Tecnologie e applicazioni web

REDIS

Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco/
DAIS - Università Ca'Foscari di Venezia
Academic year: 2021/2022

What is REDIS?

- An **in-memory** database (structure store):
  - Supports strings / hash-maps / lists / sets / sorted sets / geospatial indexes
- Cache:
  - LRU eviction
- Message broker
  - Pub/Sub messaging paradigm

Interesting features

- Data is stored in RAM memory. This allows orders of magnitude faster accesses wrt. classical databases.
- REDIS can also persist all writes to a permanent storage (immediate persistency is not guaranteed though)
- Designed to manage huge workloads
- Support for atomic operations (ex. Increment of integer values)

Interesting features

- Transactions like in classical DBs
- Single-threaded / asynchronous architecture
- Scriptable in LUA

Typical use case:

Operations involving multiple concurrent accesses to data structures and as caching layer. Note that the amount of data must fit in the server RAM memory…

Persistency

Data can be optionally stored on disk, asynchronously or immediately (risk vs. performance trade-off). REDIS supports two options:

- **RDB**: fork and dump of all data structures creating point-in-time snapshots of datasets at specified intervals. Fast but introduces an unpredictable delay
- **AOF** (append-only file): every write operation is logged in a file. File write can be asynchronous or synchronous (for maximum safety). Performance is badly affected in the latter case.

REDIS as a CACHE

One of the most popular usages is as a simple CACHE:

Two commands:

- SET <key> <value>
  - Stores a string <value> given a certain <key>
- GET <key>
  - Returns the string previously stored with a certain <key>

REDIS as a CACHE

Each read operation in the main database adds the retrieved value in the REDIS cache using the ID of the object as the key

Subsequent read operations can verify if the data is already present in the REDIS cache and avoid a new query to the database.

REDIS Pub/Sub

One interesting feature is the implementation of the Publish/Subscribe programming paradigm:

- Publishers can send messages on some **channels**
- Subscribers express interest in one or more channels, and only receive messages that are of interest, without knowledge of what (if any) publishers there are

REDIS Pub/Sub

The advantage of using this pattern is in the decoupling between entities that produce data from entities that receive (consume) it:

- Publishers don't know how many (and who are) the subscribers of a certain channel
- Subscribers don't know who produce the data, but are automatically notified at every submission

REDIS Pub/Sub

REDIS allows the subscribers to subscribe to multiple channels matching a regular expression:

Ex:

- Publishers send messages on: `it.news.topic1`, `it.news.topic2`, etc.
- A subscriber can listen to `it.news.*` to receive messages from both the topics

REDIS & Node.js

https://www.npmjs.com/package/redis

Lightweight library providing an interface to all the available REDIS commands.

Designed to be simple but extremely fast

REDIS command list:

https://redis.io/commands