Big Data Engineering

NoSQL databases

Julie Weeds

March 2019

Contents

- Why NoSQL?
- A summary of a few NoSQL databases
  - MongoDB, Cassandra,
- NewSQL

NoSQL history

- Not just a recent thing ☺
- IBM IMS (Information Management System)
  - Launched in 1968
  - Used to store the bill of materials for the Saturn V rocket
  - Hierarchical model
- Still in widespread use today

Relational databases and SQL

Structured **Q**uery **L**anguage.
Originally pronounced “Sequel”, now usually ``EssQueEll’’

- Invented by the English computer scientist Edgar F. Codd whilst working at IBM in the 1970s
- Dominant for many years and still incredibly popular / pervasive.

- **In a relational database management system the data is perceived by the user as tables (and nothing but tables).**

Tables are referred to as **relations**

Each row or record is referred to as a **tuple.** The number of tuples is referred to as the **cardinality** of the relation.

STUD#	SURNAME	COURSE#	DOB
S1	Smith	C2	31/10/1982
S2	Jones	C1	6/6/1995
S3	Smith	C1	11/10/1990

COURSE#	NAME	SCHOOL#
C1	Computer Science	S1
C2	Artificial Intelligence	S1

Each column or field is referred to as an **attribute.** The number of attributes is referred to as the **degree** of the relation.

Limitations of Relational Approach

- In the 1970s databases were conceived to store "operational data" for large enterprises (banks, universities, businesses, ...)
- This data largely fit into the relational model – that the data should be perceived by the user as tables and nothing but tables.
- The *schema* for such a database (i.e., how the database is divided into tables) could be designed once and then fixed.
- Acceptable performance on cross-table joins and transactions could be achieved by hosting the database on a single, very large and powerful server.
- Does this approach work for "big" data?

Big data database requirements

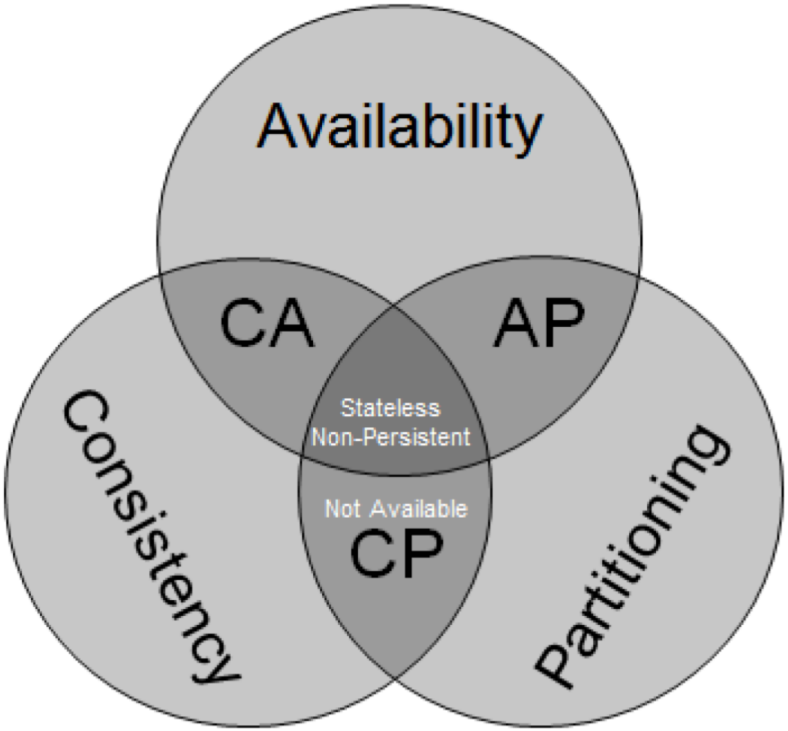
- Support for massive volumes of data ... without placing it all on a single expensive server.
- Support for variety in data forms:-
  - temporal data, spatial data, multimedia data, unstructured data, document libraries
- Support for agility in database design – want to be able to quickly and cheaply integrate a new data source
- Support for complex/structured data types
- Support for generalization and inheritance
- May only require access via primary key

Why NoSQL?

- better handling of large volumes of rapidly changing structured, semi-structured and unstructured data
- dynamic schemas which allow agile development
- use of object-oriented programming paradigm
- Horizontal scaling – increase capacity by adding more servers rather than increasing the size of a single server
- Servers can be geographically distributed, i.e., in the cloud.
- Databases automatically replicated to maintain availability in the event of localised outages or downtime.
- More appropriate balance between read/write performance

ReCAP

- You can have 2 out of three:
  - Consistent
    - ACID
  - Available
    - HA / Accessible 24x7
  - Partitioned
    - Able to split into different datacentres
    - Survive network down

