CSCU9YQ - NoSQL Databases

Lecture 9:
Column-Family Databases

Prof Gabriela Ochoa
http://www.cs.stir.ac.uk/~goc/
University of Stirling

Resources

- DataStax Enterprise
  - Website with Resources: https://www.datastax.com/resources/
  - Academy: https://academy.datastax.com/
  - Replication Strategies: https://www.youtube.com/watch?v=u7nHyzFHqMA
  - Architecht's Guide to NoSQL: https://www.datastax.com/dbas-guide-to-nosql
- Google BigTable
  - Paper: https://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf
  - Website: https://cloud.google.com/bigtable/
- Cassandra
  - Website: http://cassandra.apache.org/

Revisiting: Why NoSQL?

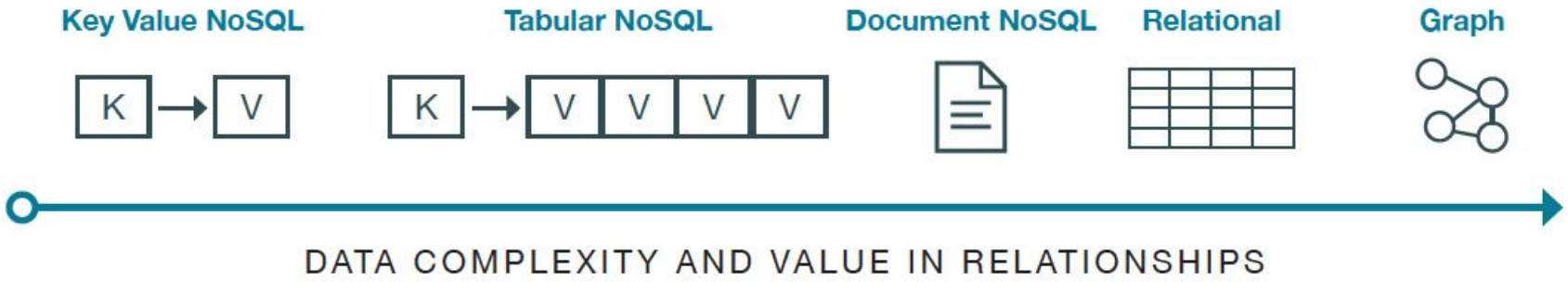
- 80's and 90's: RD standard (Oracle, Sybase, Microsoft SQL Server)
- Early 2000's: wave of Web applications, open RD (MySQl, Postgres)
- ~ 2005: The requirements of key Internet players Amazon, Facebook & Google outgrow RD for certain types of application. Need for
  - More flexible data models
  - Agile development methodologies
  - Large amounts of fast-incoming data from millions of applications and users globally
  - Maintaining extreme amounts of performance and uptime
- Nowadays:
  - Many companies using modern cloud applications.
  - Data problems encountered by Internet giants have become common issues

