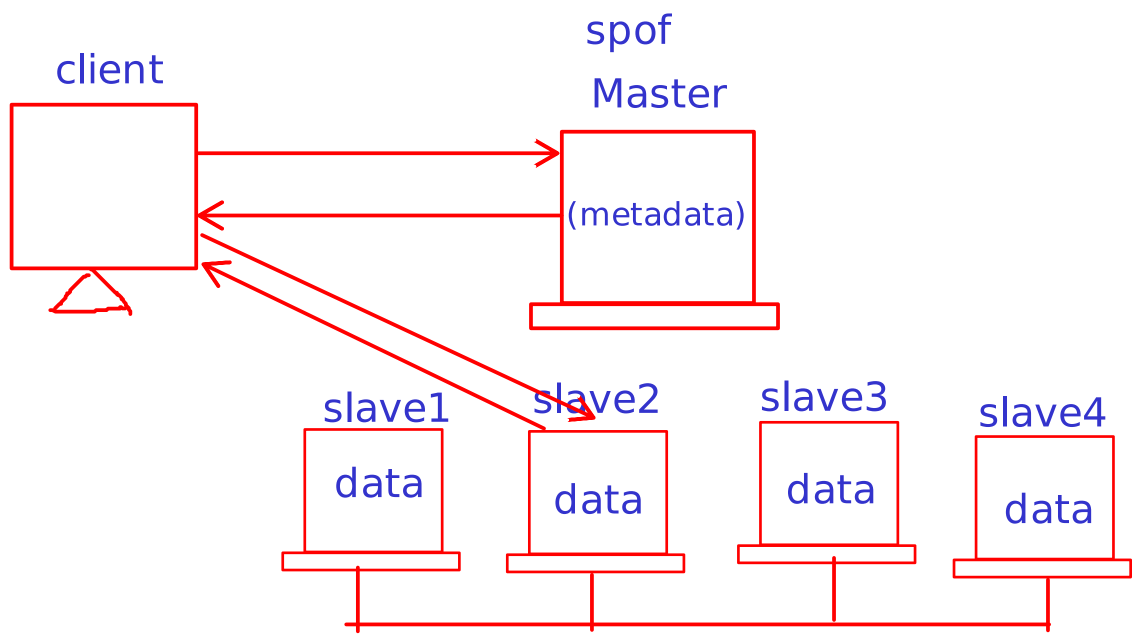
Cassandra

Introduction

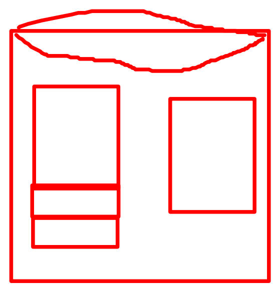
HDFS

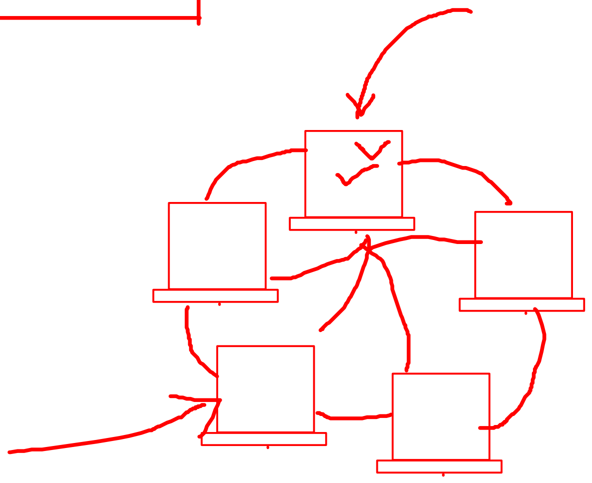
- Google BigTable

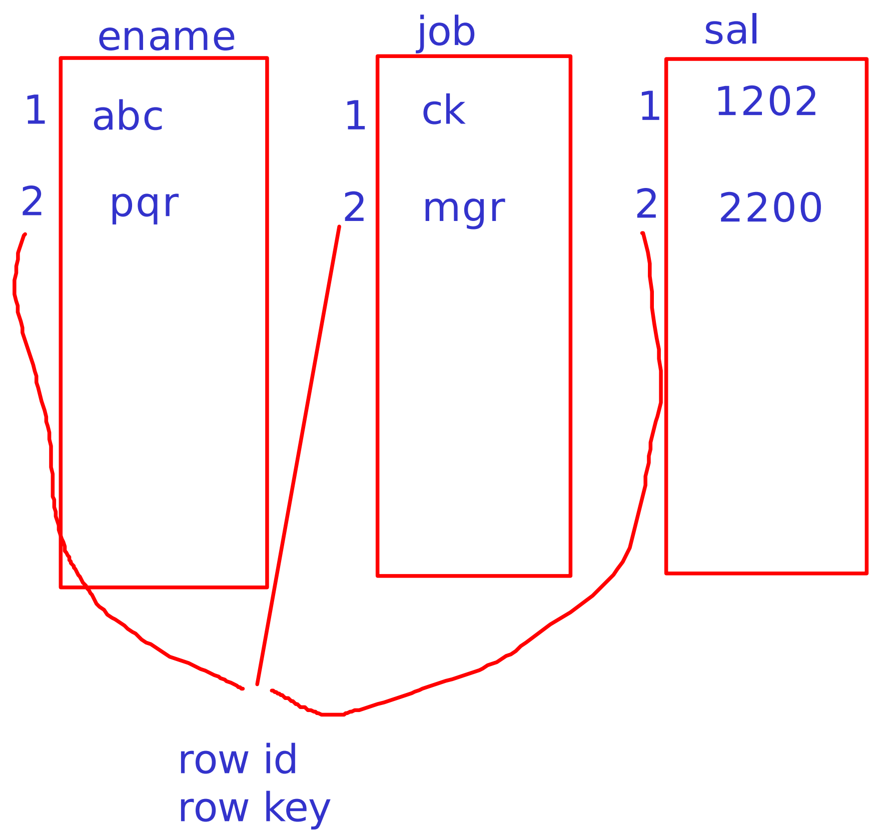
  google file system
  - High performance data storage system built on GFS and other Google technologies
  - Master-slave architecture
  - One key, multiple values
  - Columnar, SSTable (Sorted String Table) Storage, Append-only, Memtable, Compaction
- Amazon DynamoDb
  - Highly available and scalable key-value storage system
  - Decentralized peer to peer architecture
  - Compromise on consistency for better availability - Eventual consistency
  - Consistent hashing, Gossip protocol, Replication, Read repair
- Cassandra
  - Inherited from BigTable and DynamoDb
  - BigTable: Column families, Memtable, SSTable
  - DynamoDb: Consistent hashing, Partitioning, Replication



spof
client
Master
(metadata)
slave1
slave2
slave3
slave4
data
data
data
data







ename
job
sal
1
abc
1
ck
1
1202
2
pqr
2
mgr
2
2200
row id
row key

Introduction

- **Developed by**
  - Avinash Laxman (Co-inventor Amazon DynamoDb)
  - Prashant Malik (Technical Leader at Facebook)
- **Goals:**
  - Distributed NoSQL database (on commodity hardware)
  - Large amount of structured data
  - High availability
  - No single point of failure
- Basic data model is ~~rows & columns~~