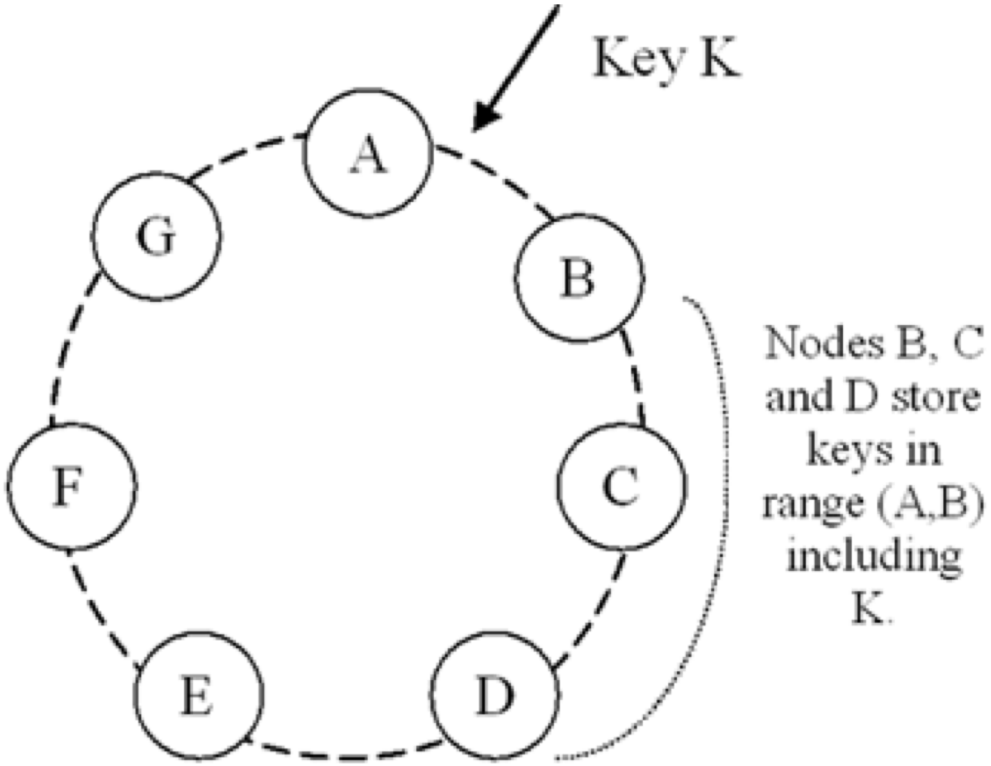


http://blog.mccrory.me/2010/11/03/cap-theorem-and-the-clouds/

NoSQL parents

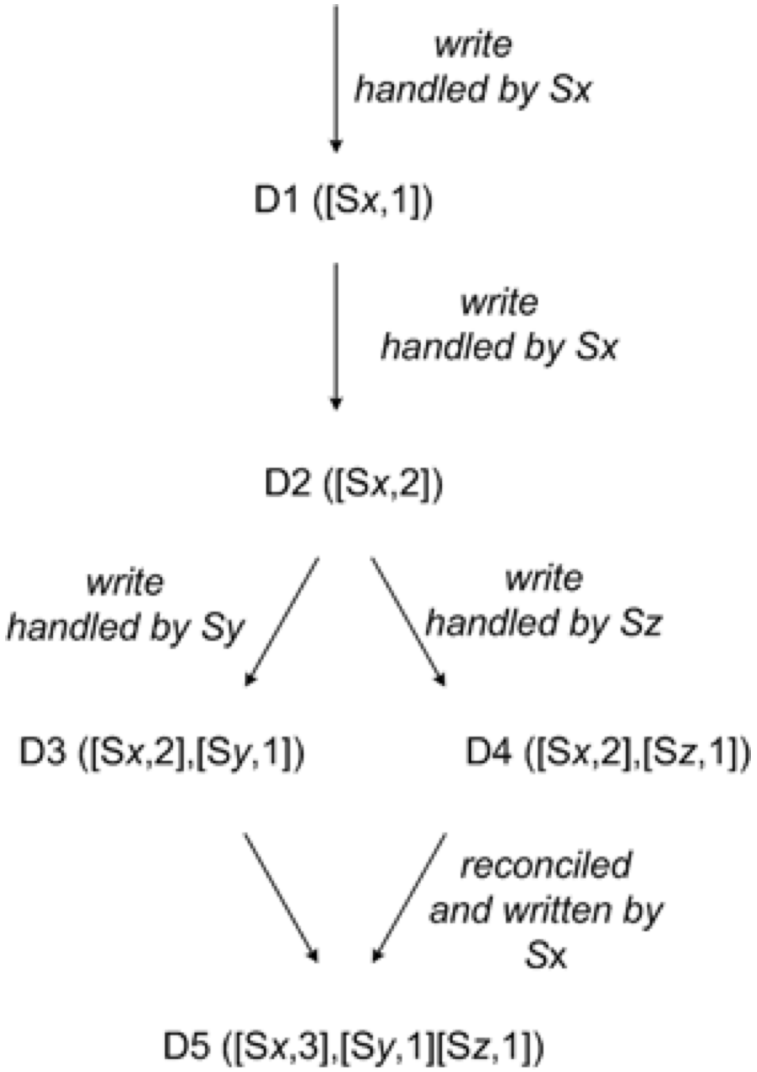
- Amazon Dynamo
  - Eventually consistent
- Google BigTable
  - Supporting very large rows
- LDM
  - Graph database

Dynamo Partitioning and Replication Model



http://www.allthingsdistributed.com/2007/10/amazons_dynamo.html

Reconciliation / Eventual Consistency



http://www.allthingsdistributed.com/2007/10/amazons_dynamo.html

Google BigTable

- Optimized to support very large data
  - Not just many rows, but rows that cannot fit into the memory of a single server
  - Column Families allow each row to live across servers
- This table dates back to 2005

