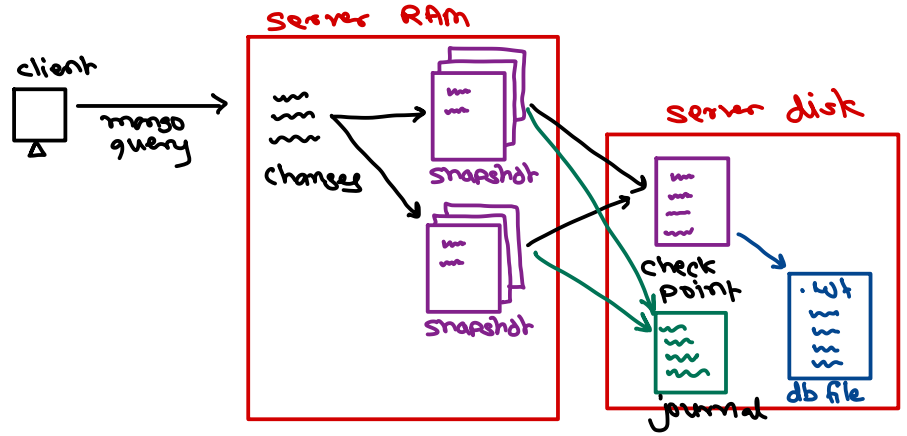
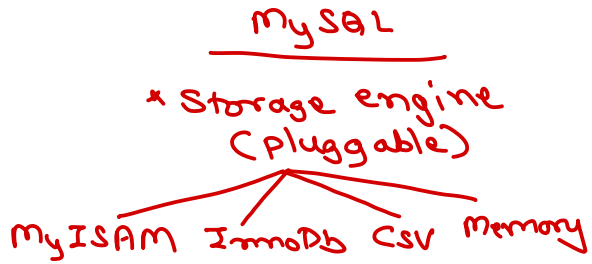
MongoDb Databases

Trainer: Mr. Nilesh Ghule

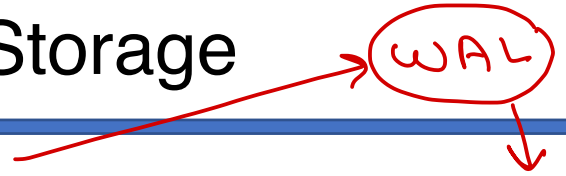
Mongo - WiredTiger Storage

- Storage engine is managing data in memory and on disk.
- MongoDB 3.2 onwards default storage engine is *WiredTiger*; while earlier version it was MMAPv1.
- WiredTiger storage engine:
  - Uses document level optimistic locking for better performance.
  - Per operation a snapshot is created from consistent data in memory.
  - The snapshot is written on disk, known as checkpoint → for recovery.
  - Checkpoints are created per 60 secs or 2GB of journal data.
  - Old checkpoint is released, when new checkpoint is written on disk and updated in system tables.
  - To recover changes after checkpoint, enable journaling.





