



# Agenda

- Cassandra Overview
- Key Cassandra Concepts
  - Consistent Hash Ring
  - Tunable Consistency
  - Eventual Consistency
  - Cassandra Data Model and Indexing
  - Disk Format
- When should you use Cassandra

## Cassandra Overview

- Distributed Key-Value store
- Big Table data model on top of Dynamo
- Handles large volumes of data, and high write load on commodity hardware
- Created by Facebook for Inbox Search
  - Facebook has a 150 node cluster, with 150 TB of data
- Open sourced on Google code July 2008
- Moved to Apache March 2009
- Latest release 0.8 June 2011
- Used by Facebook, Twitter, Digg and Netflix among others

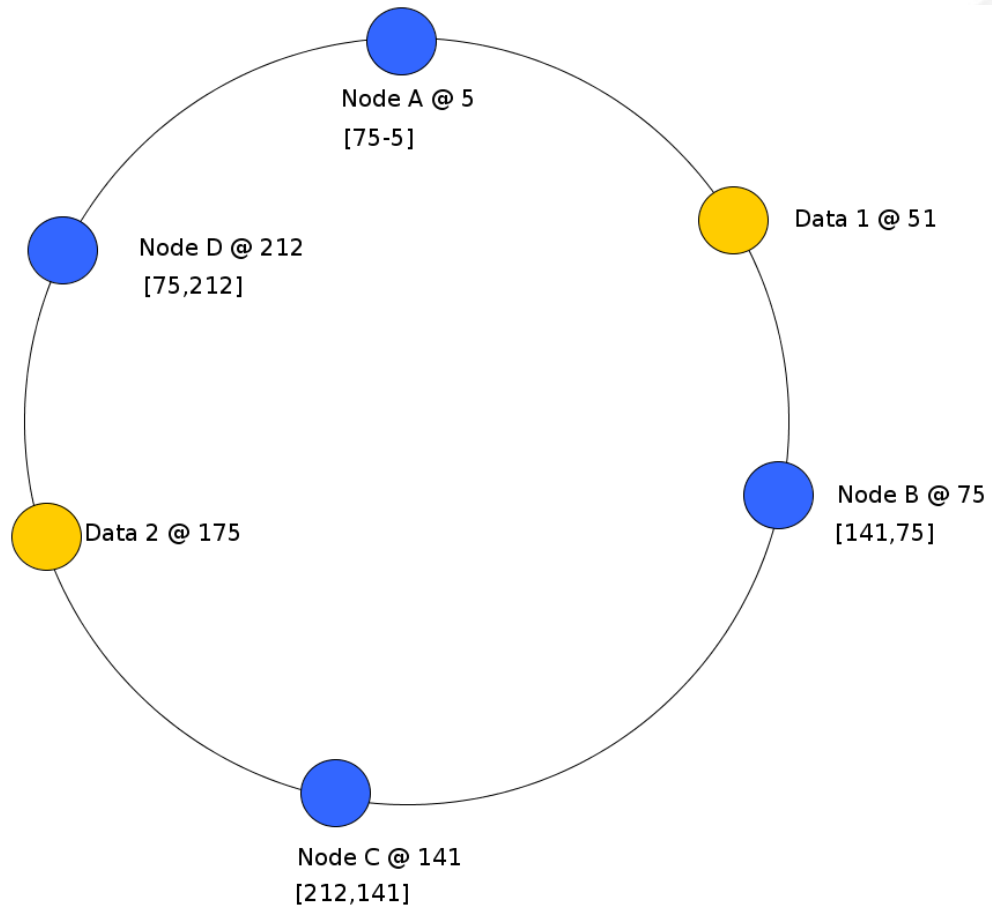
## Key Cassandra concepts

- Consistent Hashing
- Tunable Consistency
- Eventual Consistency
- Data Model

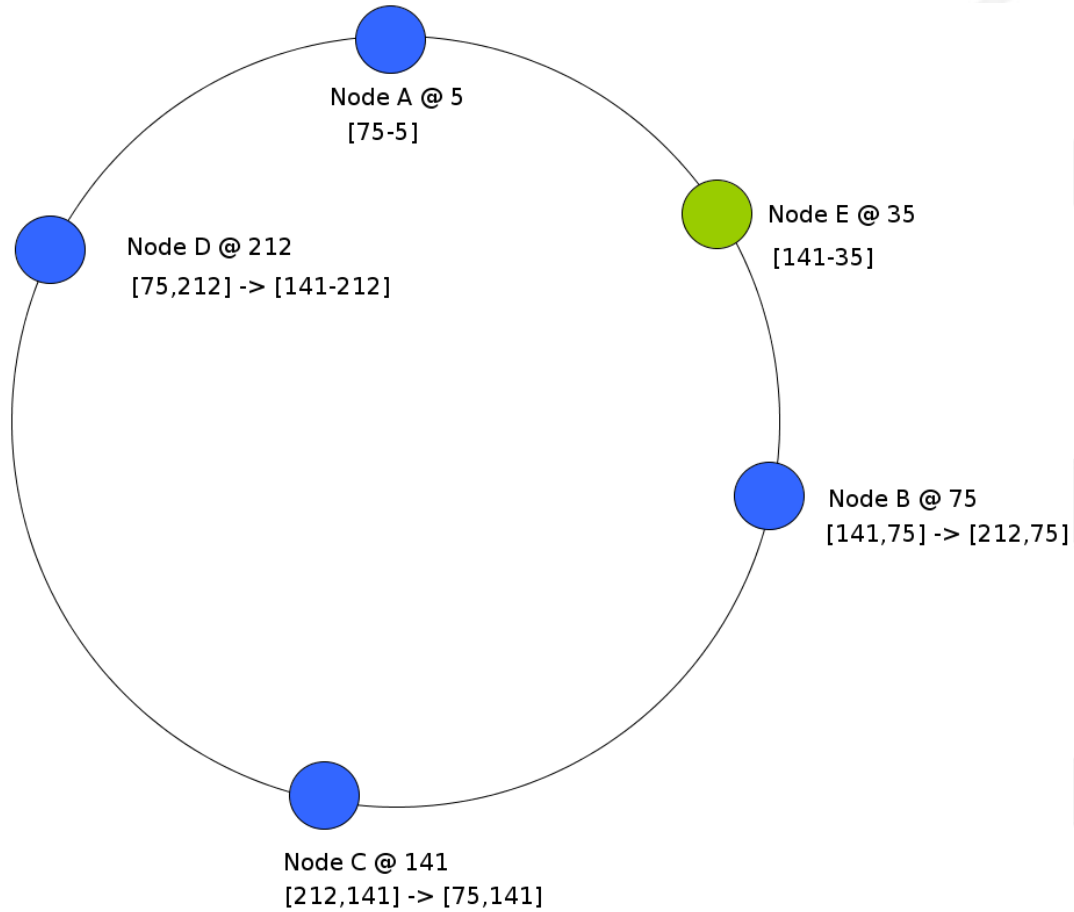
## Consistent Hash Ring

- How do you distribute data over a large number of machines?
- Big Table/HBASE – centrally track where data lives
  - Single Point of failure
  - Possible bottleneck
- N nodes, put data at  $\text{hash}(\text{key}) \% N$ 
  - Fast, Simple
  - No special node
  - When adding node N+1, you have to move most of your data
- Consistent Hashing
  - Simple
  - Adding node N+1 only requires moving a subset of data