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

Current NoSQL Databases

- Too many to list!
- Popular databases include:
  - MongoDB
  - Couchbase
  - Apache Cassandra
  - Apache HBase
  - Voldemort
  - Redis
  - Riak
  - Etc, etc

Types of NoSQL databases

Type	Description	Examples
Key-value stores	every single item in the database is stored as an attribute name (or key) together with its value	Berkeley DB Riak
Graph stores	used to store information about graphs/networks of data (e.g., social connections)	Neo4J Giraph
Document databases	pair each key with a complex data structure known as a document. Documents can contain many different key-value pairs, key-array pairs or nested documents.	MongoDB
Wide-column stores	optimised for queries over large datasets and store columns of data together, instead of rows.	Cassandra HBase

MongoDB

- Uses a document data model
- Each record and its associated data is thought of as a “document”
- Documents encoded in a JSON-like format called BSON (essentially a binary encoding of JSON)
- Advantages include
  - documents are independent units, related data is stored contiguously, making it easier to distribute data across multiple servers and maintain performance
  - application logic is written in any language which supports object-oriented programming. The object model in your application can be turned directly into a document.
  - Unstructured data is stored easily
  - Database does not need to know the schema in advance

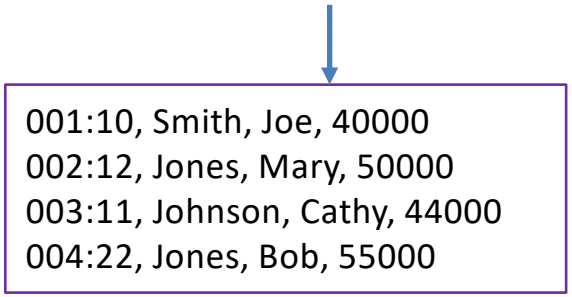
Key Value databases

- A persistent associative array or dictionary
- Simple access and fits well with programming models (especially MR)
- Indexing on other data is not often possible and can be slow

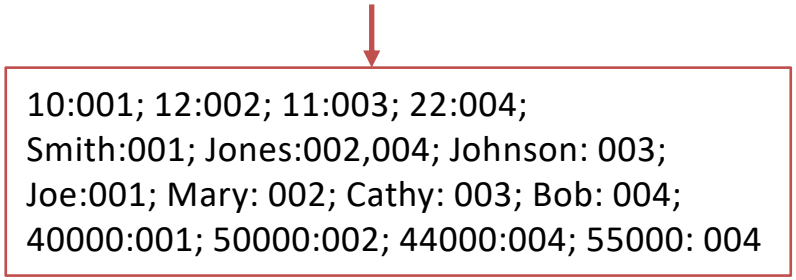
Column-oriented DBMS

- Data tables stored as columns rather than rows
- Practical use is very similar to a row-oriented DBMS
- Can use SQL to interact with data

Row-oriented systems perform well at row-based queries – e.g., returning all information about a particular object



Column-oriented systems perform well at column-based queries – e.g., find all record with a particular attribute value