Types of NoSQL

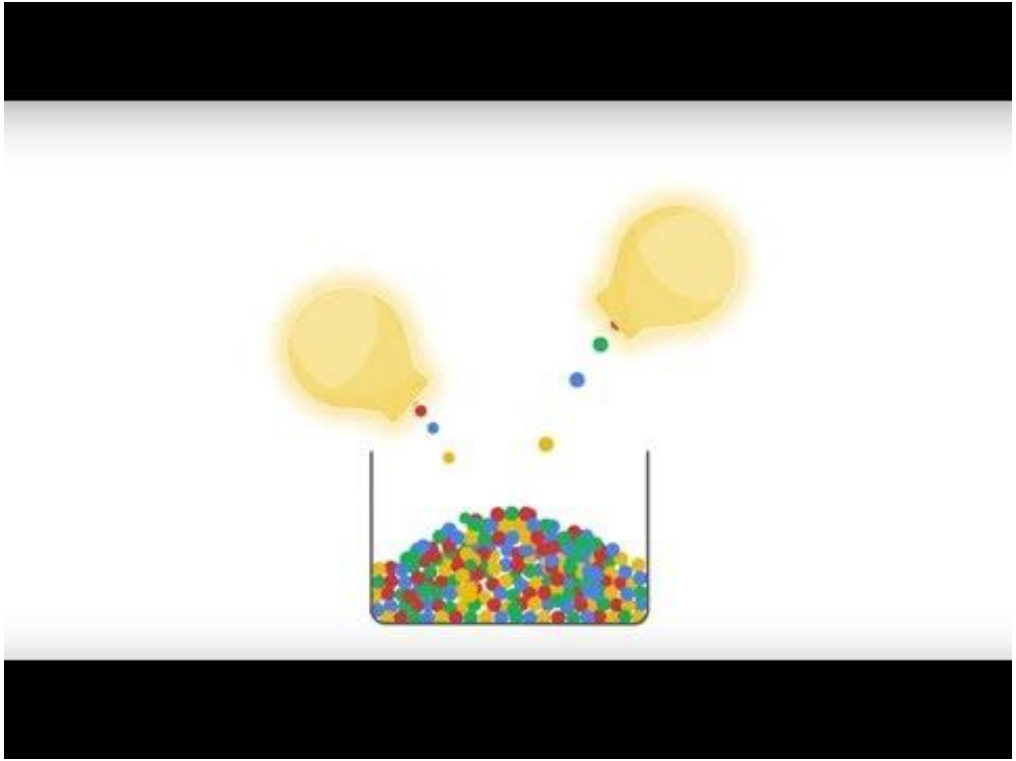
Data Model	Description	Examples
Key-Value	Most basic type Indexed key and value	Amazon Dynamo, Riak Oracle NoSQL
Column-Family	Data stored in rows Queries performed via column-based access	Cassandra, HBase, Google Big Table
Document	Documents, JSON format Document assigned a single key	MongoDB, CouchDB
Graph	Highly complex and connected data Stored as nodes, edges and propertoies	DataStax Enterprise Graph, Neo4J

The Data Model Continuum

- Some NoSQL databases require the denormalisation of data and aren't concerned with the relationships between data entities
- Others are built to handle complex and very intense data relationship scenarios:



Google BigTable



- Distributed, column-oriented data store created by Google Inc. to handle very large amounts of structured data associated with the company's Internet search and Web services operations.
- Compressed, high performance, built on Google File System, Chubby Lock Service, SSTable and a few other Google technologies.

Casandra

- Cassandra is a powerful open source distributed database system that works extremely well to handle huge volumes of records spread across multiple commodity servers.
- It was created at Facebook and later open sourced
- The data model is based on Google Bigtable
- The distributed design is based on Amazon Dynamo (Key-Value DB)
- Highly consistent, fault-tolerant, and scalable

Cassandra Data Model

- In a traditional, relational database: every column must be “satisfied” to store a row.
- Within a table, each row may or may not have relevant data for each column:

Id	FirstName	LastName	Employer	Phone
1	Orlando	Gee		245-555-0173
2	Keith	Harris	Progressive Sports	
3	Donna	Carreras	Advanced Bike Components	710-555-0173

Cassandra Data Model

In Cassandra, each row within a table essentially has its own set of columns

Row Key	FirstName	LastName	Employer	Phone
3	Donna	Carreras	Advanced Bike Components	710-555-0173

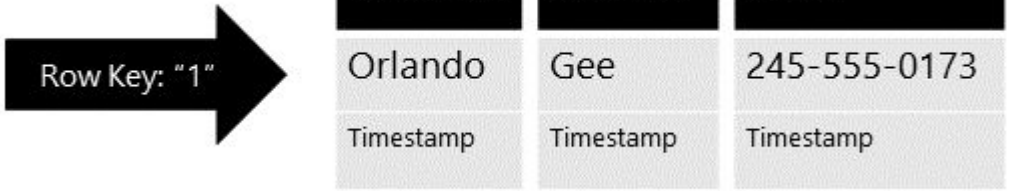
Row Key	FirstName	LastName	Employer
2	Keith	Harris	Progressive Sports

Row Key	FirstName	LastName	Phone
1	Orlando	Gee	245-555-0173

Each row does not have to share the same columns with other rows in the same table

Cassandra Data Model

- Each key in Cassandra directly points to a specific “row” in the table
- Rows are not the basic unit.
- Each row points to a collection of columns.



A column is the most basic unit and consist of a name, value and a timestamp.