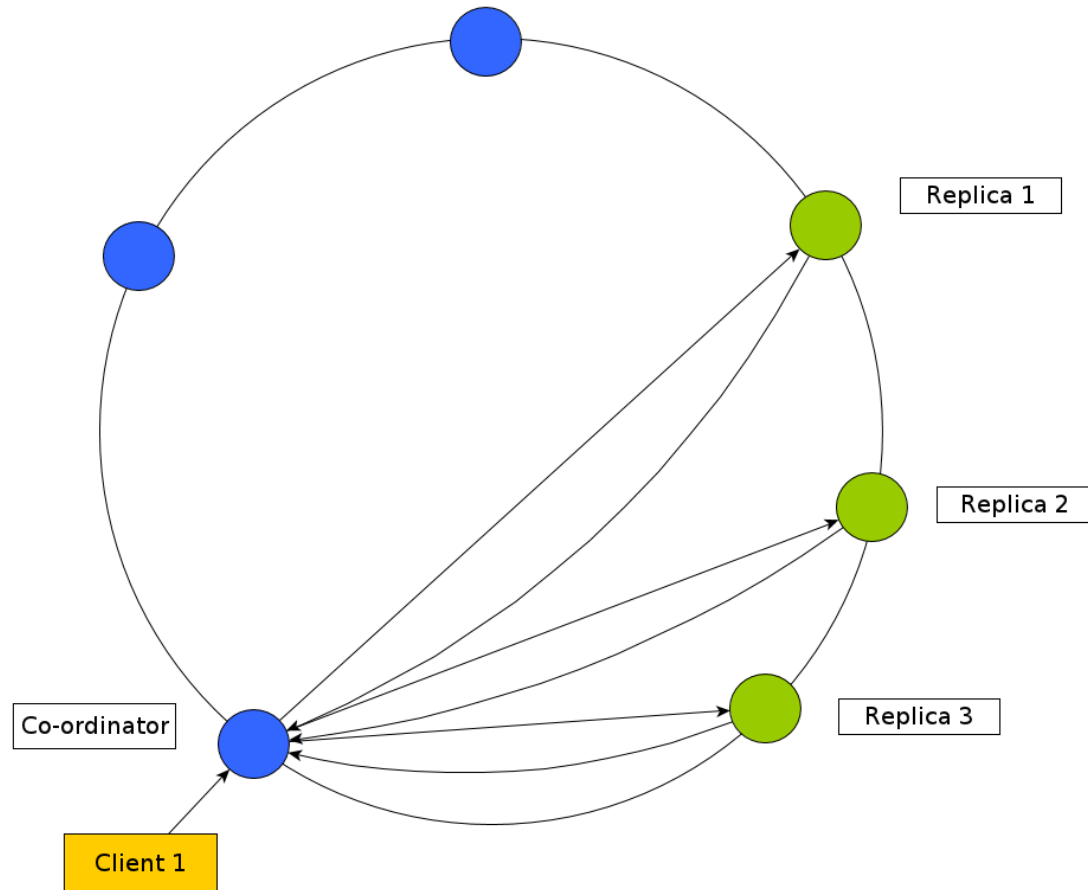
# Cassandra Ring



## Adding a new node



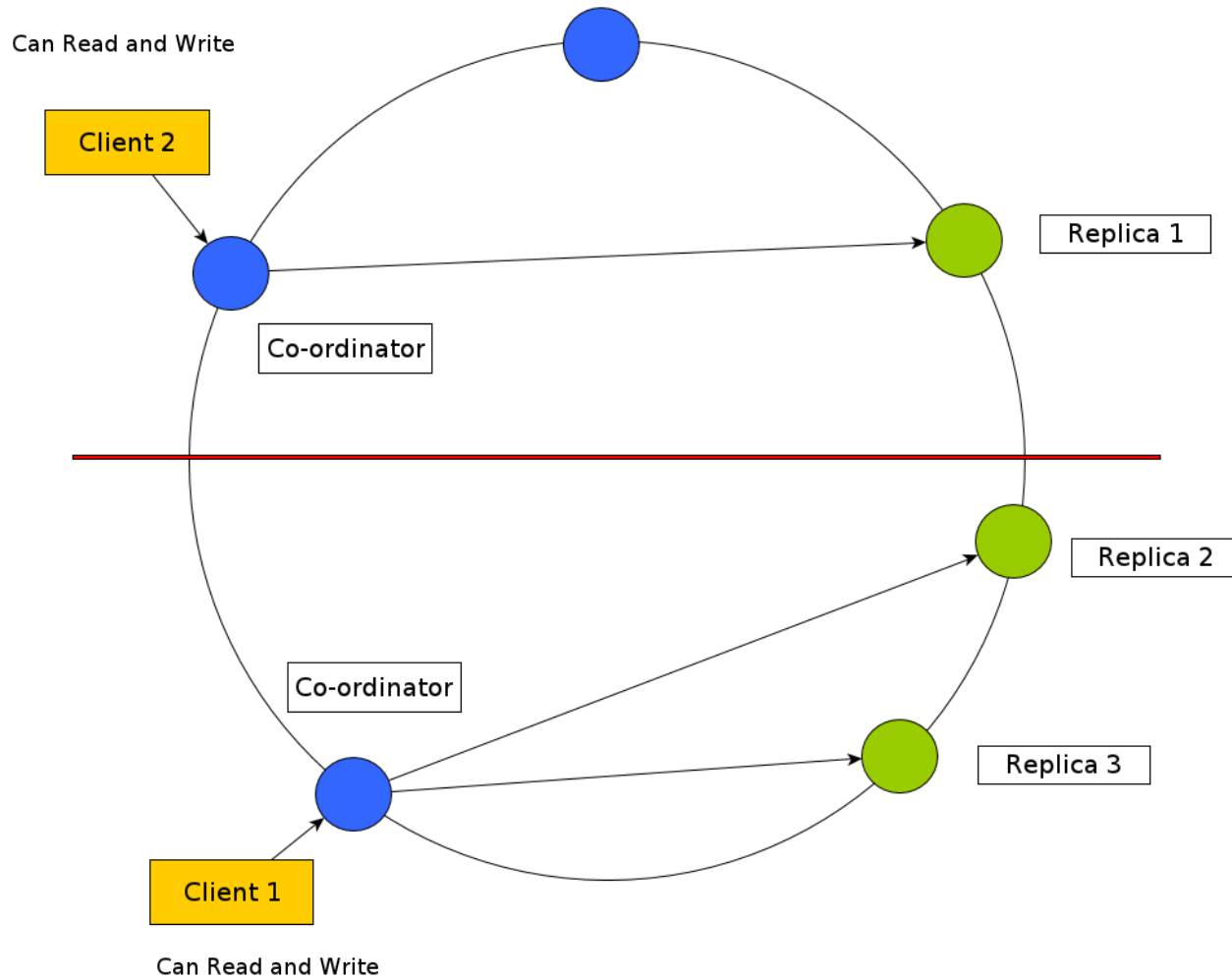
## Network Read/Write Path



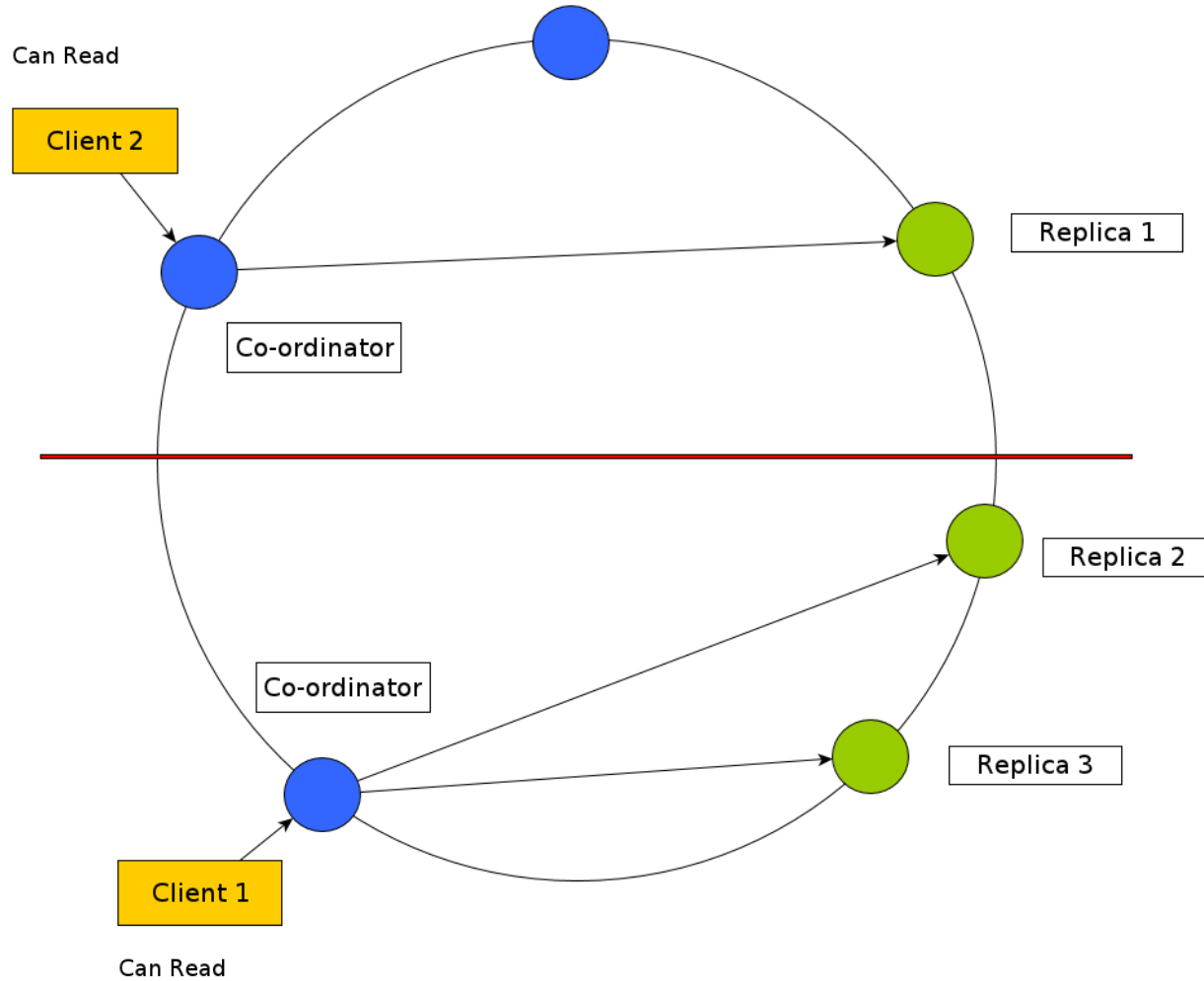
## Tunable Consistency

- Consistency level is the number of nodes that must be involved in a request
- When writing, consistency level means W nodes have persisted the write
- When reading, consistency level means that R nodes have been contacted, and the returned value is the latest
- Choices
  - ONE
  - QUORUM (majority)
  - ALL
- Consistent if  $R + W > N$

