Cassandra:
A Decentralized Structured Storage System

Presented by Arman Sepehr

August 2017

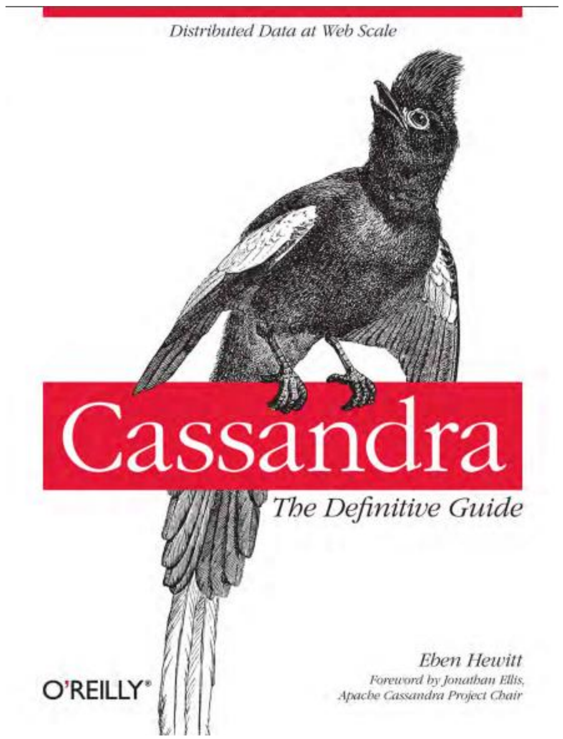
Goal

- What type of datastore is this?
- What was the driving force?
- Which Genre is proper to our application?
- Of course, columnar database
- Cassandra
  - How does it work?
  - How does it distinguish?
  - Benefit?
  - ….

Agenda

- NoSQL and types
- Scale up in Relational DBMS
- What is Cassandra?
- Who’s using Cassandra?
- Cassandra Architecture
- What’s the next phase?

