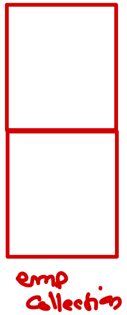
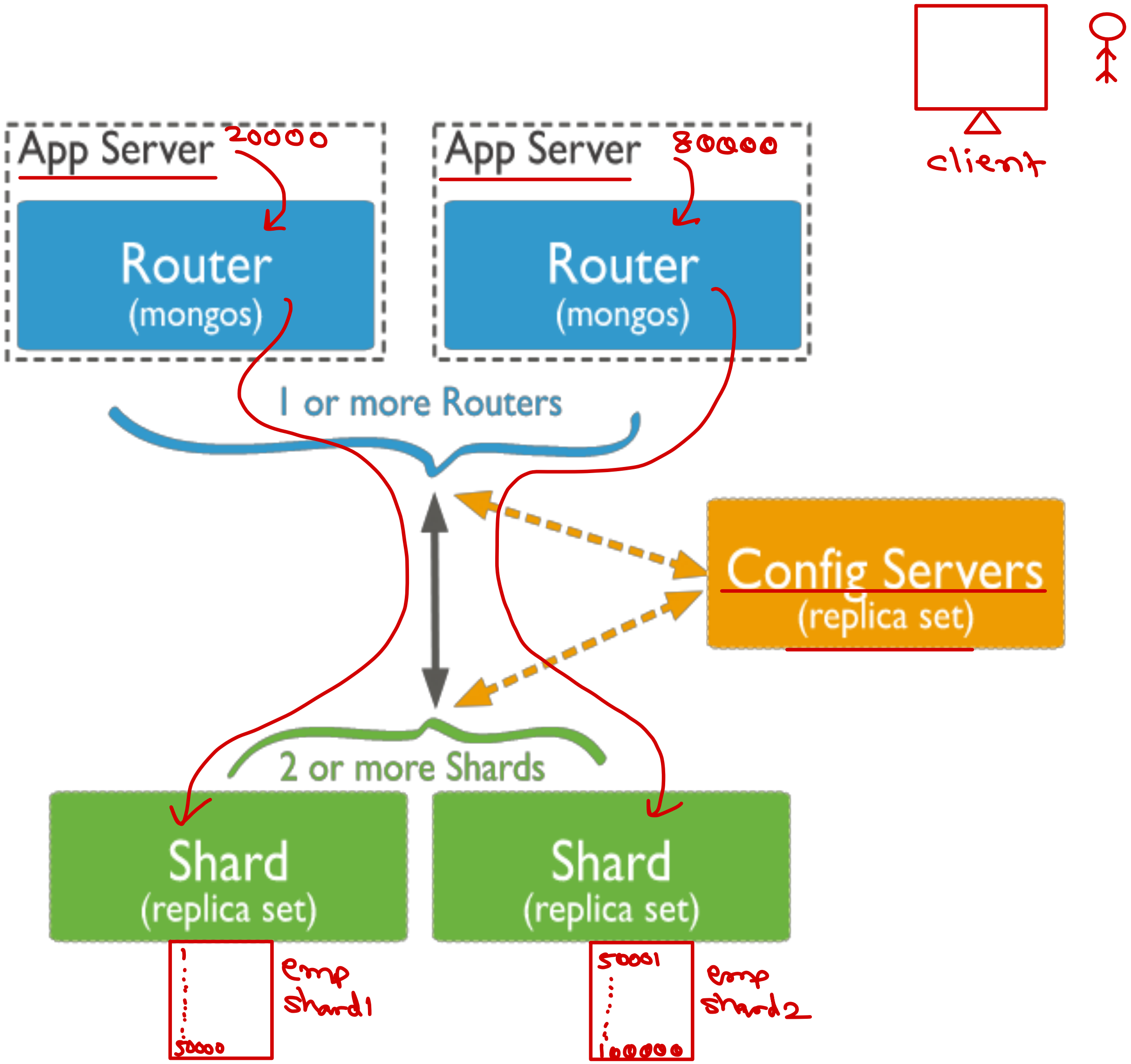
MongoDb Databases

Trainer: Mr. Nilesh Ghule

Mongo - Sharding

- Sharding is a method for distributing large data across multiple machines.
- This is mongodb approach for horizontal scaling/scaling out.
- shard: part of collection on each server (replica set).
- mongos: query router between client & cluster.
- config servers: metadata & config settings of cluster.