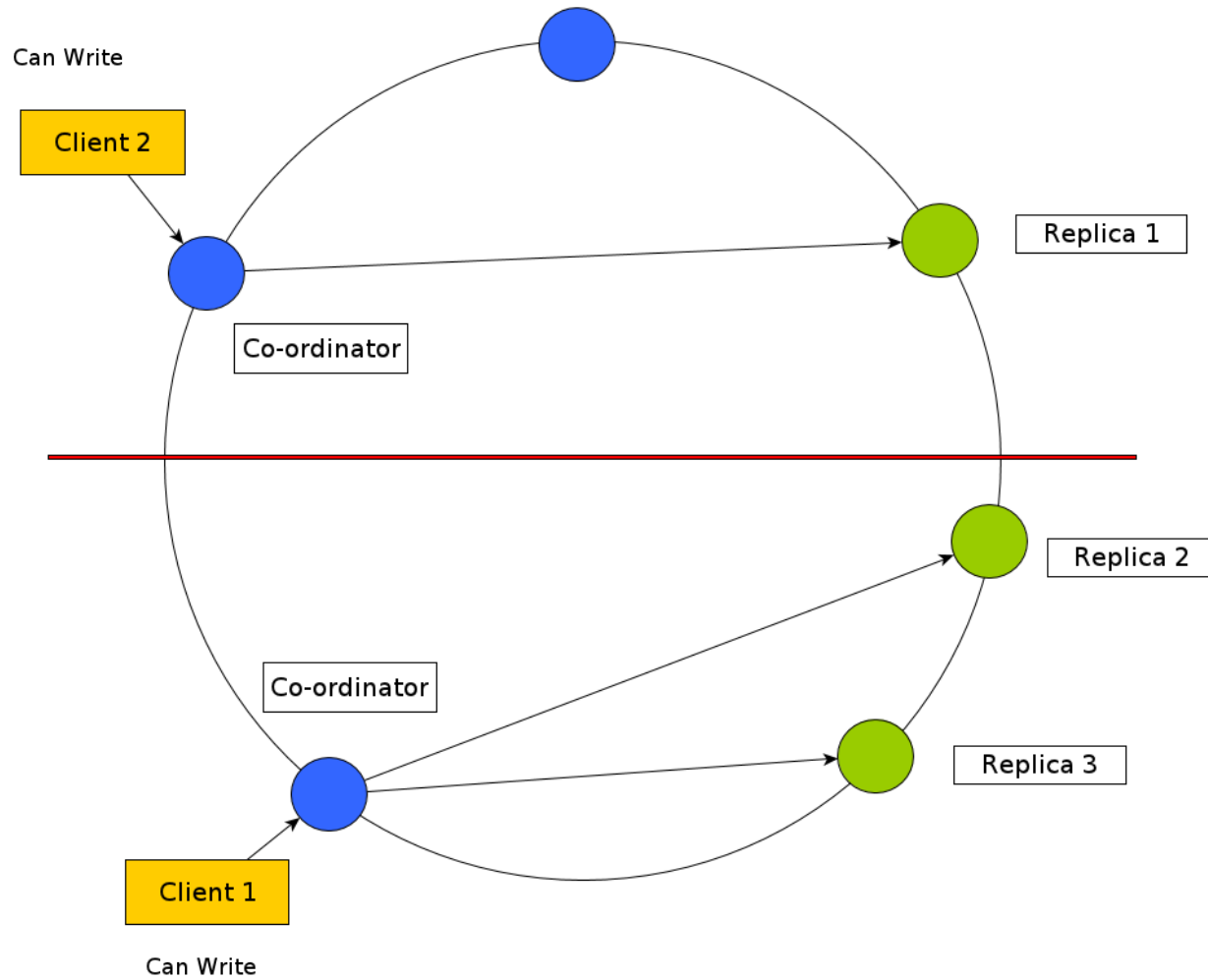
$N = 3, R = 1, W = 1, R + W < N$



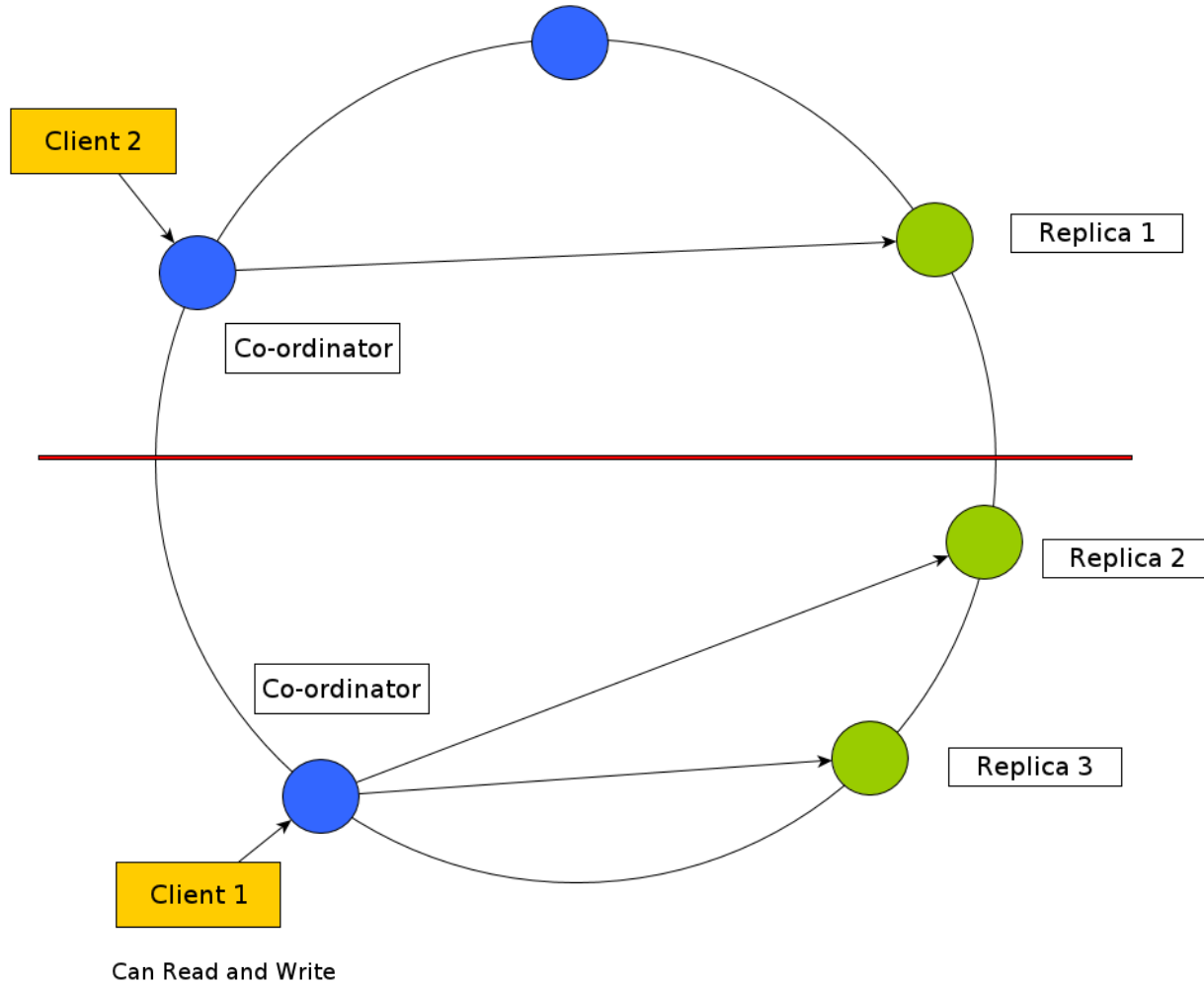
$N = 3, R = 1, W = 3, R + W > N$



$N = 3, R = 3, W = 1, R + W > N$



$N = 3, R = \text{Quorum}, W = \text{Quorum}, R + W > N$



## But what does consistency really mean?

- *"Consistency describes how and whether a system is left in a consistent state after an operation. In distributed data systems like Cassandra, this usually means that once a writer has written, all readers will see that write." \**