References


Distributed Data at Web Scale

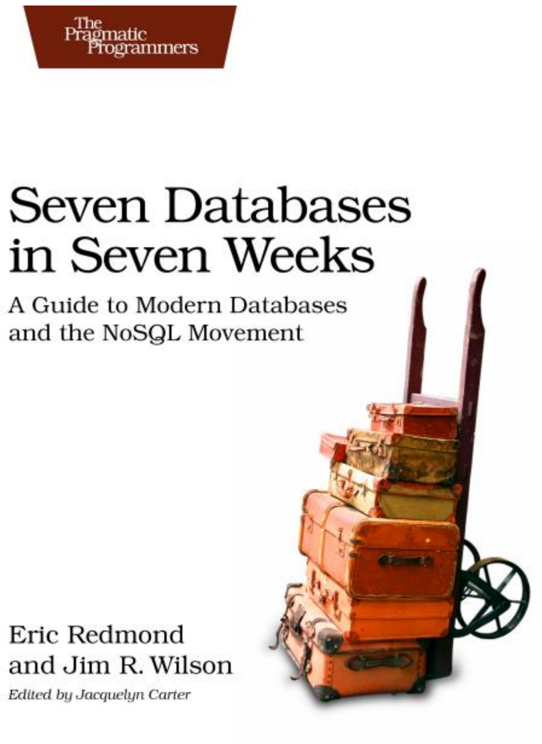
Cassandra

The Definitive Guide

O'REILLY®

Eben Hewitt

Foreword by Jonathan Ellis, Apache Cassandra Project Chair


The Pragmatic Programmers

Seven Databases in Seven Weeks

A Guide to Modern Databases and the NoSQL Movement

Eric Redmond and Jim R. Wilson

Edited by Jacquelyn Carter

DATASTAX

cassandra

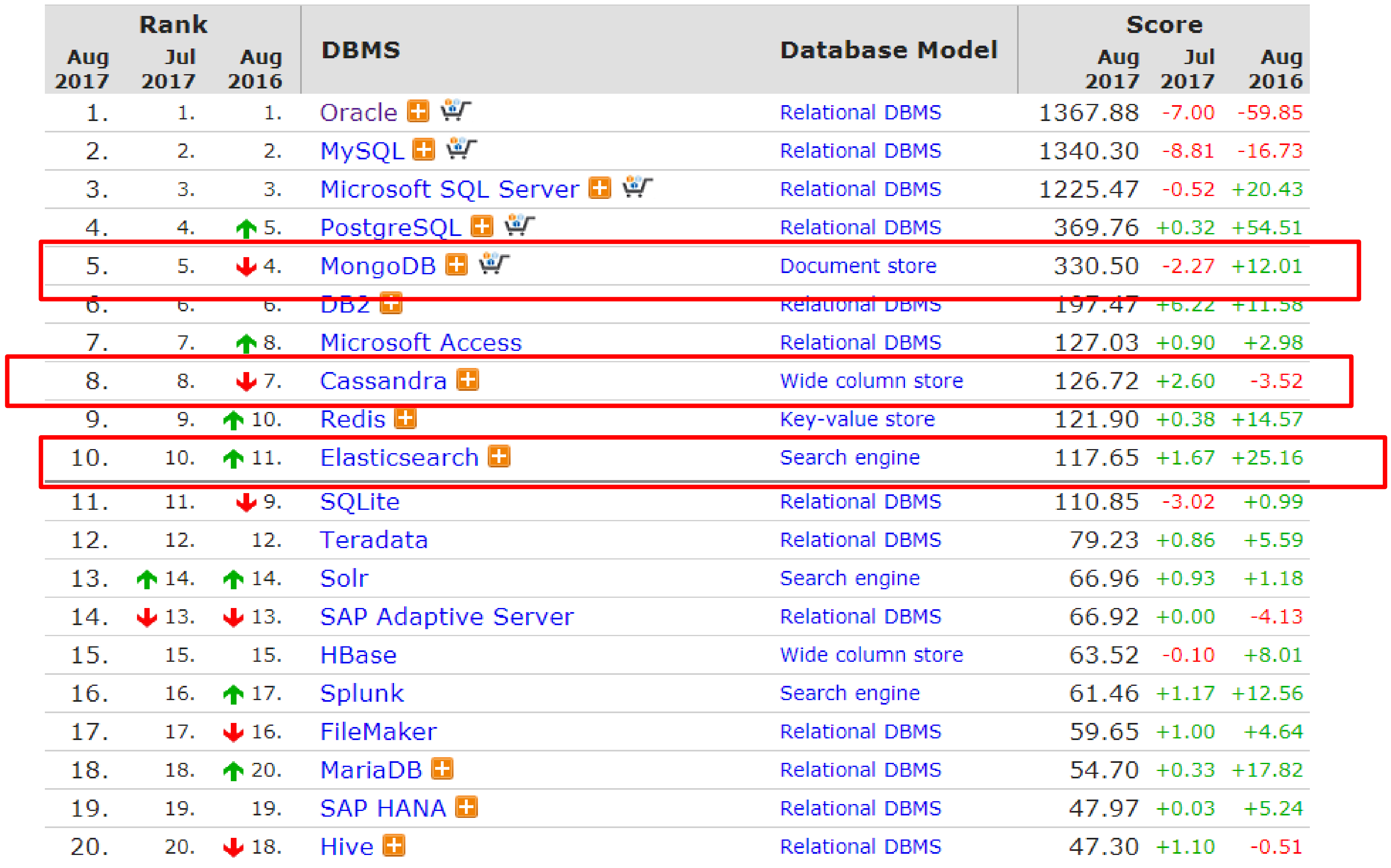
Big Picture

Big Picture: Genre

- Like music, databases can be broadly classified into one or more styles.
  - An individual song may share all of the same notes with other songs

- Genre
  - Relational database management systems(RDBMS), or row oriented
  - Key-Value Store
    - some KV implementations permits complex data types value such as lists or hashes
    - File system as KV store
  - Columnar, or column oriented
    - Between KV store and relation data base
  - Document Store: in short, a document is like a hash, with unique ID
  - Graph: Highly interconnected data
    - Do BFS, DFS is beneficial for Neo4j, How could you do in RDBMS or others?

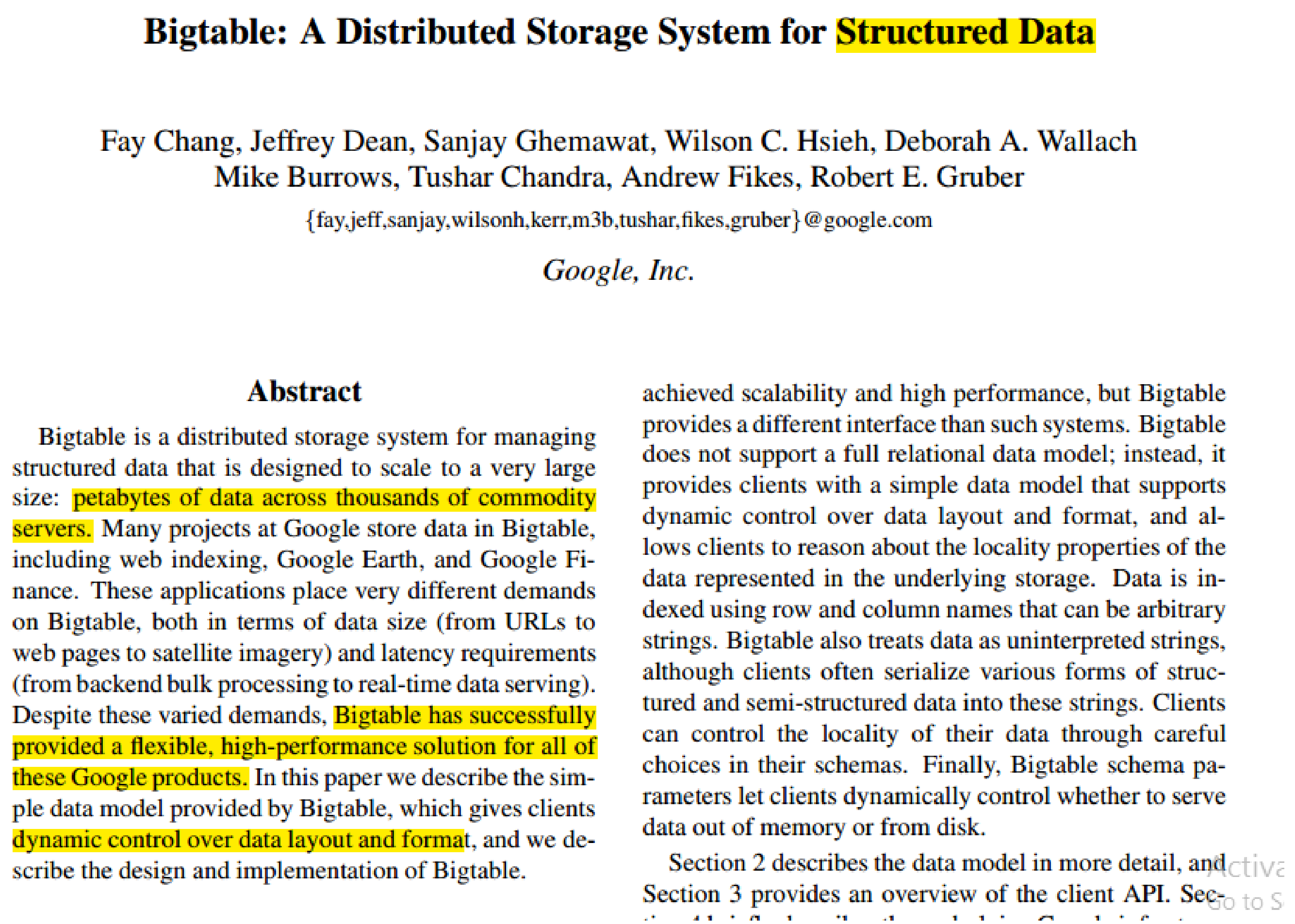
Big Picture: Market Share



Rank			DBMS	Database Model	Score		
Aug 2017	Jul 2017	Aug 2016			Aug 2017	Jul 2017	Aug 2016
1.	1.	1.	Oracle	Relational DBMS	1367.88	-7.00	-59.85
2.	2.	2.	MySQL	Relational DBMS	1340.30	-8.81	-16.73
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1225.47	-0.52	+20.43
4.	4.	↑ 5.	PostgreSQL	Relational DBMS	369.76	+0.32	+54.51
5.	5.	↓ 4.	MongoDB	Document store	330.50	-2.27	+12.01
6.	6.	6.	DB2	Relational DBMS	197.47	+6.22	+11.58
7.	7.	↑ 8.	Microsoft Access	Relational DBMS	127.03	+0.90	+2.98
8.	8.	↓ 7.	Cassandra	Wide column store	126.72	+2.60	-3.52
9.	9.	↑ 10.	Redis	Key-value store	121.90	+0.38	+14.57
10.	10.	↑ 11.	Elasticsearch	Search engine	117.65	+1.67	+25.16
11.	11.	↓ 9.	SQLite	Relational DBMS	110.85	-3.02	+0.99
12.	12.	12.	Teradata	Relational DBMS	79.23	+0.86	+5.59
13.	↑ 14.	↑ 14.	Solr	Search engine	66.96	+0.93	+1.18
14.	↓ 13.	↓ 13.	SAP Adaptive Server	Relational DBMS	66.92	+0.00	-4.13
15.	15.	15.	HBase	Wide column store	63.52	-0.10	+8.01
16.	16.	↑ 17.	Splunk	Search engine	61.46	+1.17	+12.56
17.	17.	↓ 16.	FileMaker	Relational DBMS	59.65	+1.00	+4.64
18.	18.	↑ 20.	MariaDB	Relational DBMS	54.70	+0.33	+17.82
19.	19.	19.	SAP HANA	Relational DBMS	47.97	+0.03	+5.24
20.	20.	↓ 18.	Hive	Relational DBMS	47.30	+1.10	-0.51