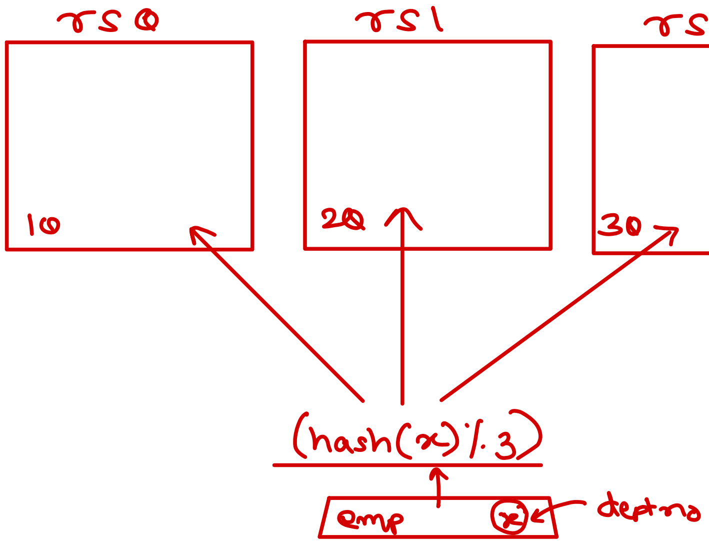


Mongo - Sharding

- Collections can be sharded across the servers based on shard keys.
- Shard keys:
  - Consist of immutable field/fields that are present in each document
  - Only one shard key. To be chosen when sharding collection. Cannot change shard key later.
  - Collection must have index starting on shard key.
  - Choice of shard key affect the performance.
- Advantages:
  - Read/Write load sharing
  - High storage capacity
  - High availability

Mongo - Sharding

- Sharding strategies:
  - Hashed sharding
    - MongoDB compute hash of shard key field's value.
    - Each chunk is assigned a range of docs based on hashed key.
    - Even data distribution across the shards. However range-based queries will target multiple shards.
  - Ranged sharding
    - Divides data into ranges based on shard key values.
    - mongos can target only those shards on which queried range is available.
    - Efficiency of sharding is based on choosing proper shard key.



Sunbeam Infotech

Redis

Redis - Introduction

key-value

- REmote DIctionary Server
- In-memory persistent open-source key-value store developed in 2009.
- Redis is maintained and developed by Salvatore Sanfilippo.
- Based on data structures: strings, ✓ hashes, ✓ sets, ✓ lists, ✓ sorted sets, ✓ geospatial indexes, hyperloglogs.
- Application/Uses:
  - Advanced key/value store as NoSQL.
  - Used as memory cache to improve application performance.
  - Message broker for real time message notifications.
  - Easy and efficient implementation of Data structures.

Redis - Features

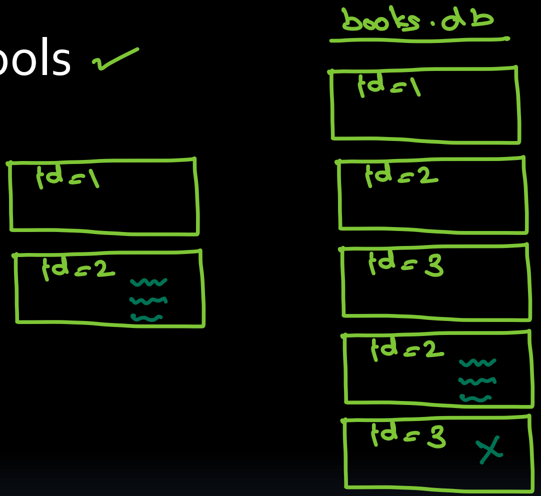
- Speed: 110000 SET/s and 81000 GET/s on entry-level Linux system.
- Pipeline: Multiple commands execution for faster execution.
- Persistence: Whole data accessed from memory, asynchronously persisted on disk with flexible policies.
- Data Structure: Based on data structures like Strings, Hashes, Sets, ...
- Atomic operations: Data is manipulated atomically by multiple clients.
- Supported Languages: Drivers available for C/C++, Java, Python, R, PHP, ...
- Master/Slave replication: Easy config and fast execution.
- Sharding: Distributing across cluster. Based on client driver capability.
- Portable: Developed in C. Work on all UNIX variants. Not supported on Win.