- Write at Quorum and the write fails? Reader reads at Quorum, What will the reader read?
  - You don't know
  - The reader may read the write, may not. The write may show up tomorrow or next week
  - No consistency guarantees when writes fail

\* <http://wiki.apache.org/cassandra/ArchitectureOverview>

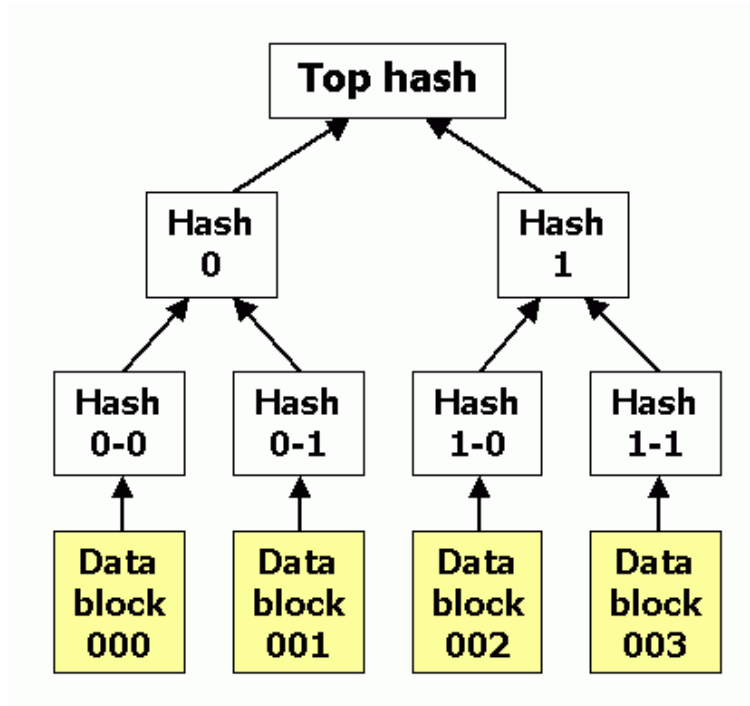
# Eventual Consistency

How does Cassandra ensure consistency in the presence of failures?

