

# 深入理解Redis持久化



六尺帐篷 (/u/f8e9b1c246f1)

2017.08.28 09:42\* 字数 4513 阅读 172 评论 0 喜欢 7

(/u/f8e9b1c246f1)

编辑文章 (/writer#/notebooks/15878374/notes/16328799)

本文翻译自官方文档

<http://redis.io/topics/persistence> (<https://link.jianshu.com?t=http://redis.io/topics/persistence>)。

## Redis 持久化

Redis 提供了多种不同级别的持久化方式：

- RDB 持久化可以在指定的时间间隔内生成数据集的时间点快照（point-in-time snapshot）。
- AOF 持久化记录服务器执行的所有写操作命令，并在服务器启动时，通过重新执行这些命令来还原数据集。AOF 文件中的命令全部以 Redis 协议的格式来保存，新命令会被追加到文件的末尾。Redis 还可以在后台对 AOF 文件进行重写（rewrite），使得 AOF 文件的体积不会超出保存数据集状态所需的实际大小。
- Redis 还可以同时使用 AOF 持久化和 RDB 持久化。在这种情况下，当 Redis 重启时，它会优先使用 AOF 文件来还原数据集，因为 AOF 文件保存的数据集通常比 RDB 文件所保存的数据集更完整。
- 你甚至可以关闭持久化功能，让数据只在服务器运行时存在。

了解 RDB 持久化和 AOF 持久化之间的异同是非常重要的，以下几个小节将详细地介绍这两种持久化功能，并对它们的相同和不同之处进行说明。

## RDB 的优点

- RDB 是一个非常紧凑（compact）的文件，它保存了 Redis 在某个时间点上的数据集。这种文件非常适合用于进行备份：比如说，你可以在最近的 24 小时内，每小时备份一次 RDB 文件，并且在每个月的每一天，也备份一个 RDB 文件。这样的话，即使遇上问题，也可以随时将数据集还原到不同的版本。
- RDB 非常适用于灾难恢复（disaster recovery）：它只有一个文件，并且内容都非常紧凑，可以（在加密后）将它传送到别的数据中心，或者亚马逊 S3 中。
- RDB 可以最大化 Redis 的性能：父进程在保存 RDB 文件时唯一要做的就是 fork 出一个子进程，然后这个子进程就会处理接下来的所有保存工作，父进程无须执行任何磁盘 I/O 操作。
- RDB 在恢复大数据集时的速度比 AOF 的恢复速度要快。

## RDB 的缺点

- 如果你需要尽量避免在服务器故障时丢失数据，那么 RDB 不适合你。虽然 Redis 允许你设置不同的保存点（save point）来控制保存 RDB 文件的频率，但是，因为 RDB 文件需要保存整个数据集的状态，所以它并不是一个轻松的操作。因此你可能



会至少 5 分钟才保存一次 RDB 文件。在这种情况下，一旦发生故障停机，你可能会丢失好几分钟的数据。

- 每次保存 RDB 的时候，Redis 都要 `fork()` 出一个子进程，并由子进程来进行实际的持久化工作。在数据集比较庞大时，`fork()` 可能会非常耗时，造成服务器在某某毫秒内停止处理客户端；如果数据集非常巨大，并且 CPU 时间非常紧张的话，那么这种停止时间甚至可能会长达整整一秒。虽然 AOF 重写也需要进行 `fork()`，但无论 AOF 重写的执行间隔有多长，数据的耐久性都不会有任何损失。