Mongo - WiredTiger Storage



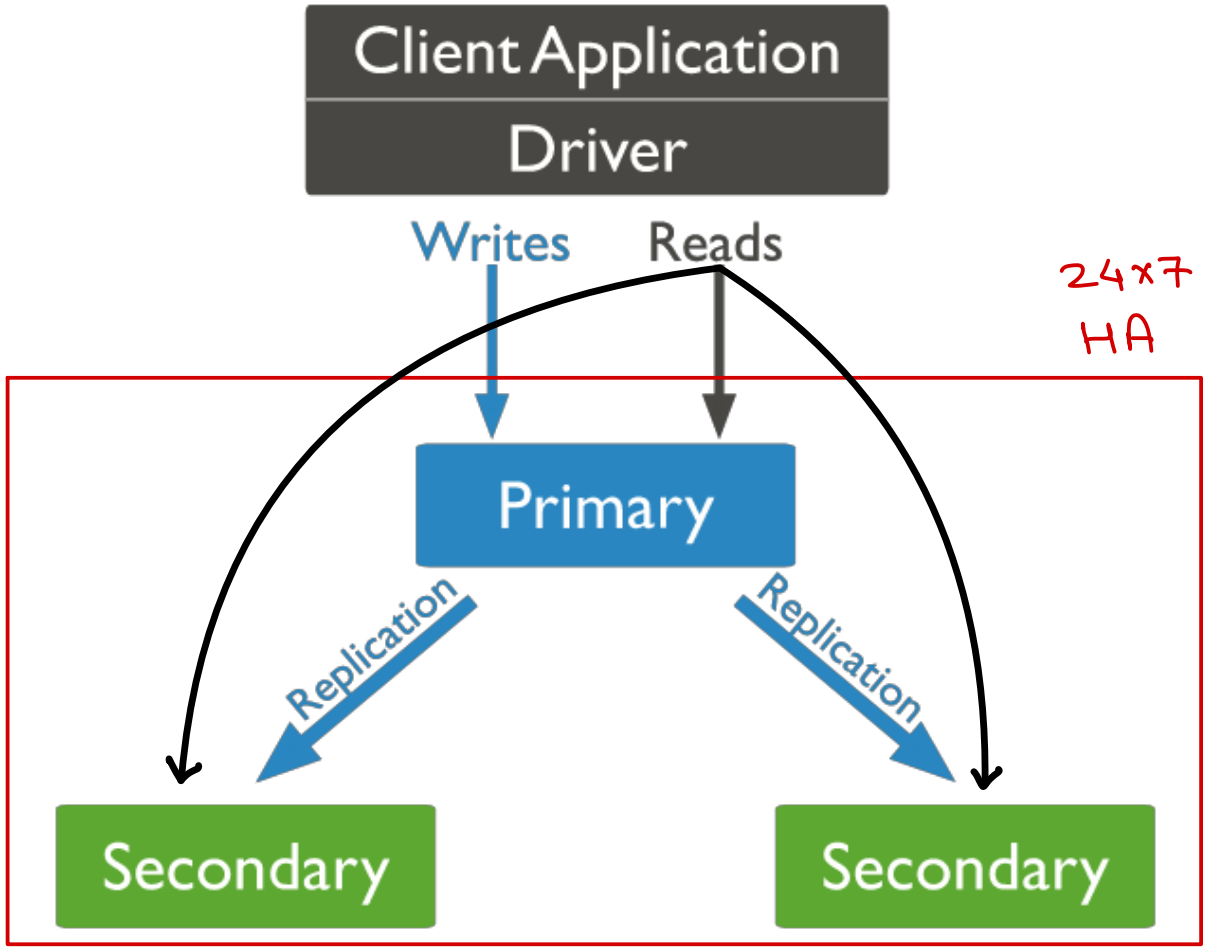
- WT uses write-ahead transaction in journal log to ensure durability.
- It creates one journal record for each client initiated write operation.
- Journal persists all data modifications between checkpoints.
- Journals are in-memory buffers that are synced on disk per 50 ms.
- WiredTiger stores all collections & journals in compressed form.
- Recovery process with journaling:
  - Get last checkpoint id from data files.
  - Search in journal file for records matching last checkpoint.
  - Apply operations in journal since last checkpoint.
- WiredTiger use internal cache with size max of 256 MB and 50% RAM - 1GB along with file system cache.

RAM 8 GB

Cache → $\frac{8\ GB}{2} - 1\ GB = 3\ GB$

Mongo - Replication

- A replica set is a group of mongod instances that maintain the same data set.
- Only one member is deemed the primary node, while other nodes are deemed secondary nodes.
- The secondaries replicate the primary's oplog.
- If the primary is unavailable, an eligible secondary will become primary.

