Modern Database Systems

Practicals: redis

Doc. RNDr. Irena Holubova, Ph.D.

holubova@ksi.mff.cuni.cz

Key-value store

Redis

- Open-source database
  - First release: 2009
  - Development sponsored by WMware
- OS: most POSIX systems like Linux, *BSD, OS X, …
  - Win32-64 experimental version
- Language: ANSI C
  - Clients for many languages: C, PHP, Java, Ruby, Perl, ...
- Not standard key-value features (rather a kind of document database):
  - Keys are binary safe = any binary sequence can be a key
  - The stored value can be any object
    - strings, hashes, lists, sets and sorted sets
  - Can do range, diff, union, intersection, … operations
    - Atomic operations
    - Not usual, not required for key-value stores

multi-model

http://redis.io/

Redis Data Types

Strings

- Binary safe = any binary sequence
  - e.g., a JPEG image
- Max length: 512 MB
- Operations:
  - Set/get the string value of a key: GET/SET, SETNX (set if not set yet)
  - String-operation: APPEND, STRLEN, GETRANGE (get a substring), SETRANGE (change a substring)
  - Integer-operation: INCR, INCRBY, DECR, DECRBY
    - When the stored value can be interpreted as an integer
  - Bit-operation: GETBIT, BITCOUNT, SETBIT

Redis Data Types

Strings – Example

```
> SET count 10
OK
> GET count
"10"
> INCR count
(integer) 11
> DECRBY count 10
(integer) 1
> DEL count
(integer) 1          // returns the number of keys removed
```

Redis Data Types

List

- Lists of strings, sorted by insertion order
- Possible to push new elements on the head (on the left) or on the tail (on the right)
- A key is removed from the key space if a list operation will empty the list (= value for the key)
- Max length: $2^{32} – 1$ elements
  - 4,294,967,295 = more than 4 billion of elements per list
- Accessing elements
  - Very fast near the extremes of the list (head, tail)
  - Slow accessing the middle of a very big list
    - *O(N)* operation

Redis Data Types

List

- Operations:
  - Add element(s) to the list:
    - LPUSH (to the head)
    - RPUSH (to the tail)
    - LINSERT (inserts before or after a specified element)
    - LPUSHX (push only if the list exists, do not create if not)
  - Remove element(s): LPOP, RPOP, LREM (remove elements specified by a value)
  - LRANGE (get a range of elements), LLEN (get length), LINDEX (get an element at index)
  - BLPOP, BRPOP remove an element or block until one is available
    - Blocking version of LPOP/RPOP

Redis Data Types

List – Example

```
> LPUSH animals dog
(integer) 1      // number of elements in the list
> LPUSH animals cat
(integer) 2
> RPUSH animals horse
(integer) 3
> LRANGE animals 0 -1 // -1 = the end
1) "cat"
2) "dog"
3) "horse"
> RPOP animals
"horse"
> LLEN animals
(integer) 2
```

Redis Data Types

Set

- Unordered collection of <u>non-repeating</u> strings
- Possible to add, remove, and test for existence of members in *O(1)*
- Max number of members: $2^{32} – 1$
- Operations:
  - Add element: SADD, remove element: SREM
  - Classical set operations: SISMEMBER, SDIFF, SUNION, SINTER
  - The result of a set operation can be stored at a specified key (SDIFFSTORE, SINTERSTORE, ...)
  - SCARD (element count), SMEMBER (get all elements)
  - Operations with a random element: SPOP (remove and return random element), SRANDMEMBER (get a random element)
  - SMOVE (move element from one set to another)

Redis Data Types

Set – Example

```
> SADD friends:Lisa Anna
(integer) 1
> SADD friends:Dora Anna Lisa
(integer) 2
> SINTER friends:Lisa friends:Dora
1) "Anna"
> SUNION friends:Lisa friends:Dora
1) "Lisa"
2) "Anna"
> SISMEMBER friends:Lisa Dora
(integer) 0
> SREM friends:Dora Lisa
(integer) 1
```

Redis Data Types

Sorted Set

- Non-repeating collection of strings
- Every member is associated with a score
  - Used in order to make the set ordered
    - From the smallest to the greatest
  - May have repeated values
    - Then lexicographical order
- Possible to add, remove, or update elements in *O(log N)*
- Operations:
  - Add element(s): ZADD, remove element(s): ZREM, increment the score of a member: ZINCRBY
  - Number of elements in a set: ZCARD
  - Elements with a score in a specified range: ZCOUNT (count), ZRANGEBYSCORE (get the elements)
  - Set operations (store result at a specified key): ZINTERSTORE, ZUNIONSTORE , …

Redis Data Types

Sorted Set – Example

```
> ZADD articles 1 Anna 2 John 5 Tom
(integer 3)
> ZCARD articles
(integer) 3
> ZCOUNT articles 3 10 // members with score 3-10
(integer) 1
> ZINCRBY articles 1 John
"3"                     // returns new John's score
> ZRANGE articles 0 -1 // outputs all members
1) "Anna"     // sorted according score
2) "John"
3) "Tom"
```

Redis Data Types

Hash

- Maps between string fields and string values
- Max number of field-value pairs: $2^{32} - 1$
- Optimal data type to represent objects
  - e.g., a user with fields name, surname, age, …
- Operations:
  - HSET key field value (set a value to the field of a specified key), HMSET (set multiple fields)
  - HGET (get the value of a hash field), HMGET, HGETALL (get all fields and values in a hash)
  - HKEYS (get all fields), HVALS (get all values)
  - HDEL (delete one or more hash fields), HEXISTS, HLEN (number of fields in a hash)

Redis Data Types

Hash – Example

```
> HSET users:sara id 3
(integer) 1
> HGET users:sara id
"3"
> HMSET users:sara login sara group students
OK
> HMGET users:sara login id
1) "sara"
2) "3"
> HDEL users:sara group
(integer) 1
> HGETALL users:sara
1) "id"
2) "3"
3) "login"
4) "sara"
```

Redis Cache-like Behaviour

Example

```
> SET cookie:google hello
OK
> EXPIRE cookie:google 30
(integer) 1
> TTL cookie:google           // time to live
(integer) 23
> GET cookie:google
"hello"                       // still some time to live
> TTL cookie:google
(integer) -1                  // key has expired
> GET cookie:google
(nil)                         // and was deleted
```

Assignment

- Chose your unique problem domain
  - E.g., the results of football matches of various teams
- For your selected problem domain think about an application that uses the advanced data structures of Redis
  - Hashes, lists, sets and sorted sets
- Submit a script with respective commands for Redis + explanatory comments

References

- Eric Redmond – Jim R. Wilson: **Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement**
- Pramod J. Sadalage – Martin Fowler: **NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence**
- Karl Seguin: **The Little Redis Book** http://openmymind.net/2012/1/23/The-Little-Redis-Book/
- Data Types: http://redis.io/topics/data-types
- Commands: http://redis.io/commands