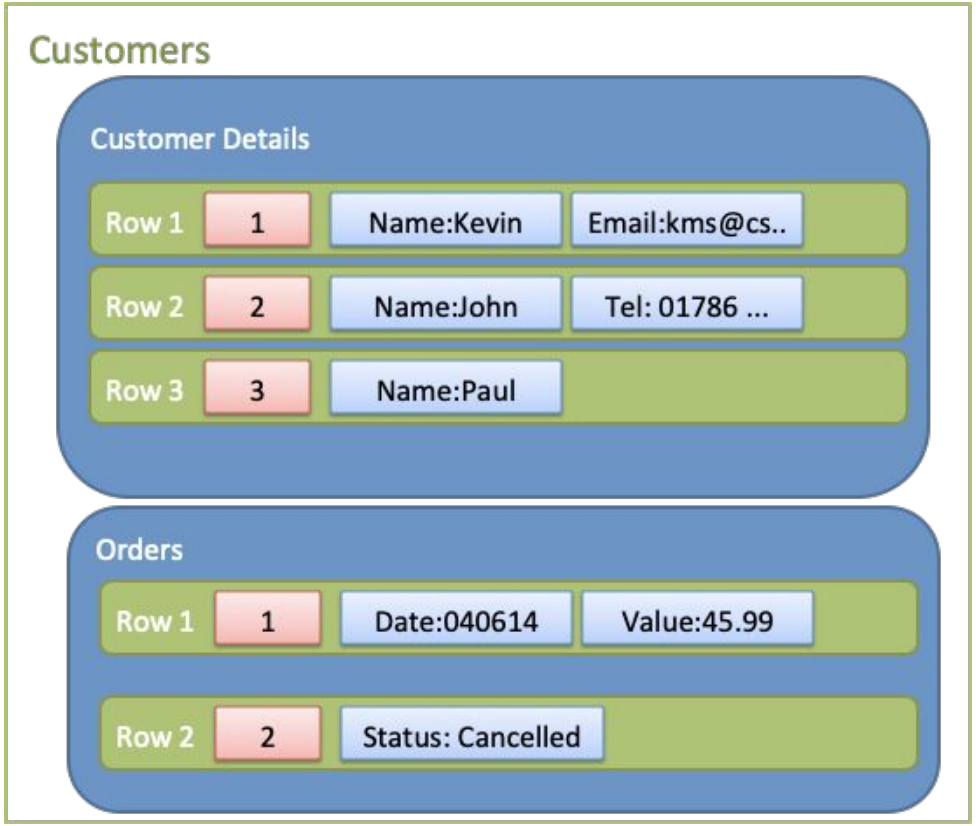
The Row-key: “1” points to to collection of 3 columns.

- Cassandra can store individual columns in a distributed manner to optimise for specific queries.
- Columns that are often queried together can be placed on the same node to optimise the performance of the specific query.

Column-family Databases: The Concept

Example

- Two column families
- Row key to identify each row
- Each row can have different set of columns
- if a row does not contain a value for a certain column, that column is simply missing
- Variable length lists of unique values, saves a lot of space



Queries in Cassandra (CQL)

- Own query language named Cassandra Query Language (CQL)
- You can use the cassandra query shell (cqlsh) to interact with Cassandra and issue queries directly against the database.
- Many of the **TABLE**, **INSERT** and **SELECT** queries are the same between SQL and Cassandra.
- CQL has many features for querying data, but not all that SQL has
- CQL does not allow joins or subqueries, WHERE clauses are simple.

Create a Table

Use an existing “database” and then use a CREATE TABLE command

```
USE contoso;

        CREATE TABLE products(
            product_id int,
            name text,
            is_available boolean,
            PRIMARY KEY (product_id)
        )
```

Note that the CREATE TABLE syntax uses a different PRIMARY KEY syntax and simpler data types than you would see in a traditional SQL Database.

SELECT

```
SELECT cols FROM col_family WHERE condition
```

- cols can be comma separated list: name, email
- or range name .. email
- or *
- or FIRST n (gets first n columns)
- Examples
  - `SELECT * FROM products`
  - `SELECT COUNT(*) FROM products`

INSERT

```
INSERT INTO col_family (col, col, ...) VALUES (val, val, ...)
USING option ...
```

- VALUES: a literal, a list, a set or a JSON style array of literals: {k:v, k:v, ...}
- Option can be CONSISTENCY, TIMESTAMP, TTL (time to live)
- Examples
  - `INSERT INTO products (product_id, name, is_available) VALUES (8, "Helmet", true);`
  - `INSERT INTO users (user_id, first_name, last_name, emails) VALUES('123', Kev', Swingler', {'kms@cs.stir.ac.uk', 'kms7@stir.ac.uk'});`
  - `INSERT INTO users (user_id, phone) VALUES('123','467676')`

