Roanoke Code Camp

High Availability Web Applications with Cassandra

May 17 2014

About Me

http://www.tildedave.com/

This Talk

What is Cassandra? How Does It Work?

Why Cassandra is Great For Web Applications

Storing Sessions in Cassandra

Demo Time with Docker

This Talk

What is Cassandra?

How Does It Work?

Apache Cassandra

“The Apache Cassandra database is the right choice when you need scalability and high availability without compromising performance. Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data. Cassandra's support for replicating across multiple datacenters is best-in-class, providing lower latency for your users and the peace of mind of knowing that you can survive regional outages.”

- http://cassandra.apache.org/

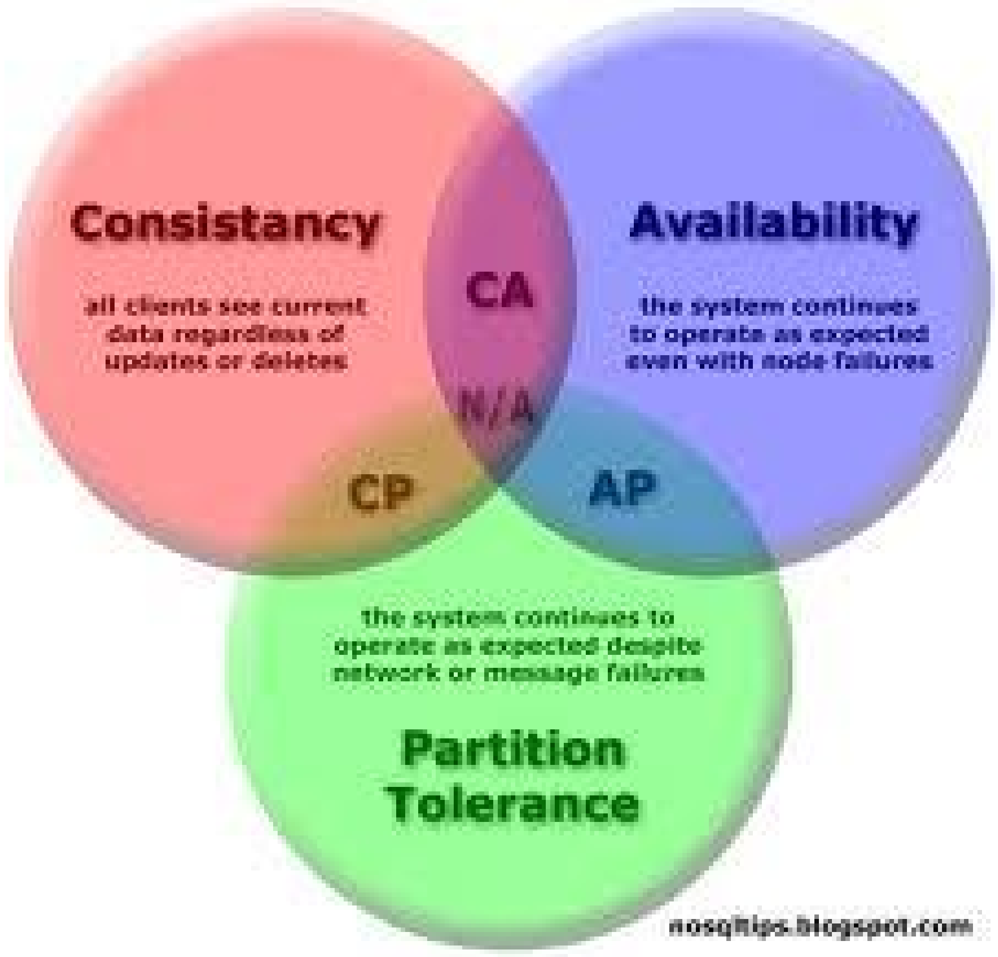
A Quick History of Apache Cassandra

- 2007 - Dynamo Paper (Amazon)
- 2008 - Facebook implementation open sourced
- February 2010: Graduates from Apache incubator
- April 2010: 0.6 released
- October 2011: 1.0 released
- September 2013: 2.0 released

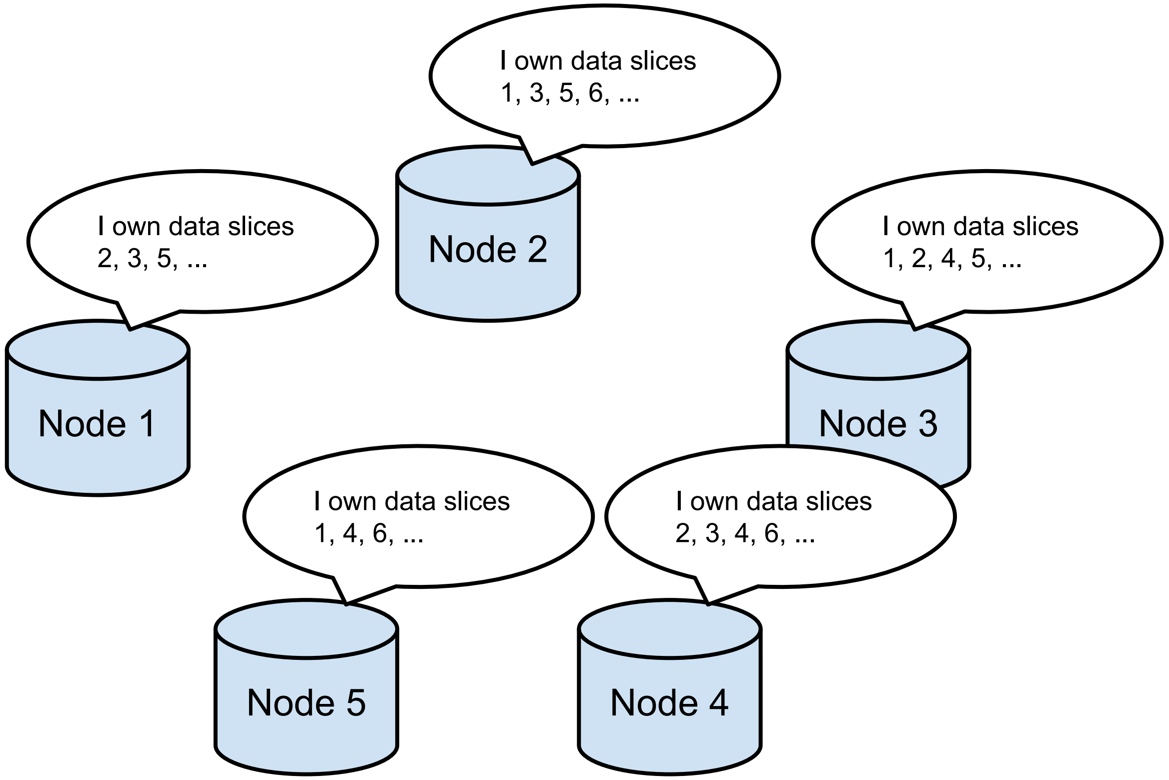
Cassandra Chooses:

Availability

Partition Tolerance

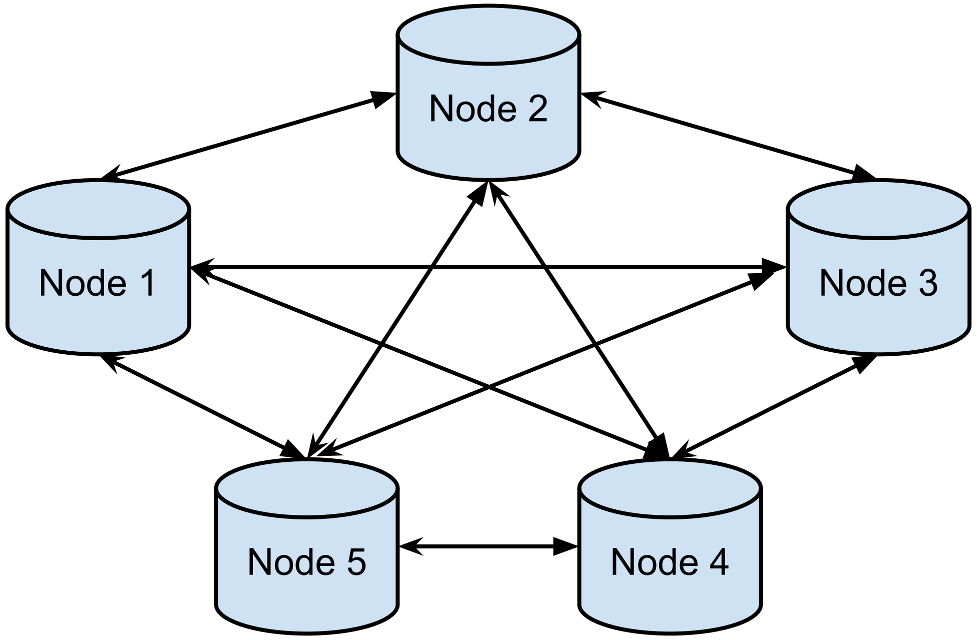

Consistancy
all clients see current data regardless of updates or deletes
CA
Availability
the system continues to operate as expected even with node failures
N/A
CP
AP
the system continues to operate as expected despite network or message failures
Partition Tolerance
nosqltips.blogspot.com

