redislabs

Home of Redis

Durability

Data persistence

Takeaways

Redis is an in-memory database. In a nutshell:

1. "NoSQL" is a way to think about managing data
2. All data is always written to and read from RAM[e]
3. Data may be persisted to and recovered from durable storage

[e] Redis[e] Flash uses Flash as a RAM extension

What's fast, expensive and volatile?

(hint: not the SpaceX rocket)

What's fast, expensive and volatile?

RAM, which Redis uses to store all data:

- The fastest storage available
- Supports random access
- Provides similar latencies for both reads and writes

But RAM:

- Is expensive compared to SSD and spinning metal
- Loses data without power

Redis as a cache

Because RAM is volatile, using Redis as a cache for transient data is popular use case.

Types of volatile information include:

- Data copied from a different source of truth, e.g. DB
- Precomputed values, e.g. rendered HTML
- Ephemeral data, e.g. session store

Redis-as-a-Cache

- Data persistence is optional and tunable
- Configurable data memory footprint (maxmemory)
- Volatile keys that expire after a settable TTL
- OOM eviction policies (maxmemory-policy)
  - allkeys-lru, allkeys-lfu, allkeys-random, noeviction
  - volatile-lru, volatile-lfu, volatile-random, volatile-ttl
  - Are approximated for performance (maxmemory-samples)
- Keyspace_misses and keyspace_hits in INFO

transient data != availability

While the data in the cache isn't "important", ensuring its availability usually is.

- No cache means increase in application latency
- Cold cache means it needs to be warmed
- Beware of thundering herds toppling your infrastructure
- Life happens - recovery could take a while...

Redis as a primary database

(i.e. the data in it exists nowhere else)

Redis as a primary database

For some data Redis is the single source of truth.

Popular common examples are job queues, session metrics, stream analytics…

But even entire apps, for example Muut or Spot.IM

You really don't want to loose the data then.

Data durability

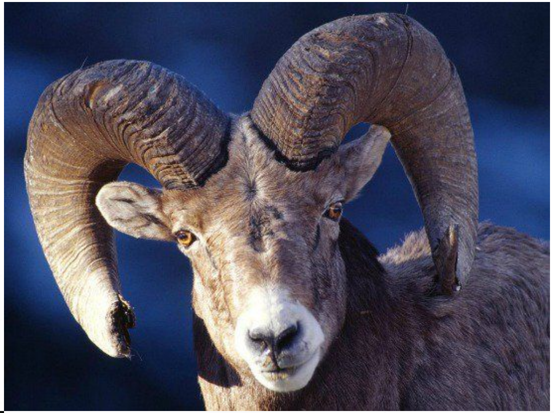
- Durability is the ability to recover the data after a failure, e.g. a power outage
- RAM is volatile, which makes it the undurable
- SSD and spinning metal disks do persist data
- Redis has two independent data persistence options: RDB and AOF
- Redis data persistence is 🐟-able:
  - Think different
  - Persistence impacts performance - costs CPU, RAM and IO

Dump files, a.k.a. RDB

- The dump file contains a serialization of all data
- Essentially a snapshot in time
- Can be created with SAVE (blocking!) or BGSAVE (not)
- Configurable background jobs with the `save` directive:
  - Syntax: `save <seconds> <changes>`
  - Defaults:

    ```
    save 900 1
    save 300 10
    save 60 10000
    ```
- Snapshots are atomic

BG: thread forking, RAM & Copy-on-Write



Append-only files

- The AOF is a log, write commands are added to it
- Enabled by setting `appendonly yes`
- Flush to disk configurable via `appendfsync`:
  - Every command (slow, not suppprted by Redis^e): `always`
  - Every second (default): `everysec`
  - The OS' consideration: `none`
- Can grow indefinitely, but manageable with:
  - Manual call to `BGREWRITEAOF`
  - `auto-aof-rewrite-percentage 100`
  - `auto-aof-rewrite-min-size 64mb`

RDB and/or AOF

- RDB files are
  - Compact in size and require minimal time to load
  - Atomic
  - Good for backups and disaster recovery
- AOF files are
  - Potentially large and time consuming to load
  - Asynchronously written
  - For making recent changes durable
- Use one, the other or both according to requirements

persisted data != availability

Ensuring that data is recoverable from disk after failure increases, but does not guarantee, availability.

- Reading serialized data from disk takes recovery time
- Replaying logged commands takes recovery time
- During that period the database isn't available
- That's what high-availability (replication and failover) is for