KEY-VALUE DATABASE

REDIS

Yunghans Irawan (yirawan@nus.edu.sg)

Agenda

- Key-Value Database
- Redis
- Data Manipulation
- Transaction support
- Persistence
- Replication and Failover
- Summary

Key-Value Database

- Non relational database that stores data as key and value pair
  - Analogy: primary key and the entire record
- Simple structure allow various tradeoffs towards other quality attributes
  - Performance
  - Scalability

Key Value Databases

Name	Initial Release	Latest Version	License
Redis	2009	4.0.10 June 2018	Open Source
Memcached	2003	1.5.9 July 2018	Open Source
Riak	2009	2.1.0 Apr 2015	Open Source
Ehcache	2009	3.3.0 Feb 2017	Open Source

Redis

- REmote DIctionary Server
- Written in C by Salvatore Sanfilippo in 2006

Redis keys

- Can use any binary sequence as key
  - string
  - the content of a JPEG file
  - empty string
- The longer the key, the more expensive its key-comparisons
- Keys should be readable by sticking to some schema
  - Example: "user:1000:followers"
- The maximum allowed key size is 512 MB.

Redis Feature

- Fast read and write operations are very fast
- Support data persistence to disk
- Configurable key expiration
- Support Transactions
- Support Publish/ Subscribe
- Provides Lua scripting to extend Redis to create new commands

Data Types

- Strings
  - Can be treated
- Lists
  - collections of string elements sorted according to the order of insertion
- Sets
  - collections of unique, unsorted string elements.
- Sorted sets
  - similar to Sets but where every string element is associated to a floating number value, called *score*. The elements are always taken sorted by their score, so unlike Sets it is possible to retrieve a range of elements

Data Types

- Hashes
  - maps composed of fields associated with values
- Bit arrays
  - you can set and clear individual bits, count all the bits set to 1, find the first set or unset bit, and so forth
- HyperLogLogs
  - probabilistic data structure which is used in order to estimate the cardinality of a set

Commands (Strings)

- SET

```
MSET first "First Key value" second "Second Key value“
OK
```

- GET

```
MGET first second
1) "First Key value“
2) "Second Key value"
```

- EXPIRE

```
EXPIRE current_chapter 10
(integer) 1
```

Commands (Strings)

- TTL
  - Returns how many more seconds to live
  - -2 means key is expired or does not exist
  - -1 no expiration time set

```
TTL current_chapter
(integer) 3
```

- INCR

```
> SET counter 100
OK
> INCR counter
(integer) 101
```

- INCRBY

```
> INCRBY counter 5
(integer) 106
```

Commands (Strings)

- DECR and DECRBY

```
> DECR counter
(integer) 105
> DECRBY counter 100
(integer) 5
```

- INCRBYFLOAT
  - Can pass negative value

```
> INCRBYFLOAT counter 2.4
"7.4"
> INCRBYFLOAT counter -0.3
"7.1"
```

Commands (List)

- LPUSH
  - Insert data at the beginning of a list

```
> LPUSH books "Clean Code“
(integer) 1
```

- RPUSH
  - Insert data at the end of a list

```
> RPUSH books "Code Complete"
(integer) 2
> LPUSH books "Peopleware“
(integer) 3
```

- LLEN
  - Return the length of a list

```
> LLEN books
(integer) 3
```

Commands (List)

- LINDEX
  - Return elements in a given list

Index	Returns
0	First element
1	Second element
-1	Last element
-2	Second last element

```
> LINDEX books 1
"Clean Code"
```

- LRANGE
  - returns an array with all elements from a given index range

```
> LRANGE books 0 1
1) "Peopleware"
2) "Clean Code“
> LRANGE books 0 -1
1) "Peopleware"
2) "Clean Code“
3) "Code Complete"
```

Commands (List)

- LPOP and RPOP
  - Read and removes data at the start/end of a list

```
> LPOP books
"Peopleware"
> RPOP books
"Code Complete“
> LRANGE books 0 -1
1) "Clean Code"
```

Commands (Hash)