Many copies of the data are stored throughout the cluster

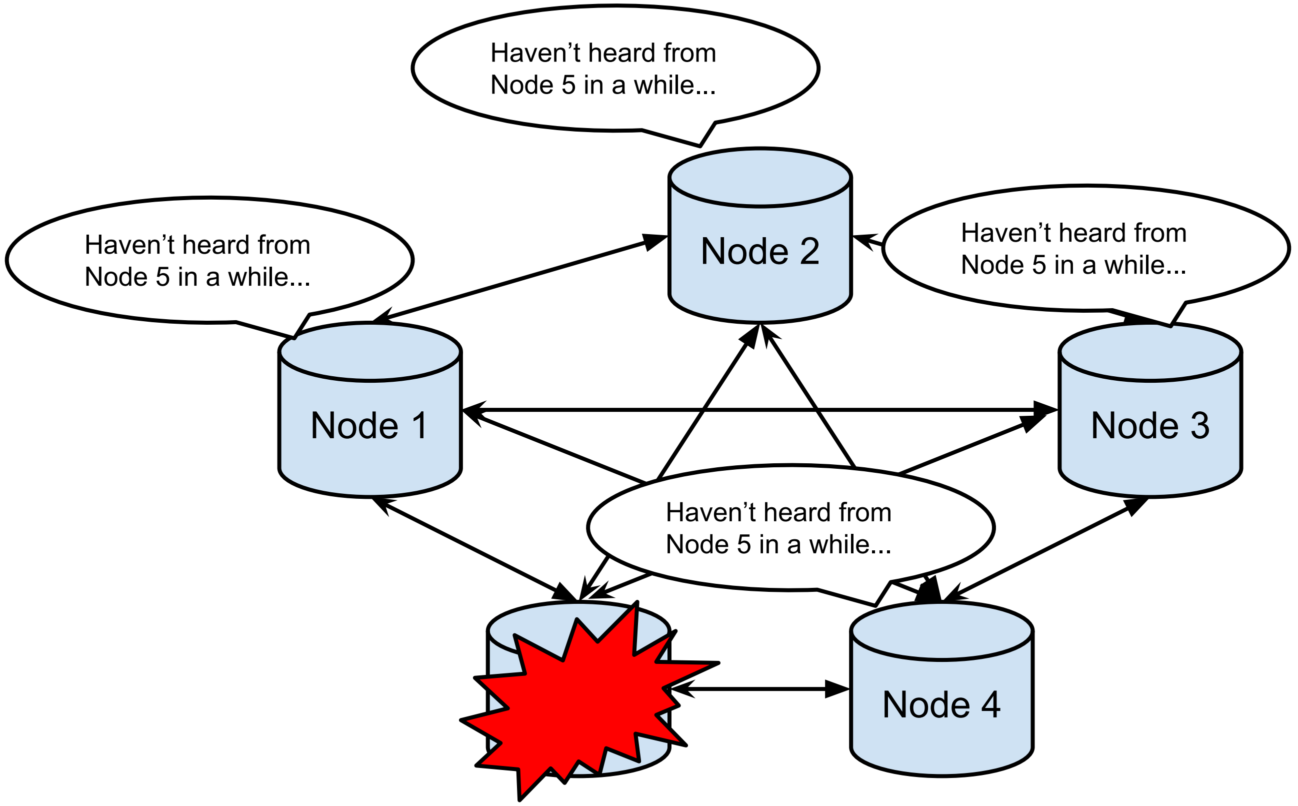

I own data slices 1, 3, 5, 6, ...
Node 2
I own data slices 2, 3, 5, ...
Node 1
I own data slices 1, 2, 4, 5, ...
Node 3
I own data slices 1, 4, 6, ...
Node 5
I own data slices 2, 3, 4, 6, ...
Node 4

How Cassandra Handles Failure: Gossip

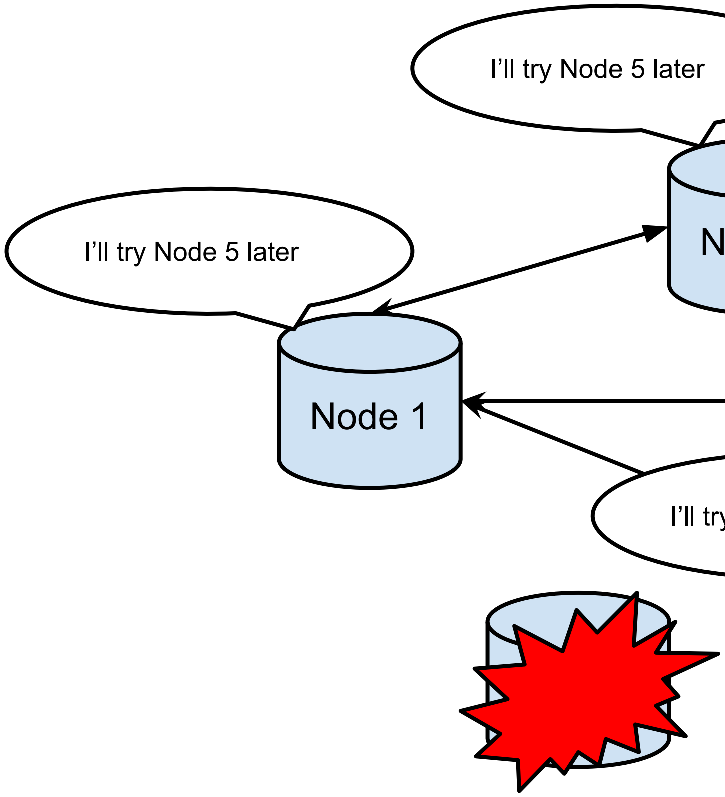
Cluster is constantly communicating through gossip



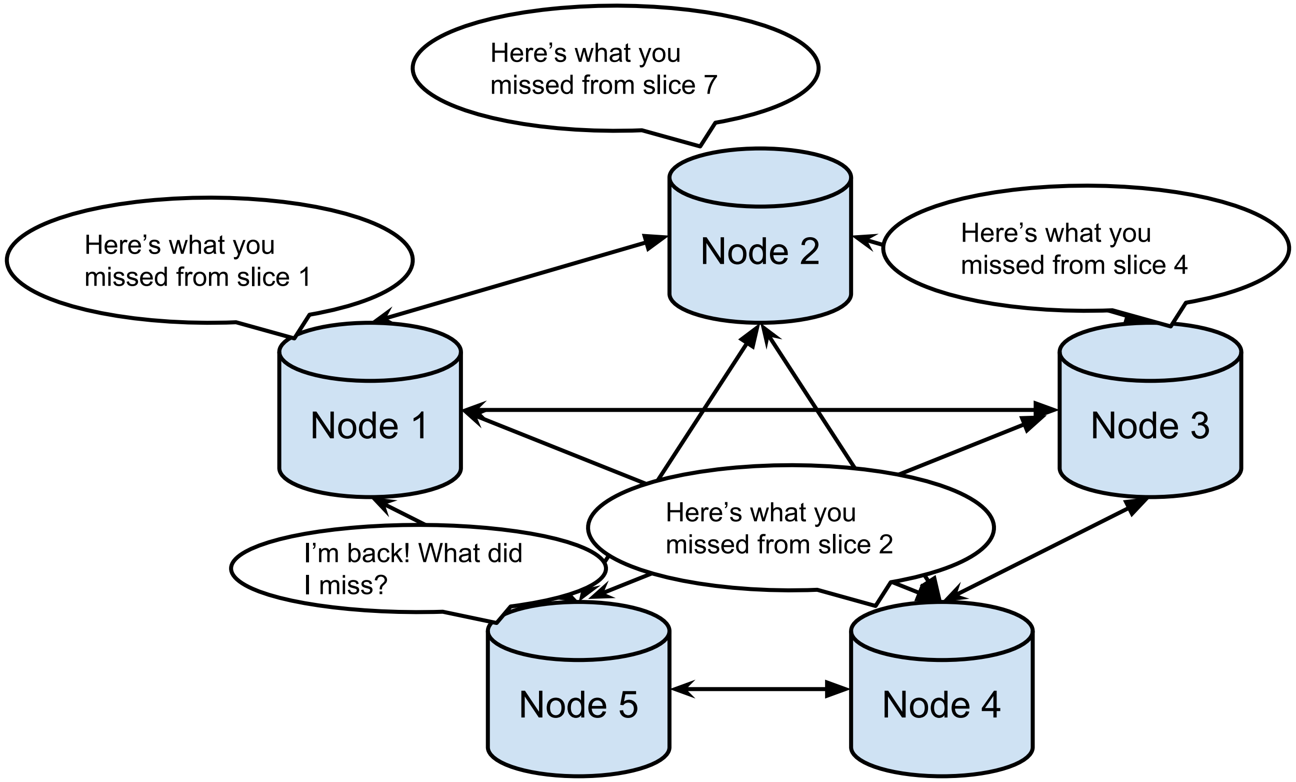
Node failures are detected by the cluster