## AOF 的优点

- 使用 AOF 持久化会让 Redis 变得非常耐久 (much more durable)：你可以设置不同的 `fsync`策略，比如无 `fsync`，每秒钟一次 `fsync`，或者每次执行写入命令时 `fsync`。
- AOF 的默认策略为每秒钟 `fsync` 一次，在这种配置下，Redis 仍然可以保持良好的性能，并且就算发生故障停机，也最多只会丢失一秒钟的数据（`fsync` 会在后台线程执行，所以主线程可以继续努力地处理命令请求）。
- AOF 文件是一个只进行追加操作的日志文件 (append only log)，因此对 AOF 文件的写入不需要进行 `seek`，即使日志因为某些原因而包含了未写入完整的命令（比如写入时磁盘已满，写入中途停机，等等），`redis-check-aof` 工具也可以轻易地修复这种问题。
- Redis 可以在 AOF 文件体积变得过大时，自动地在后台对 AOF 进行重写：重写后的新 AOF 文件包含了恢复当前数据集所需的最小命令集合。整个重写操作是绝对安全的，因为 Redis 在创建新 AOF 文件的过程中，会继续将命令追加到现有的 AOF 文件里面，即使重写过程中发生停机，现有的 AOF 文件也不会丢失。而一旦新 AOF 文件创建完毕，Redis 就会从旧 AOF 文件切换到新 AOF 文件，并开始对新 AOF 文件进行追加操作。
- AOF 文件有序地保存了对数据库执行的所有写入操作，这些写入操作以 Redis 协议的格式保存，因此 AOF 文件的内容非常容易被读懂，对文件进行分析 (parse) 也很轻松。导出 (export) AOF 文件也非常简单：举个例子，如果你不小心执行了 `FLUSHALL` (<https://link.jianshu.com?t=http://doc.redisfans.com/server/flushall.html#flushall>) 命令，但只要 AOF 文件未被重写，那么只要停止服务器，移除 AOF 文件末尾的 `FLUSHALL` (<https://link.jianshu.com?t=http://doc.redisfans.com/server/flushall.html#flushall>) 命令，并重启 Redis，就可以将数据集恢复到 `FLUSHALL` (<https://link.jianshu.com?t=http://doc.redisfans.com/server/flushall.html#flushall>) 执行之前的状态。

## AOF 的缺点

- 对于相同的数据集来说，AOF 文件的体积通常要大于 RDB 文件的体积。根据所使用的 `fsync`策略，AOF 的速度可能会慢于 RDB。在一般情况下，每秒 `fsync`的性能依然非常高，而关闭 `fsync`可以让 AOF 的速度和 RDB 一样快，即使在高负荷之下也是如此。不过在处理巨大的写入载入时，RDB 可以提供更有保证的最大延迟时间 (latency)。
- AOF 在过去曾经发生过这样的 bug：因为个别命令的原因，导致 AOF 文件在重新载入时，无法将数据集恢复成保存时的原样。（举个例子，阻塞命令 `BRPOPLPUSH` (<https://link.jianshu.com?t=http://doc.redisfans.com/list/brpoplpush.html#brpoplpush>) 就曾经引起过这样的



bug。) 测试套件里为这种情况添加了测试：它们会自动生成随机的、复杂的数据集，并通过重新载入这些数据来确保一切正常。虽然这种 bug 在 AOF 文件中并不常见，但是对比来说，RDB 几乎是不可能出现这种 bug 的。

## RDB 和 AOF，我应该用哪一个？

一般来说，如果想达到足以媲美 PostgreSQL 的数据安全性，你应该同时使用两种持久化功能。

如果你非常关心你的数据，但仍然可以承受数分钟以内的数据丢失，那么你可以只使用 RDB 持久化。

有很多用户都只使用 AOF 持久化，但我们并不推荐这种方式：因为定时生成 RDB 快照 (snapshot) 非常便于进行数据库备份，并且 RDB 恢复数据集的速度也要比 AOF 恢复的速度要快，除此之外，使用 RDB 还可以避免之前提到的 AOF 程序的 bug。

因为以上提到的种种原因，未来我们可能会将 AOF 和 RDB 整合成单个持久化模型。（这是一个长期计划。）

接下来的几个小节将介绍 RDB 和 AOF 的更多细节。

## RDB 快照

在默认情况下，Redis 将数据库快照保存在名字为 dump.rdb 的二进制文件中。

你可以对 Redis 进行设置，让它在“N秒内数据集至少有 M个改动”这一条件被满足时，自动保存一次数据集。

你也可以通过调用 SAVE (<https://link.jianshu.com?t=http://doc.redisfans.com/server/save.html#save>) 或者 BGSAVE (<https://link.jianshu.com?t=http://doc.redisfans.com/server/bgsave.html#bgsave>)，手动让 Redis 进行数据集保存操作。

