Redis persistence in practice

Eugene Fidelin, 2014
Jaludo B.V.

Redis key features

- ❏ **All datasets are stored in memory**
  extremly fast read/write operations.
- ❏ **Datasets can be saved to disk**
  RDB and AOF - two ways to achieve persistence.
- ❏ **Redis forks**
  for long tasks Redis creates child processes.

RDB snaphots and AOF logs

Persistence in Redis is a matter of configuration, balancing the trade-off between performance, disk I/O, and data durability.

- ❑ **RDB** is a very compact single-file point-in-time representation of the Redis dataset.
- ❑ **AOF** is a simple text log of write operations.

RDB advantages

- ★ RDB is a very compact and perfect for backups and for disaster recovery (~2 times smaller than AOF log file).
- ★ RDB maximizes Redis performances since the only work the Redis parent process needs to do in order to persist is forking a child that will do all the rest. The parent instance will never perform disk I/O or alike.
- ★ RDB allows faster restarts with big datasets compared to AOF (up to 2 times faster).

RDB disadvantages (1/2)

- ➔ RDB is **not** good to minimize the chance of data loss in case Redis unexpectedly stops working - in this case latest minutes of data can be lost.
- ➔ RDB needs to fork() a child process to persist on disk. Fork() can be time consuming if the dataset is big and the CPU performance not great. This may result in Redis to stop serving clients for some millisecond.

RDB disadvantages (2/2)

- ➔ RDB uses fork() that implements a copy-on-write semantic of memory pages. Additional memory consumtion is proportional to the number of changes the dataset will get while snapshoting is in progress. In a very write-heavy application Redis may use up to 2x the memory normally used, so memory swapping becomes risky.
- ➔ RDB snapshoting generates a higher volume of disk writes because whole file is being rewritten each time.

AOF advantages (1/2)

- ★ AOF log is an append only log, there are no seeks, nor corruption problems if there is a power outage.
- ★ AOF generates less blocks writes to disk because data is only appended to file.
- ★ Using AOF Redis is much more durable: there are different fsync policies: no fsync at all (fsynk done by OS, usually each 30s on Linux), fsync every second (default config), fsync at every query (very slow).

AOF advantages (2/2)

- ★ Redis is able to automatically rewrite the AOF in background when it gets too big.
- ★ AOF needs fork() only to rewrite logs, you can tune how often you want to without any trade-off on durability.
- ★ AOF contains a log of all the operations one after the other in an easy to understand and parse format.

AOF disadvantages

- ➔ AOF takes more time to load in memory on server restart.
- ➔ AOF files are usually bigger (1.5 - 3 times) than the equivalent RDB files for the same dataset.
- ➔ AOF can be slower than RDB. With fsync set to every second performance is still very high, and with fsync disabled it exactly as fast as RDB even under high load.
- ➔ AOF file can be damaged, because of unexpected shutdown, but this can be easily fixed.

RDB and AOF usage

- RDB persistence is enabled by default.
- AOF and RDB can be enabled at the same time. The AOF file will be used to reconstruct the original dataset on restart, since it is guaranteed to be the most complete.
- Redis makes sure to avoid triggering an AOF `BGREWRITEAOF` and RDB `BGSAVE` at the same time.
- Persistence can be disabled at all.

RDB configuration (1/2)

`save <seconds> <changes>`

Will save the DB if both the given number of seconds and the number of write operations occurred.
Multiple `save` settings can be specified.

`save ""`

Disable RDB snapshoting.

`dbfilename <filename>.rdb`

The filename where to dump the DB

RDB configuration (2/2)

`stop-writes-on-bgsave-error <yes/no>`
Redis will stop accepting writes if RDB snapshots are enabled and the latest background save failed.

`rdbcompression <yes/no>`
Compress string objects using LZF in RDB dump

`rdbchecksum <yes/no>`
Add a CRC64 checksum at the end of the file. Makes it more resistant to corruption but there is a performance hit to pay (around 10%) when saving/loading RDB files.

RDB related commands

`BGSAVE`

Save the DB in background. Redis forks, the parent continues to serve the clients, the child saves the dataset on disk then exits.

`SAVE`

Perform a synchronous save of the dataset. Other clients are blocked - **never use in production**!

`LASTSAVE`

Return the Unix time of the last successfull DB save.

AOF configuration (1/2)

```
appendonly <yes/no>
```
Enable or disable AOF persistence.

```
appendfilename <filename>.aof
```
The name of the append only file.

```
auto-aof-rewrite-percentage <value>
auto-aof-rewrite-min-size <size>
```
Automatic rewrite the AOF log file when the log size grows by the specified percentage.

```
auto-aof-rewrite-percentage 0
```
Disable automatical AOF rewrite.

AOF configuration (2/2)

`appendfsync <always/everysec/no>`
Specify how often actually write data on disk instead of waiting for more data in the output buffer.
`no`: don't fsync, let the OS flush the data when it wants (usually every 30 sec.). Faster.
`always`: fsync after every write. Slow, Safest.
`everysec`: fsync one time every second. Compromise.

`no-appendfsync-on-rewrite <yes/no>`
Prevent fsync() from being called in the main process while a **`BGSAVE`** or **`BGREWRITEAOF`** is in progress

AOF related commands

`BGREWRITEAOF`

Instruct Redis to start an AOF rewrite process. The rewrite will create a small optimized version of the current AOF log.

If `BGREWRITEAOF` fails, no data gets lost as the old AOF will be untouched

Persistence related INFO. RDB (1/5)

`rdb_changes_since_last_save`
Number of operations that produced some kind of changes in the dataset since the last time either **`SAVE`** or **`BGSAVE`** was called.

`rdb_bgsave_in_progress`
Flag indicating whether an RDB save is on-going.

`rdb_last_save_time`
Unix timestamp of last successful RDB save (same as the **`LASTSAVE`** command).

Persistence related INFO. RDB (2/5)

`rdb_last_bgsave_status`
Status of the last RDB save operation (should be ‘ok’).

`rdb_last_bgsave_time_sec`
Duration of the last RDB save operation in seconds.

`rdb_current_bgsave_time_sec`
Duration of the on-going RDB save operation if any.

Persistence related INFO. AOF (3/5)

`aof_enabled`
Flag indicating whether an AOF logging is activated.

`aof_rewrite_in_progress`
Flag indicating whether an AOF rewrite operation is on-going.

`aof_rewrite_scheduled, aof_pending_rewrite`
Flags indicating whether an AOF rewrite operation will be scheduled once the on-going RDB save is complete.

`aof_last_rewrite_time_sec`
Duration of the last AOF rewrite operation in seconds.

Persistence related INFO. AOF (4/5)

`aof_current_rewrite_time_sec`
Duration of the on-going AOF rewrite operation if any.

`aof_last_bgrewrite_status`
Status of the last AOF rewrite operation.

`aof_current_size`
AOF current file size (in bytes).

`aof_base_size`
AOF file size on latest startup or rewrite (in bytes).

Persistence related INFO. AOF (5/5)

`aof_buffer_length`
Size of the AOF buffer.

`aof_rewrite_buffer_length`
Size of the AOF rewrite buffer.

`aof_pending_bio_fsync`
Number of fsync pending jobs in background I/O queue.

`aof_delayed_fsync`
Delayed fsync counter.

Redis persistence and pools

Pools are multiple Redis servers running on the same machine but using different ports.

★ **More effective memory usage**. When RDB snapshoting is performed only DBs from one pool are copied in memory.

★ **More CPUs can be used**. While each Redis instance can use only one CPU, different pools can use different cores.

★ **More fine-tuning**. Each pool can be configured and restarted independently.

Persistence config for pools (1/2)

- ❑ Each pool should have different configuration for `dbfilename` and `appendfilename`.
- ❑ Because pools can’t be synchronized between each other, it is possible that RDB snapshot and/or AOF rewrite could start on several pools at the same time.
- ❑ In case RDB or RDB + AOF persistence is used, this can cause high CPU, memory and disk usage.
  To avoid this: disable Redis automatic and use manual control over `BGSAVE` and/or `BGREWRITEAOF` operations.

Persistence config for pools (2/2)

Example configuration for AOF + RDB:

- ❏ disable automatic Redis RDB snapshots and AOF rewrite: **`save ""`**, **`auto-aof-rewrite-percentage 0`**
- ❏ configure cron jobs for **`BGSAVE`** and **`BGREWRITEAOF`** of each pool at different time:

```
m h * * * redis-cli -p <port> BGSAVE
m h */4 * * redis-cli -p <port> BGREWRITEAOF
```

`m` and **`h`** should be different for every pool. **`BGSAVE`** can be run 1-4 times per day, **`BGREWRITEAOF`** once per 2-4 days (calculate how fast AOF file grows for 100%)

Persistence and replication

RDB snapshots are used when performing a master-slave synchronization (even if RDB persistence is disabled):

- the master starts background saving, and starts to buffer all new commands received that will modify the dataset.
- when the background saving is complete, the master transfers the database file (RDB) to the slave.
- the slave saves file on disk, and then loads it into memory.
- the master then sends all buffered commands to the slave.

Persistence on master

If Redis has more read operations than writes - the master server can be used to perform persistence:

- ❏ enable only AOF logging on master,
- ❏ disable RDB and AOF on all slaves.

- ★ Backups will be automatically used to restore dataset in memory after restart.
- ★ AOF doesn’t use fork() that could cause Redis to stop serving clients for some millissecond.
- ★ If necessary, RDB snapshots could be executed by cron, when the machine is less busy.

Persistence on slave

If Redis has lot of write operations or disk system is slow (like at AWS EC2) - persistence-only slave can be used:

- ❑ enable RDB snaphosts on persistence-only slave,
- ❑ disable RDB and AOF on master and other slaves.

- ★ Master does not need to perform any background disk operations and is fully dedicated to serve client requests, except for a slave connection.
- ★ RDB allows faster restarts with big datasets, but snapshots should be moved to master server manualy.

Persistence tips and tricks

- ★ Remove all AOF and RDB files from slave before restart. Redis will start syncing with master faster without restoring existing backup.
- ★ Temporary disable AOF on master when slave is syncing. This will prevent RDB and AOF running at the same time and will decrease IO wait on master.
- ★ Try to avoid Redis (or all Redis pools) grow beyond 80% of memory. After that, you do start seeing high IO behavior, not necessarily tied directly to swapping

Useful links

- ❑ Official technical description of Redis persistence: http://redis.io/topics/persistence
- ❑ Wider overview of Redis persistence: http://oldblog.antirez.com/post/redis-persistence-demystified.html
- ❑ How to configure multiple Redis servers on one machine: http://chrislaskey.com/blog/342/running-multiple-redis-instances-on-the-same-server/
- ❑ Implementing persistence in Redis (article and book): http://www.packtpub.com/article/implementing-persistence-in-redis-intermediate
- ❑ Tool to parse RDB files and analyse memory usage: https://github.com/sripathikrishnan/redis-rdb-tools

Contacts

Eugene Fidelin

- Email: eugene.fidelin@gmail.com
- Twitter: @EugeneFidelin
- Facebook: facebook.com/eugene.fidelin
- Github: github.com/eugef