- Client Retries
  - All writes to Cassandra are idempotent, so client libraries will generally retry on failure
- Hinted Handoff
  - If a write cannot go to the right node, store a hint to send the write when the node goes online
- Read Repair
  - If during a read the returned data is not consistent, send the latest data to the inconsistent node
- Node Repair

## Node Repair

- A node creates a merkel tree and compares it with its neighbors, sending differences between them
- Expensive, involves compacting and reading all data on a node



## Cassandra Data Model (Ignoring Super columns\*)

Cassandra	DB Equivalent
Key Space	Schema
Column Family	Table
Key	Row
Column	Column

\*[wtf is a super column](#)

## Cassandra Data Model Example

```
#In KeySpace Hockey
"scores" : { #Column Family
  "game-3681" : { #key
    "away"      : "Vancouver" #Column
    "date"      : "2010-10-28" #Column
    "home"      : "Colorado" #Column
    "score"     : "1-2" #Column
    "shootout"  : "true" #Column
  }
  "game-4314" : { #key
    "away"      : "Toronto" #Column
    "date"      : "2010-10-13" #Column
    "home"      : "Vancouver" #Column
    "score"     : "5-2" #Column
  }
  ...
}
```

## Getting your data out

- You can only quickly look up data by key
- When you can only read by key, all indexes are columns on a key
- For the hockey schema, how do you look up games by date, games by team, shootout wins?
  - Denormalize, and maintain your own indexes

## Games by date

```
"games-date-index" : { #Column Family

  "2010-10-20" : { #Key
    "game-3681" : "" #Column
    "game-3689" : "" #Column
    "game-3675" : "" #Column
  }

  "2010-10-21" : { #Key
    "game-3878" : "" #Column
    "game-3912" : "" #Column
  }
}
```

Index query only returns the id. If you want more, lookup individual keys, or push data into the index.

## Games by team

```
"games-by-team" : { #Column Family
  "Vancouver" : { #Key
    "1970-8-14-game-581" : "" #Column
    ...
    "2010-10-13-game-4313" : "" #Column
    "2010-10-15-game-4398" : "" #Column
    "2010-10-19-game-4425" : "" #Column
    ...
  }
  "Toronto" : #Key
  ...
}
```

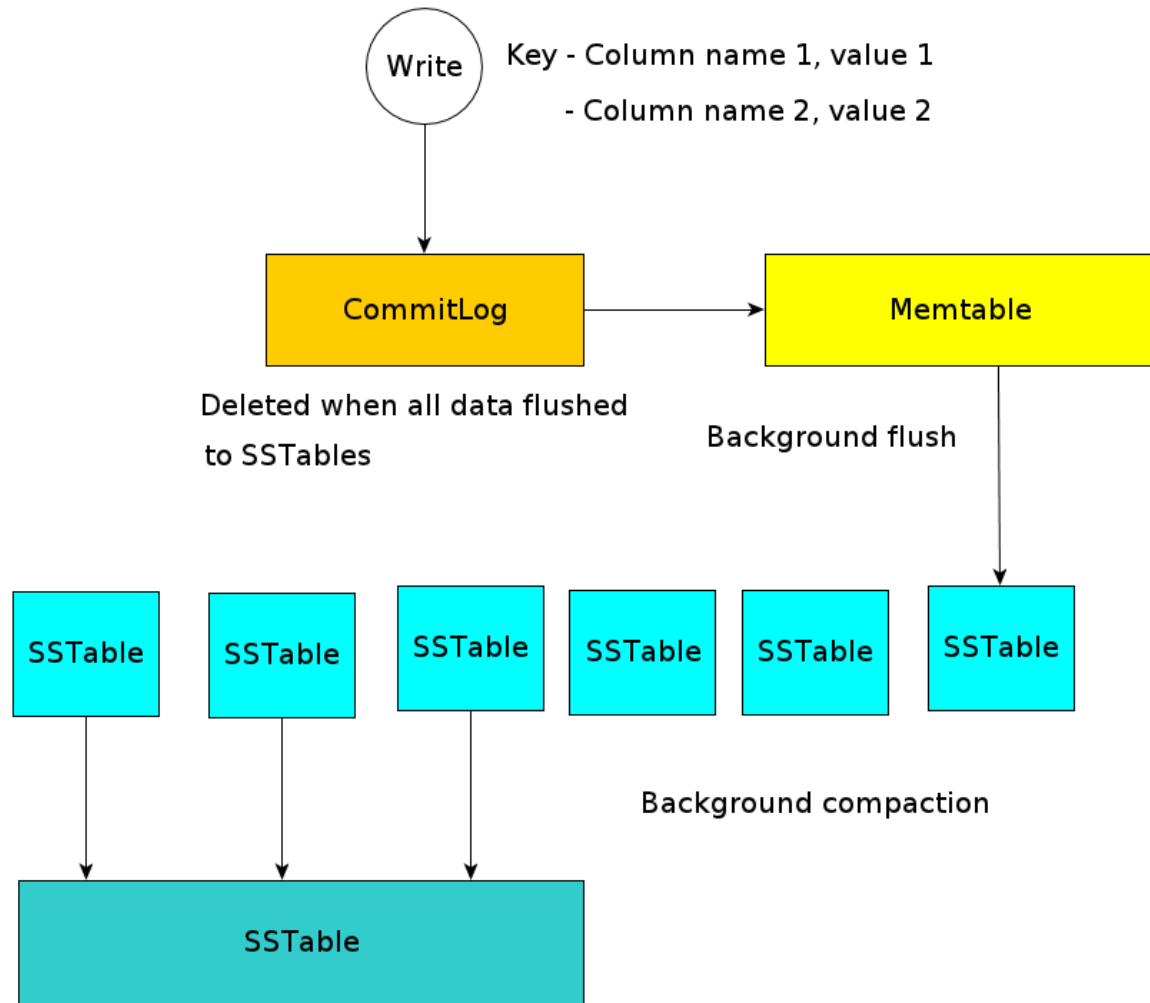
When you have a new query, run a map reduce job to create a new index