db-engines.com @ August 2017

Big Picture: What's Big-table?



Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber
{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com
Google, Inc.

Abstract

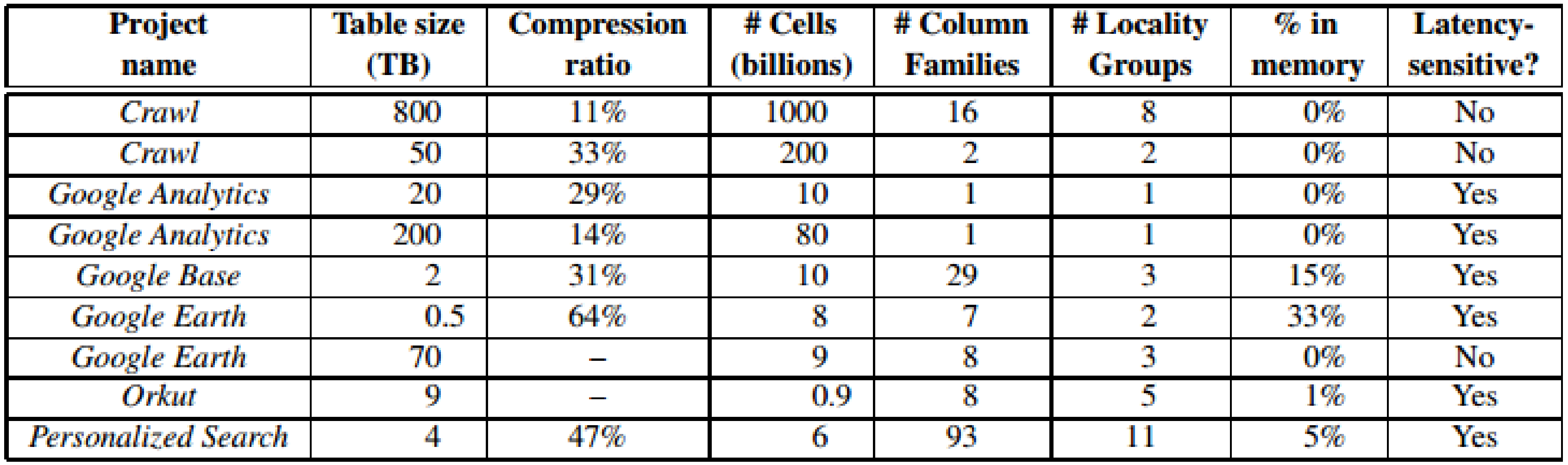
Bigtable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. Many projects at Google store data in Bigtable, including web indexing, Google Earth, and Google Finance. These applications place very different demands on Bigtable, both in terms of data size (from URLs to web pages to satellite imagery) and latency requirements (from backend bulk processing to real-time data serving). Despite these varied demands, Bigtable has successfully provided a flexible, high-performance solution for all of these Google products. In this paper we describe the simple data model provided by Bigtable, which gives clients dynamic control over data layout and format, and we describe the design and implementation of Bigtable.

achieved scalability and high performance, but Bigtable provides a different interface than such systems. Bigtable does not support a full relational data model; instead, it provides clients with a simple data model that supports dynamic control over data layout and format, and allows clients to reason about the locality properties of the data represented in the underlying storage. Data is indexed using row and column names that can be arbitrary strings. Bigtable also treats data as uninterpreted strings, although clients often serialize various forms of structured and semi-structured data into these strings. Clients can control the locality of their data through careful choices in their schemas. Finally, Bigtable schema parameters let clients dynamically control whether to serve data out of memory or from disk.

Section 2 describes the data model in more detail, and Section 3 provides an overview of the client API. Sec-

Chang, Fay, et al. "Bigtable: A distributed storage system for structured data." *ACM Transactions on Computer Systems (TOCS)* 26.2 (2008): 4.

Big Picture: What's Big-table?



Project name	Table size (TB)	Compression ratio	# Cells (billions)	# Column Families	# Locality Groups	% in memory	Latency-sensitive?
Crawl	800	11%	1000	16	8	0%	No
Crawl	50	33%	200	2	2	0%	No
Google Analytics	20	29%	10	1	1	0%	Yes
Google Analytics	200	14%	80	1	1	0%	Yes
Google Base	2	31%	10	29	3	15%	Yes
Google Earth	0.5	64%	8	7	2	33%	Yes
Google Earth	70	–	9	8	3	0%	No
Orkut	9	–	0.9	8	5	1%	Yes
Personalized Search	4	47%	6	93	11	5%	Yes

Table 2: Characteristics of a few tables in production use. *Table size* (measured before compression) and *# Cells* indicate approximate sizes. *Compression ratio* is not given for tables that have compression disabled.

Chang, Fay, et al. "Bigtable: A distributed storage system for structured data." *ACM Transactions on Computer Systems (TOCS)* 26.2 (2008): 4.

Big Picture: What's Next?