Redis - Highlights

- Key-value DB, where values can store complex data types with atomic ops.
- Value types are basic data structures made available to programmers without layers of abstraction.
- It is in-memory but persistent store i.e. whole database is maintained in server RAM, only changes are updated on disk for backup.
- The data storage in disk is in append-only data files.
- Maximum data size is limited to the RAM size.
- On modern systems if Redis is going out of memory, it will start swapping and slow down the system.
- Max memory limit can be configured to raise error on write or evict keys.

Redis - Installation

- Install: sudo apt-get install redis-server redis-tools ✓
- Run server: sudo systemctl start redis ✓
- Run client: redis-cli ✓
- redis> ping → PONG
- redis> INFO
- redis> CONFIG GET *
- redis> CONFIG GET loglevel ← key
- redis> CONFIG SET loglevel notice
  - loglevels: 0. debug, 1. verbose, 2. notice, 3. warning
- redis> KEYS *



Redis - Data Types & Commands

- Keys
  - Any binary sequence as key i.e. any string to any binary file.
  - Max key size is 512 MB. Very large key size is not good.
  - Set up convention for key e.g. users:1001:posts.november
- Data Types:
  - ✓ String: Basic type. (SET, GET, DEL) ✓
  - ✓ List: Ordered collection. (LPUSH, RPUSH, LPOP, RPOP, LREM, LRANGE).
  - ✓ Set: Ordered collection. Unique values. (SADD, SMEMBERS, SISMEMBER).
  - ✓ Sorted Set: Sorted collection. Unique. Each value have score value (float) for sorting. (ZADD, ZRANGE).
  - ✓ Hashes: Object with multiple fields. (HMSET, HGETALL, HMGET)

Redis - Publish/Subscribe

- PSUBSCRIBE channel-pattern
  - receive notifications from given channels. e.g. b?g, b*g, b[ai]g
- PUBLISH channel "message"
  - send message to channel
- PUNSUBSCRIBE channel-pattern
  - stop receiving notifications from given channels.
- UNSUBSCRIBE channel
  - stop receiving notifications from given channel.
- PUBSUB *command*
  - monitor pub-sub subsystem
  - e.g. PUBSUB channels

Redis-Transactions & Pipeline

- Transaction:
  - Puts multiple commands in a queue and execute them at once.

    at server side
  - MULTI: begin transaction
  - All commands after this are queued.
  - EXEC: execute all commands from start of transaction
  - DISCARD: discard all commands from start of transaction
- Pipeline:
  - Client sends multiple commands to server in a batch.
  - Saves network round-trip each time.
  - All commands may not execute in a transaction.
  - echo -en "PING\r\nSET key value\r\nGET key\r\n" | nc localhost 6379

    netcat

192 . 168 . 0 . 1

xxx . xxx . xxx . xxx 32-bit : IPv4

0-255

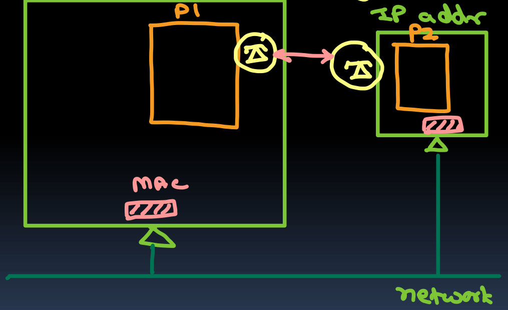
128-bit : IPv6

IP addr

socket = ip addr + port

std code region

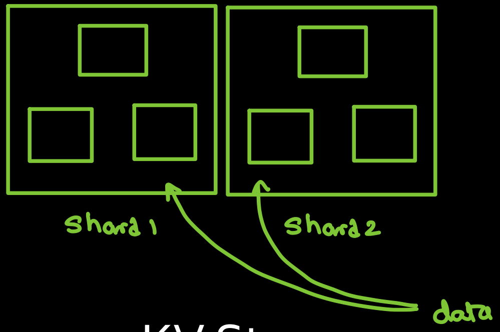
phone no.



