REDIS

Finalist NoSQL techtalks

INTRODUCTION

- Wikipedia: *Redis is an open-source, networked, in-memory, key-value data store with optional durability. It is written in ANSI C. The development of Redis is sponsored by VMware.*
- Easy!

KEY-VALUE STORE?

- No, 5 different data structures
  - Strings (typical key-value)
  - Hashes
  - Lists
  - Sets
  - Sorted Sets
- Yes, each data structure uses key and values

KEY-VALUE STORE?

- Relational databases only have tables, generic solution
  - Adds complexity and impacts perfomance
- Redis is not a one-size-fits-all solution.
- Use Redis data structures specific to your problem
  - Similar to the way you code...

SERVER / CLIENT

- Spin up a server using `redis-server`
- Connect to it with `redis-cli`
- Simple!

DATABASES

- Server has multiple database, referenced by number.
- Switch using SELECT index

DATA STRUCTURES

STRINGS: KEY-VALUE

- SET key value
- GET key
- Also (and more...)
  - MSET, MGET
  - GETSET
  - SETNX

STRINGS: BITS

- SETBIT key offset value
- 2.6+
  - BITCOUNT key [start] [end]
  - BITOP operation destkey key
- See interesting metrics use case at: http://blog.getspool.com/2011/11/29/fast-easy-realtime-metrics-using-redis-bitmaps/

STRINGS: COUNTERS

- INCR key
- INCRBY key increment
- DECR key
- DECRBY key decrement

HASH

- HSET key field value
- HGET key field
- Also
  - HKEYS
  - HGETALL
  - HINCRBY

LIST

- LPUSH en LPOP
- RPUSH en RPOP
- LRANGE
- RPOPLPUSH en BRPOPLPUSH (wtf?)

SET

- Set operations (!)
- SADD key member [member...]
- SMEMBERS key
- SISMEMBER key
- Also...
  - SDIFF, SINTER and SUNION

SORTED SET

- ZADD key score member [score] [member]
- ZRANGE key start stop [WITHSCORES]
- ZRANGEBYSCORE key min max

MORE

- Publish / Subscribe
- Expiration
- Transactions
- Finding keys

PERFORMANCE

WHY?

- Specialized data structures, for example:
  - ISMEMBER command has O(1) complexity
- See command reference for complexity values of different commands: http://redis.io/commands

WHY?

- In-memory
- Redis does not care about values (also means: no queries)

HOW FAST?

- Bulk import demo...

IS THAT ALL?

WE NEED MORE POWER!

NORMAL MODE

- Client: `INCR X`
- Server: 1

- Client: `INCR X`
- Server: 2
- Client: `INCR X`
- Server: 3

PIPELINING!

- Client: `INCR X`
- Client: `INCR X`
- Client: `INCR X`
- Server: 1
- Server: 2
- Server: 3

HOW FAST?

- Pipelining enhanced bulk import demo...

IS THAT ALL?

REDIS-CLI --PIPE

- Mass insertion from command line.
- Commands are checked!

MORE CAPACITY

REDIS CLUSTER

- Currently work in progress.
- Ruby driver offers distributed Redis capabilities:
  - Add extra (master) servers for more capacity
  - Keys are distributed over servers
  - Client calculates where keys are stored
    - CPU :-(

HOW FAST?

- Clustered bulk import demo... (no pipelining)

PERSISTENCE

WHAT'S THE CATCH?

- Data structures
- Speed
- In-memory
- There must be something wrong with persistence!

MEMORY-BOUND

- Database size bound to memory size
- Virtual memory support deprecated

READ IT

- Must read post by antirez on Redis persistence: http://oldblog.antirez.com/post/redis-persistence-demystified.html

DURABILITY

- 1: Client sends a write command to database
  Data is in clients memory
- 2: Database receives the write
  Data is in server memory
- 3: The database calls the system call that writes the data on disk *Data is in kernel's buffer*
- 4: The OS transfers the write buffer to the disk controller
  Data is in the disk cache
- 5: The disk controller actually writes the data into a physical media.
  Persisted!

DURABILITY

- ~~1: Client sends a write command to database~~
  ~~*Data is in clients memory*~~
- ~~2: Database receives the write~~
  ~~*Data is in server memory*~~
- 3: The database calls the system call that writes the data on disk *Data is in kernel's buffer*
- 4: The OS transfers the write buffer to the disk controller
  Data is in the disk cache
- ~~5: The disk controller actually writes the data into a physical media.~~
  ~~*Persisted!*~~

DURABILITY

- Data to disk
- We can control:
  - Data written to OS kernel using `write` call
  - Data committed to disk using `fsync` call

DURABILITY

- Redis has two persistence options:
  - RDB
  - AOF

RDB

RDB

- Snapshotting
- Specify at what intervals you want a snapshot to be taken.
- Not very durable
- Easy backups

AOF

APPEND-ONLY-FILE

- Much more durable, but slower (in default mode)
- Buffer flushed (`write`) after each command (when re-entering the event loop)
- Different `fsync` policies (from worst to best):
  - none (OS decides)
  - every second (default)
  - every command (slow, but best durability in any db systeem)
- Append only == No corruption

MASTER-SLAVE

MASTER-SLAVE

- Master can have multiple slaves
- Slaves can have slaves as well
- Replication is non-blocking on master as well as slave

HANDS-ON

TWITTER

- `git clone https://github.com/robinroestenburg/redis-techtalk`
- `cd sinatra-twitter`
- `bundle`
- `rackup`
- Go to http://localhost:9292 and sign up/in.

TWITTER

- Implement all methods that are now returning dummy or empty results.

- Extra points:
  - Implement favoriting a Tweet and viewing favorites.
  - When following a user, all his tweets must be visible in the followers' timeline (in the right order).