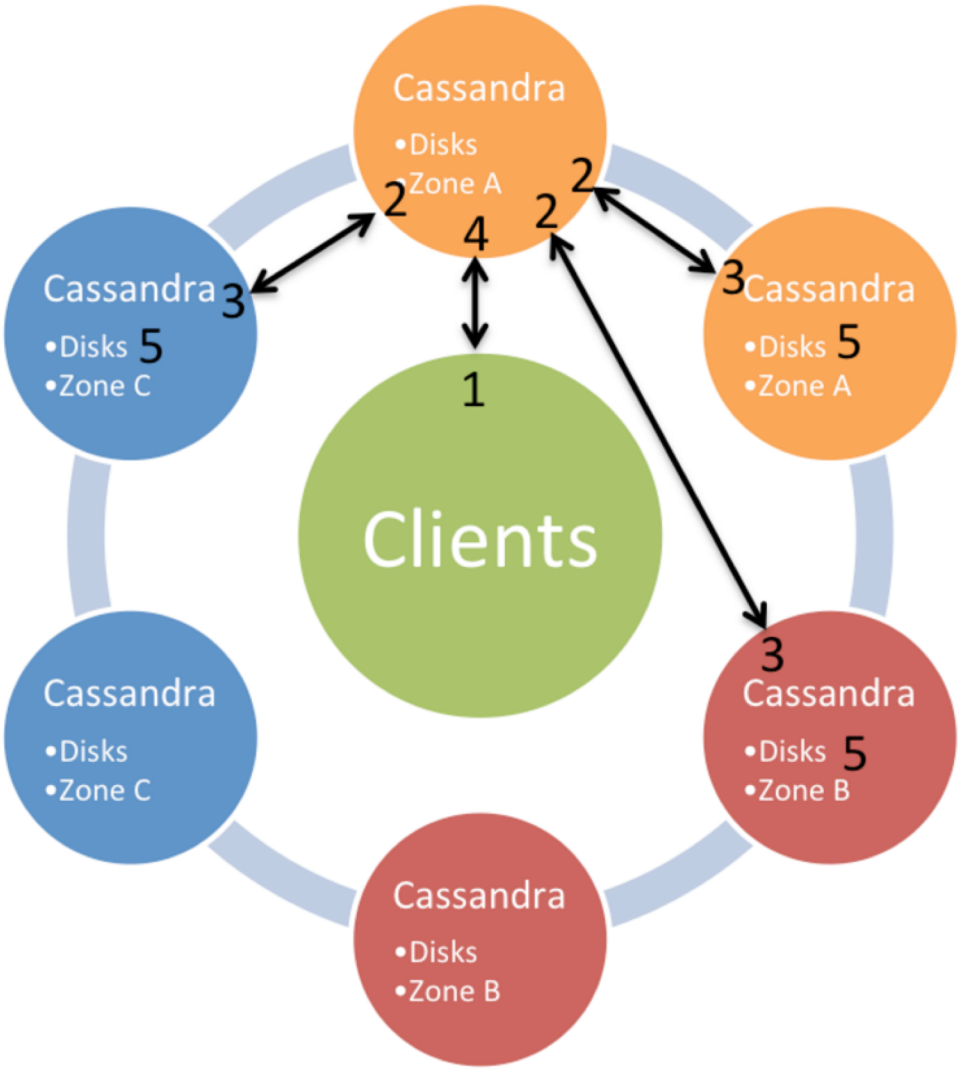
Apache Cassandra

- Hybrid between key-value and column-oriented DBMS
- Uses Cassandra Query Language (CQL) as an alternative to SQL
- Developed by Facebook and released as open-source project in 2008
- Main features:
  - Data is distributed across a cluster (no single point of failure)
  - Supports replication, fault-tolerant
  - Horizontal scaling
  - Tunable consistency
  - Hadoop integration with MapReduce support (+Apache Pig and Apache Hive)

Cassandra Write Model

Single Datacentre

1. Client Writes to any Cassandra Node
2. Coordinator Node replicates to nodes and Zones
3. Nodes return ack to coordinator
4. Coordinator returns ack to client
5. Data written to internal commit log disk



If a node goes offline, hinted handoff completes the write when the node comes back up.

Requests can choose to wait for one node, a quorum, or all nodes to ack the write

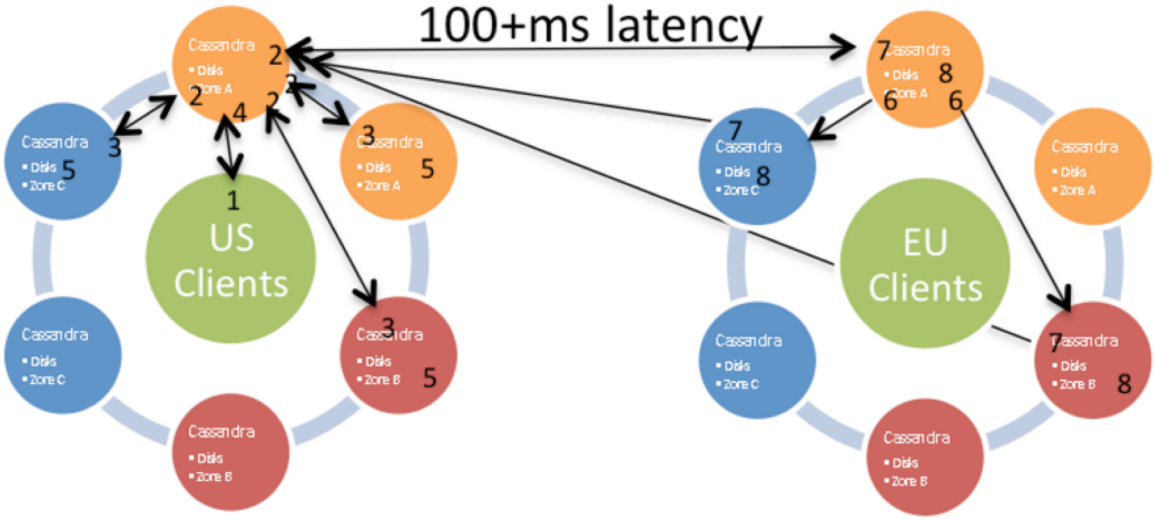
SSTable disk writes and compactions occur asynchronously

Source: Netflix

Multi Datacentre Writes

1. Client Writes to any Cassandra Node
2. Coordinator node replicates to other nodes Zones and regions
3. Local write acks returned to coordinator
4. Client gets ack when 2 of 3 local nodes are committed
5. Data written to internal commit log disks
6. When data arrives, remote node replicates data
7. Ack direct to source region coordinator
8. Remote copies written to commit log disks

