<http://www.bjpowernode.com/tutorial_baseinterviewquestions/225.html>

## 8、Redis的持久化是什么？

RDB持久化：该机制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。

AOF和RDB的同时应用：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。

## RDB的优缺点？

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。

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，你就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。

## AOF的优缺点？

● 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 onlylog），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。

● 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。