Haven't heard from Node 5 in a while...
Haven't heard from Node 5 in a while...
Haven't heard from Node 5 in a while...
Haven't heard from Node 5 in a while...
Node 1
Node 2
Node 3
Node 4

If a node is unavailable, cluster continues to function

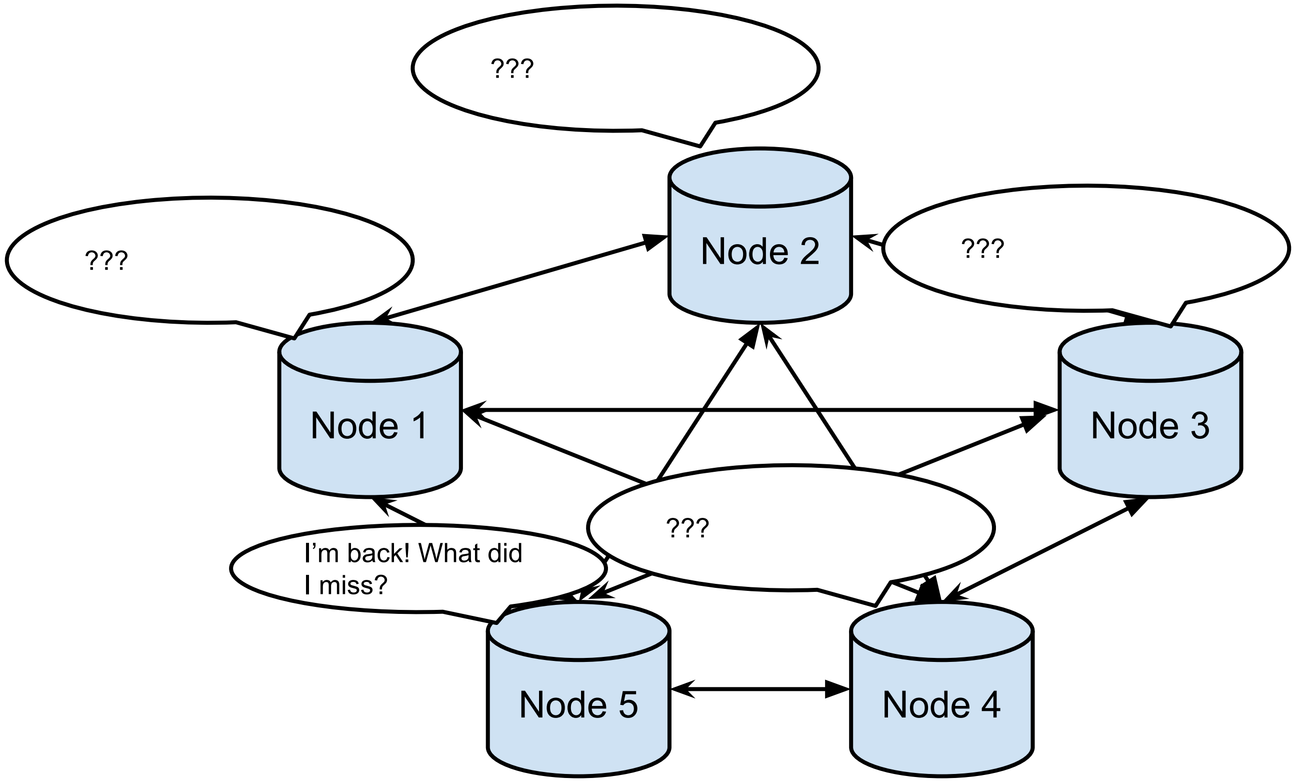


On recovery, node gets “up to speed” with missed data (hints)


Here’s what you missed from slice 7
Node 2
Here’s what you missed from slice 1
Here’s what you missed from slice 4
Node 1
Node 3
Here’s what you missed from slice 2
I’m back! What did I miss?
Node 5
Node 4

How Cassandra Handles Failure: Gossip

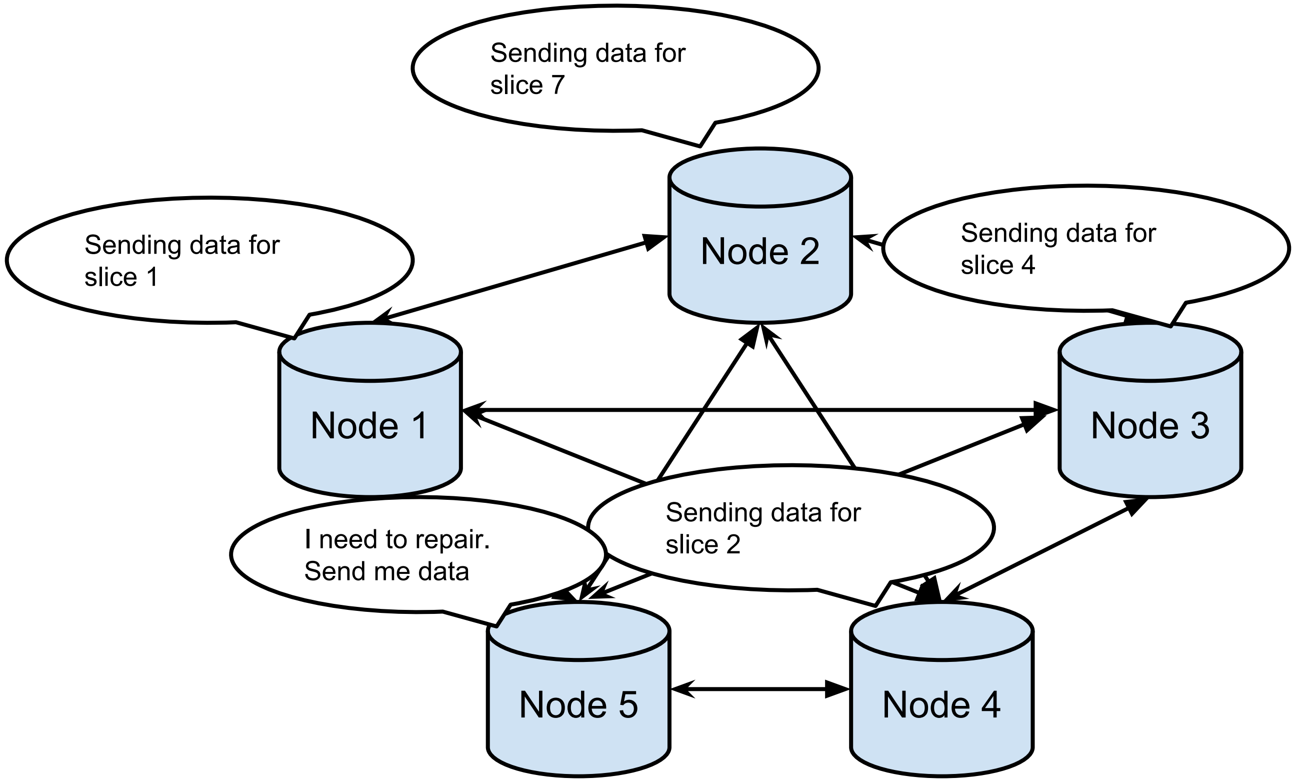
If a node has been gone too long, hints expire and it is broken. It needs to be repaired