- HSET
  - Set a value to a field of an element of given key

```
> HSET movie "title" "The Godfather"
(integer) 1
```

- HMSET
  - Set multiple field values for a key

```
> HMSET movie "year" 1972 "rating" 9.2 "watchers" 10000000
OK
```

- HINCRBY
  - Increment a field by a given integer

```
> HINCRBY movie "watchers" 3
(integer) 10000003
```

Commands (Hash)

- HSET
  - Set a value to a field of an element of given key

```
> HSET movie "title" "The Godfather"
(integer) 1
```

- HMSET
  - Set multiple field values for a key

```
> HMSET movie "year" 1972 "rating" 9.2 "watchers" 10000000
OK
```

- HINCRBY
  - Increment a field by a given integer

```
> HINCRBY movie "watchers" 3
(integer) 10000003
```

- HINCRBYFLOAT
  - Increment a field by a float

Commands (Set)

- SADD
  - Add one or many members to a set

```
> SADD user:max:favorite_artists "Arcade Fire" "Arctic
Monkeys" "Belle & Sebastian" "Lenine"
(integer) 4
> SADD user:hugo:favorite_artists "Daft Punk" "The Kooks"
"Arctic Monkeys"
(integer) 3
```

- SINTER
  - Return the intersection of two sets

```
> SINTER user:max:favorite_artists user:hugo:favorite_artists
1) "Arctic Monkeys"
```

Commands (Set)

- SDIFF
  - Returns all members in the first set that does not exist in the sets that follows it

```
> SDIFF user:max:favorite_artists user:hugo:favorite_artists
1) "Belle & Sebastian"
2) "Arcade Fire“
3) "Lenine“
> SDIFF user:hugo:favorite_artists user:max:favorite_artists
1) "Daft Punk"
2) "The Kooks"
```

Commands (Set)

- SUNION
  - Returns all members of all sets

```
> SUNION user:max:favorite_artists user:hugo:favorite_artists
1) "Lenine"
2) "Daft Punk“
3) "Belle & Sebastian“
4) "Arctic Monkeys“
5) "Arcade Fire“
6) "The Kooks"
```

Commands (Set)

- SRANDMEMBER
  - Add one or many members to a set

```
> SRANDMEMBER user:max:favorite_artists
"Arcade Fire"
```

- SISMEMBER
  - Whether a member exists in a set

```
> SISMEMBER user:max:favorite_artists "Arctic Monkeys"
(integer) 1
```

- SREM
  - Removes and returns members from a set

```
> SREM user:max:favorite_artists "Arctic Monkeys"
(integer) 1
```

Commands (Set)

- SMEMBERS
  - Returns all members of a set

```
> SMEMBERS user:max:favorite_artists
1) "Belle & Sebastian"
2) "Arcade Fire“
3) "Lenine"
```

Commands (Sorted Set)

- ZADD
  - Add a score and a value to a set

```
> ZADD leaders 100 "Alice"
(integer) 1
> ZADD leaders 100 "Zed“
(integer) 1
> ZADD leaders 102 "Hugo“
(integer) 1
> ZADD leaders 101 "Max“
(integer) 1
```

Commands (Sorted Set)

- ZRANGE
  - Returns elements from the lowest to the highest score and then the value in ascending order
- ZREVRANGE
  - Return elements from the highest to the lowest score and then the value in descending order

```
> ZREVRANGE leaders 0 -1
1) "Hugo"
2) "Max“
3) "Zed“
4) "Alice“
```

```
> ZREVRANGE leaders 0 -1 WITHSCORES
1) "Hugo"
2) "102“
3) "Max“
4) "101“
5) "Zed“
6) "100“
7) "Alice“
8) "100"
```

Commands (Sorted Set)

- ZRANGEBYLEX and ZREVRANGEBYLEX
  - Returns elements based on its value in ascending and descending order respectively
- ZRANGEBYSCORE and ZREVRANGEBYSCORE
  - Returns elements based on its score in ascending and descending order respectively
- ZREM
  - Removes a member from a sorted set

```
> ZREM leaders "Hugo“
(integer) 1
> ZREVRANGE leaders 0 -1
1) "Max"
2) "Zed“
3) "Alice"
```

