetting the node that no longer exists.

Cassandra will also remove hints for dropped tables.

Repair and the fine print

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 means that

1. We lose "historical" data for which the write has already finished, so there is nothing to tell the rest of the cluster exactly what data has gone missing
2. We can also lose hints-not-yet-replayed from requests the failed node coordinated

With sufficient dedication, you can get by with "only run repair after hardware failure and rely on hinted handoff the rest of the time," but as your clusters grow (and hardware failure becomes more common) performing repair as a one-off special case will become increasingly difficult to do perfectly. Thus, we continue to recommend running a full repair weekly.

DataStax has many ways for you to advance in your career and knowledge.

You can take free classes, get certified, or read one of our many white papers.