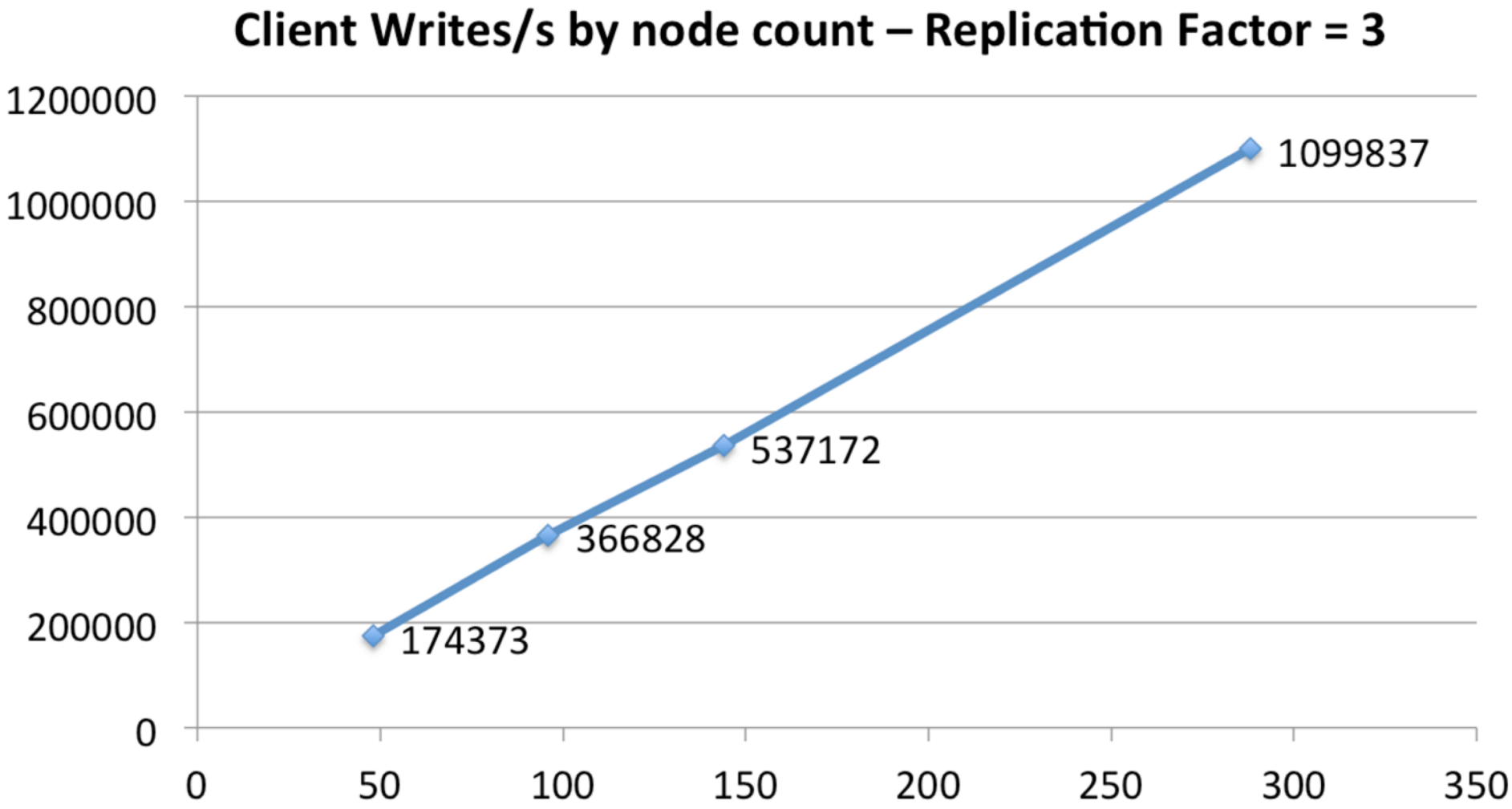
If a node or region goes offline, hinted handoff completes the write when the node comes back up. Nightly global compare and repair jobs ensure everything stays consistent.



Source: Netflix

Cassandra Scale Up

In Amazon EC2



Source: Netflix

Cassandra Model

- **Keyspaces** are roughly equivalent to SQL Databases
  - Encapsulate replication strategies
- **Column Families** roughly equivalent to SQL tables
- Generally a different approach vs SQL
  - Writes are cheap
  - Indexes are expensive
  - Normalization is not the goal

Cassandra Model cont.

- Inserts are the same as updates
  - No read first
- Data can be marked with a Time to Live (TTL)
  - Automatically deleted
- Deletes are not instant
  - Deleted rows are marked with a tombstone
  - Eventually cleaned up
  - Can re-appear if you do not run node repair after a node failure

CQL

- A variant of SQL written specifically for Cassandra
  - The preferred model of access
  - Replaces the old “Thrift” API
- Attempts to have some compatibility with normal SQL
  - e.g. you can use either KEYSPACE or TABLE interchangeably

CQL examples

SELECT name, occupation FROM users WHERE userid IN (199, 200, 207);

However, some queries are not permitted:

```
SELECT firstname, lastname FROM users WHERE
birth_year = 1981 AND country = 'FR';
```

Requires a large scan of the database and cannot give a predictable time response:

ALLOW FILTERING will make this run anyway

INSERT / UPDATE

```
INSERT INTO NerdMovies (movie, director, main_actor, year)
      VALUES ('Serenity', 'Joss Whedon', 'Nathan Fillion', 2005)
            USING TTL 86400;
```

- Every row can have a specified expiry time
- Inserts work even if the data is already there, unless you specify:

```
INSERT INTO NerdMovies (movie, director, main_actor, year)
      VALUES ('Serenity', 'Joss Whedon', 'Nathan Fillion', 2005)
       IF NOT EXISTS
            USING TTL 86400;
```