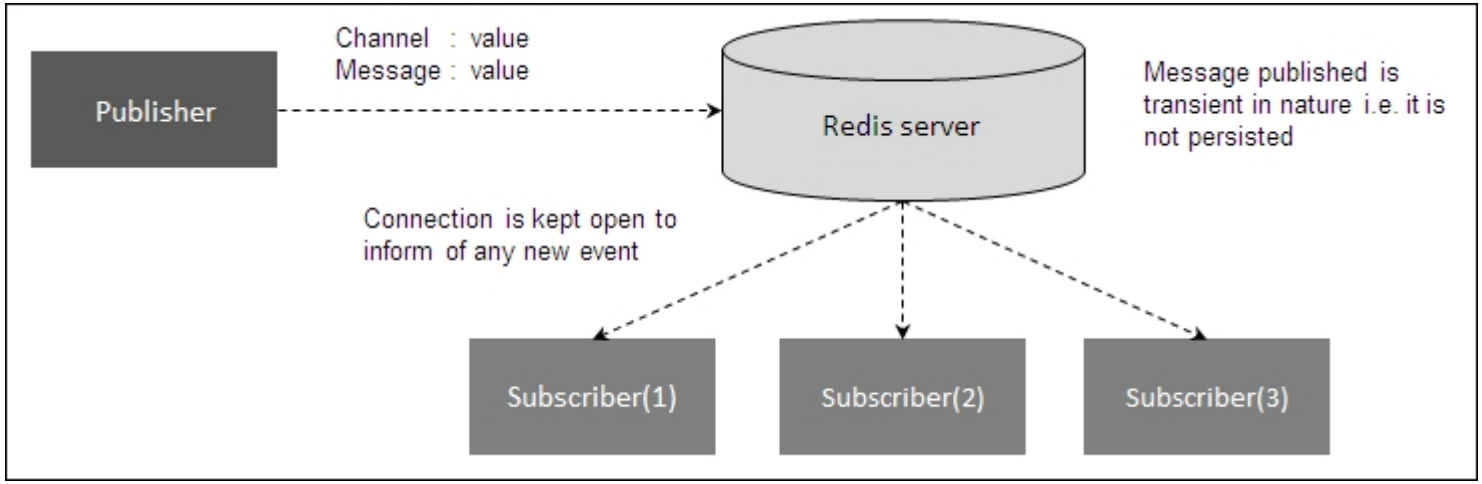
Commands (Sorted Set)

- ZSCORE
  - returns the score of a member.
- ZRANK
  - returns the member rank (or index) ordered from low to high. The member with the lowest score has rank 0.
- ZREVRANK
  - returns the member rank (or index) ordered from high to low. The member with the highest score has rank 0.

```
> ZSCORE leaders "Max"
"101"
> ZRANK leaders "Max"
(integer) 2
> ZREVRANK leaders "Max"
(integer) 0
```

Commands (Publish/Subscribe)

- Redis support publish/subscribe feature
- It does not store the message after delivering it
- It does not store the message if the client (subscriber) was unable to consume it