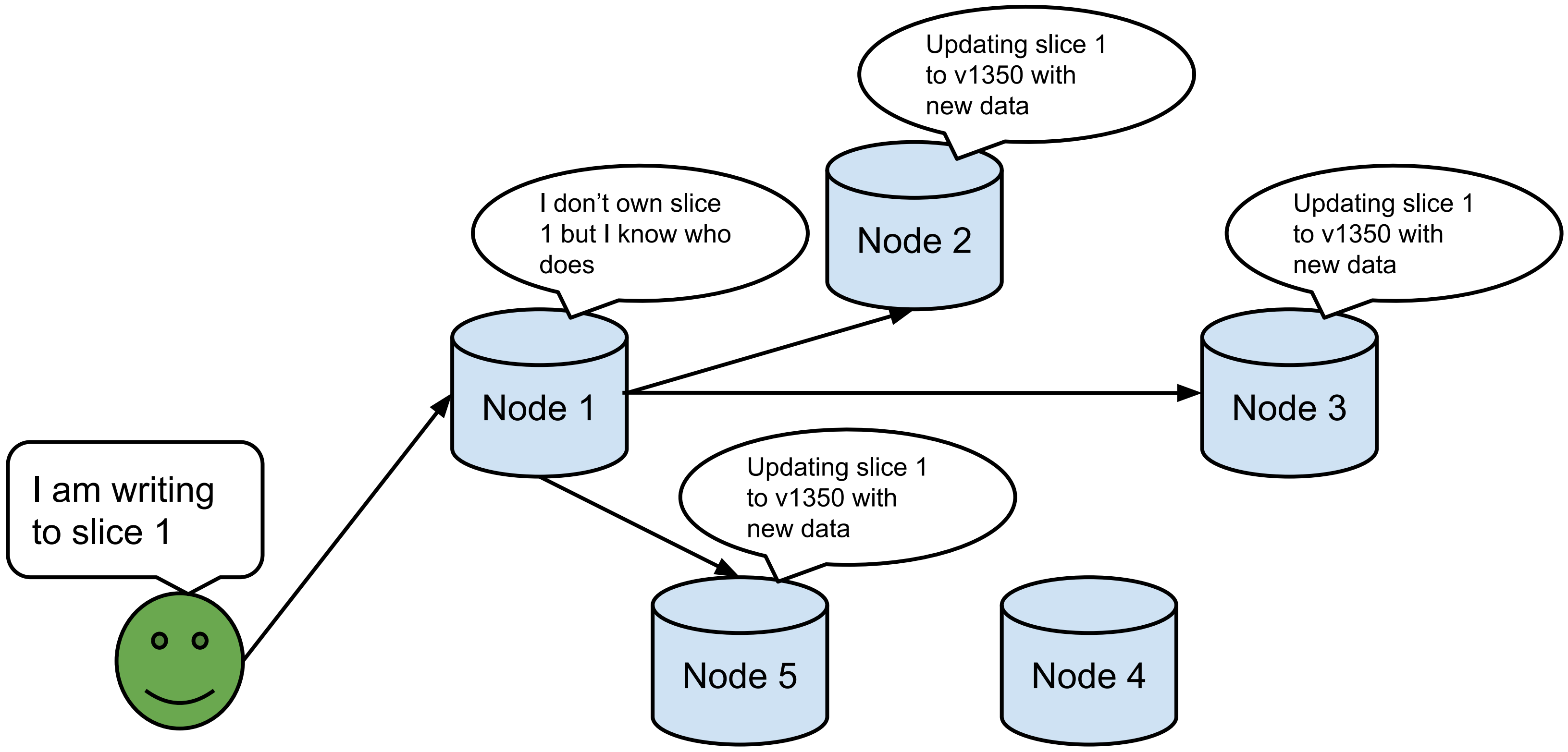
How Cassandra Handles Failure: Gossip

Repairing a node removes data inconsistencies

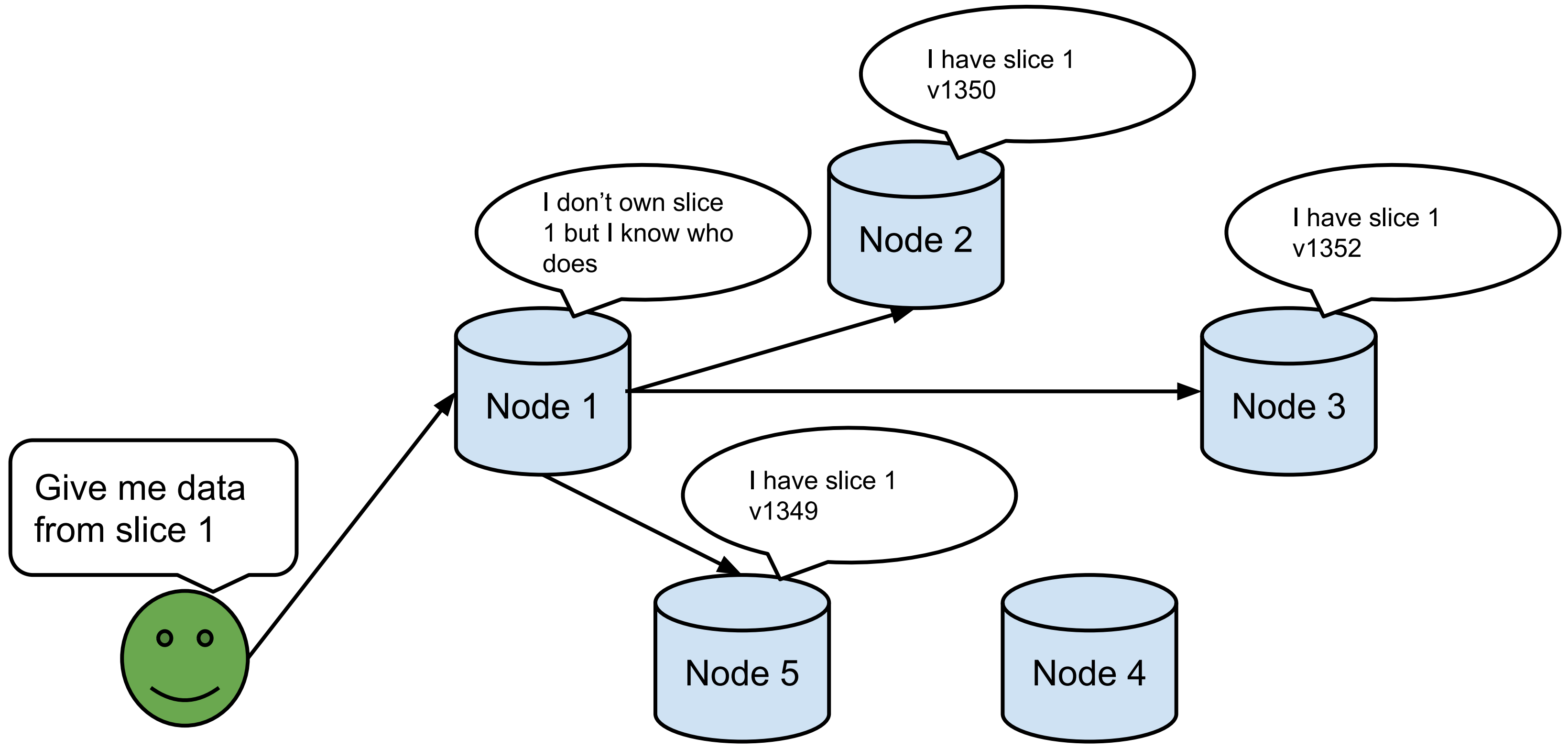
Uses a data structure called a Merkle Tree



Writes propagate throughout the cluster


I am writing to slice 1
I don't own slice 1 but I know who does
Updating slice 1 to v1350 with new data
Updating slice 1 to v1350 with new data
Updating slice 1 to v1350 with new data
Node 1
Node 2
Node 3
Node 4
Node 5

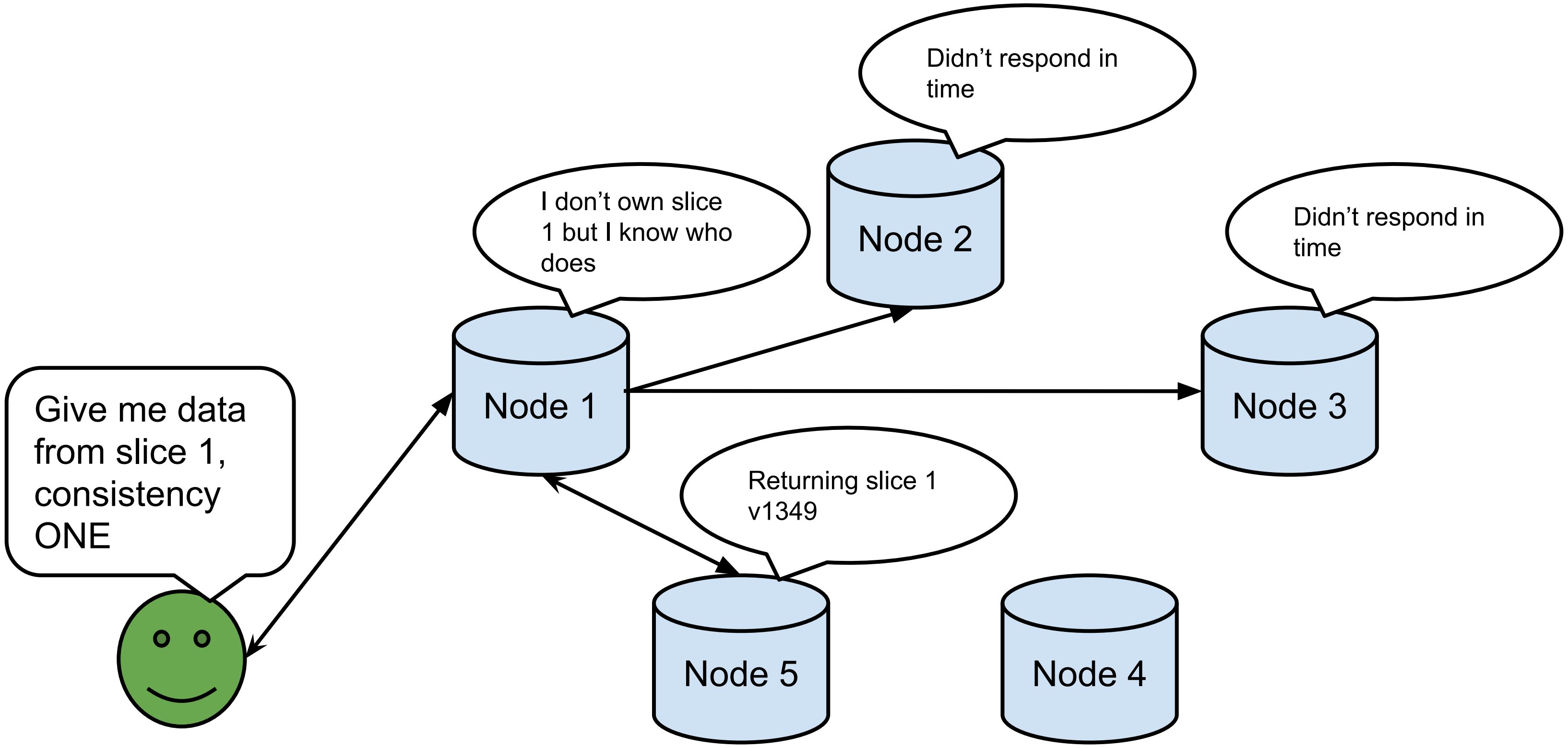
Queries may reach a cluster that is inconsistent
What to do here?



