一、什么是数据持久化
二、redis持久化方式
三、redis持久化方式的优缺点
五、总结

一、什么是数据持久化

1、从字面来理解: 持久可以理解为持续多久。因此，数据持久化可以理解为，数据可以保存多久

2、从关系型和非关系型数据库的角度来理解:

2.1、关系型: 往数据库操作数据时，数据的最终结果都是保存在物理磁盘

2.2、非关系型(redis): 往数据库操作数据时，数据先到内存，然后再到物理磁盘。因此，从这个角度来看，数据持久化应该是内存与物理磁盘的相互映射

3、从逻辑角度来理解

3.1、可以减少访问数据库数据次数

3.2、代码重用性高，能够完成大部分数据库操作

3.3、松散耦合，使持久化不依赖于底层数据库和上层业务逻辑实现，更换数据库时只需修改配置文件而不用修改代码

二、redis持久化方式

1、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-timesnapshot）。

2、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。

3、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。

三、redis持久化方式的优缺点

1、rdb方式

1.1、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（disasterrecovery）：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
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。

1.2、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。 **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。**

2、aof方式

2.1、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（appendonly 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人读懂，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 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 命令， 并重启 Redis ， 就可以将数据集恢复到 FLUSHALL 执行之前的状态。

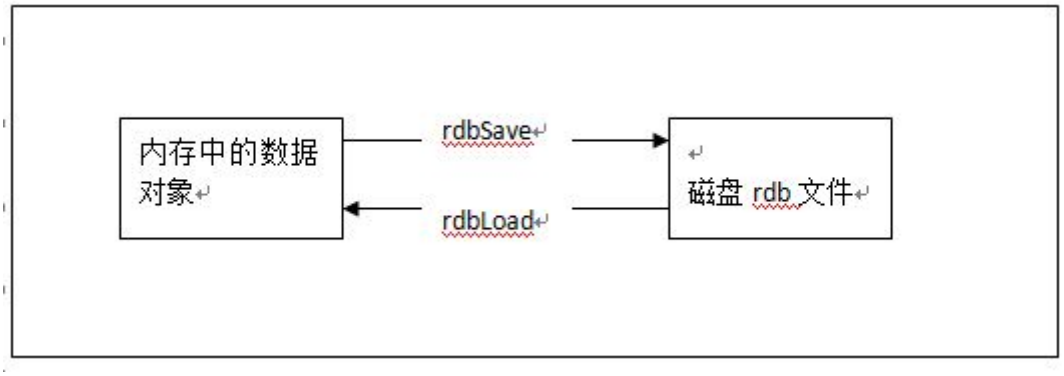
2.2、缺点

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。

四、redis持久化运行实例

1、RDB

在 Redis 运行时， RDB 程序将当前内存中的数据库快照保存到磁盘文件中，在 Redis 重启动时， RDB 程序可以通过载入 RDB 文件来还原数据库的状态。RDB 功能最核心的是 rdbSave 和 rdbLoad 两个函数， 前者用于生成 RDB 文件到磁盘， 而后者则用于将 RDB 文件中的数据重新载入到内存中：



1.1、运作方式

当 Redis 需要保存 dump.rdb 文件时， 服务器执行以下操作：

Ø Redis 调用 fork() ，同时拥有父进程和子进程。

Ø 子进程将数据集写入到一个临时 RDB 文件中。

Ø 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。

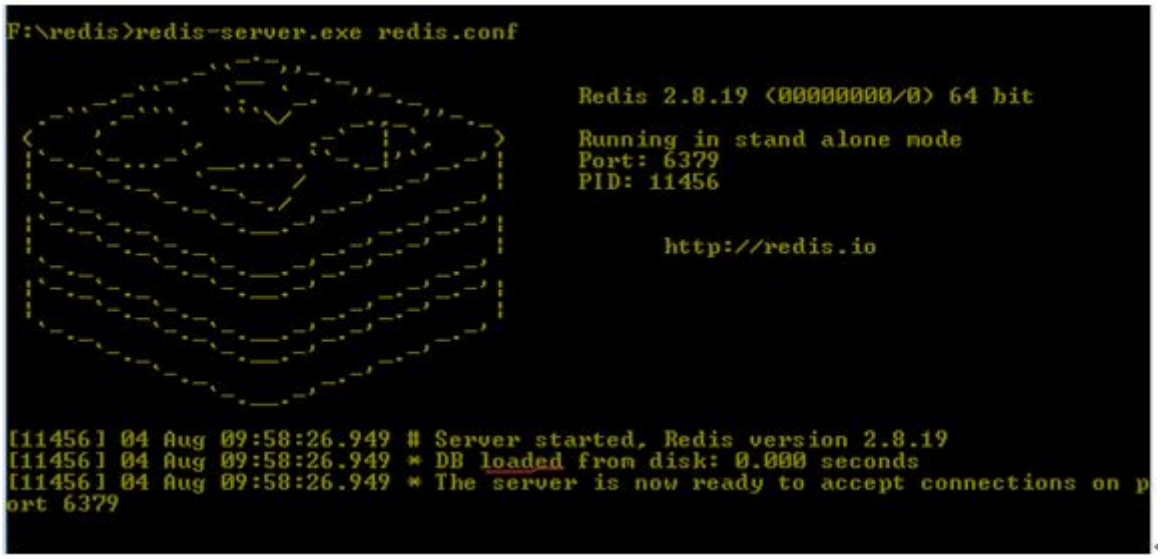
SAVE 和 BGSAVE 两个命令都会调用 rdbSave 函数，但它们调用的方式各有不同：

Ø SAVE 直接调用 rdbSave ，阻塞 Redis 主进程，直到保存完成为止。在主进程阻塞期间，服务器不能处理客户端的任何请求。

Ø BGSAVE 则 fork 出一个子进程，子进程负责调用 rdbSave ，并在保存完成之后向主进程发送信号，通知保存已完成。因为rdbSave 在子进程被调用，所以 Redis 服务器在 BGSAVE 执行期间仍然可以继续处理客户端的请求。

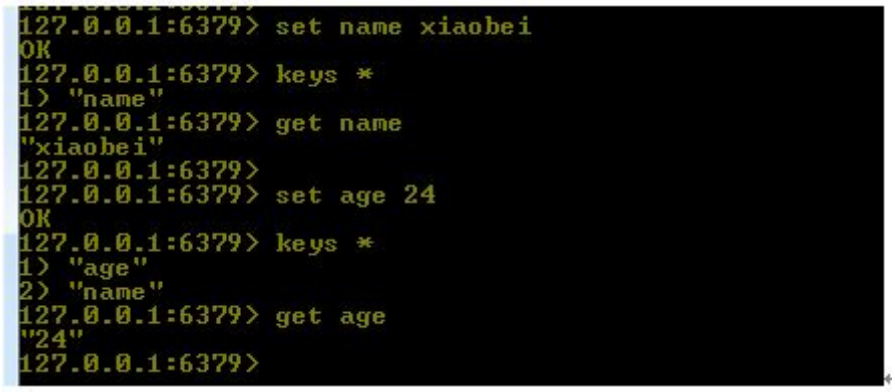
1.2、运行例子

a、启动redis



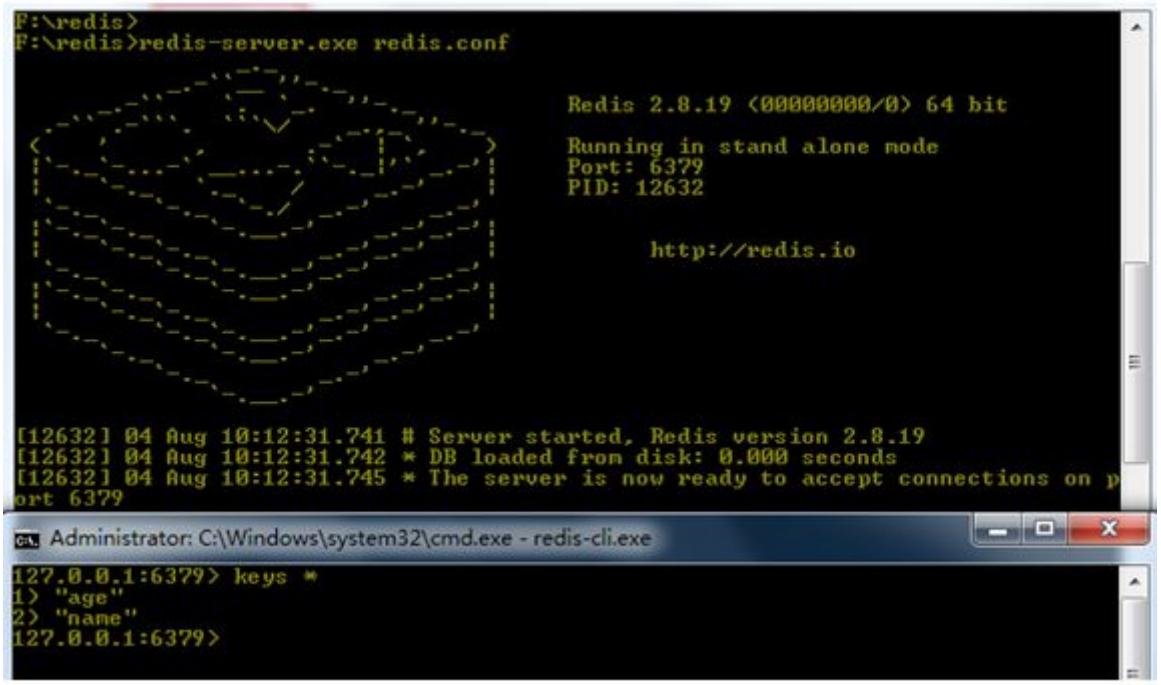
b、物理磁盘

c、内置客户端操作



d、终止客户端

e、重启redis



2、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。

1.1、运作方式

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。

1.2、保存模式

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 策略可以兼顾速度和安全性。

1.3、读取和还原数据

Redis 读取 AOF 文件并还原数据库的详细步骤如下：
Ø 创建一个不带网络连接的伪客户端（fakeclient）。
Ø 读取 AOF 所保存的文本，并根据内容还原出命令、命令的参数以及命令的个数。
Ø 根据命令、命令的参数和命令的个数，使用伪客户端执行该命令。
Ø 执行 2 和 3 ，直到 AOF 文件中的所有命令执行完毕。
完成第 4 步之后， AOF 文件所保存的数据库就会被完整地还原出来。

1.4、例子

a、修改redis配置文件(这里是关闭rdb，而且运行模式为每次更新一次aof文件)

```
appendonly yes

# The name of the append only file (default: "appendonly.aof")
appendfilename "appendonly.aof"
```

b、重启redis

```
F:\redis>redis-server.exe redis.conf
                Redis 2.8.19 (00000000/0) 64 bit
                Running in stand alone mode
                Port: 6379
                PID: 2024

                    http://redis.io

[2024] 04 Aug 10:34:18.043 # Server started, Redis version 2.8.19
[2024] 04 Aug 10:34:18.043 * The server is now ready to accept connections on po
rt 6379
```

c、客户端进行连接并操作

```
127.0.0.1:6379> set name xiaobei
OK
127.0.0.1:6379> get name
"xiaobei"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> set name 'hello world'
OK
127.0.0.1:6379> get name
"hello world"
127.0.0.1:6379>
```

d、终止redis并重启

```
[2024] 04 Aug 10:39:20.558 # User requested shutdown...
[2024] 04 Aug 10:39:20.558 * Calling fsync() on the AOF file.
[2024] 04 Aug 10:39:20.560 # Redis is now ready to exit, bye bye...
```

```
F:\redis>redis-server.exe redis.conf
                                Redis 2.8.19 (00000000/0) 64 bit
                                Running in stand alone mode
                                Port: 6379
                                PID: 13364

                                      http://redis.io

[13364] 04 Aug 10:39:41.163 # Server started, Redis version 2.8.19
[13364] 04 Aug 10:39:41.164 * DB loaded from append only file: 0.001 seconds
[13364] 04 Aug 10:39:41.167 * The server is now ready to accept connections on p
ort 6379
```