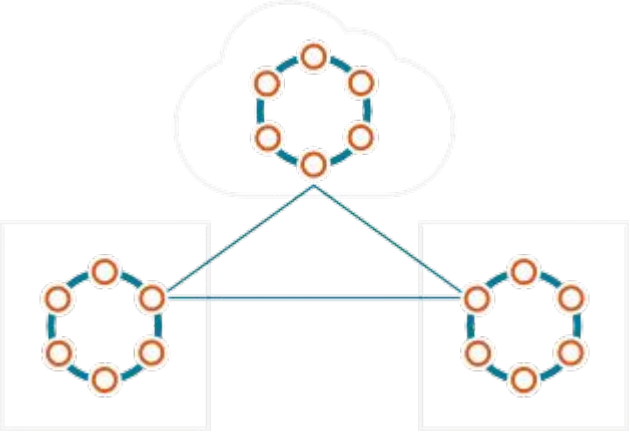
Creating a Database

```
CREATE KEYSPACE contoso
     WITH REPLICATION = {'class':'SimpleStrategy', 'replication_factor' : 3};
```

- Tables (column families) are grouped together into a KEYSPACE (analogous to a single database in RDBs)
- The keyspace defines the replication strategy across nodes
  - Replication class: simple or Network topology
  - Replication factor: total number of data copies that are replicated. For example
    - RF = 1 means only one copy of each row in a cluster
    - RF = 3 means 3 copies of the data are stored across the cluster

Cassandra Architecture

- Database can scale and perform with no downtime (continuous availability).
- Masterless (peer-to-peer) “ring” distributed architecture
- To add more capacity, simply add new nodes in an online fashion to an existing cluster.
- No single point of failure, true continuous availability



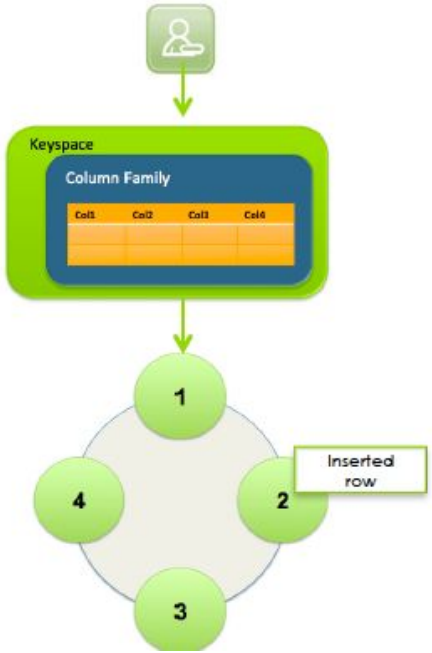
All nodes are the same

- there is **no** master node,
- all nodes communicating with each other via a *gossip* protocol

Gossip is a peer-to-peer communication protocol in which nodes periodically exchange state information about themselves and about other nodes they know about.

Automatic Data Distribution (Partition)

- Cassandra automatically distributes and maintains data across the nodes in a cluster
- Uses a *partitioner*, a hashing mechanism that takes a table row's primary key, computes a numerical token for it, and assigns it to one of the nodes in a cluster.
- Two main strategies for partitioning
  1. Random partitioning – this is the default and recommended strategy. Partitions data as evenly as possible across all nodes
  2. Ordered partitioning – stores column family row keys in sorted order across the nodes in a DB cluster



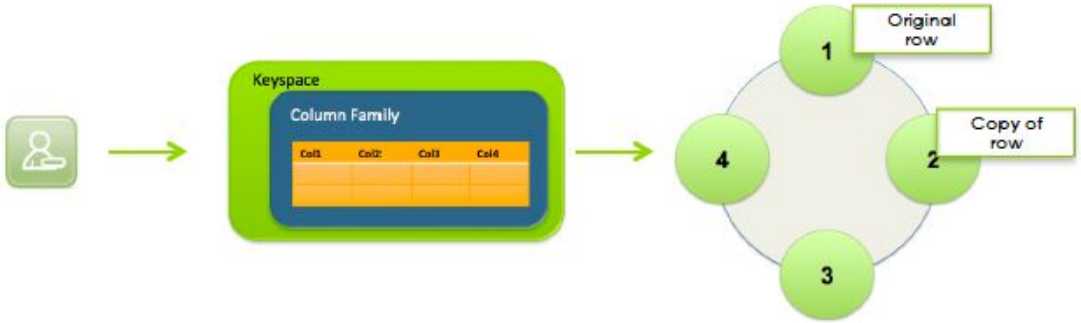
Data is inserted and assigned a row key in a column family