Querying with Consistency Level ONE

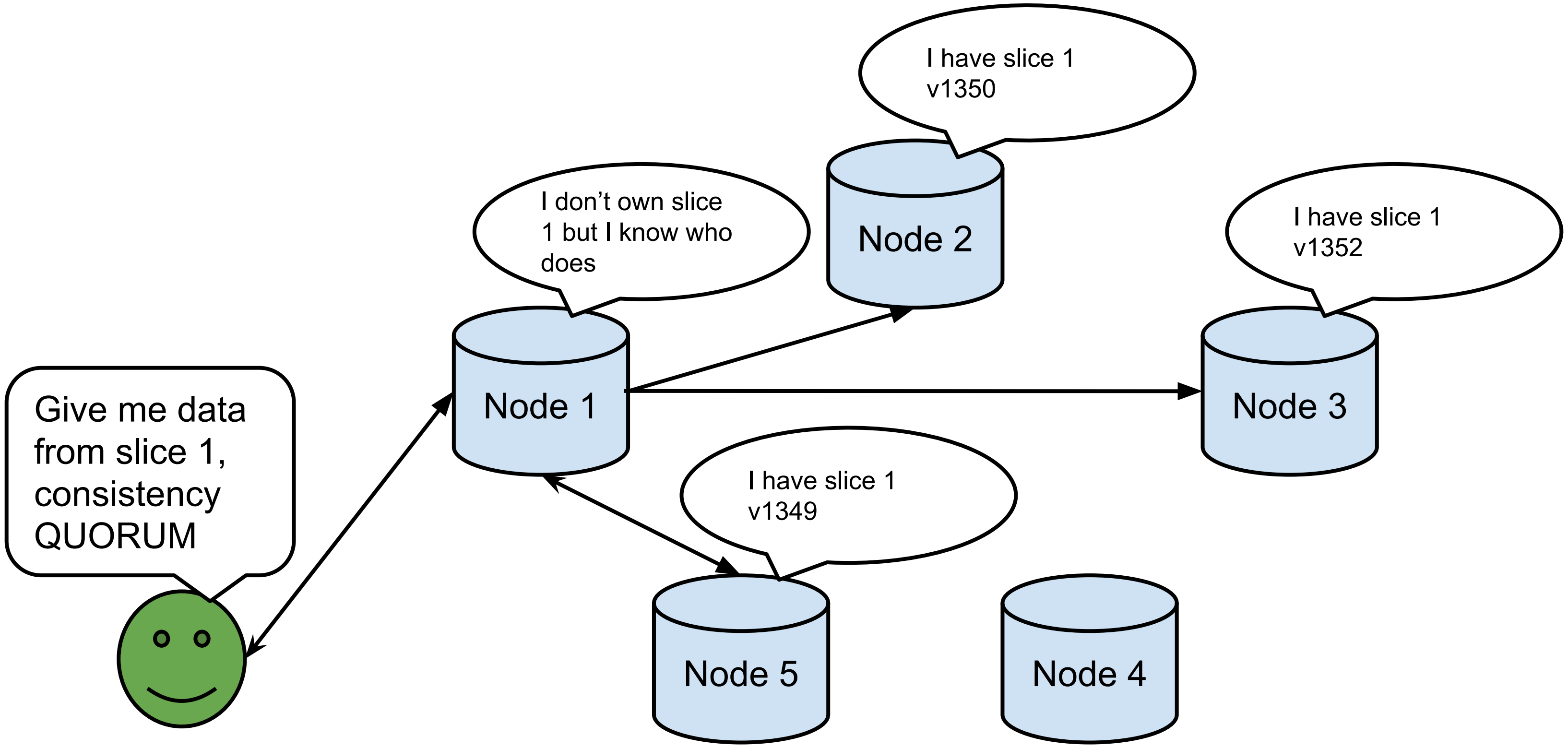
Client specifies a consistency level

ONE consistency only asks for one node to respond



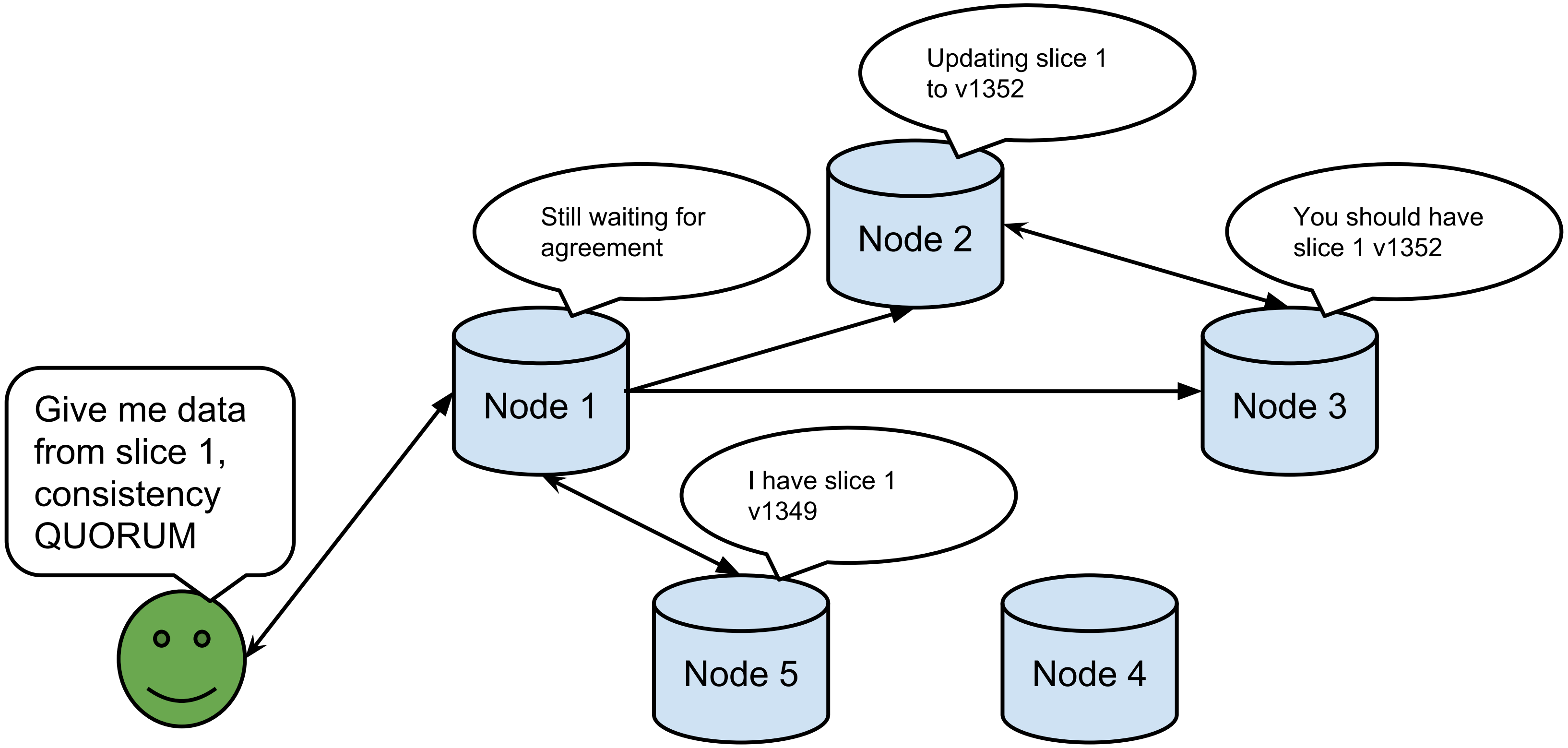
Querying with Consistency Level QUORUM

QUORUM requires a majority of nodes that own a slice to respond (here: 3 copies of slice 1, 2 must respond)