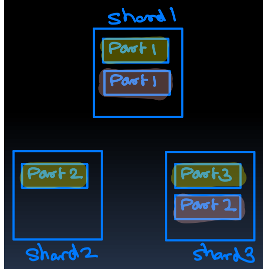
SunBeam Infotech

Oracle NoSQL - KVStore

Introduction

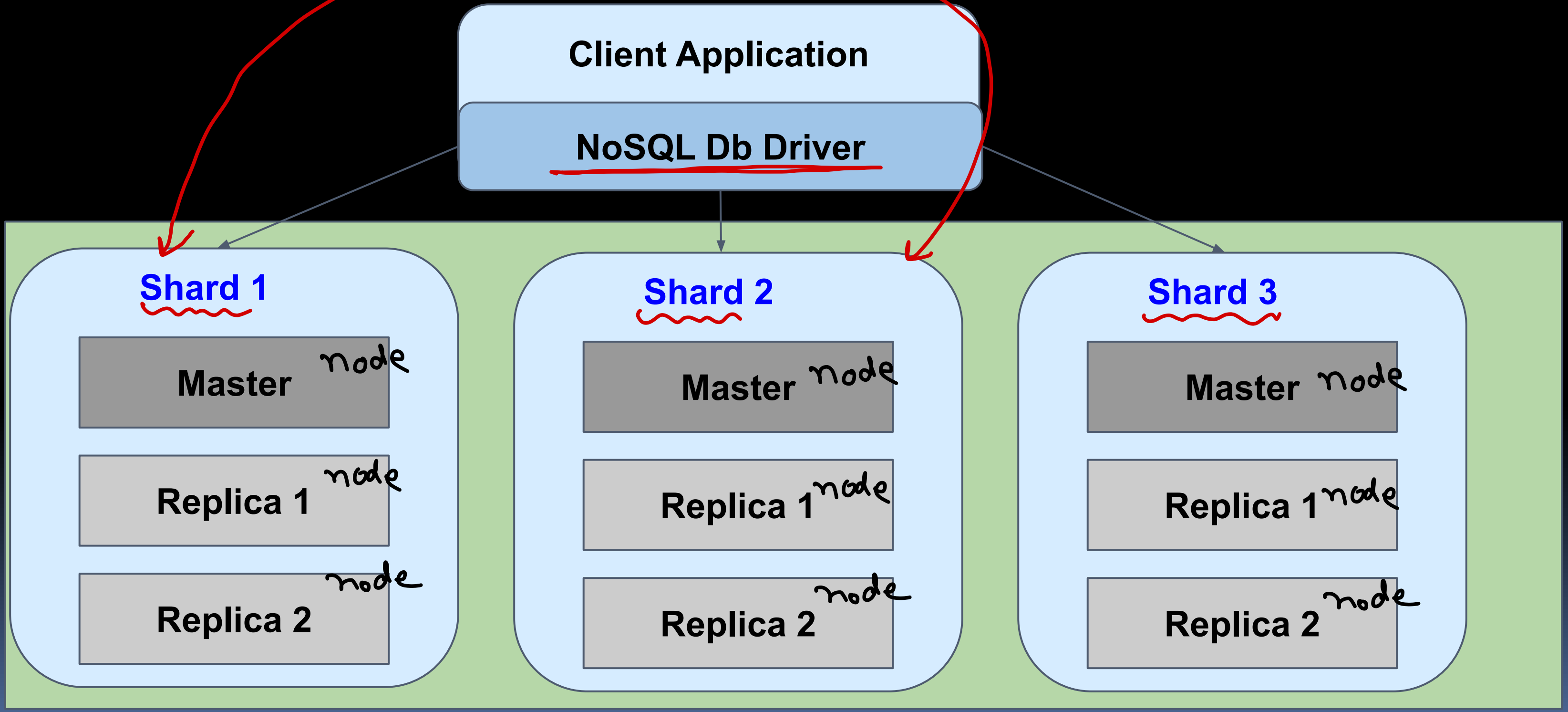


- Multi-terabyte distributed key-value pair storage → KV Store
- High performance, scalable, Eventual consistency, Durable.
- User defined read/write performance levels.
- Terminologies: string /dbda/01/-/name /dbda/01/-/email "James Bond" "james@bond.com" value any thing
  - KV Pair → Key : Major & Minor keys, Value : byte array
  - KV Store → Container of KV pairs
  - Partition → Hashed Set of Records (on major keys)
  - Shard → Set of partitions. Group of machines for replication. Shard is chosen transparently i.e. auto selected by oracle nosql db.
  - Replication factor → Number of replicas. Default is 3.
  - Storage node → Physical machine for storing data (CPU+RAM+Disk).



Architecture

Key | Value
/dbda/1/-/name | Jamey
/dbda/1/-/addr | UK
/dbda/2/-/name | Bill



Consistency

- Related to update operation.
- Eventual consistency.
- Trade-off between : Speed & Availability
- Write transaction durability consists of Sync Policy & Replica Ack:
    - Sync Policy:
        - ✓ Sync (to disk) → Most Durable
        - ✗ Write No Sync (to OS buffer) → Moderate
        - ✗ No Sync (local log buffer - flush when convenient) → Fastest
