Redis persistence

How Redis writes data to disk

1. **RDB** (Redis Database): RDB persistence performs point-in-time snapshots of your dataset at specified intervals.
2. **AOF** (Append Only File): AOF persistence logs every write operation received by the server.
3. **No persistence**: You can disable persistence completely. This is sometimes used when caching.
4. **RDB + AOF**: You can also combine both AOF and RDB in the same instance.

RDB advantages

- RDB is a very compact single-file point-in-time representation of your Redis data.
- RDB files are perfect for backups.
- RDB is very good for disaster recovery, being a single compact file that can be transferred to far data centers, or onto Amazon S3 (possibly encrypted).
- 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 process will never perform disk I/O or alike.
- RDB allows faster restarts with big datasets compared to AOF.

RDB disadvantages

- RDB is NOT good if you need to minimize the chance of data loss in case Redis stops working (for example after a power outage).
- RDB needs to fork() often in order to persist on disk using a child process. fork() can be time consuming if the dataset is big, and may result in Redis stopping serving clients for some millisecond or even for one second if the dataset is very big and the CPU performance is not great.

AOF advantages

- Using AOF Redis is much more durable
- The AOF log is an append-only log, so there are no seeks, nor corruption problems if there is a power outage.
- Even if the log ends with a half-written command for some reason (disk full or other reasons) the `redis-check-aof` tool is able to fix it easily.
- Redis is able to automatically `rewrite` the AOF in background when it gets too big
- The rewrite is completely safe as while Redis continues appending to the old file, a completely new one is produced with the minimal set of operations needed to create the current data set, and once this second file is ready Redis switches the two and starts appending to the new one.
- AOF contains a log of all the operations one after the other in an easy to understand and parse format.
- You can even easily export an AOF file.

- For instance even if you've accidentally flushed everything using the `FLUSHALL` command, as long as no rewrite of the log was performed in the meantime, you can still save your data set just by stopping the server, removing the latest command, and restarting Redis again.

AOF disadvantages

- AOF files are usually bigger than the equivalent RDB files for the same dataset.
- AOF can be slower than RDB depending on the exact `fsync policy.`
  - In general with fsync set to *every second* performance is still very high
  - and with fsync disabled it should be exactly as fast as RDB even under high load.

Ok, so what should I use?

If you care a lot about your data, but still can live with a few minutes of data loss in case of disasters, you can simply use RDB alone.

There are many users using AOF alone, but we discourage it since to have an RDB snapshot from time to time is a great idea for doing database backups, for faster restarts, and in the event of bugs in the AOF engine.

How RDB Persistence Works:

- **Triggering RDB Persistence**
  - **SAVE | BGSAVE**
  - **Automatic Trigger**
- **Creating the RDB Snapshot**
- **Swapping the RDB Snapshot**
- **Restoring from RDB Snapshot**

- Redis **forks**. We now have a child and a parent process.
- The child starts to write the dataset to a temporary RDB file.
- When the child is done writing the new RDB file, it replaces the old one.

Advantages

- **Fast Recovery**:
- **Compact Storage**
- **Portability**

Disadvantages

- **Potential Data Loss**

How AOF Persistence Works:

- **Writing Operations to the AOF File**
- **AOF Rewrite**
- **Replaying AOF File**
- **Safety and Durability**

Advantages

- **Data Integrity and Recovery**
- **Flexibility**

Disadvantages

- **Larger File Size**
- **Recovery Time**

Backing up Redis data

Create a cron job in your server creating hourly snapshots of the RDB file in one directory, and daily snapshots in a different directory.

Every time the cron script runs, make sure to call the `find` command to make sure too old snapshots are deleted:
