大数据库系统

3.5 Redis的持久化

3.5 Redis的持久化

- 主要内容

  3.5.1 RDB持久化

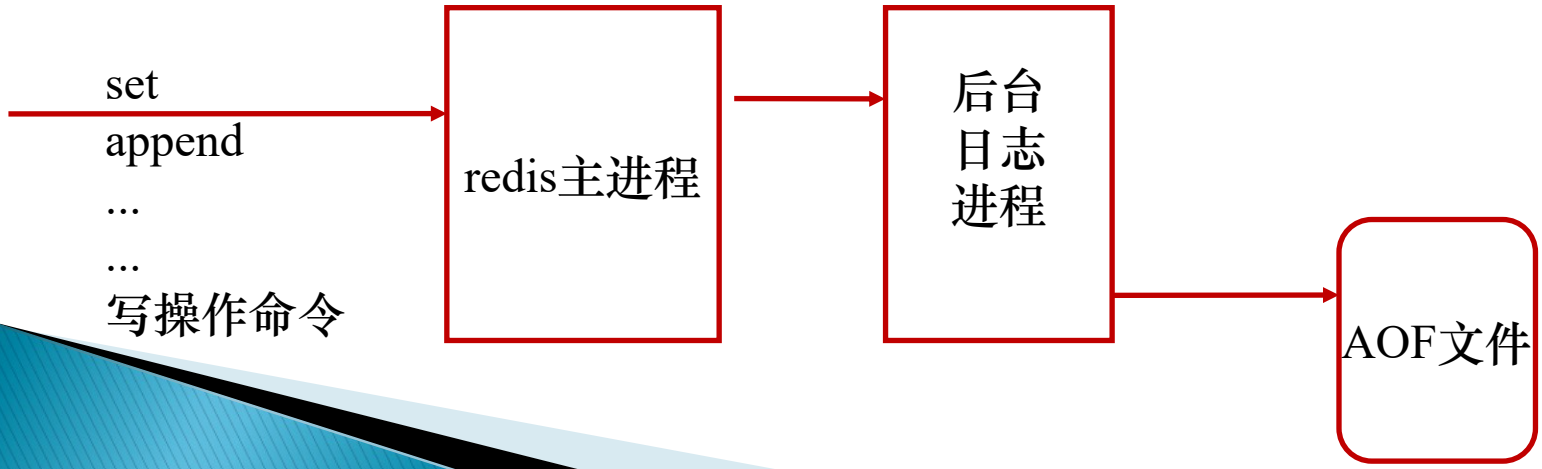
  3.5.2 AOF持久化

3.5.2 AOF持久化

- RDB的缺陷
  - 如果在两个RDB保存点之间发生了断电，那么将会损失两个保存点之间时间段写入的数据
  - 内存数据量大时，影响性能
  - Redis在2.4版本以后，增加了一种新的持久化方式，即AOF持久化方式Append of File，也叫日志持久化
  - 可以两者配合使用，也可以只启用一个

3.5.2 AOF持久化

- AOF（Append Only File）持久化
  - 保存服务器执行的所有写操作命令到单独的日志文件中
  - 在服务器重启时，通过加载日志文件中的这些命令并执行来恢复数据
    - ✓ 实时性更好，也就是当进程意外退出时，丢失的数据更少



3.5.2 AOF持久化

- AOF的相关配置项

  AOF的配置项同样在redis.conf目录中

- appendonly

  是否开启AOF持久化，设置为yes表示开启AOF持久化

```
appendonly no
```

在默认情况下，AOF持久化没有被开启，如果要开启AOF需要把该选项设置为YES

3.5.2 AOF持久化

- appendfsync

设定持久化策略

```
appendfsync everysec
```

该配置项的参数有如下三种：

- appendfsync always　# 每1个命令，都立即同步到AOF，安全，速度慢
- appendfsync everysec # 折衷方案，每秒写1次，丢数据顶多丢1s数据
- appendfsync no　　# 写入工作交给操作系统，由操作系统判断缓冲区大小，统一写入到AOF，同步频率低，速度快（操作系统根据自身缓冲区到一定大小自动写入磁盘，最多30s）

◆ Appendfilename

AOF文件名，可以修改它，默认为appendonly.aof

```
appendonly yes

                                                    aof
appendfilename appendonly.aof
```

注意：默认下这个Appendfilename是注释状态的，去掉“#”；

```
# The working directory.
#
# The DB will be written inside this directory, with the filename specified
# above using the 'dbfilename' configuration directive.
#
# The Append Only File will also be created inside this directory.
#
# Note that you must specify a directory here, not a file name.
dir /usr/local/redis/rdbaof
```

1、高版本的aof文件与rdb文件放在同一个目录，均由“dir”配置项配置目录

2、如果更改AOF和RDB的存放目录，不要忘了需要更改目录的权限，可使用chmod 777 更改目录权限，例如：sudo chmod 777 /usr/local/redis/rdbaof/