## Twitter Raincloud – Stat service built on Cassandra

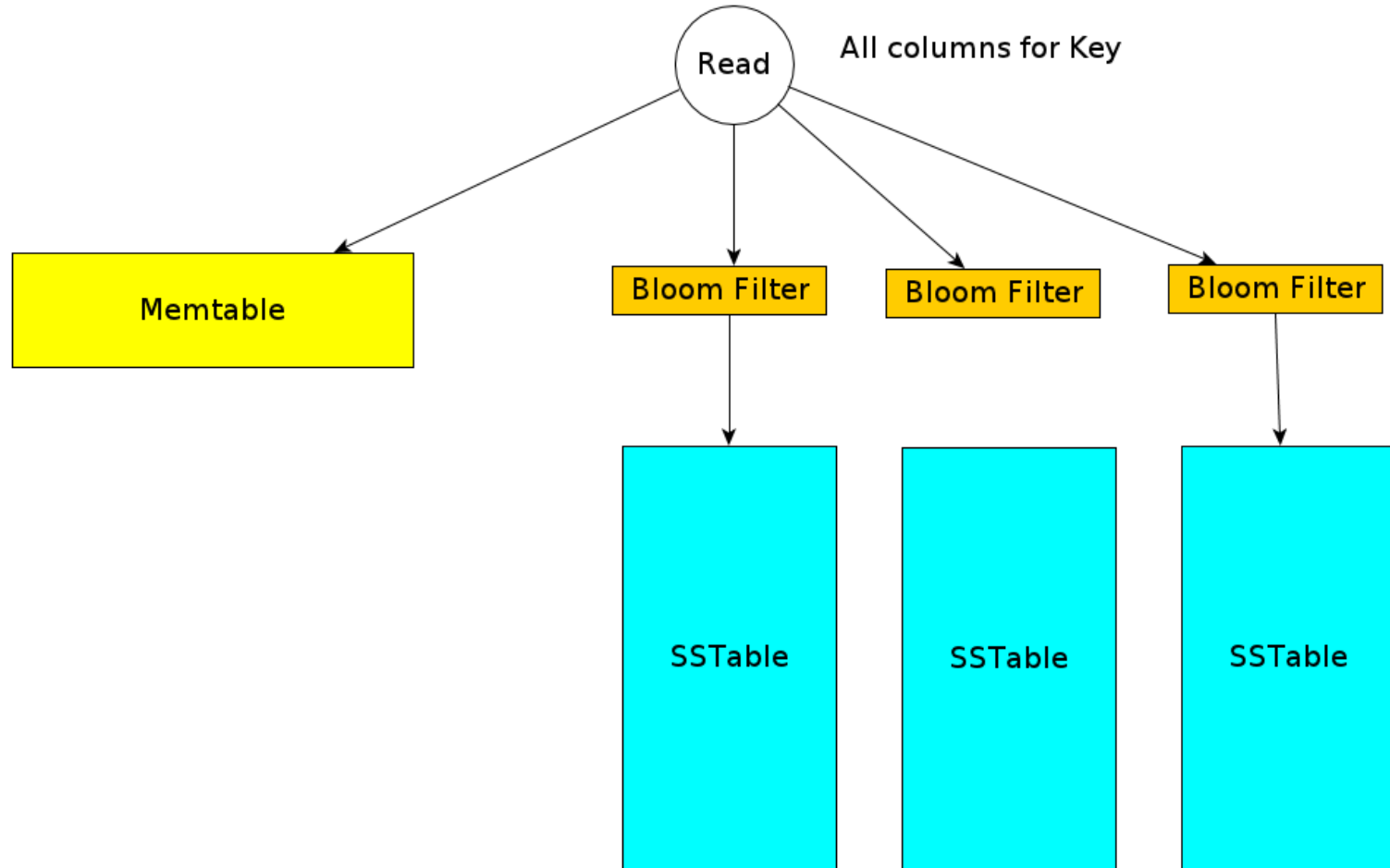
1 click on com.example.blog/foo results in 16 updates:

- com (all time)
- com.example (all time)
- com.example.blog (all time)
- com.example.blog /foo (all time)
- com (1st Feb 2011)
- com.example (1st Feb 2011)
- com.example.blog (1st Feb 2011)
- com.example.blog /foo (1st Feb 2011)
- com (11am-12 on 1st Feb)
- com.example (11am-12 on 1st Feb)
- com.example.blog (11am-12 on 1st Feb)
- com.example.blog /foo (11am-12 on 1st Feb)
- com (11:41-42 on 1st Feb)
- com.example (11:41-42 on 1st Feb)
- com.example.blog (11:41-42 on 1st Feb)
- com.example.blog /foo (11:41-42 on 1st Feb)