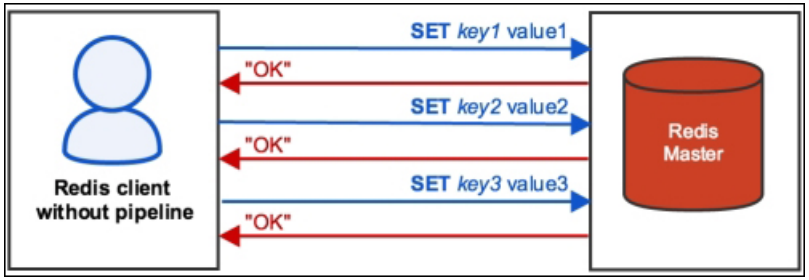


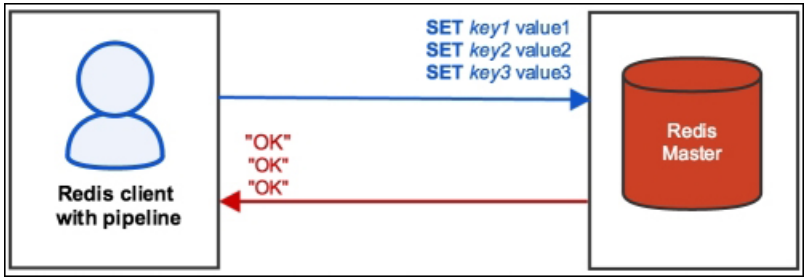
Transaction support

- Sequence of commands executed in order and atomically.
- MULTI marks the beginning
- EXEC marks the end of the transaction
- DISCARD to cancel the transaction

Pipelines

- Sending multiple commands to Redis server without waiting for individual replies
- The commands do not run as a transaction
- Redis libraries may use pipelines by default and automatically choose the number of commands to go into each pipeline.





Multiple databases

- You can choose between databases using command **SELECT <dbid>**
- Not recommended to be used in production
- The best practice is to launch multiple instances of Redis servers if you want to have multiple databases on the same machine

Key Naming Convention

- Redis do not have any form of namespacing
- To prevent key collision, namespace should be used
- Example:

```
music-online:song:1
music-online:song:2
music-online:album:10001:metadata
music-online:album:10001:songs
music-online:author:123
```

Persistence

- There are two persistence model
  - RDB (Redis Database)
  - AOF (Append Only File)

RDB

- By default, the database is being persisted into disk in the background
  - Known as snapshotting
- RDB is not 100% guaranteed data recovery approach
  - There is a chance of data loss

Persistence Example

- For example, if we have the following in redis.conf

```
save 900 1
save 300 10
save 60 10000
```

- The above means
  - Save an .rdb file every 900 seconds if at least one write operation happens
  - Save an .rdb file every 200 seconds if at least 10 write operation happen
  - Save an .rdb file every 60 seconds if at least 10,000 write operations happen

AOF

- When Redis receives a command that changes the dataset,
  - append that command to the AOF (Append-only File)
- When Redis is restarted:
  - restore the data by executing all commands in AOF preserving the order and rebuild the state of the dataset
- Fully durable
- Trade-off: performance and disk space

AOF vs RDB

- RDB is faster to restore
- AOF is fully durable
- If both RDB and AOF enabled, AOF will take precedence because of its durability guarantees
- Decision tree:
  - Do not need persistence, disable both
  - Can torelate data loss, use RDB
  - Requires fully durable persistence, use both