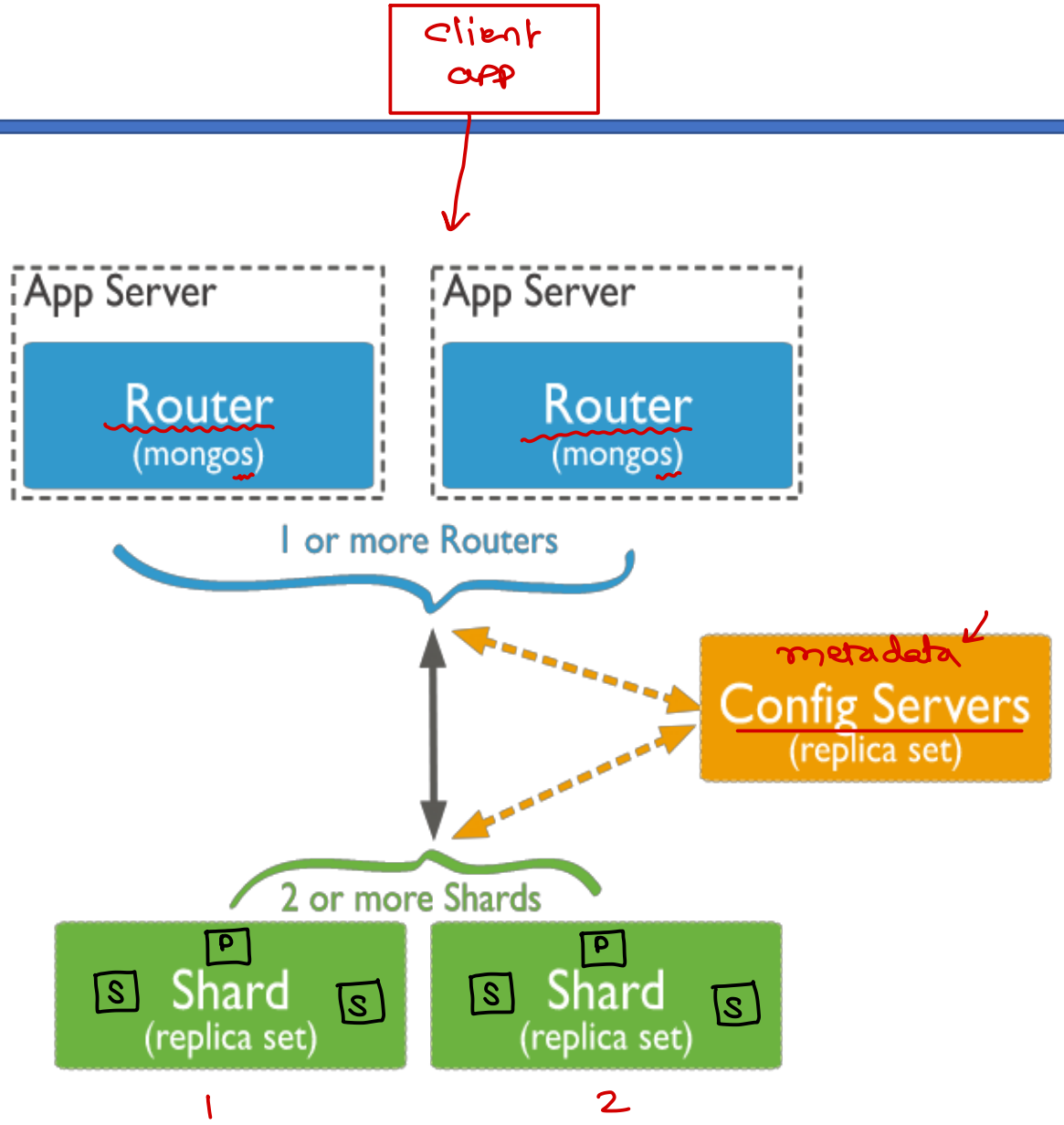


Mongo - Replication

- Secondary servers communicate with each other via heart-beat.
- Secondary applies operations from primary asynchronously.
- When primary cannot communicate a secondary for more than 10 seconds, secondary will hold election to elect itself as new primary. This automatic failover process takes about a minute.
- An arbiter (do not store data) can be added in the system (with even number of secondaries) to maintain quorum in case of election.
- By default client reads from primary, but can set read preference from secondary. Reading from secondary may not reflect state of primary; as read from primary may read before data is durable.

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 → replica

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

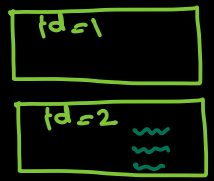
- Speed: 110000 SET/s and 81000 GET/s on entry-level Linux system.
- Pipeline: Multiple commands execution for faster execution.
- Persistence: Whole data accessed from memory, asynchronously persisted on disk with flexible policies.
- Data Structure: Based on data structures like Strings, Hashes, Sets, ...
- 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.