- Column-oriented, Decentralized peer to peer & follow Eventual consistency
- Datastax company develop and support commercial edition of Cassandra

$$\begin{tikzcd}
\text{Facebook} \arrow[d] & & \\
\text{open-source (apache)} \arrow[rr] \arrow[dd] & & \text{commodity edition} \\
& & \text{cassandra} \arrow[u] \arrow[d] \\
\text{Datastax} \arrow[rr] \arrow[dr, bend right] & & \text{enterprise edition} \\
& \text{cloud (astra)} &
\end{tikzcd}$$

Cassandra Development

- Developed in Java ✓
- 2007-2008 - Developed at Facebook ✓
- July 2008 - Open sourced by Facebook ✓
- March 2009 - Apache Incubator project ✓
- February 2010 - Apache Top-level project ✓
- 2011 - version 0.8 - Added CQL ✓
- 2013 - version 2.0 - Added light-weight transactions, Triggers ✓
- 2015 - version 3.0 - Storage engine improved, Materialized views ✓
- 2017 -version 3.11 – bug fix from the last lelease ✓
- 2021 - version 4.0.5 ---> we r using this version ✓
- 2022 - version 4.1.0 - Latest release

Installation

- Prerequisite
  - Java 8 (Java 11 experimental)
- Can be installed through apt or yum tool (Ubuntu/CentOS)
- Manual installation
  - Download Cassandra 3.11.x (.tar.gz) and extract it
  - set CASSANDRA_HOME to Cassandra directory
  - set JAVA_HOME to JDK 8 directory
  - Install python 2.7 (for cqlsh)
  - set CASSANDRA_HOME/bin into PATH variable
  - Start Cassandra
- terminal1> cassandra
- terminal2> cqlsh