Data placed on node based on its column family row key

Replication

Controlled by two parameters

- Replication factor: total number of data copies that are replicated.
- Replication strategy:
  - Simple
  - Network topology



Simple Strategy

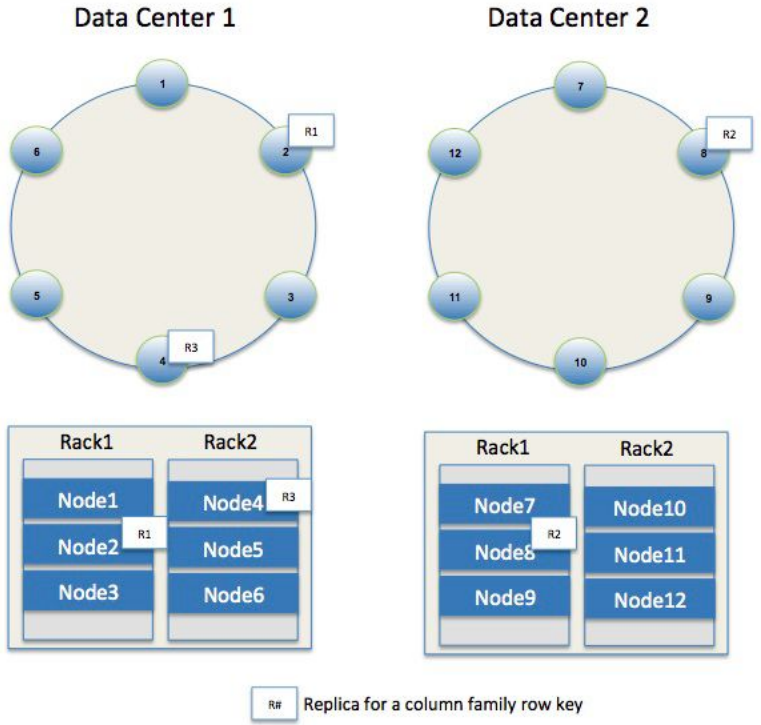
Places the original row on a node determined by the partitioner.

Additional replica rows are placed on the next nodes clockwise in the ring without considering rack or data center location.

Replication Strategy

Network Topology Strategy: allows for replication between different racks in a data center and/or between multiple data centers.

This strategy provides more control over where replica rows are placed.