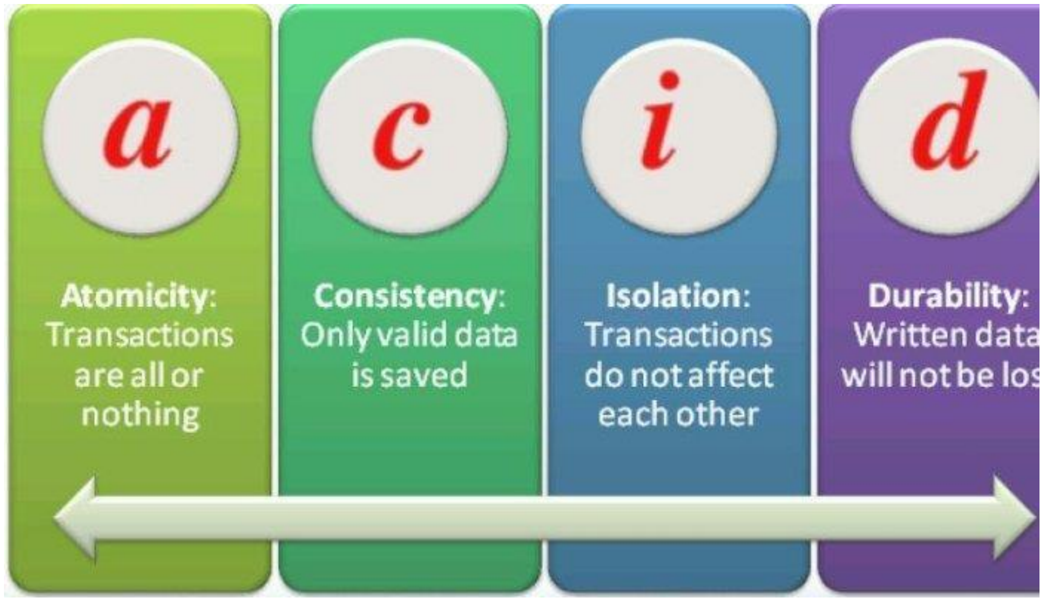
- We got what's NoSQL Genre
- We got different types for NoSQL
- Big Questions?
  - NoSQL is transient topic! Gently, scale up in RDBMS
  - What's our dream database?
  - And, Cassandra…

Relational DBMS: Small and Medium Data

- **XML, SQLite for Small Data**
- **Since 1970**
- **Fits on 1 machine**
- **RDBMS is fine**
  - **Postgres**
  - **Mysql**
- **Supports hundreds of concurrent users**
- **Scale Vertically**
- **Excellent for applications such as management (accounting, reservations, staff management, etc)**

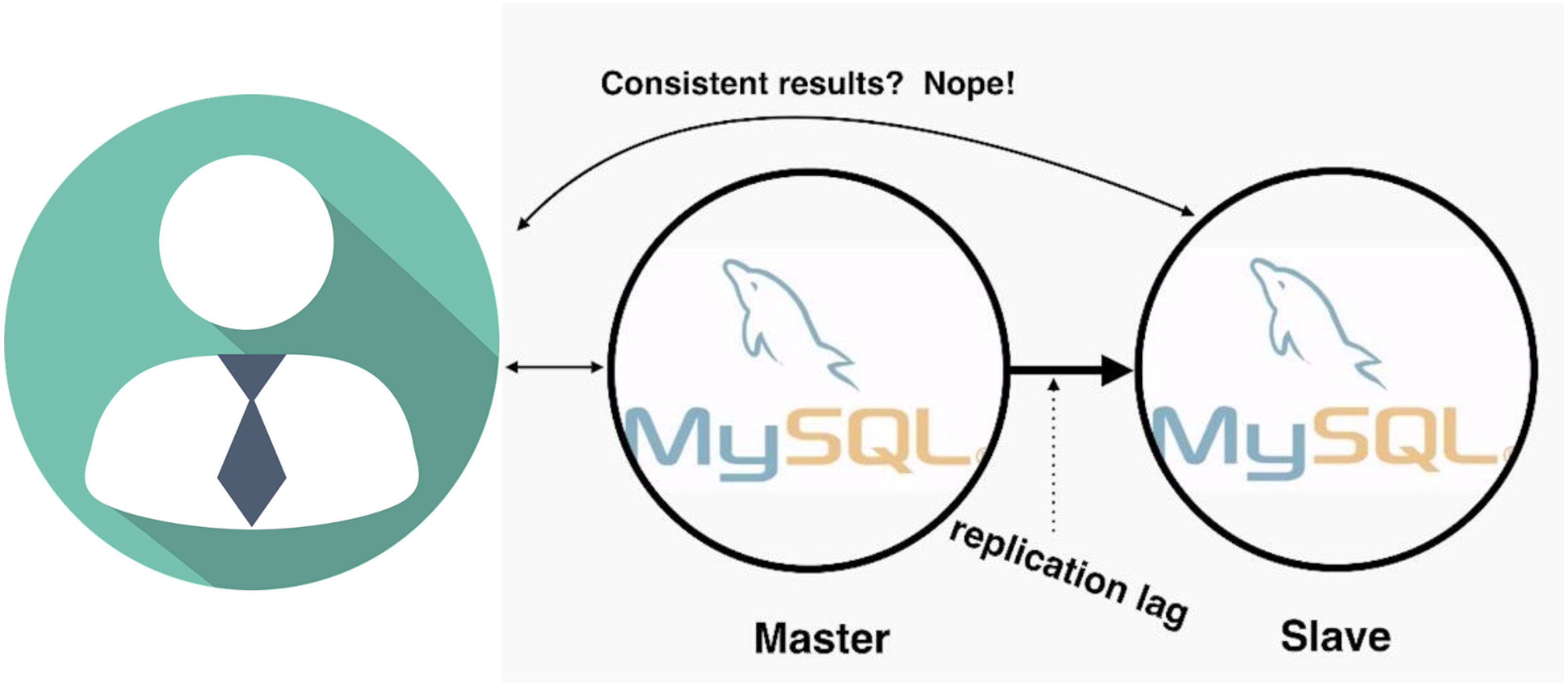
Relational DBMS

- ACID makes us feel good
- Schemas aren't designed for sparse data
- <u>Databases are simply not designed to be distributed,</u> Really?


a
c
i
d
Atomicity: Transactions are all or nothing
Consistency: Only valid data is saved
Isolation: Transactions do not affect each other
Durability: Written data will not be los

