Redis - a Flexible Key/Value Datastore

An Introduction

CIRCL
Computer Incident
Response Center
Luxembourg

Alexandre Dulaunoy

AIMS 2011

MapReduce and Network Forensic

- MapReduce is an old concept in computer science
  - The **map** stage to perform isolated computation on independent problems
  - The **reduce** stage to combine the computation results
- Network forensic computations can easily be expressed in map and reduce steps:
  - parsing, filtering, counting, sorting, aggregating, anonymizing, shuffling...

Concurrent Network Forensic Processing

- To allow concurrent processing, a non-blocking data store is required
- To allow flexibility, a schema-free data store is required
- To allow fast processing, you need to scale horizontally and to know the cost of querying the data store
- To allow streaming processing, write/cost versus read/cost should be equivalent

Redis: a key-value/tuple store

- Redis is key store written in C with an extended set of data types like lists, sets, ranked sets, hashes, queues
- Redis is usually in memory with persistence achieved by regularly saving on disk
- Redis API is simple (telnet-like) and supported by a multitude of programming languages
- `http://www.redis.io/`

Redis: installation

- Download Redis 2.2.9 (stable version)
- tar xvfz redis-2.2.9.tar.gz
- cd redis-2.2.9
- make

Keys

- Keys are free text values (up to 2^{31} bytes) - newline not allowed
- Short keys are usually better (to save memory)
- Naming convention are used like keys separated by colon

Value and data types

- binary-safe strings
- lists of binary-safe strings
- sets of binary-safe strings
- hashes (dictionary-like)
- pubsub channels

Running redis and talking to redis...

- screen
- cd ./src/ && ./redis-server
- new screen session (crtl-a c)
- redis-cli
- DBSIZE

Commands available on all keys

Those commands are available on all keys regardless of their type

- TYPE [key] $\rightarrow$ gives you the type of key (from string to hash)
- EXISTS [key] $\rightarrow$ does the key exist in the current database
- RENAME [old new]
- RENAMENX [old new]
- DEL [key]
- RANDOMKEY $\rightarrow$ returns a random key
- TTL [key] $\rightarrow$ returns the number of sec before expiration
- EXPIRE [key ttl] or EXPIRE [key ts]
- KEYS [pattern] $\rightarrow$ returns all keys matching a pattern (!to use with care)

Commands available for strings type

- SET [key] [value]
- GET [key]
- MGET [key1] [key2] [key3]
- MSET [key1] [valueofkey1] ...
- INCR [key] — INCRBY [key] [value] $\rightarrow$! string interpreted as integer
- DECR [key] — INCRBY [key] [value] $\rightarrow$! string interpreted as integer
- APPEND [key] [value]

Commands available for sets type

- SADD [key] [member] $\rightarrow$ adds a member to a set named key
- SMEMBERS [key] $\rightarrow$ return the member of a set
- SREM [key] [member] $\rightarrow$ removes a member to a set named key
- SCARD [key] $\rightarrow$ returns the cardinality of a set
- SUNION [key ...] $\rightarrow$ returns the union of all the sets
- SINTER [key ...] $\rightarrow$ returns the intersection of all the sets
- SDIFF [key ...] $\rightarrow$ returns the difference of all the sets
- S....STORE [destkey key ...] $\rightarrow$ same as before but stores the result

Commands available for list type

- RPUSH - LPUSH [key] [value]
- LLEN [key]
- LRANGE [key] [start] [end]
- LTRIM [key] [start] [end]
- LSET [key] [index] [value]
- LREM [key] [count] [value]

Sorting

- SORT [key]
- SORT [key] LIMIT 0 4

Commands availabled for sorted set type

- ZADD [key] [score] [member]
- ZCARD [key]
- ZSCORE [key] [member]
- ZRANK [key] [member] $\rightarrow$ get the rank of a member from bottom
- ZREVRANK [key] [member] $\rightarrow$ get the rank of a member from top

Atomic commands

- GETSET [key] [newvalue] $\rightarrow$ sets newvalue and return previous value
- (M)SETNX [key] [newvalue] $\rightarrow$ sets newvalue except if key exists (useful for locking)

MSETNX is very useful to update a large set of objects without race condition.

Database commands

- SELECT [0-15] $\rightarrow$ selects a database (default is 0)
- MOVE [key] [db] $\rightarrow$ move key to another database
- FLUSHDB $\rightarrow$ delete all the keys in the current database
- FLUSHALL $\rightarrow$ delete all the keys in all the databases
- SAVE - BGSAVE $\rightarrow$ save database on disk (directly or in background)
- DBSIZE
- MONITOR $\rightarrow$ what's going on against your redis datastore (check also redis-stat)