3、RDB+AOF

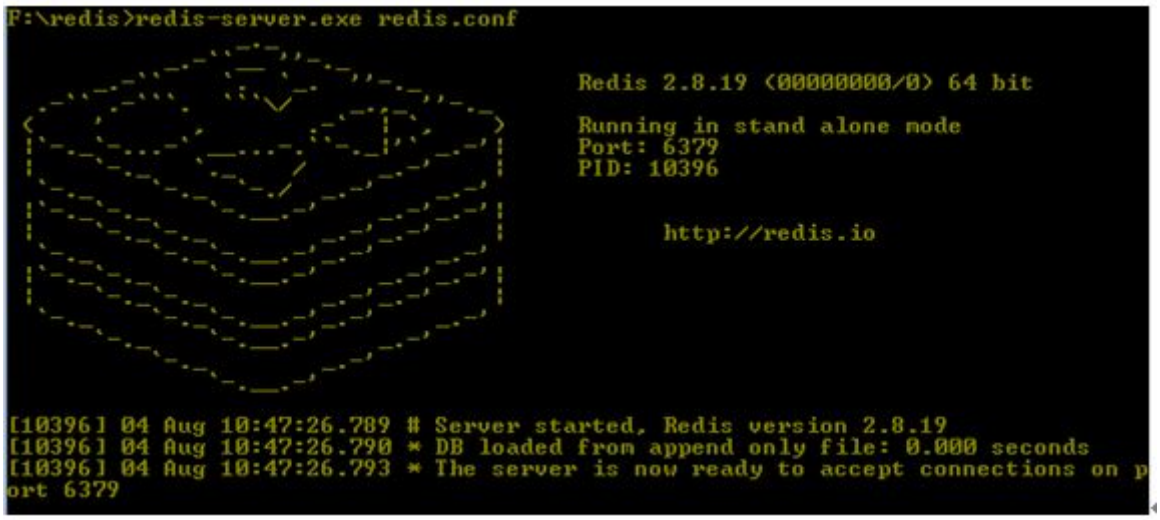
Redis同时运行着rdb和aof两种模式

那么当 Redis 启动时，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

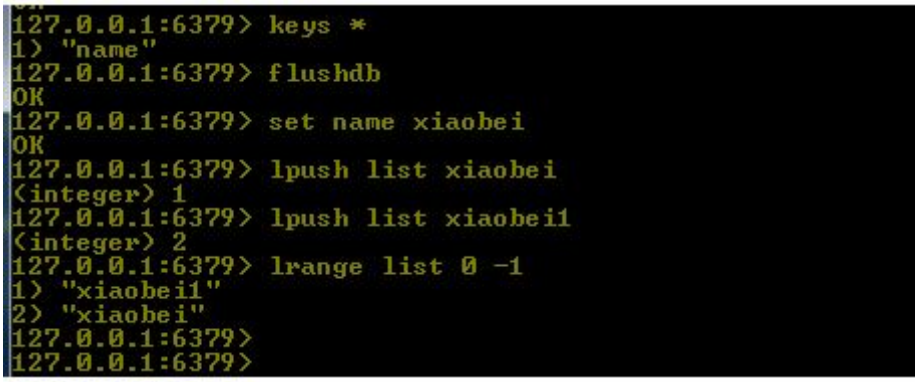
1、运行例子

a、修改redis配置文件，设置同时启动rdb和aof

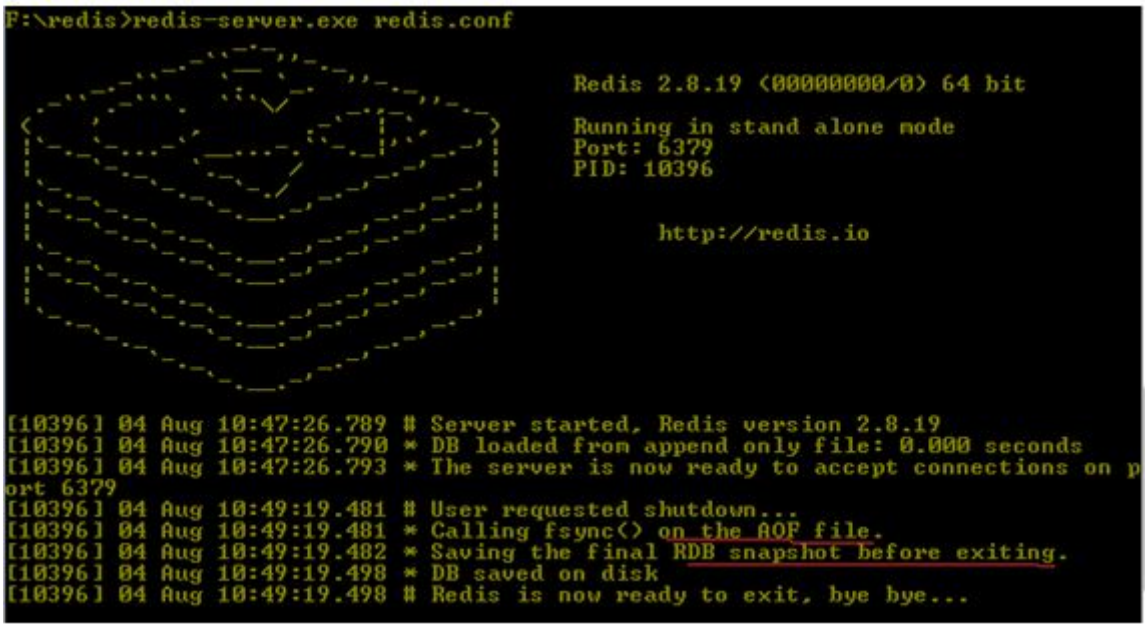
b、启动redis



c、客户端操作数据



d、终止redis



五、总结

1、redis 支持rdb和aof等两种持久方式，如果都关闭rdb和aof等方式。则可以把redis看成是内存缓存

2、如果rdb和aof都开启，则优先考虑aof

3、rdb可以看成是保存数据结果，而aof则是记录修改/写入等操作

4、根据rdb和aof的作用，可以用rdb作为完全备份，而把aof作为增量备份