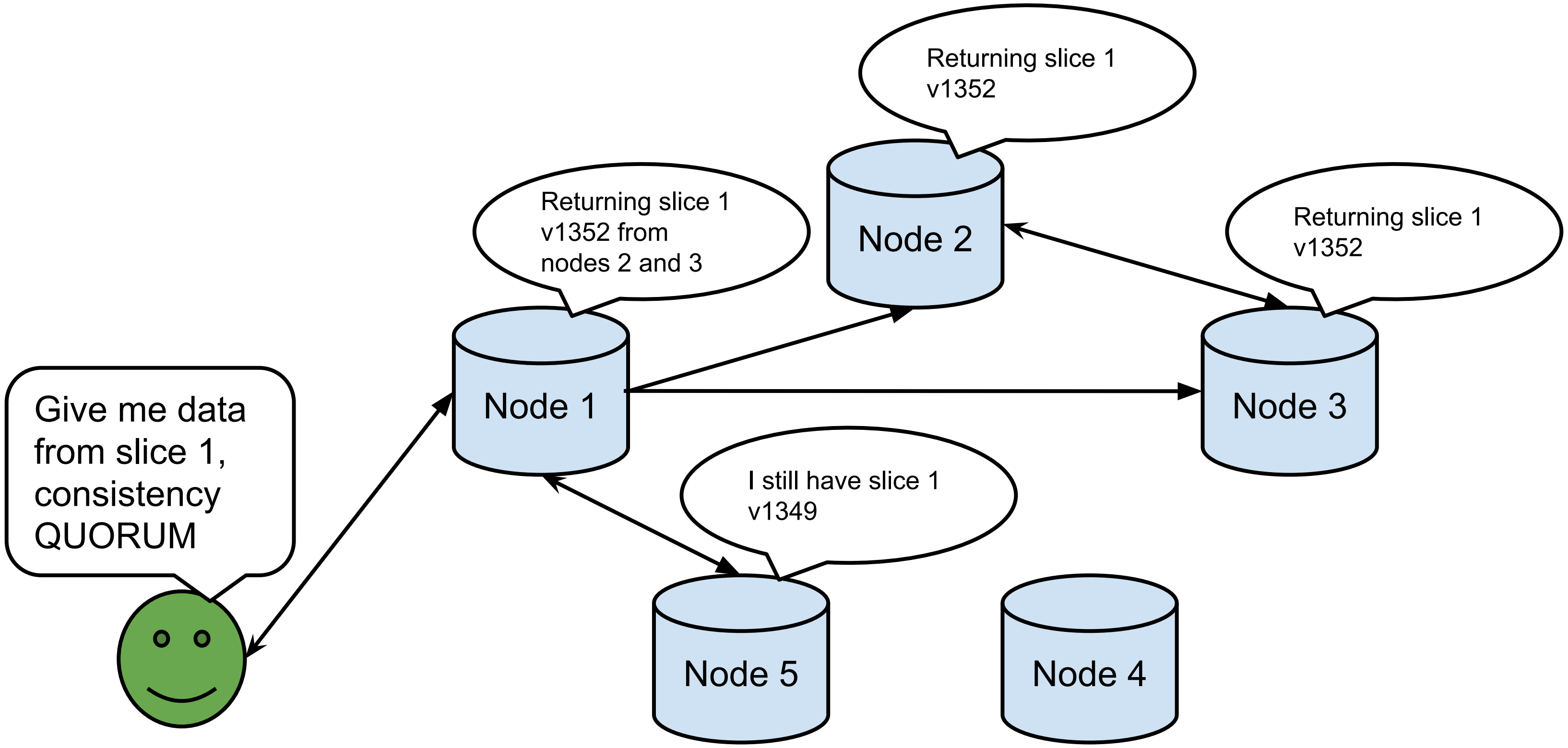
Querying with Consistency Level QUORUM

In the event of disagreement, client waits for gossip to update the cluster to agree


Updating slice 1 to v1352
Still waiting for agreement
Node 2
You should have slice 1 v1352
Give me data from slice 1, consistency QUORUM
Node 1
Node 3
I have slice 1 v1349
Node 5
Node 4