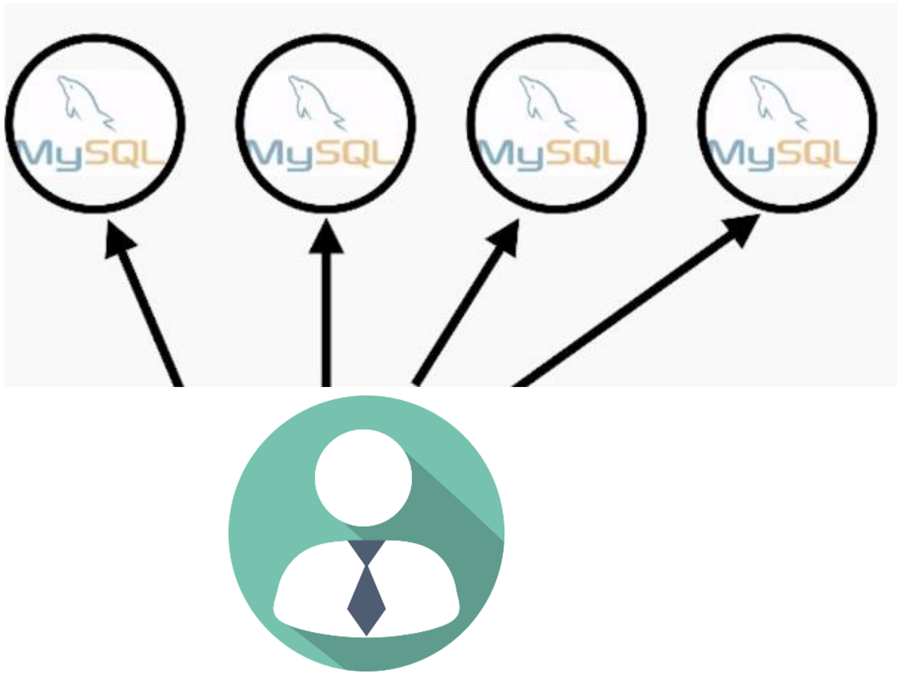
Can RDBMS work for Big data?

- **Master / Slave : ACID is lie**



Can RDBMS work for Big data?

- Sharding is actually a Nightmare
  - Data is all over the place
  - No more joins
  - Schema changes over all nodes manually



  - Adding shards requires manually moving data

Summary of Failure: Just Say Yes to NoSQL

- **Scaling is a pain**
  - **Vertically or Horizontally?**

- **ACID is naïve at best**
  - **You aren't consistent**
- **Re-sharding is a manual process**
- **We're going to de-normalize for performance**
- **High availability is complicated, requires additional operational overhead**

Lessons Learned: Dream DB

- **Consistency is not practical**

  So we git it up.
- **Manual sharding and rebalancing is hard**

  So let's build in automatically
- **Every moving part makes systems more complex**

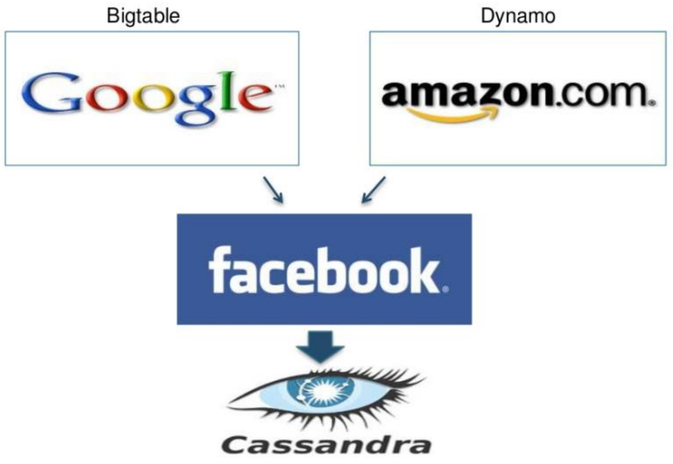
  So simplify out architecture – no more master/slave
- **Scaling up is expensive**

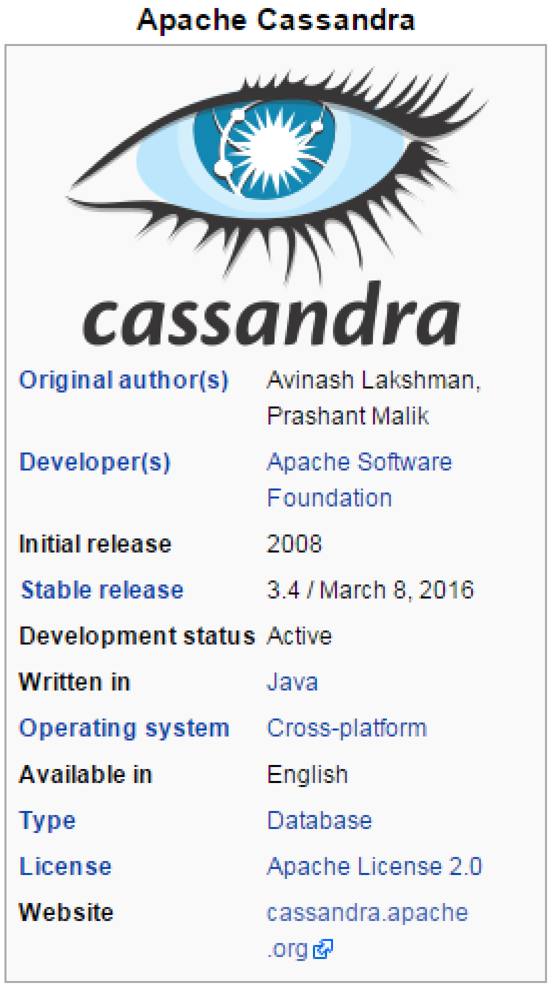
  Scale in horizon – we want commodity hardware
- **Scatter / gather no good**

  Goal is to always hit 1 machine

What's Cassandra?

- Apache Cassandra was initially developed at Facebook to power their Inbox Search
- Originally designed at Facebook, Cassandra came from Amazon's highly available Dynamo and Google's BigTable data model