This can have unpredictable timing because it requires read-before-write

Non-SQL data types

- Sets
  - CREATE TABLE cycling.cyclist_career_teams (id UUID PRIMARY KEY, lastname text, teams **set**<text>);
- Lists
  - CREATE TABLE cycling.upcoming_calendar (year int, month int, events **list**<text>, PRIMARY KEY (year, month));
- Maps
  - CREATE TABLE cycling.cyclist_teams (id UUID PRIMARY KEY, lastname text, firstname text, teams **map**<int,text>);
- Tuples
  - CREATE TABLE cycling.popular (rank int PRIMARY KEY, cinfo **tuple**<text,text,int>);

Direct support for JSON

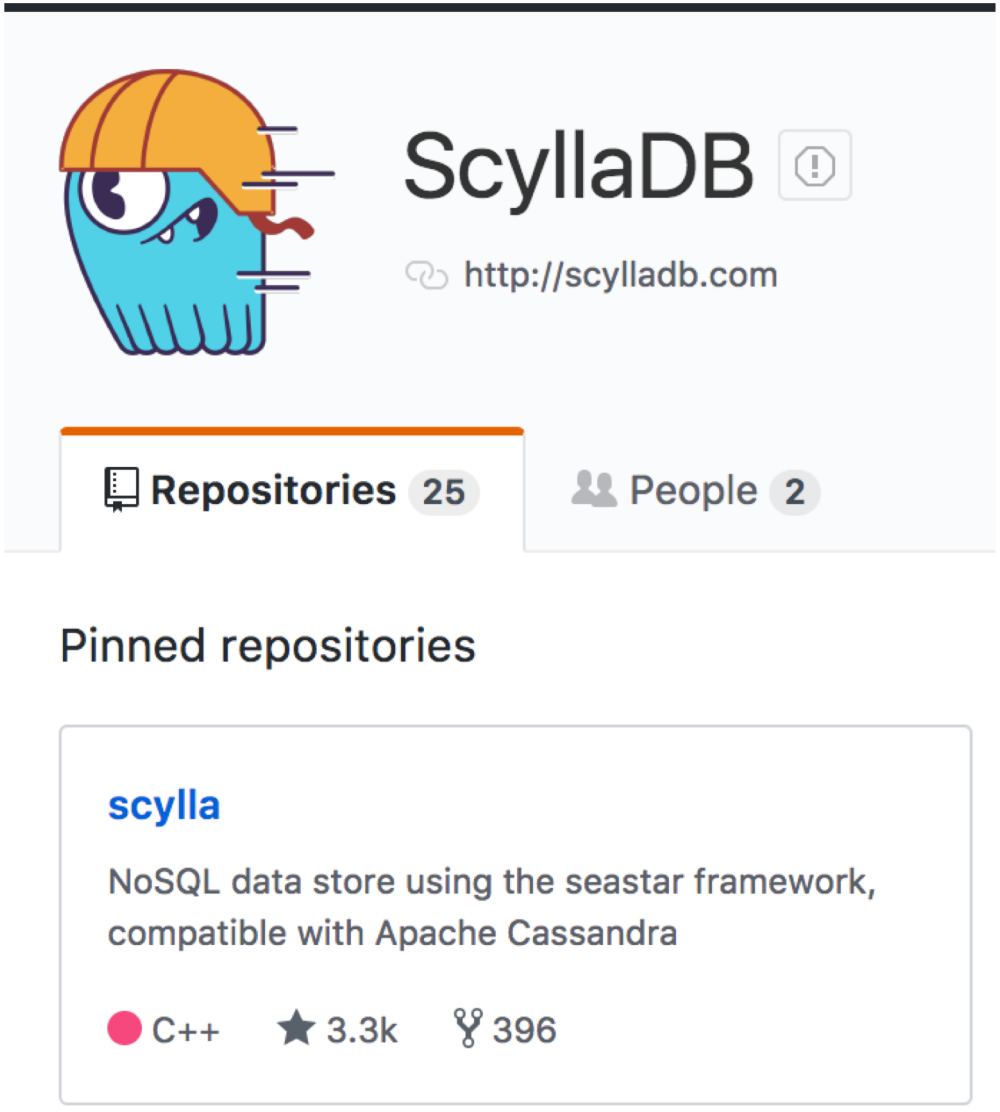
```
INSERT INTO cycling.cyclist_category JSON '{
  "category" : "GC",
  "points" : 780,
  "id" : "829aa84a-4bba-411f-a4fb-38167a987cda",
  "lastname" : "SUTHERLAND" }';
```

cassandra.yaml

- Configuration of the major parts of the system
  - Datacentres, Racks, Cluster name
  - Authentication and Authorization
  - Partitioner
  - Data Storage location
  - Cacheing
  - Network topology and ports
  - Etc, etc

ScyllaDB

- A C++ “clone” of Cassandra
- Also Open Source
- Claims to be significantly faster