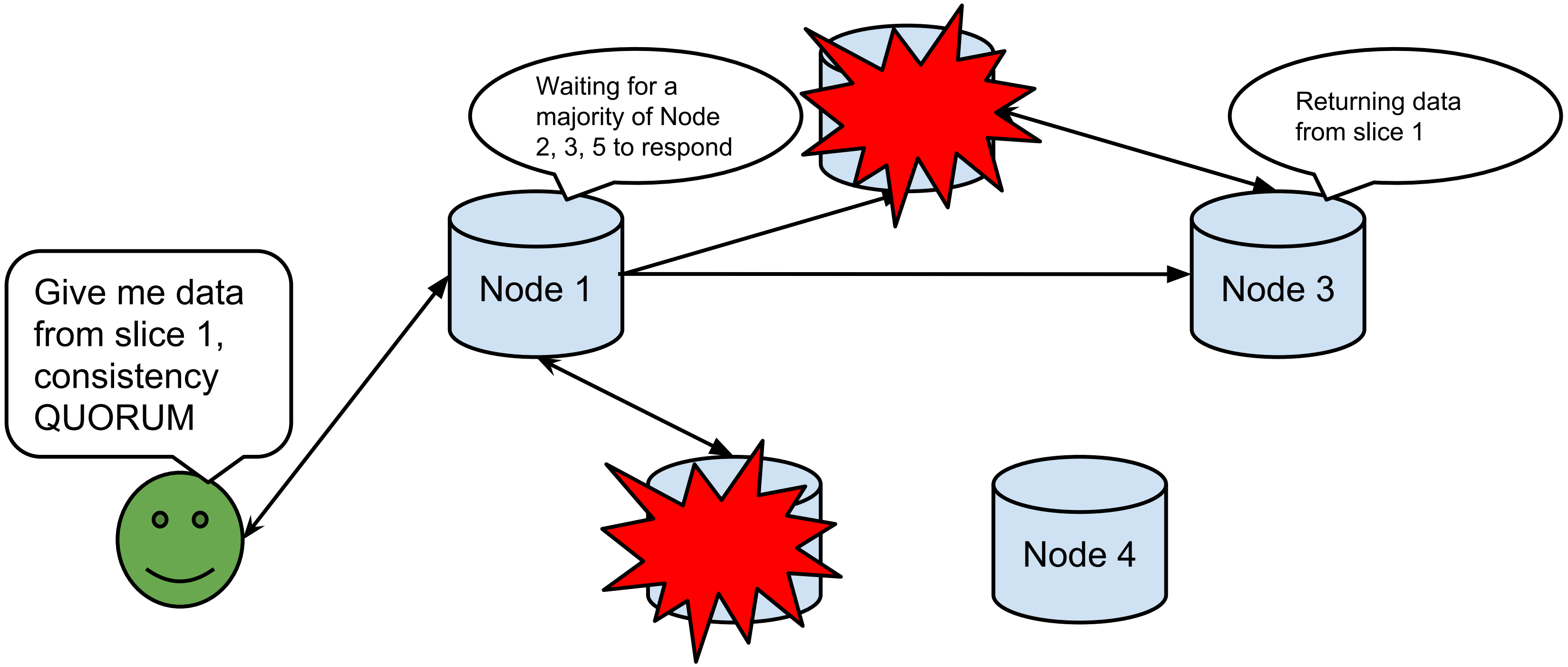
Querying with Consistency Level QUORUM

Once a majority of nodes owning data agree, the query can be satisfied

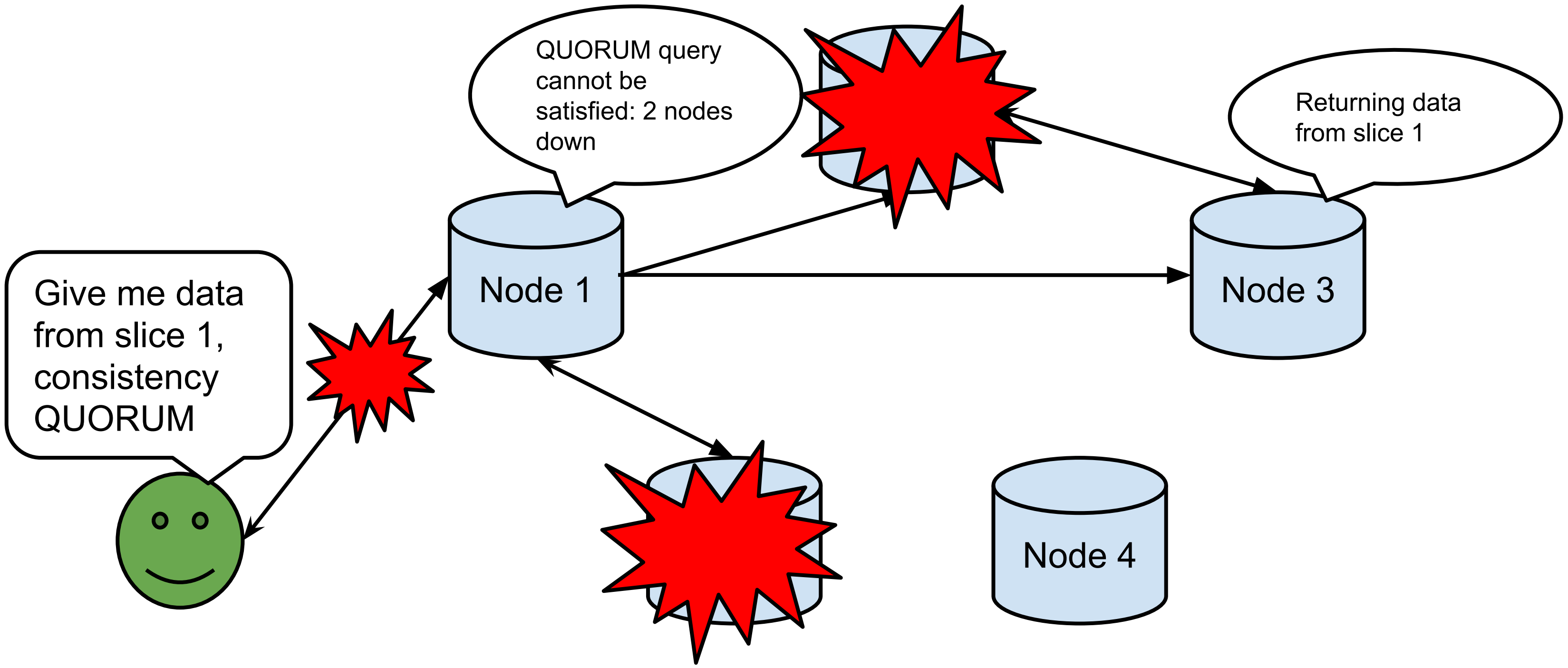

Give me data from slice 1, consistency QUORUM
Returning slice 1 v1352 from nodes 2 and 3
Returning slice 1 v1352
Returning slice 1 v1352
I still have slice 1 v1349
Node 1
Node 2
Node 3
Node 4
Node 5

Querying with Consistency Level QUORUM

If a consistency level cannot be satisfied, the query returns an error


Waiting for a majority of Node 2, 3, 5 to respond
Returning data from slice 1
Give me data from slice 1, consistency QUORUM
Node 1
Node 3
Node 4

If a consistency level cannot be satisfied, the query errors out


QUORUM query cannot be satisfied: 2 nodes down
Returning data from slice 1
Give me data from slice 1, consistency QUORUM
Node 1
Node 3
Node 4

Cassandra Consistency Levels

Queries make a tradeoff between availability and consistency

ONE: One node responds
QUORUM: A majority of nodes responsible for the data responds
LOCAL_QUORUM: Majority of nodes in datacenter agree
EACH_QUOURM: Majority of nodes in every datacenter agree
ALL: Every node responsible for the data responds

Cassandra Column Families

Cassandra stores data in Column Families

You can only make queries that match an index

In SQL-terms, every column is nullable.

No joins. Denormalize your data

```
CREATE COLUMNFAMILY IF NOT EXISTS sessions (
   session_key               text,
   session_data              text,
   PRIMARY KEY               (session_key)
)
```

CQL3 is a “SQL-like” language for executing queries

```
INSERT INTO api_cache
  (identity_provider, ddi, username, api_path, json,
   last_modified)
VALUES
  (%(identity_provider)s, %(ddi)s, %(username)s,
   %(api_path)s, %(json)s, %(last_modified)s)
USING TTL {ttl}
```

Why Cassandra is Great For Web Applications

Our Use Case - Rackspace Cloud Control Panel

Rackspace Cloud Control Panel: mycloud.rackspace.com

Manage your hosted Cloud Resources

Our main needs:

- uptime
- disaster recovery (handling loss of a datacenter)