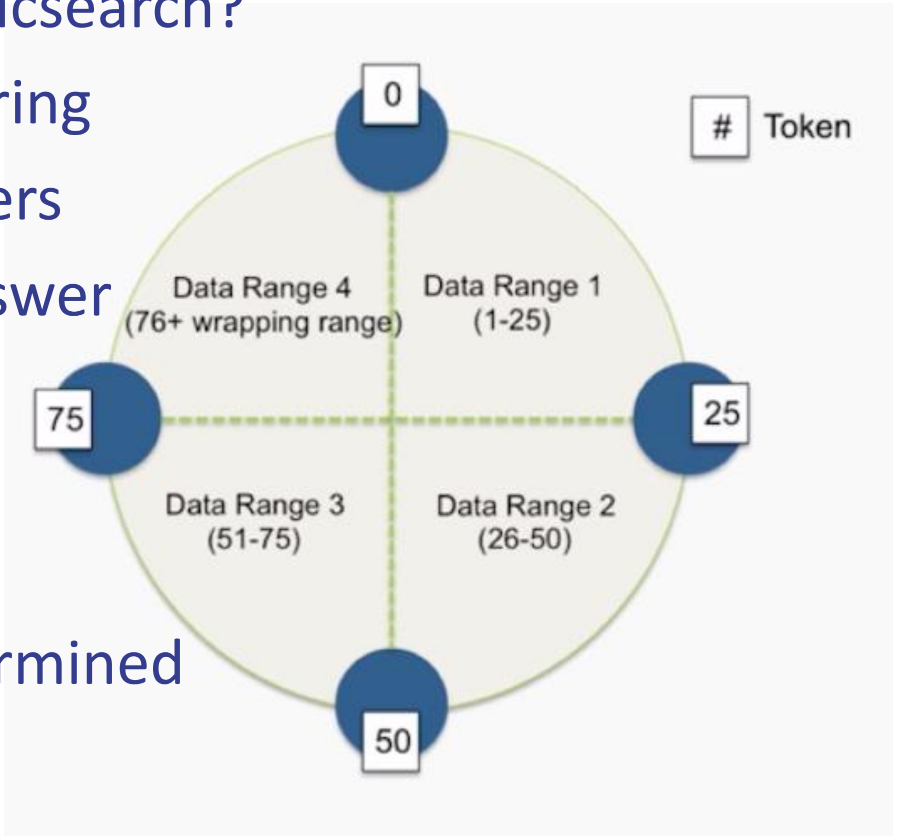

Apache Cassandra

Original author(s)	Avinash Lakshman, Prashant Malik
Developer(s)	Apache Software Foundation
Initial release	2008
Stable release	3.4 / March 8, 2016
Development status	Active
Written in	Java
Operating system	Cross-platform
Available in	English
Type	Database
License	Apache License 2.0
Website	cassandra.apache.org

Think Conceptually, What's Cassandra?

- No master/slave
- No config servers, zookeepers

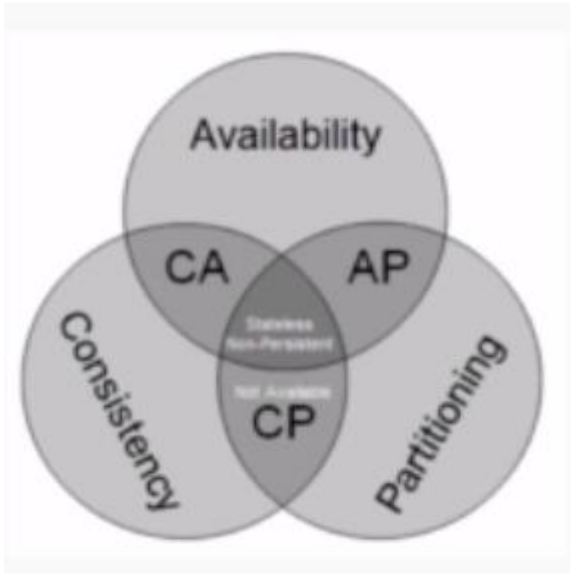
  what about hbase or elasticsearch?
- Data is partitioned around the ring
- Data is replicated to RF=N servers
- All nodes hold data and can answer queries (both reads and writes)



- Location of data on ring is determined by partition key

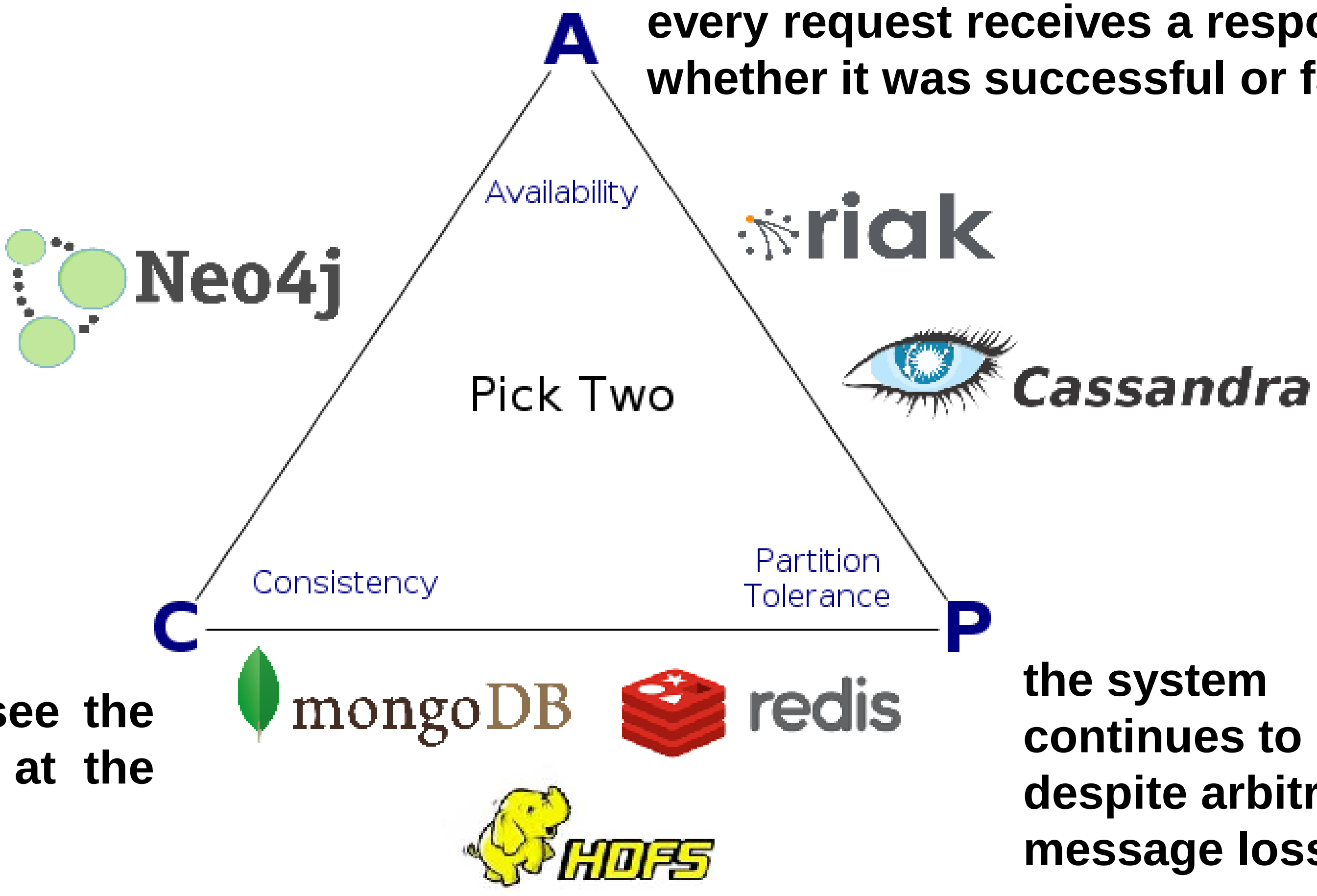
Theoretical Aspect: CAP Theory

- Impossible to be both consistent and highly available during a network partition
- Latency between data centers also makes consistency impractical
- Cassandra chooses Availability and Partition Tolerance over Consistency
- Down time is awful for several application