Replication

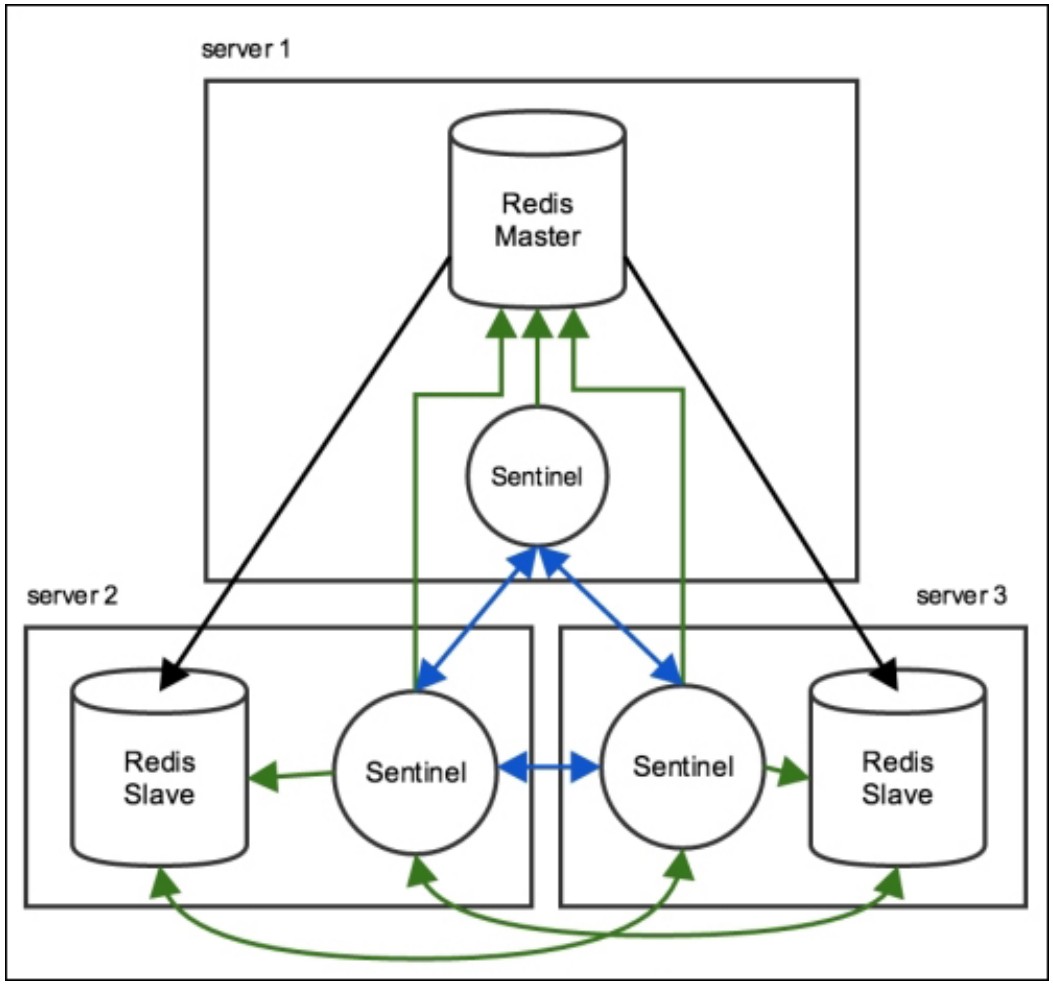
- Replication allow a **master** Redis instance to replicate its data to its **slaves**
- One master can have multiple slaves
- Write to slaves can be done asynchronously

```
$ redis-server --port 5555

$ redis-server --port 6666 --slaveof 127.0.0.1 5555

$ redis-server --port 7777 --slaveof 127.0.0.1 5555
```

Failover

- When a master node fails, one of its slave nodes should be promoted to be the new master and update all other slaves
- Redis Sentinel is a system to automatically promotes a Redis slave to master if the existing master fails
- Need to use Redis client that support Sentinel
  - Need to query a sentinel to find out which Redis instance it is going to connect to