比如说，以下设置会让 Redis 在满足“60秒内有至少有 1000个键被改动”这一条件时，自动保存一次数据集：

```
save 60 1000
```

这种持久化方式被称为快照 (snapshot)。

## 快照的运作方式

当 Redis 需要保存 dump.rdb 文件时，服务器执行以下操作：

- Redis 调用 fork()，同时拥有父进程和子进程。
- 子进程将数据集写入到一个临时 RDB 文件中。
- 当子进程完成对新 RDB 文件的写入时，Redis 用新 RDB 文件替换原来的 RDB 文件，并删除旧的 RDB 文件。

这种工作方式使得 Redis 可以从写时复制 (copy-on-write) 机制中获益。

## 只进行追加操作的文件 (append-only file, AOF)

快照功能并不是非常耐久 (durable)：如果 Redis 因为某些原因而造成故障停机，那么服务器将丢失最近写入、且仍未保存到快照中的那些数据。

尽管对于某些程序来说，数据的耐久性并不是最重要的考虑因素，但是对于那些追求完全耐久能力 (full durability) 的程序来说，快照功能就不太适用了。



从 1.1 版本开始，Redis 增加了一种完全耐久的持久化方式：AOF 持久化。

你可以通过修改配置文件来打开 AOF 功能：

```
appendonly yes
```

从现在开始，每当 Redis 执行一个改变数据集的命令时（比如 *SET* (<https://link.jianshu.com?t=http://doc.redisfans.com/string/set.html#set>)），这个命令就会被追加到 AOF 文件的末尾。

这样的话，当 Redis 重新启时，程序就可以通过重新执行 AOF 文件中的命令来达到重建数据集的目的。

## AOF 重写

因为 AOF 的运作方式是不断地将命令追加到文件的末尾，所以随着写入命令的不断增加，AOF 文件的体积也会变得越来越大。

举个例子，如果你对一个计数器调用了 100 次 *INCR* (<https://link.jianshu.com?t=http://doc.redisfans.com/string/incr.html#incr>)，那么仅仅是为了保存这个计数器的当前值，AOF 文件就需要使用 100 条记录（entry）。

然而实际上，只使用一条 *SET* (<https://link.jianshu.com?t=http://doc.redisfans.com/string/set.html#set>) 命令已经足以保存计数器的当前值了，其余 99 条记录实际上都是多余的。

为了处理这种情况，Redis 支持一种有趣的特性：可以在不打破服务客户端的情况下，对 AOF 文件进行重建（rebuild）。

执行 *BGREWRITEAOF* (<https://link.jianshu.com?t=http://doc.redisfans.com/server/bgrewriteaof.html#bgrewriteaof>) 命令，Redis 将生成一个新的 AOF 文件，这个文件包含重建当前数据集所需的最少命令。

Redis 2.2 需要自己手动执行 *BGREWRITEAOF* (<https://link.jianshu.com?t=http://doc.redisfans.com/server/bgrewriteaof.html#bgrewriteaof>) 命令；Redis 2.4 则可以自动触发 AOF 重写，具体信息请查看 2.4 的示例配置文件

## AOF 有多耐久？

你可以配置 Redis 多久才将数据 fsync 到磁盘一次。

有三个选项：

- 每次有新命令追加到 AOF 文件时就执行一次 fsync：非常慢，也非常安全。
- 每秒 fsync 一次：足够快（和使用 RDB 持久化差不多），并且在故障时只会丢失 1 秒钟的数据。
- 从不 fsync：将数据交给操作系统来处理。更快，也更不安全的选择。

推荐（并且也是默认）的措施为每秒 fsync 一次，这种 fsync 策略可以兼顾速度 and 安全性。

总是 fsync 的策略在实际使用中非常慢，即使在 Redis 2.0 对相关的程序进行了改进之后仍是如此——频繁调用 fsync 注定了这种策略不可能快得起来。