3.5.2 AOF持久化

- 创建AOF文件并查看

1、使用pkill -9命令关闭redis，vim进入redis.conf进行配置，打开AOF功能，设置AOF文件目录为“/var/rdb/”,no-appendfsync-on-rewrite设置为no，appendfsync设置为everysec，其余选项均为默认值，保存并退出

```
[root@localhost redis]# pkill -9 redis
```

```
appendonly yes

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

appendfilename "appendonly.aof"
# Note that you must specify a directory here, not a file name.
dir /var/rdb/

# appendfsync always
appendfsync everysec
# appendfsync no

no-appendfsync-on-rewrite no
```

2、终端1启动redis服务器，终端2进入/var/rdb/查看里面的AOF文件内容

```
[root@localhost redis]# ls /var/rdb/
appendonly.aof  dump.rdb
[root@localhost redis]# more /var/rdb/appendonly.aof
[root@localhost redis]#
```

终端2

可以看到/var/rdb目录下不仅有上节课的rdb文件，也有了appendonly.aof文件

3、终端2连接到服务器，往数据库中写入两个键值site和name，由于之前AOF触发条件为每秒1次，此时肯定已经写入AOF文件了

```
redis 127.0.0.1:6379> keys *
(empty list or set)
redis 127.0.0.1:6379> set site www.zixue.it
OK
redis 127.0.0.1:6379> set name lily
OK
redis 127.0.0.1:6379>
```

终端2

4、打开终端3，查看此时AOF文件的内容

```
[root@localhost redis]# more /var/rdb/appendonly.aof
*2
$6
SELECT
$1
0
*3
$3
set
$4
site
$12
www.zixue.it
*3
$3
set
$4
name
$4
lily
```

可以看到，此时AOF文件里面记录了刚刚我们输入的两个键值命令

终端3

3.5.2 AOF持久化

5、在终端2键值name由“lily”改成“lucy”，并在终端1强行用pkill关闭服务器，模拟突然断电，再重启redis服务器

```
redis 127.0.0.1:6379> set name lily
OK
redis 127.0.0.1:6379> set name lucy
OK
redis 127.0.0.1:6379>
```

终端2

```
[root@localhost redis]# pkill -9 redis
[root@localhost redis]#
```

终端1

6、终端2连入客户端，查看断电前更改的键值name的内容

```
redis 127.0.0.1:6379> get name
"lucy"
redis 127.0.0.1:6379>
```

终端2

可以发现即使服务器断电了，也能成功地恢复了原来的数据

如果靠rdb，则这些数据都将丢失，而AOF可以在秒级保证数据完整性，顶多丢失1秒的数据

3.5.2 AOF持久化

问题：

在上述第2步中看到，/var/rdb目录下存在有上次课留下来的RDB，存有site键，可是服务器启动后，在第3步中的key *命令后显示数据库是空的，为什么没有恢复呢?

上次课的数据库

- 5、启动客户端，连上服务器，分别查询两个键值是否存在

```
redis 127.0.0.1:6379> get site
"www.zixue.it"
```

3.5.2 AOF持久化

◆ AOF重写

在讲AOF重写前，先看这么一个例子

1、先flushdb，在终端2添加键值age

```
redis 127.0.0.1:6379> set age 1
OK
redis 127.0.0.1:6379>
```

终端2

2、在终端3查看AOF文件

```
[root@localhost redis]# tail /var/rdb/appendonly.aof
SELECT
$1
0
*3
$3
set
$3
age
$1
1
```

终端3

3、在终端2执行多次incr指令增加键值age里的数值，incr了N次

```
redis 127.0.0.1:6379> incr age
(integer) 2
redis 127.0.0.1:6379> incr age
(integer) 3
redis 127.0.0.1:6379> incr age
(integer) 4
redis 127.0.0.1:6379> incr age
(integer) 5
```

终端2

3.5.2 AOF持久化

4、到终端3再次查看AOF文件的内容，由于incr数量多，AOF内容也就多，我们查看AOF文件的末尾10行的内容

```
[root@localhost redis]# tail /var/rdb/appendonly.aof
*2
$4
incr
$3
age
*2
$4
incr
$3
age
[root@localhost redis]#
```

终端3

可以看到在AOF中反复多次地写入了我们刚刚键入的incr命令

- 想象一下，如果对一个key反复incr或者反复set1000次，实际上对于key而言，它只有一个最终值，而对于AOF文件而言，则需要记录1000多条命令，这会导致什么问题?
- 1、AOF文件冗余大；2、恢复时非常慢；

3.5.2 AOF持久化

- 解决方法：

  由于所有key在内存中都只有1个具体状态，因此可以将内存中的所有key，都当成新key以最终value状态存入

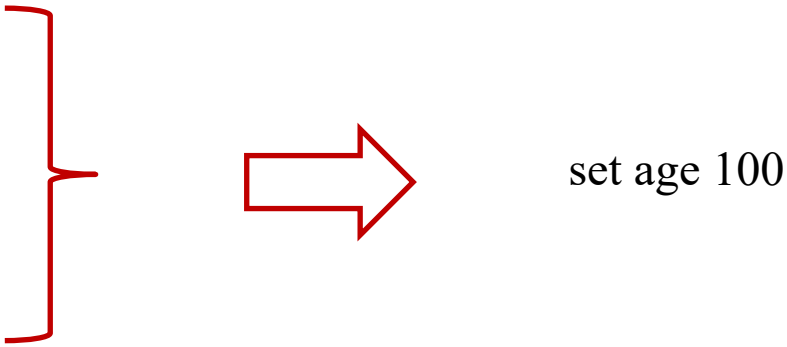
如：

set age -1

set age 0

incr age

……(100次incr)