    - Replica Ack Policy:
        - All → slower
        - Simple Majority (majority of nodes)
        - None → fastest

Consistency

- Read consistency:
  - Absolute (from Master)
    - Most Consistent : Most recent version
  - Time based (from replica within time-interval of Master)
    - Data of known version or later
  - Version (from replica with current/higher version of transaction token)
    - Recent data for given time
  - None (any replica)
    - Fastest : Can read stale data

Installation

- Download kv-ce-4.3.11.tar.gz and extract to some directory → kv-ce-4.3.11
- Edit ~/.bashrc
  - export KVHOME=<path to kv-ce-4.3.11>
  - export KVROOT=<path to kv-ce-4.3.11/kvroot>
- Start kvstore and test it.
  - java -jar $KVHOME/lib/kvstore.jar kvlite -verbose -root $KVROOT -store kvstore -host $HOSTNAME -port 5000 -secure-config disable
  - java -jar $KVHOME/lib/kvstore.jar ping -verbose -host $HOSTNAME -port 5000
  - java -jar $KVHOME/lib/kvstore.jar runadmin -verbose -host $HOSTNAME -port 5000 -store kvstore

KV CLI :: kv ->

- show versions
- show topology
- verify
- history
- put kv -key <key> -value <value>
- get kv -key <key>
- get kv -key <key> -all
- delete kv -key <key>
- delete kv -key <key> -all