→ mapping

mysql	cassandra
database	keyspace
table	table
row	row
column	column

Features

- Peer to peer architecture no master-slave SPOF
- Linear scale performance capacity $\propto$ nodes
- High Performance $\longrightarrow$ high speed in read/write
- Simplified deployment and maintenance -> linux
- Less expensive => horiz scalability
- Supports multiple programming languages $\longrightarrow$ py , java ,c++ , Node.js, C#....., &REST service
- Operational and Development simplicity
- Cloud Availability AWS, Azur,GCP
- Ability to deploy across data centers
- Fault tolerant
- Configurable consistency (tight or eventual)
- Flexible data model
- Column family store

Limitations

- Aggregation operations are not supported
- Range queries on partition key are not supported
- Not good for too many joins
- Not suitable for transactional data ✓
- During compaction performance / throughput slows down
- Not designed for update-delete

Performance

- Performance measures
  - Throughput (operations per second) ↑
  - Latency (time required for one operation) ↓
- Cassandra vs MySQL
  - MySQL (more than 50GB data)
    - Write speed: 300 ms ✓
    - Read speed: 350 ms ✓
  - Cassandra (more than 50GB data)
    - Write speed: 0.12 ms ←
    - Read speed: 15 ms ←

Applications

v v v

- Applications
  - Product catalog/Playlist ✓
  - Recommendation/Personalization engine ✓
  - Sensor/IoT data ✓
  - Messaging/Time-series data ✓
  - Fraud detection
- Customers
  - Facebook, Netflix, eBay, Apple, Walmart, GoDaddy
- Application requirements
  - Store and handle time-series data ✓
  - Store and handle large volume of data ✓
  - Scale predictably (Linear Scaling) ✓
  - High availability ✓

uuid ->universally unique identifier
128bit -> 16byte

time-uuid = time stamp + uuid
(helpful in sorting
on time field)

Data Model

- Cassandra provides the Cassandra Query Language (CQL), an SQL-like language, to create and update database schema and access data
- CQL allows users to organize data within a cluster of Cassandra nodes using:
  - **Keyspace**
    - Defines how a dataset is replicated, per datacenter
    - Replication is the number of copies saved per cluster. Keyspaces contain tables
  - **Table**
    - Defines the typed schema for a collection of partitions. Tables contain partitions, which contain rows, which contain columns. Cassandra tables can flexibly add new columns to tables with zero downtime.
  - **Partition**
    - Defines the mandatory part of the primary key all rows in Cassandra must have to identify the node in a cluster where the row is stored
    - All performant queries supply the partition key in the query.
  - **Row**
    - Contains a collection of columns identified by a unique primary key made up of the partition key and optionally
    - additional clustering keys
  - **Column**
    - A single datum with a type which belongs to a row

CQL

- Users can access Cassandra through its nodes using Cassandra Query Language (CQL)
- CQL treats the database (Keyspace) as a container of tables
- Programmers use cqlsh: a prompt to work with CQL or separate application language drivers

Data Types

- ascii: US-ascii character string
- bigint: 64-bit signed long ints
- blob: Arbitrary bytes in hexadecimal
- boolean: True or False
- counter: Distributed counter values 64 bit
- decimal: Variable precision decimal
- double: 64-bit floating point
- float: 32-bit floating point
- frozen: Tuples, collections, $\boxed{\text{UDT}}$ containing CQL types
- inet - IP address in ipv4 or ipv6 string format
- int: 32 bit signed integer
- list: Collection 01 elements
- map: JSON style collection of elements
- set: Sorted collection of elements
- text: UTF-8 encoded strings
- timestamp: ID generated with
- date+time: as int/string
- timeuuid: Type 1 uuid
- tuple: A group of 2,3 fields
- uuid: Standard uuid (128-bit)
- varchar: UTF-8 encoded string
- varint: Arbitrary precision integer

Cassandra vs MongoDB: Differences

- Cassandra
  - Dev In java ✓
  - Column based ✓
  - Cassandra uses a traditional model with a table structure, using rows and columns. ✓
  - Cassandra offers an assortment of peers node
  - Used in-house query language, CQL ✓
  - no internal aggregation framework ✓✗
  - According to the CAP theorem Cassandra is an AP system.

- MongoDB
  - Dev in C++ ✓
  - Document based ✓
  - MongoDB employs an objective-oriented or data-oriented model
  - MongoDB uses a single master node.
  - queries are structured into JSON fragments
  - own aggregation framework ✓
  - According to the CAP theorem, MongoDB is a CP system