We are deployed in three datacenters

Cloud Deployments

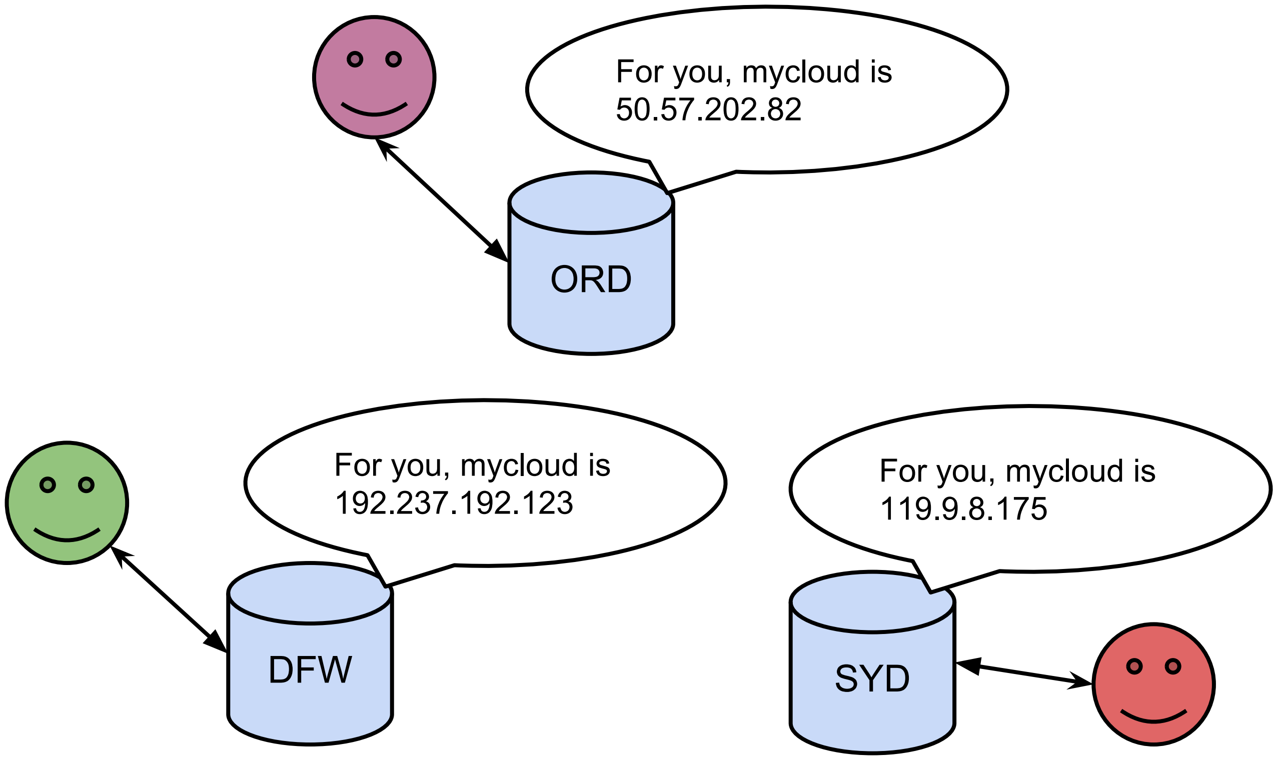
Deployed on Rackspace Public Cloud

Cloud deployments must assume failure as part of their architecture

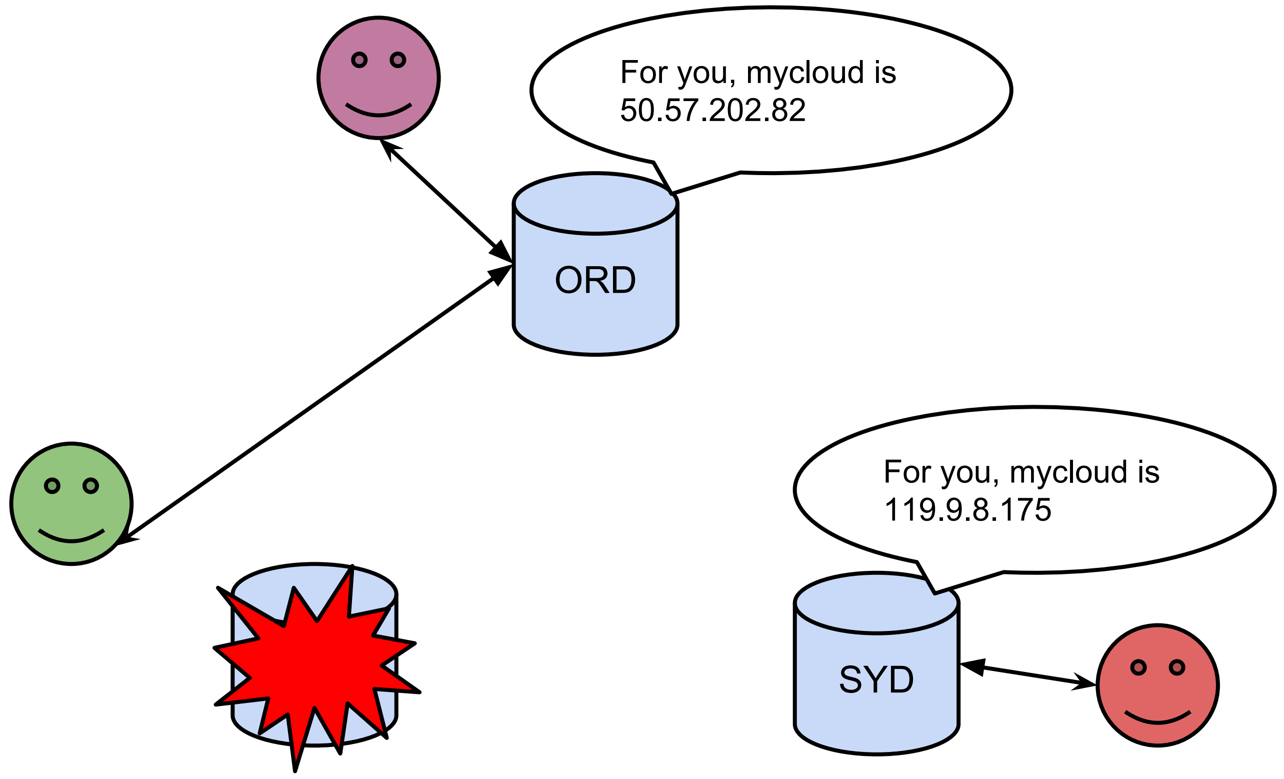
- EC2 Instance Retirement
- Scheduled network maintenances
- Datacenter/Availability Zone outages

DNS is our primary tool to handle datacenter outages

NS records delegate DNS control for mycloud.rackspace.com to F5 Global Traffic Managers


For you, mycloud is 50.57.202.82
ORD
For you, mycloud is 192.237.192.123
DFW
For you, mycloud is 119.9.8.175
SYD

On failure customers are directed to a different install


For you, mycloud is 50.57.202.82
ORD
For you, mycloud is 119.9.8.175
SYD

Datacenter Affinity

5 Cassandra nodes per data center

Because of DNS, clients “stick” to their datacenter

Client stickiness allows us to avoid inconsistency issues with Cassandra LOCAL_QUORUM

- If you write to our databases in Dallas, you immediately read from Dallas
- Data propagates to other datacenters in the background

Traditional Session Storage

In contrast, web applications that use traditional session storage...

Traditional Web Applications

1) No global session storage, users sticky to one server

- Cannot survive loss of individual server
- Cannot restart or upgrade server without downtime

Traditional Web Applications

2) Session storage in single-master SQL system

- Cannot reboot MySQL master without downtime
- Cannot take MySQL master out of rotation without manual master failover
- Cannot survive loss of a datacenter

3) Session storage in Multi-Master SQL

- Must manually restart replication if it fails
- Does not elegantly scale to *n* masters

Webapp Sessions in Cassandra + Delegated DNS <u>means</u>:

- Direct users to any datacenter without downtime
- Any datacenter can fail and your web application continues to function

Why Cassandra is better

Since switching to Cassandra:

- 100% uptime
- Many manually-induced datacenter failovers because of …
  - Cloud Networking issues
  - Per-DC nightly scheduled maintenance
  - Upgrading libraries
  - Upgrading Cassandra (yes)

Some of Our Pains with Cassandra

Pains with Cassandra: Cross-Ocean Install

Bug in Cassandra in high-latency situations meant some nodes refused to acknowledge other nodes as up

Hints built up

Repairs fail to complete, alerts go off

We upgraded to Cassandra 2.0.5

Pains with Cassandra: Driver Startup Time

We stop and recreate <u>every thread</u> on code deployment

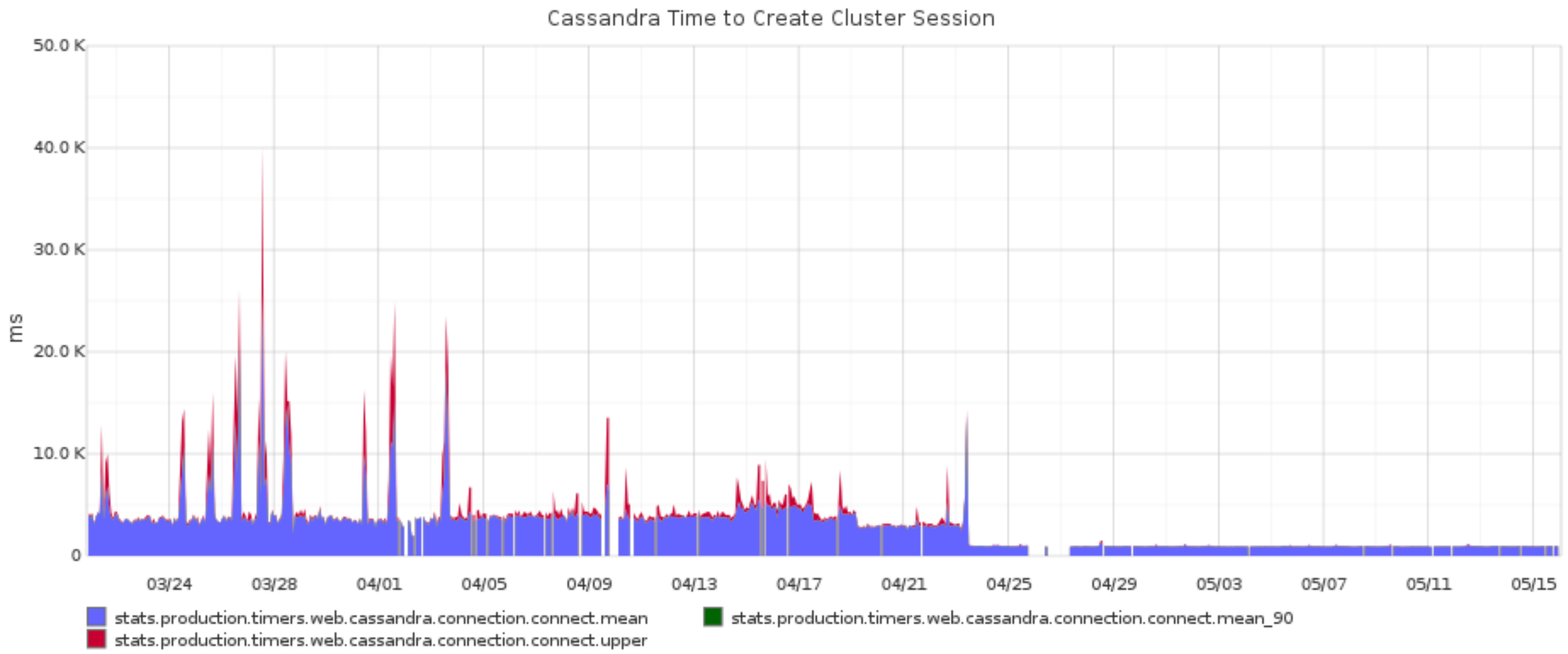
up to 8 code deployments a day

Bug in Cassandra caused this recreation to be <u>really slow</u>

e.g. 10+ seconds
e.g. health checks failed
e.g. alerts flapped

Pains with Cassandra: Driver Startup Time

Cassandra 2.0.7 (and working closely with a Cassandra contributor) saved us



Pains with Cassandra: Private Networks

Cassandra nodes gossip on public network

Queries connect on private network

Private network has different failure characteristics than public network

Cassandra uptime doesn’t save you if Cassandra can still gossip

Pains with Cassandra: Private Networks

We use a thread-based approach in our application

If a thread tries to talk over PrivateNet and fails enough it gets “poisoned”

Some threads healthy, some threads not

Health checks flap

Customers see errors, Datacenters fail over, etc.

Demo Using Web Application Sessions with Cassandra

Example: Sessions with Flask (Python Web Framework)

flask.sessions.SessionInterface

Requires you to implement

- open_session(app, request)
- save_session(app, session, response)

Quick Cassandra: https://github.com/nicolasff/docker-cassandra

Demo code is up on [Github]

Key queries:

```
SELECT * FROM sessions WHERE session_key = %s

DELETE FROM sessions WHERE session_key = %s

INSERT INTO sessions (session_key, session_data)
   VALUES (%s, %s)
   USING TTL %s
```

Final Thoughts

Final Thoughts

Cassandra is great

I would encourage all multi-DC applications that need "always on" guarantees to use it

Being able to fail datacenters over without downtime is *transformative* for a team

Demo Time With Docker