major minor

/dbda/1/-/name

James

/dbda/1/-name

/dbda/1/

cassandra

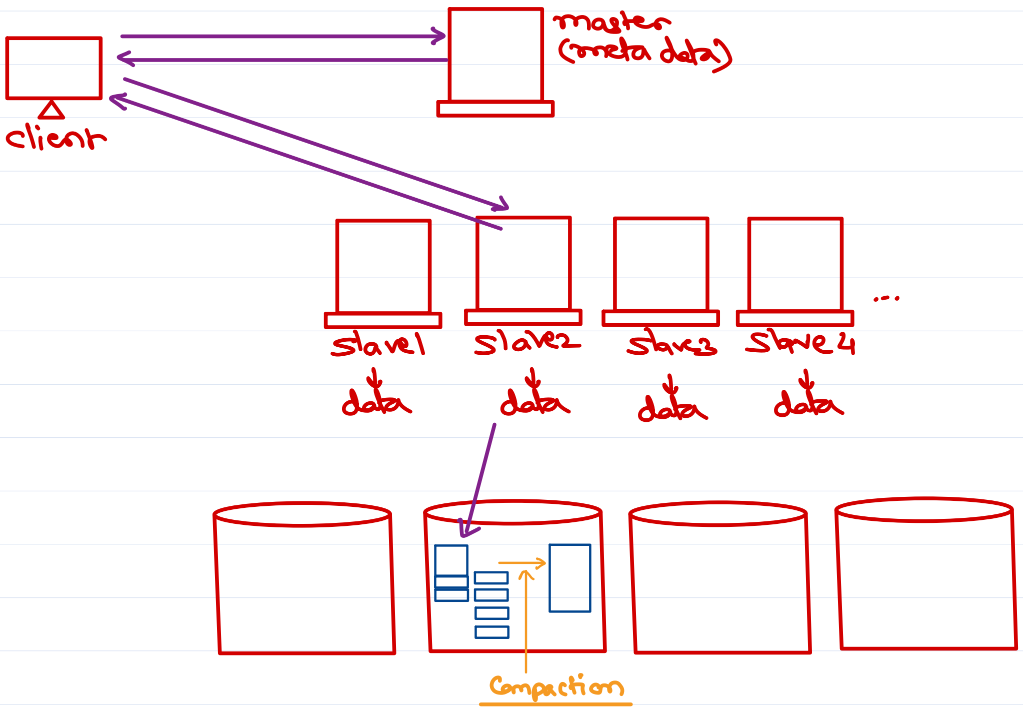
Sunbeam Infotech

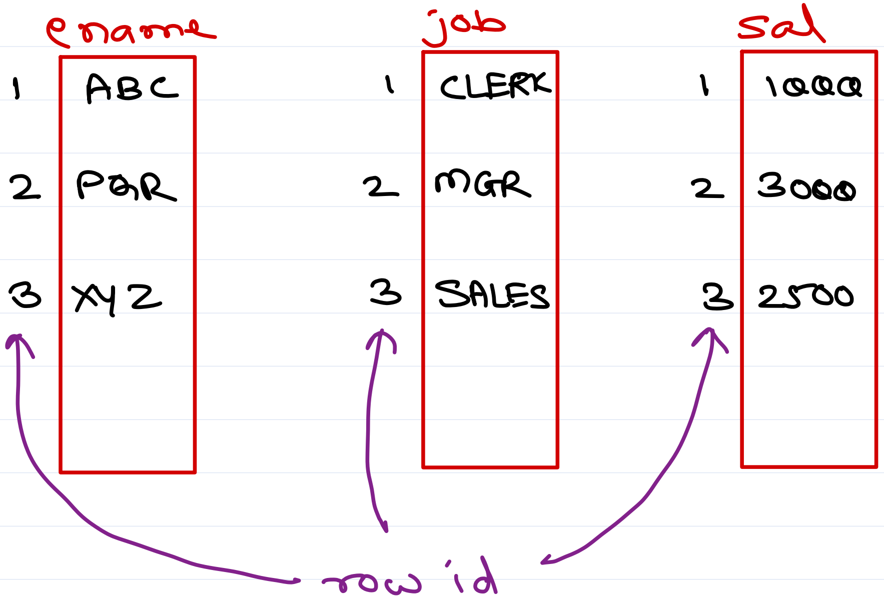
Introduction

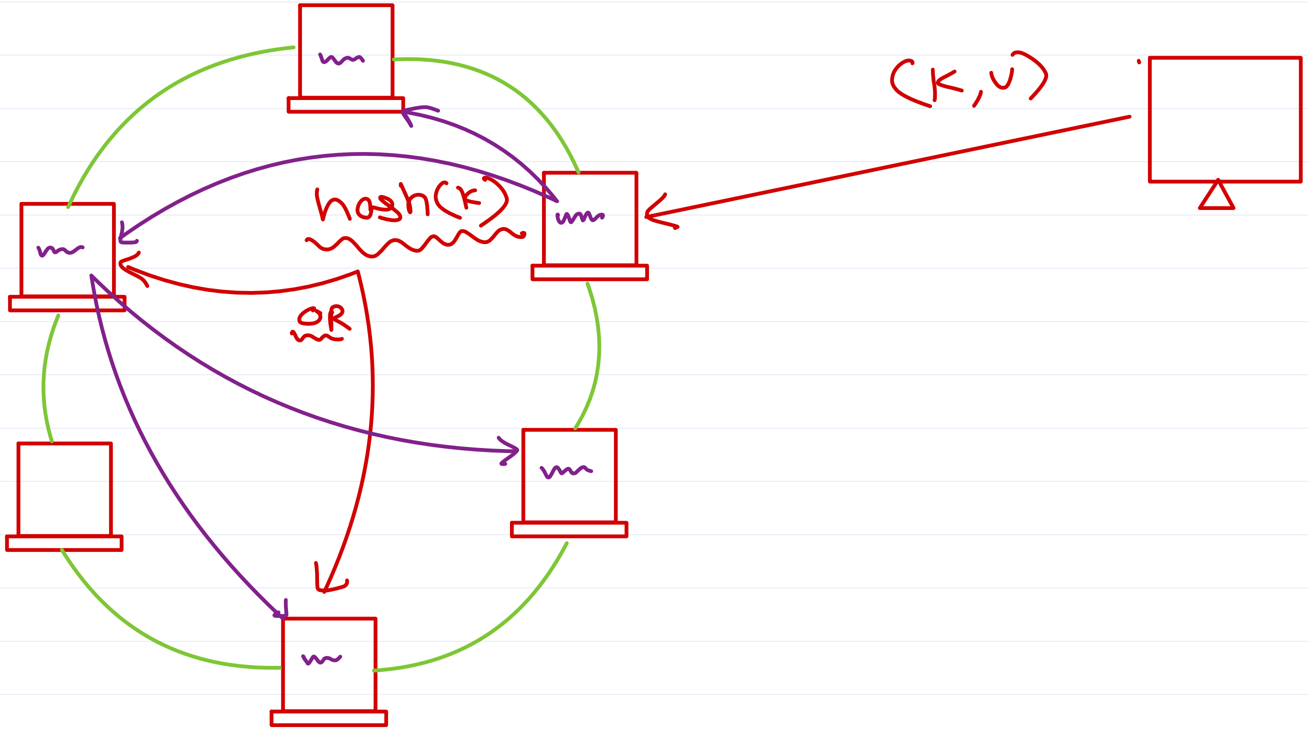
- Google BigTable

  Google File System (Distributed File System)
  HDFS follow GFS concepts.

  - 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

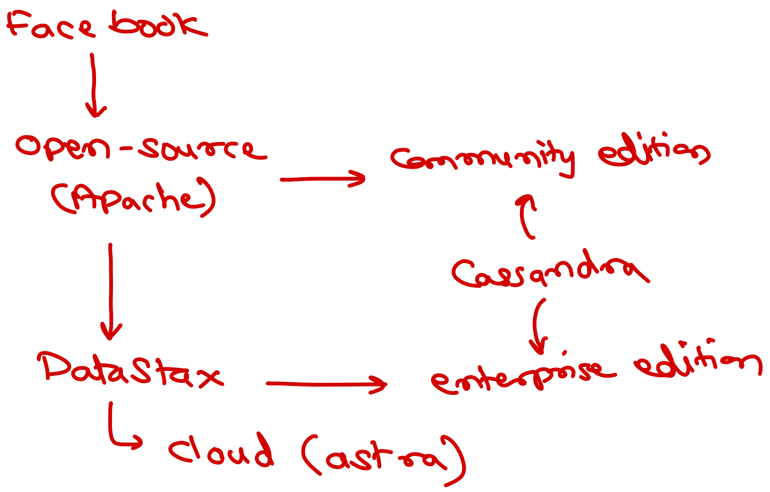

master
(meta data)
client
slave1
slave2
slave3
slave4
...
data
data
data
data
Compaction


ename
job
sal
1
ABC
2
PQR
3
XYZ
1
CLERK
2
MGR
3
SALES
1
1000
2
3000
3
2500
row id


(K,V)
hash(K)
OR

Concept

- 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.



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.
- 2020 - version 3.11 - Latest release.

Cassandra 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
    - terminal 1> cassandra
    - terminal 2> cqlsh

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>