## 如果 AOF 文件出错了，怎么办？

(<http://doc.redisfans.com/topic/persistence.html#id9>  
(<https://link.jianshu.com?t=http://doc.redisfans.com/topic/persistence.html#id9>)

服务器可能在程序正在对 AOF 文件进行写入时停机，如果停机造成了 AOF 文件出错（corrupt），那么 Redis 在重启时会拒绝载入这个 AOF 文件，从而确保数据的一致性不会被破坏。

当发生这种情况时，可以用以下方法来修复出错的 AOF 文件：

- 为现有的 AOF 文件创建一个备份。
- 使用 Redis 附带的 redis-check-aof 程序，对原来的 AOF 文件进行修复。

```
$ redis-check-aof --fix
```

- （可选）使用 diff -u 对比修复后的 AOF 文件和原始 AOF 文件的备份，查看两个文件之间的不同之处。
- 重启 Redis 服务器，等待服务器载入修复后的 AOF 文件，并进行数据恢复。

## AOF 的运作方式

AOF 重写和 RDB 创建快照一样，都巧妙地利用了写时复制机制。

以下是 AOF 重写的执行步骤：

- Redis 执行 fork()，现在同时拥有父进程和子进程。
- 子进程开始将新 AOF 文件的内容写入到临时文件。
- 对于所有新执行的写入命令，父进程一边将它们累积到一个内存缓存中，一边将这些改动追加到现有 AOF 文件的末尾：这样即使在重写的中途发生停机，现有的 AOF 文件也还是安全的。
- 当子进程完成重写工作时，它给父进程发送一个信号，父进程在接收到信号之后，将内存缓存中的所有数据追加到新 AOF 文件的末尾。

搞定！现在 Redis 原子地用新文件替换旧文件，之后所有命令都会直接追加到新 AOF 文件的末尾。

## 怎么从 RDB 持久化切换到 AOF 持久化

在 Redis 2.2 或以上版本，可以在不重启的情况下，从 RDB 切换到 AOF：

- 为最新的 dump.rdb 文件创建一个备份。
- 将备份放到一个安全的地方。
- 执行以下两条命令：

```
redis-cli> CONFIG SET appendonly yes  
redis-cli> CONFIG SET save ""
```



- 确保命令执行之后，数据库的键的数量没有改变。
- 确保写命令会被正确地追加到 AOF 文件的末尾。

步骤 3 执行的第一条命令开启了 AOF 功能：Redis 会阻塞直到初始 AOF 文件创建完成为止，之后 Redis 会继续处理命令请求，并开始将写入命令追加到 AOF 文件末尾。

步骤 3 执行的第二条命令用于关闭 RDB 功能。这一步是可选的，如果你愿意的话，也可以同时使用 RDB 和 AOF 这两种持久化功能。

## RDB 和 AOF 之间的相互作用

在版本号大于等于 2.4 的 Redis 中，`BGSAVE` (<https://link.jianshu.com?t=http://doc.redisfans.com/server/bgsave.html#bgsave>) 执行的过程中，不可以执行 `BGREWRITEAOF` (<https://link.jianshu.com?t=http://doc.redisfans.com/server/bgrewriteaof.html#bgrewriteaof>)。反过来说，在 `BGREWRITEAOF` (<https://link.jianshu.com?t=http://doc.redisfans.com/server/bgrewriteaof.html#bgrewriteaof>) 执行的过程中，也不可以执行 `BGSAVE` (<https://link.jianshu.com?t=http://doc.redisfans.com/server/bgsave.html#bgsave>)。

这可以防止两个 Redis 后台进程同时对磁盘进行大量的 I/O 操作。

如果 `BGSAVE` (<https://link.jianshu.com?t=http://doc.redisfans.com/server/bgsave.html#bgsave>) 正在执行，并且用户显示地调用 `BGREWRITEAOF` (<https://link.jianshu.com?t=http://doc.redisfans.com/server/bgrewriteaof.html#bgrewriteaof>) 命令，那么服务器将向用户回复一个 OK 状态，并告知用户，`BGREWRITEAOF` (<https://link.jianshu.com?t=http://doc.redisfans.com/server/bgrewriteaof.html#bgrewriteaof>) 已经被预定执行：一旦 `BGSAVE` (<https://link.jianshu.com?t=http://doc.redisfans.com/server/bgsave.html#bgsave>) 执行完毕，`BGREWRITEAOF` (<https://link.jianshu.com?t=http://doc.redisfans.com/server/bgrewriteaof.html#bgrewriteaof>) 就会正式开始。