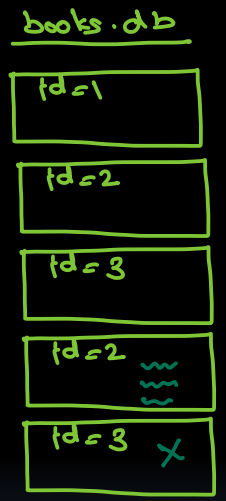
Redis - Highlights

- 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
- redis> INFO
- redis> CONFIG GET *
- redis> CONFIG GET loglevel
- redis> CONFIG SET loglevel notice
  - loglevels: 0. debug, 1. verbose, 2. notice, 3. warning
- redis> KEYS *





Redis - Data Types & Commands

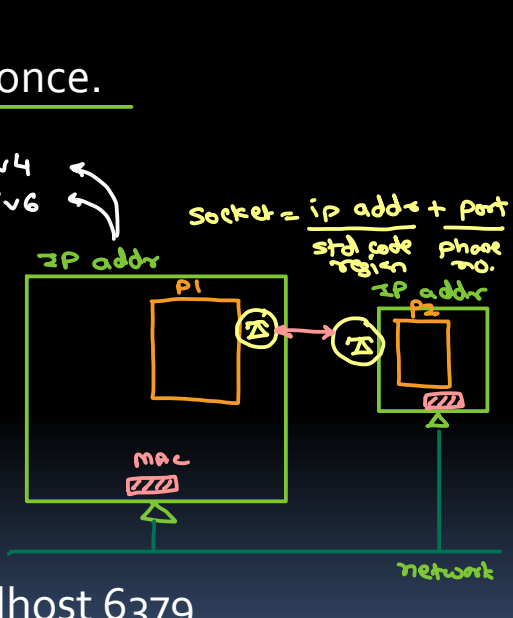
- Keys
  - Any binary sequence as key i.e. any string to any binary file.
  - Max key size is 512 MB. Very large key size is not good.
  - Set up convention for key e.g. users:1001:posts.november
- Data Types:
  - String: Basic type. (SET, GET, DEL)
  - List: Ordered collection. (LPUSH, RPUSH, LPOP, 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

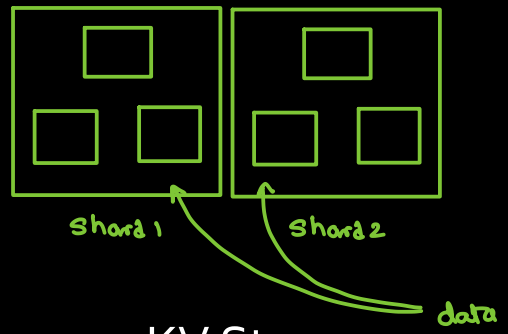
- Transaction:
  - Puts multiple commands in a queue and execute them at once.
  - MULTI: begin transaction
  - All commands after this are queued.
  - EXEC: execute all commands from start of transaction
  - DISCARD: discard all commands from start of transaction
- Pipeline:
  - Client sends multiple 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