Theoretical Aspect: CAP Theory

every request receives a response about whether it was successful or failed



all nodes see the same data at the same time

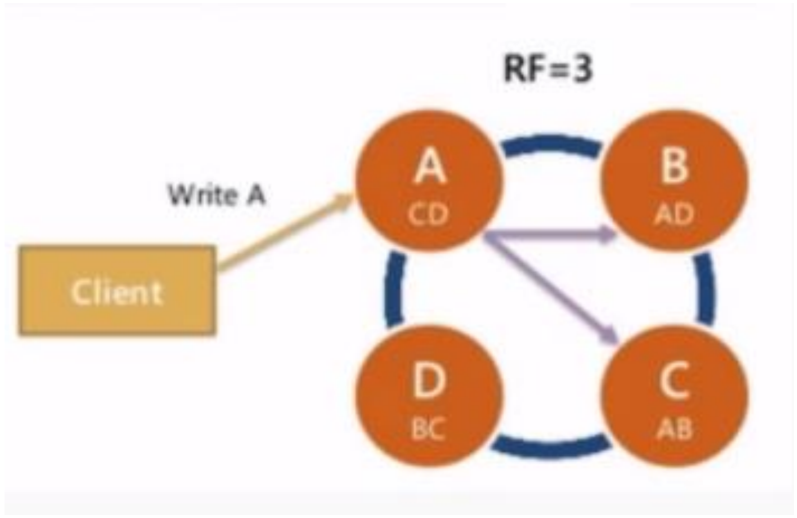
the system continues to operate despite arbitrary message loss

Consistency Level

- **Strong (Sequential): After the update completes any subsequent access will return the updated value**
- **Weak (weaker than Sequential): The system does not guarantee that subsequent accesses will return the updated value**
- **Eventual: All updates will propagate throughout all of the replicas in a distributed system, but that this may take some time. Eventually, all replicas will be consistent.**

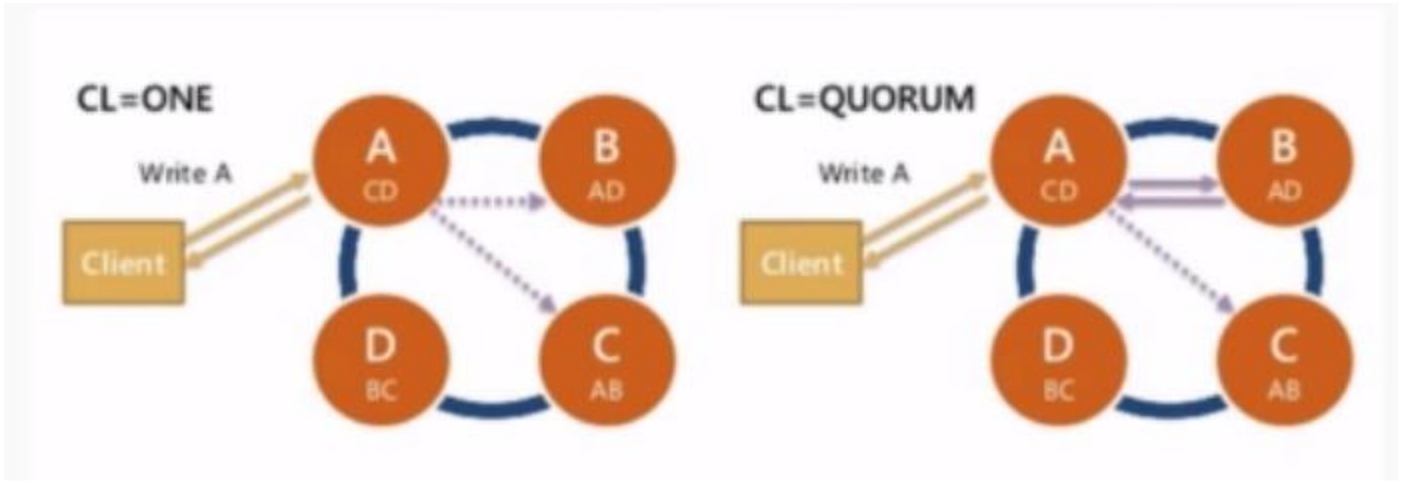
Replication

- Data is replicated automatically
- you pick number of severs
- Called “replication factor” or RF
- Data is ALWAYS replication to each replica
- If a machine is down, missing data is replayed via hinted handoff