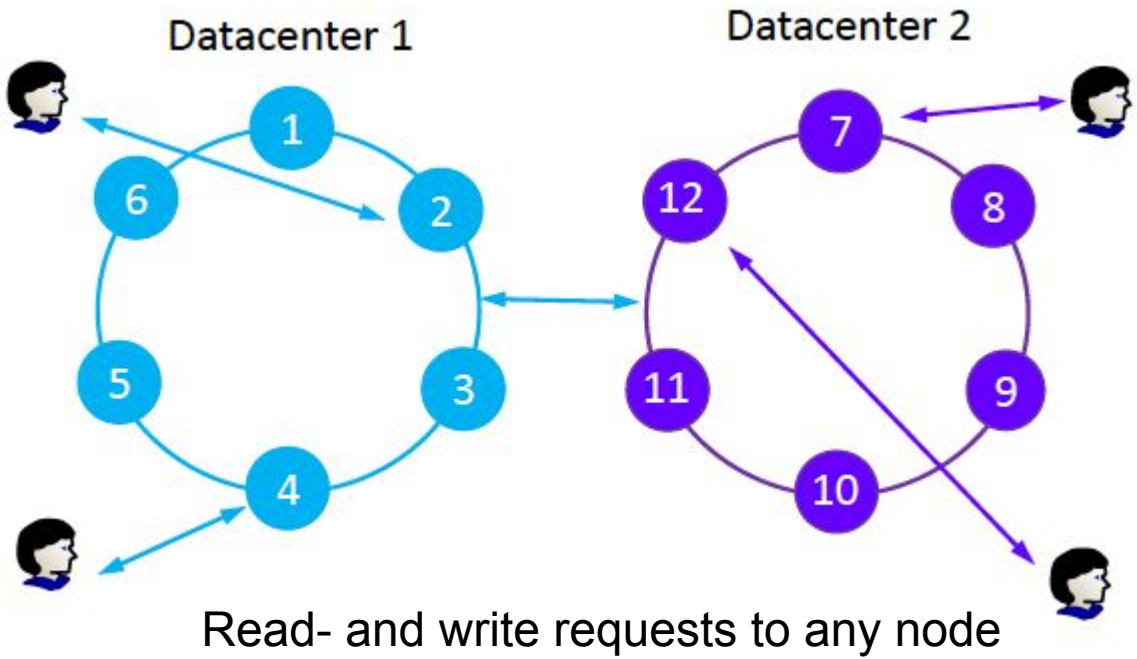


Cassandra is Distributed and Decentralised

Distributed: Capable of running on multiple machines

Decentralised: No single point of failure

Single cluster may run across geographically dispersed data centers



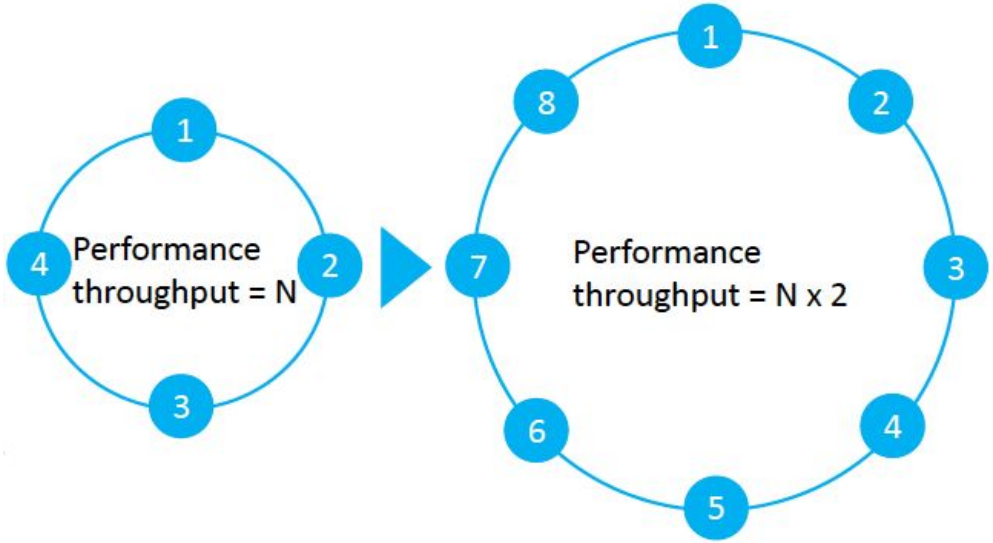
Read- and write requests to any node

Elastic Scalability

Scales horizontally, adding more machines

Adding nodes increase performance throughput (transactions/secs) linearly

De-/ and increasing the nodecount happens seamlessly (Automatic distribution & maintenance)



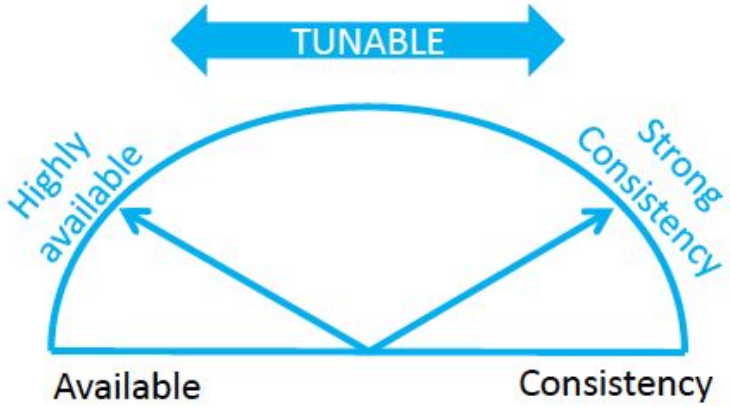
Linear scale performance. Addition of nodes produce predictable increase in performance