SunBeam Infotech

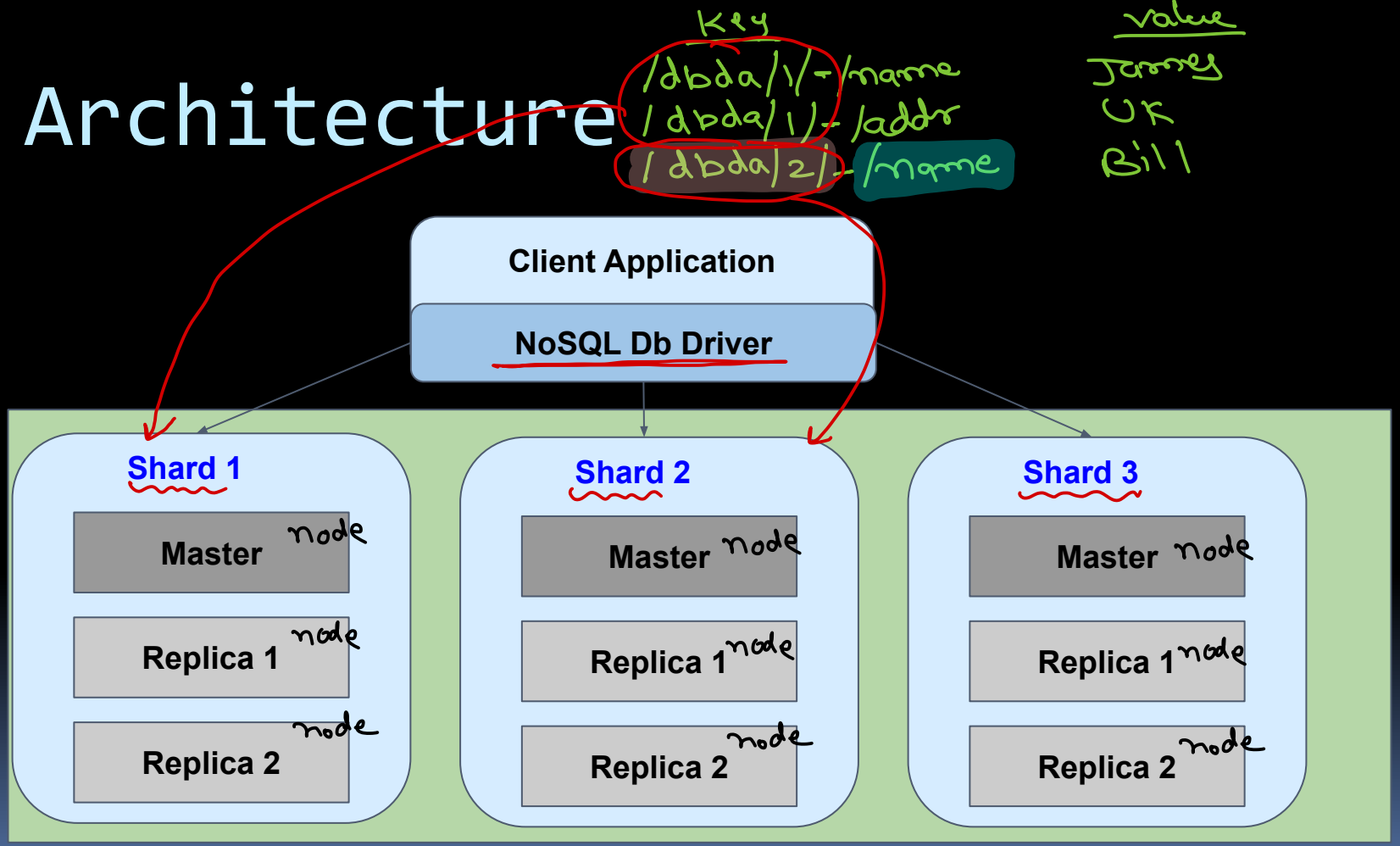
Oracle NoSQL - KVStore

Introduction



- Multi-terabyte distributed key-value pair storage → KV Store
- High performance, scalable, Eventual consistency, Durable.
- User defined read/write performance levels.
- Terminologies:
  - 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



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
    - Simple Majority (majority of nodes)
    - None

Consistency

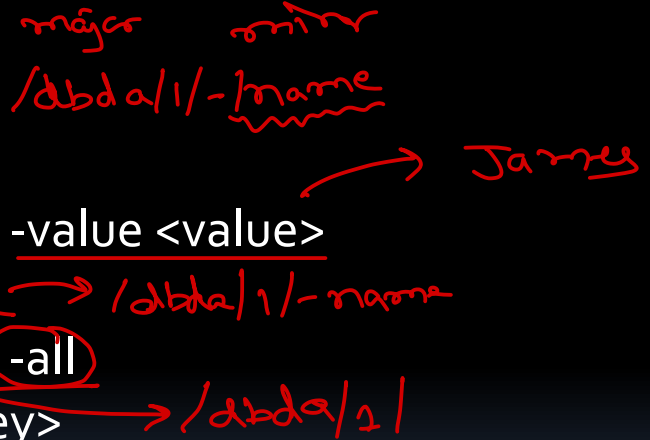
- Read consistency:
  - Absolute (from Master)
    - Most Consistent : Most recent version
  - Time based (from replica within time-interval of Master)
    - Data of known version or later
  - Version (from replica with current/higher version of transaction token)
    - Recent data for given time
  - 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



Nilesh Ghule

Thank you!