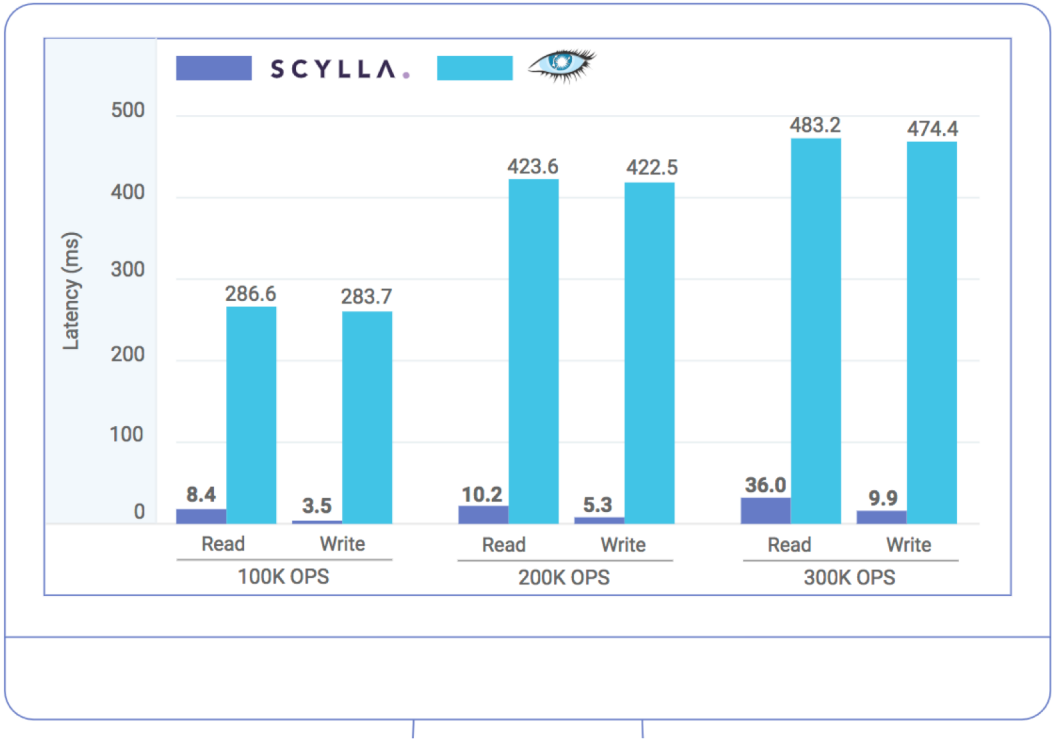


Scylla vs Cassandra

4-node Scylla 2.2 (i3.metal) vs 40-node Cassandra 3.11 (i3.4xlarge)

- 2.5X reduction in AWS EC2 costs
- Dramatic improvement in availability through 10X reduction in cluster size
- Up to 11X improvement in 99th percentile latency
- Amazon EC2 Bare Metal Instances are a great platform for Scylla

SEE THE FULL REPORT →



Amazon EC2 I3 Instances

Storage optimised for high transaction workloads

Product Details

Model	vCPU	Memory (GiB)	Networking Performance	Storage (TB)
i3.large	2	15.25	Up to 10 Gigabit	1 x 0.475 NVMe SSD
i3.xlarge	4	30.5	Up to 10 Gigabit	1 x 0.95 NVMe SSD
i3.2xlarge	8	61	Up to 10 Gigabit	1 x 1.9 NVMe SSD
i3.4xlarge	16	122	Up to 10 Gigabit	2 x 1.9 NVMe SSD
i3.8xlarge	32	244	10 Gigabit	4 x 1.9 NVMe SSD
i3.16xlarge	64	488	25 Gigabit	8 x 1.9 NVMe SSD
i3.metal	72*	512	25 Gigabit	8 x 1.9 NVMe SSD

Top ten databases 2019

343 systems in ranking, February 2019

Rank Feb 2019	Rank Jan 2019	Rank Feb 2018	DBMS	Database Model	Score Feb 2019	Score Jan 2019	Score Feb 2018
1.	1.	1.	Oracle	Relational, Multi-model	1264.02	-4.82	-39.26
2.	2.	2.	MySQL	Relational, Multi-model	1167.29	+13.02	-85.18
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1040.05	-0.21	-81.98
4.	4.	4.	PostgreSQL	Relational, Multi-model	473.56	+7.45	+85.18
5.	5.	5.	MongoDB	Document	395.09	+7.91	+58.67
6.	6.	6.	IBM Db2	Relational, Multi-model	179.42	-0.43	-10.55
7.	7.	↑ 8.	Redis	Key-value, Multi-model	149.45	+0.43	+22.43
8.	8.	↑ 9.	Elasticsearch	Search engine, Multi-model	145.25	+1.81	+19.93
9.	9.	↓ 7.	Microsoft Access	Relational	144.02	+2.41	+13.95
10.	10.	↑ 11.	SQLite	Relational	126.17	-0.63	+8.89

ranked by popularity:

- number of results in web searches
- Stack Overflow & DBA Stack Exchange
- Google Trends
- job advertisements
- professional networks
- social networks