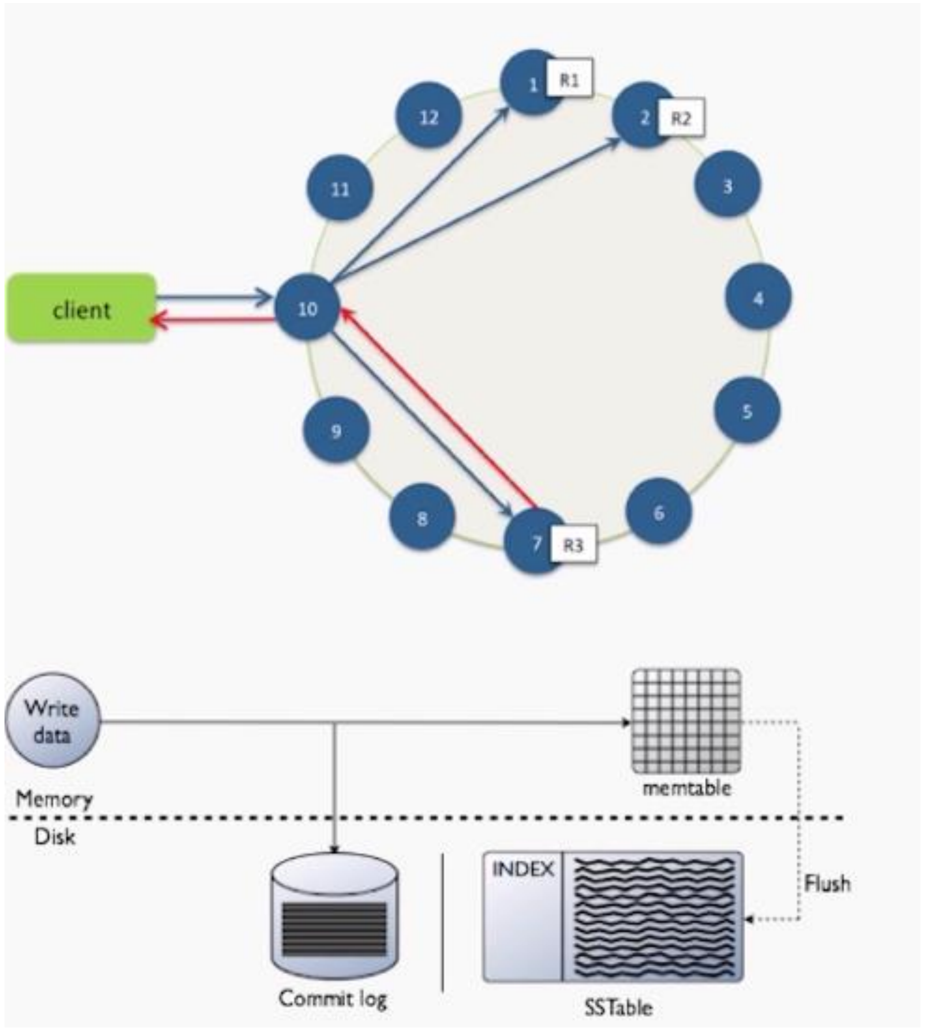
Tunable Data Consistency

- **Per query consistency**
- **ALL, QUORUM, ONE**
- **How many replica for query to respond OK**



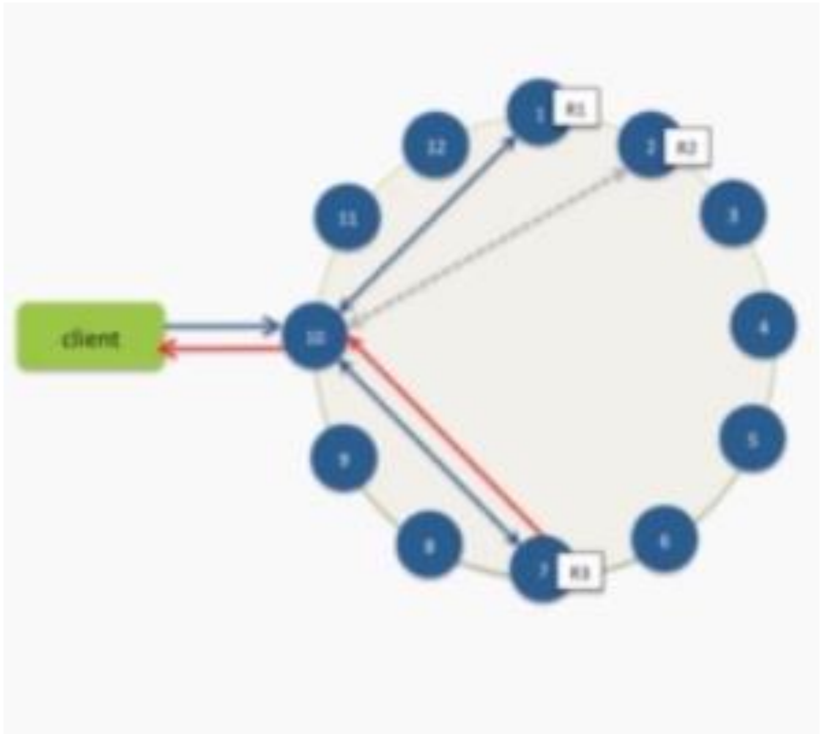
Architecture Overview

- **The Writes Path**
  - **Writes are written to any nodes in the cluster**
  - **Writes are written to commit log, the to memtable**
  - **Every write includes a timestamps**
  - **Memtable flushed to disk periodically (sstable)**



Architecture Overview

- **The Read Path**
  - **Any servers may be queried, it acts as the coordinator**
  - **Contacts nodes with the requested key**
  - **On each node data is pulled from sstables and merged**
  - **Consistency<ALL perfors read repair in backgroud**

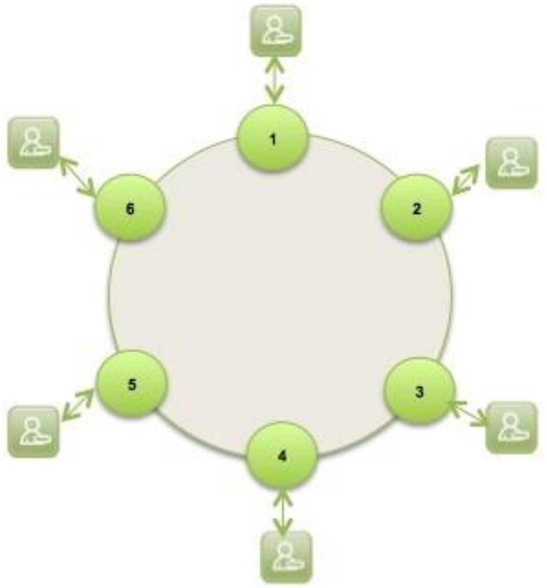


Why Cassandra?

- **Gigabyte to Petabyte scalability**
- **Linear performance gains through adding nodes**
- **No single point of failure**
- **Easy replication / data distribution**
- **Tunable data consistency**
- **Flexible schema design**
- **CQL language (like SQL)**

No Single Point of Failure

- All nodes the same
- Customized replication affords tunable data redundancy
- Read/write from any node
- Can replicate data among different physical data center racks
- What about Hbase or Elasticsearch?



Use-case: Facebook Inbox Search

- **Cassandra developed to address this problem.**
- **50+TB of user messages data in 150 node cluster on which Cassandra is tested.**
- **Search user index of all messages in 2 ways.**
  - **Term search : search by a key word**
  - **Interactions search : search by a user id**

Comparison with MySQL

- MySQL > 50 GB Data
  Writes Average : ~300 ms
  Reads Average : ~350 ms

- Cassandra > 50 GB Data
  Writes Average : 0.12 ms
  Reads Average : 15 ms

- Stats provided by Authors using facebook data.

Who’s Using Cassandra?

- **Use Cases:**
  - **High write throughput**
- **When Not to use Cassandra?**
  - **ACID Compliant with no roll back**
- **My Experience:**
  - **Hbase works great in analyzing or reading**
  - **Redis is perfect for in-memory database**
  - **If you need complicated query, you must use either Solr or Elasticsearch**
  - **Cassandra is not using HDFS, and CFS is more reliable than the other.**

Assignment

- **Reading Assignment:**
  - **Ellasandra**
  - **Twissandra**

Thank You