# Write Path

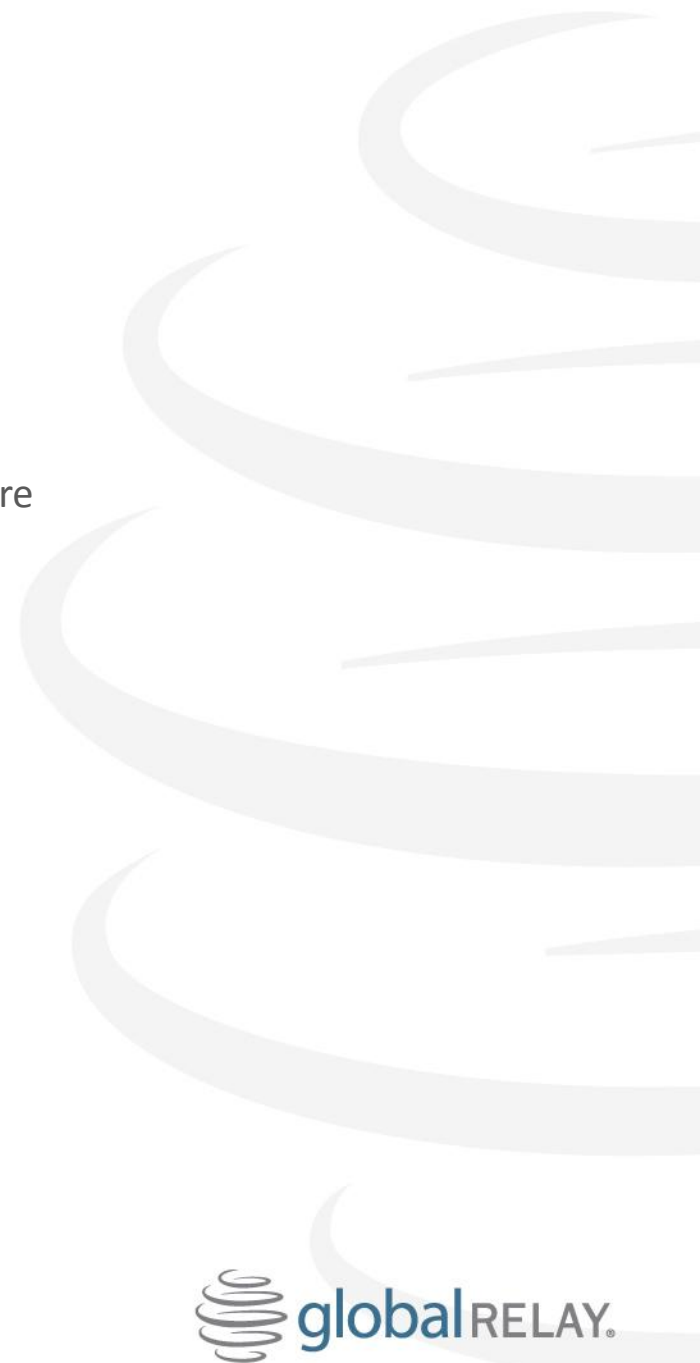


# Read Path



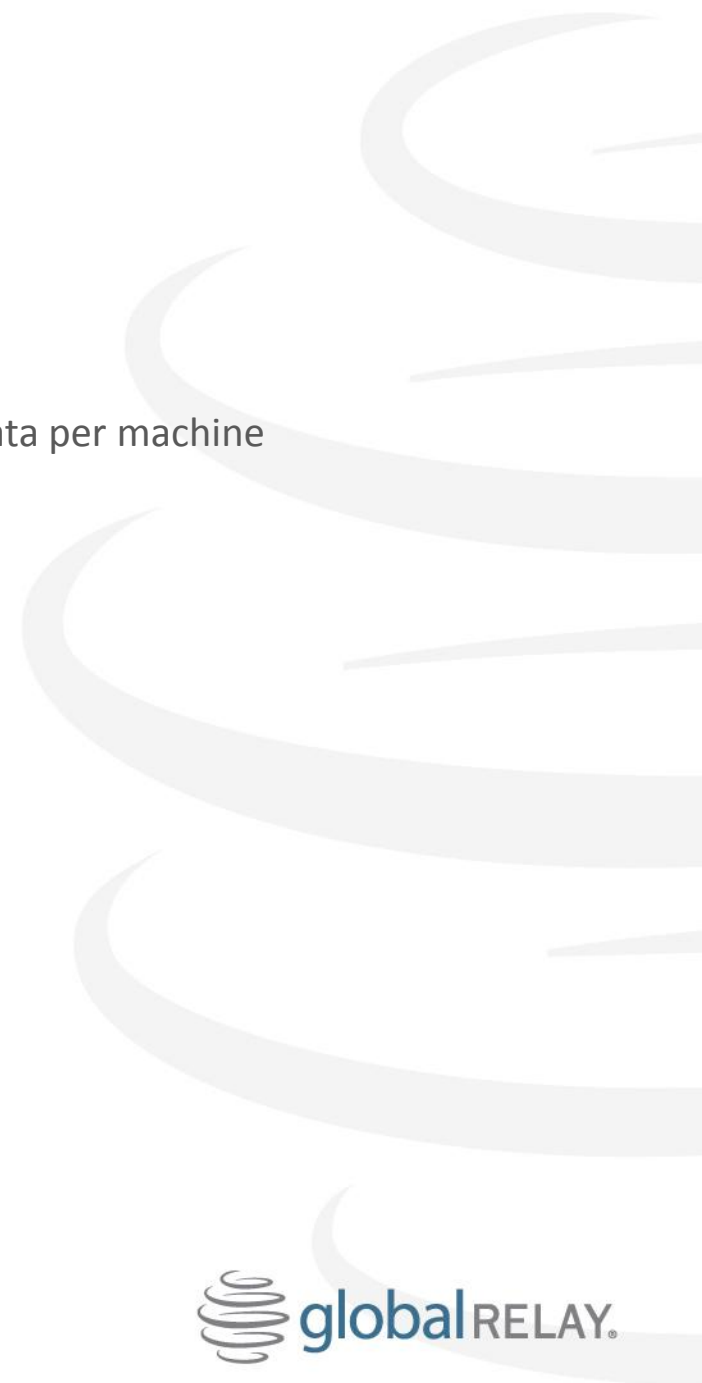
## When to use Cassandra

- Your data doesn't fit on a single machine
- You need high reliability
- You need a Key-Value store
- You need something slightly better than a Key-Value store
- You don't need to perform ad hoc queries
- Very high write load



## Potential Problems with Cassandra

- Immaturity (compared to databases)
- Compaction can destroy read performance
- Cassandra becomes difficult to manage with > 1TB of data per machine



## What I didn't mention

- Counters
- Secondary Indexes
- Order Preserving Partitioner
- Cross Data Center replication
- Super Columns
- TTL columns
- Map Reduce



## We're hiring

- Sr. Java Developers
- Java Developer (Load/Performance Testing)
- Experienced .NET/C# Developer
- JavaScript GUI Developer
- Int./Sr. QA Analysts
- Desktop Support
- System Administrator

Full list of jobs at [globalrelay.com](http://globalrelay.com)

My email : [sean.bridges@gmail.com](mailto:sean.bridges@gmail.com)