Tunable Data Consistency

Supports transactions with strong or eventual consistency

Adjustable for read-and write-operations separately

Conflicts are solved during reads, as focus lies on write-performance



Choose consistency according to application needs

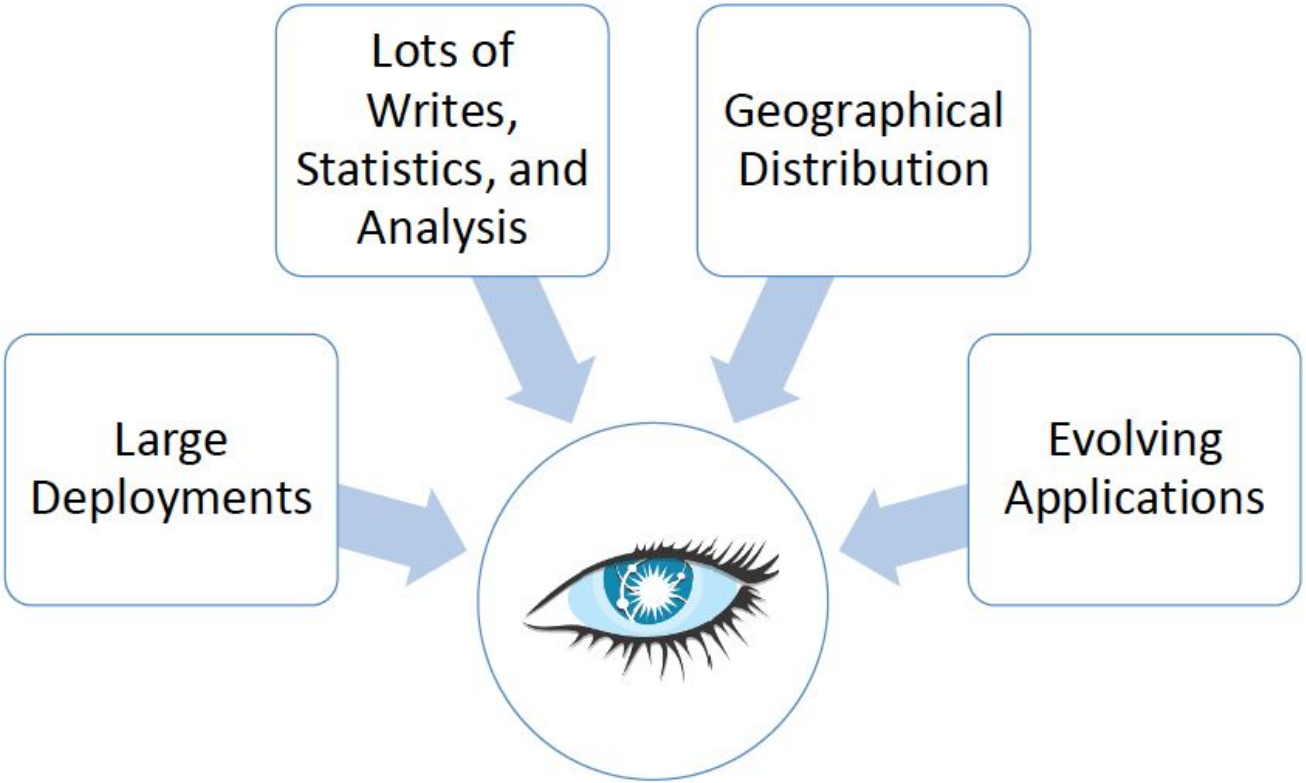
RDBMS	NoSQL DATABASE LIKE CASSANDRA
Moderate velocity data	High velocity data (devices, sensors, etc.)
Data coming in from one/few locations	Data coming in from many locations
Primarily structured data	Structured, with semi-unstructured
Complex/nested transactions	Simple transactions
Protect uptime via failover/log shipping	Protect uptime via architecture
High availability	Continuous availability
Deploy app central location/one server	Deploy app everywhere/many servers
Primary concern: scale reads	Scale writes and reads
Scale up for more users/data	Scale out for more users/data
Maintain data volumes with purge	High data volumes; retain forever

Comparing Characteristics Relational DBs vs Cassandra

Who is Using Cassandra?

Cassandra is being used by some of the biggest companies such as Facebook, Twitter, Cisco, Rackspace, ebay, Twitter, Netflix, and more.

Use cases



Cloud Applications

Internet of Things (IOT)

Product catalogs and retail apps

User activity tracking and monitoring

Messaging

Social media analytics and recommendation engines

Other time series based applications

Summary

- Revisiting
  - Why NoSQL?
  - Types of Data Model
- Origins of Column-family databases
- Casandra
  - Data Model
  - Query Language (CQL)
  - Architecture
  - Data Distribution (Partition)
  - Data Replication
  - Other features: elastic escalability, tunable consistency
  - Use cases