http://db-engines.com/en/ranking

Next 20

11.	11.	10.	Cassandra	Wide column	123.37	+0.39	+0.59
12.	13.	17.	MariaDB	Relational, Multi-model	83.42	+4.60	+21.77
13.	12.	13.	Splunk	Search engine	82.81	+1.39	+15.55
14.	14.	12.	Teradata	Relational	75.97	-0.22	+2.98
15.	15.	18.	Hive	Relational	72.29	+2.38	+17.23
16.	16.	14.	Solr	Search engine	60.96	-0.52	-2.91
17.	17.	16.	HBase	Wide column	60.28	-0.12	-1.43
18.	18.	19.	FileMaker	Relational	57.79	+0.64	+3.43
19.	19.	20.	SAP HANA	Relational, Multi-model	56.55	-0.09	+9.19
20.	21.	15.	SAP Adaptive Server	Relational	55.75	+0.71	-7.74
21.	20.	21.	Amazon DynamoDB	Multi-model	54.95	-0.15	+15.07
22.	22.	22.	Neo4j	Graph	47.86	+1.06	+8.04
23.	23.	23.	Couchbase	Document	35.58	+0.98	+3.83
24.	24.	24.	Memcached	Key-value	29.45	-0.09	+0.51
25.	25.	26.	Microsoft Azure SQL Database	Relational, Multi-model	27.12	-0.07	+3.34
26.	26.	25.	Informix	Relational, Multi-model	26.35	-0.41	-2.03
27.	27.	31.	Microsoft Azure Cosmos DB	Multi-model	24.86	+0.47	+8.67
28.	28.	28.	Vertica	Relational, Multi-model	22.81	+1.03	+2.84
29.	30.	33.	Amazon Redshift	Relational	20.99	+0.94	+7.87
30.	32.	27.	CouchDB	Document	20.00	+0.68	-0.30

Top ten databases 2015

283 systems in ranking, November 2015

Rank			DBMS	Database Model	Score		
Nov 2015	Oct 2015	Nov 2014			Nov 2015	Oct 2015	Nov 2014
1.	1.	1.	Oracle	Relational DBMS	1480.95	+13.99	+28.82
2.	2.	2.	MySQL	Relational DBMS	1286.84	+7.88	+7.77
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1122.33	-0.90	-97.87
4.	4.	↑ 5.	MongoDB ⊞	Document store	304.61	+11.34	+59.87
5.	5.	↓ 4.	PostgreSQL	Relational DBMS	285.69	+3.56	+28.33
6.	6.	6.	DB2	Relational DBMS	202.52	-4.28	-3.71
7.	7.	7.	Microsoft Access	Relational DBMS	140.96	-0.87	+2.12
8.	8.	↑ 9.	Cassandra ⊞	Wide column store	132.92	+3.91	+40.93
9.	9.	↓ 8.	SQLite	Relational DBMS	103.45	+0.78	+8.17
10.	10.	↑ 11.	Redis ⊞	Key-value store	102.41	+3.61	+20.06

http://db-engines.com/en/ranking

Next 20

11.	11.	↓ 10.	SAP Adaptive Server	Relational DBMS	83.71	-1.93	-0.91
12.	12.	12.	Solr	Search engine	79.77	+0.71	+2.96
13.	13.	13.	Teradata	Relational DBMS	77.08	+3.64	+9.85
14.	14.	↑ 16.	Elasticsearch	Search engine	74.77	+4.55	+31.71
15.	15.	15.	HBase	Wide column store	56.46	-0.78	+9.49
16.	16.	↑ 17.	Hive	Relational DBMS	54.91	+1.35	+18.30
17.	17.	↓ 14.	FileMaker	Relational DBMS	51.73	+1.95	+0.39
18.	18.	↑ 20.	Splunk	Search engine	44.61	+1.11	+12.44
19.	19.	↑ 21.	SAP HANA	Relational DBMS	39.62	+0.52	+11.26
20.	20.	↓ 18.	Informix	Relational DBMS	38.46	+0.12	+2.88
21.	21.	↑ 23.	Neo4j ⊞	Graph DBMS	34.04	+0.63	+9.39
22.	22.	↓ 19.	Memcached	Key-value store	32.39	+0.82	-0.20
23.	↑ 25.	↑ 27.	MariaDB ⊞	Relational DBMS	26.64	+2.01	+9.35
24.	↓ 23.	↓ 22.	CouchDB	Document store	26.37	-0.49	+0.57
25.	↓ 24.	↓ 24.	Couchbase ⊞	Document store	25.82	-0.37	+4.33
26.	26.	26.	Firebird	Relational DBMS	22.56	-0.33	+4.95
27.	↑ 28.	↓ 25.	Netezza	Relational DBMS	21.84	+0.60	+3.76
28.	↓ 27.	↑ 31.	Amazon DynamoDB	Multi-model ⓘ	21.75	+0.42	+9.43
29.	↑ 30.	↓ 28.	Microsoft Azure SQL Database	Relational DBMS	19.46	+0.62	+4.85
30.	↓ 29.	↓ 29.	Vertica	Relational DBMS	19.41	+0.45	+5.46

2015 → 2019

Gaining popularity

- Redis — key-value
- Elasticsearch — search engine
- Splunk — search engine

Losing popularity

- Cassandra — wide column stores
- Hbase — wide column stores

Relational databases (e.g., SQL) still seem dominant

“NewSQL”

- ACID databases that aim to provide high availability and scalabilility
  - VoltDB
  - NuoDB
  - Google Spanner
  - MemSQL
- Generally assume
  - online transaction processing (rather than analysis)
  - individual transactions touch on a small portion of the database
  - transactions are repetitive (with different values)

In Memory Databases

- Memory is relatively much cheaper than it used to be
- Uses snapshots or transaction logs to ensure durability
- *Some NoSQL, some NewSQL*
  - SAP Hana
  - Redis
  - VoltDB
  - MemSQL
  - Apache Geode

Questions?