Redis Sentinel

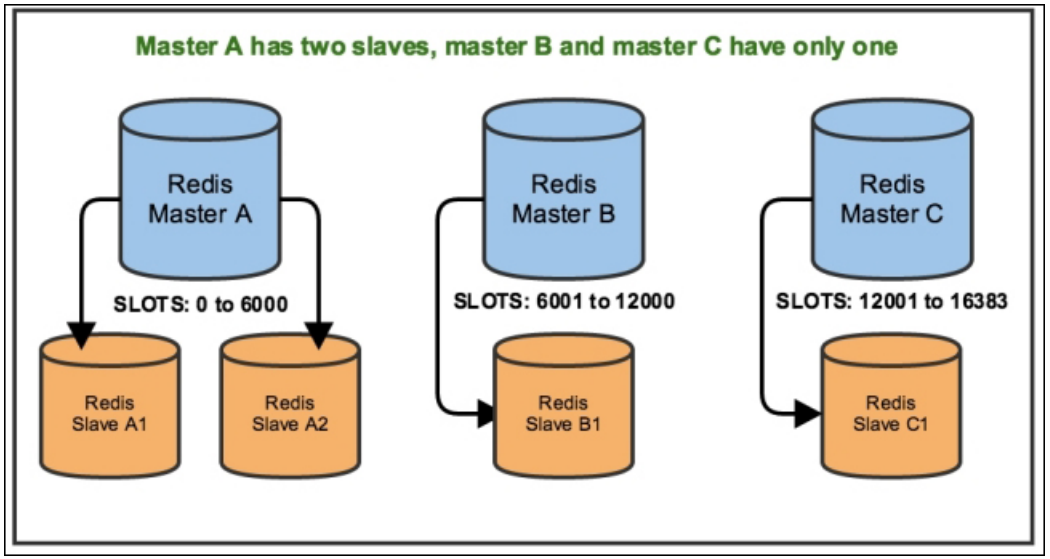


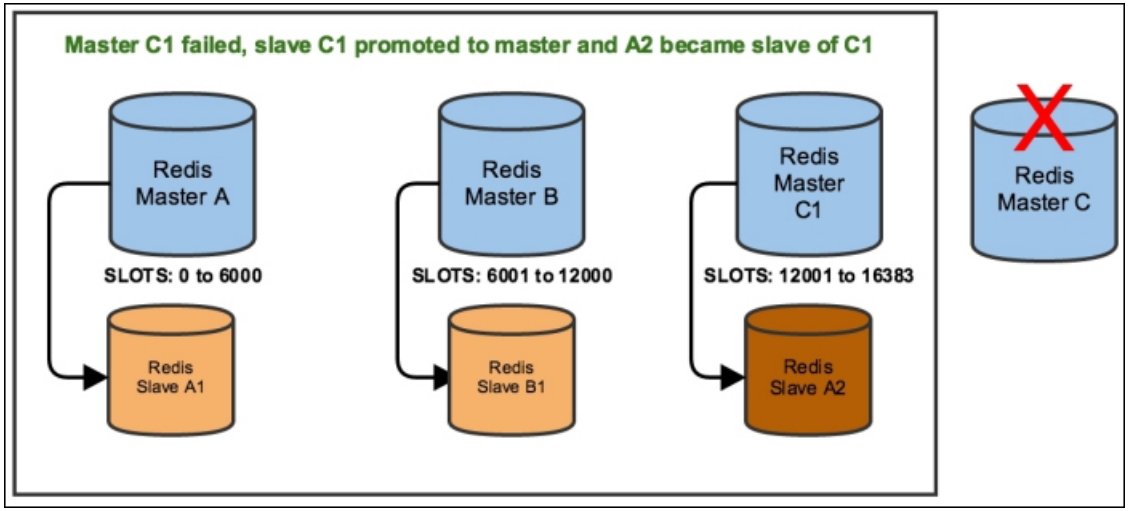
Redis Cluster

- Automatically shard data across different Redis instances
  - Provide some degree of availability
- Requires at least 3 masters
- All data is sharded across the masters and replicated to the slaves
- Each master own a portion of the 16,384 slots

```
HASH_SLOT = CRC16(key) mod 16384
```

Redis Cluster





Redis Cluster Configuration

```
cluster-enabled yes
cluster-config-file cluster.conf
cluster-node-timeout 2000
cluster-slave-validity-factor 10
cluster-migration-barrier 1
cluster-require-full-coverage yes
```

- Cluster is enabled
- Configuration file for the cluster will be cluster.conf
- A node can be unavailable for 2000 ms without being considered as failing
- A slave will be promoted to be a master if the master is unavailable for cluster-node-timeout x cluster-slave-validity-factor
- Minimum number of slaves for a master, otherwise masters with less slaves will borrow spare slaves from other masters
- If any of the hash slot is unavailable (because the master is down) then the entire cluster is unavailable

Further exploration

- twemproxy
  - Open source proxy developed by Twitter with similar purpose as Redis Cluster

Summary

- Redis is one of popular key-value store
  - Some argues otherwise
  - Not as simple as one may think
- Caching is one of popular use case for Redis
- Support for set, map and atomic increment operation can be a very useful arsenal in the design of your application
- Be aware of the limitation of the persistence model of RDB and AOF

References

- Redis Documentation
  - http://redis.io/documentation
- Redis Essentials
  - Hugo Lopes Tavares, Maxwell Dayvson Da Silva, Pack Publishing, September 2015
- The Little Redis Book
  - Karl Seguin
  - http://openmymind.net/redis.pdf
- Redis Cheat Sheet
  - https://www.cheatography.com/tasjaevan/cheat-sheets/redis/