当 Redis 启动时，如果 RDB 持久化和 AOF 持久化都被打开了，那么程序会优先使用 AOF 文件来恢复数据集，因为 AOF 文件所保存的数据通常是最完整的。

## 备份 Redis 数据

在阅读这个小节前，先将下面这句话铭记于心：一定要备份你的数据库！

磁盘故障，节点失效，诸如此类的问题都可能让你的数据消失不见，不进行备份是非常危险的。

Redis 对于数据备份是非常友好的，因为你可以在服务器运行的时候对 RDB 文件进行复制：RDB 文件一旦被创建，就不会进行任何修改。当服务器要创建一个新的 RDB 文件时，它先将文件的内容保存在一个临时文件里面，当临时文件写入完毕时，程序才使用 `rename(2)` 原子地用临时文件替换原来的 RDB 文件。

这也就是说，无论何时，复制 RDB 文件都是绝对安全的。

以下是我们的建议：

- 创建一个定期任务 (cron job)，每小时将一个 RDB 文件备份到一个文件夹，并且每天将一个 RDB 文件备份到另一个文件夹。
- 确保快照的备份都带有相应的日期和时间信息，每次执行定期任务脚本时，使用 `find` 命令来删除过期的快照：比如说，你可以保留最近 48 小时内的每小时快照，还可以保留最近一两个月的每日快照。



- 至少每天一次，将 RDB 备份到你的数据中心之外，或者至少是备份到你运行 Redis 服务器的物理机器之外。

Redis (/nb/15878374)

© 著作权归作者所有



六尺帐篷 (/u/f8e9b1c246f1)

写了 241233 字，被 15271 人关注，获得了 1455 个喜欢 (/u/f8e9b1c246f1)


喜欢 7





(http://cwb.assets.jianshu.io/notes/images/1632879)

被以下专题收入，发现更多相似内容


投稿管理

- + 收入我的专题
- 

Android... (/c/5139d555c94d?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

Android开发 (/c/d1591c322c89?utm\_source=desktop&utm\_medium=notes-included-collection)
- 


程序员 (/c/NEt52a?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

Android知识 (/c/3fde3b545a35?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

secret... (/c/612ecccbfb5e?utm\_source=desktop&utm\_medium=notes-included-collection)


Redis 持久化 (persistence) 【转】 (/p/3cf6ee170ef7?utm\_campaign=...

本文档翻译自<http://redis.io/topics/persistence>。这篇文章提供了 Redis 持久化的技术性描述，推荐所有 Redis 用户阅读。要更广泛地了解 Redis 持久化，以及这种持久化所保证的耐久性 (durability)，请参...

 daos (/u/b5c4f3f1bf53?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)


redis持久化RDB和AOF (/p/42ee31529682?utm\_campaign=maleskine&...

Redis持久化: 提供了多种不同级别的持久化方式:一种是RDB,另一种是AOF. RDB 持久化可以在指定的时间间隔内生成数据集的时间点快照 (point-in-time snapshot)。AOF 持久化记录服务器执行的所有写操作命令...

 不姓马的小马哥 (/u/85bb71cb6bc9?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)

Redis持久化 (/p/8117b436a4a5?utm\_campaign=maleskine&utm\_conte...

Redis 持久化: 常用的两种持久化 提供了多种不同级别的持久化方式:一种是RDB,另一种是AOF. RDB 持久化可以在指定的时间间隔内生成数据集的时间点快照 (point-in-time snapshot)。相关配置 redis.conf配置...

 边学边记 (/u/bf237bd74005?utm\_campaign=maleskine&utm\_content=user&utm\_medium=seo\_notes&utm\_source=recommendation)