set age 100

在某个时间点，对AOF进行上述过程处理，是AOF重写的一种方法

3.5.2 AOF持久化

- AOF重写
  - AOF持久化的实现是通过保存被执行的写命令来保存数据库数据
  - 随着运行时间的增加，AOF文件的内容增大，占空间变多
  - 太大的AOF文件会影响服务器的正常运行，在执行数据恢复时，将会耗费更多的时间

AOF重写是指把内存中的数据，逆化成命令，写入到AOF日志里，以解决AOF日志过大的问题

3.5.2 AOF持久化

- AOF重写方式
  - AOF文件重写功能会丢弃过期的数据，也就是过期的数据不会被写入AOF文件中
  - AOF文件重写功能会丢弃无效的命令，无效的命令将不会被写入AOF文件中无效命令包括重复设置某个键值对时的命令、删除某些数据的命令、过期数据的命令等
  - AOF文件重写功能可以将多条命令合并为一条命令，然后写入AOF文件中

3.5.2 AOF持久化

- AOF文件重写的触发方式

1、当前AOF文件同时满足redis.conf中的auto-aof-rewrite-percentage和auto-aof-rewrite-min-size设定的条件时触发重写

2、手动重写方式：执行BGREWRITEAOF命令触发AOF文件重写

3.5.2 AOF持久化

- auto-aof-rewrite-percentage

指定Redis重写AOF文件的条件，默认为100

```
auto-aof-rewrite-percentage 100
```

- 如果当前AOF文件的增长量大于上次AOF文件的100%，就会触发重写操作；
- 如果将该选项设置为0，则不会触发重写操作

- 如果只有这个选项会有什么问题?

  刚开始AOF文件很小，可能很快就需要重写aof文件，刚写了1个命令就重写。。。

3.5.2 AOF持久化

- auto-aof-rewrite-min-size

  指定触发重写操作的AOF文件的大小，默认为64MB

```
auto-aof-rewrite-min-size 64mb
```

  - 如果当前AOF文件的大小低于该值，此时就算当前文件的增量比例达到了auto-aof-rewrite-percentage选项所设置的条件，也不会触发重写操作
  - 只有同时满足auto-aof-rewrite-min-size和auto-aof-rewrite-percentage这两个选项所设置的条件，才会触发重写操作

3.5.2 AOF持久化

- no-appendfsync-on-rewrite

在AOF重写的过程中，是否停止主进程对磁盘的操作

```
no-appendfsync-on-rewrite yes
```

- 默认值为NO
- 如果参数为YES，则表示重写AOF时，停止主进程写磁盘操作，则可以减轻重写AOF文件时CPU和硬盘的负载，但可能会丢失重写AOF文件过程中的修改数据
- 如果参数为NO，则表示重写AOF时，允许主进程写磁盘

3.5.2 AOF持久化

◆ 怎么理解no-appendfsync-on-rewrite

- 由于AOF重写由一个子进程执行，并且操作会涉及大量的磁盘操作，而主进程接收用户的写操作、写入RDB等也需要占用磁盘；如果AOF文件非常大，则重写时负载非常大，造成主进程出现阻塞的情形

- 如果参数为YES，则表示子进程重写AOF时，停止主进程写磁盘，此时主进程会将命令暂时写入缓存中，等AOF重写完，再根据缓存中的内容写入磁盘，如果此时断电，则缓存中数据丢失，造成数据损失

- 如果参数为NO，则表示重写AOF时，允许主进程写磁盘，能保证数据不会丢失，但如果AOF文件较大，则会造成负载较高甚至进程发生阻塞

- 需要在负载性能与安全性之间根据实际应用情况进行平衡

3.5.2 AOF持久化

- 查看AOF重写效果

1、关闭redis服务器，修改redis.conf文件中的AOF相关配置项，为了能尽快看到AOF重写，将auto-aof-rewrite-percentage值改小，再重新开启服务器

```
auto-aof-rewrite-percentage 100
auto-aof-rewrite-min-size 32M
```

终端1

./src/redis-server redis.conf

2、如何快速增加AOF的大小，即大量的命令操作?

再终端2使用redis-benchmark命令，产生50000条指令

./src/redis-benchmark –n 50000　　终端2

3、在终端3不断观察AOF文件的大小

```
[root@localhost redis]# ll /var/rdb/
total 44
-rw-r--r-- 1 root root   275 Oct  3 21:32 appendonly.aof
-rw-r--r-- 1 root root 40096 Oct  3 19:41 dump.rdb
[root@localhost redis]# ll /var/rdb/
total 576
-rw-r--r-- 1 root root 546598 Oct  3 21:45 appendonly.aof
-rw-r--r-- 1 root root  40096 Oct  3 19:41 dump.rdb
[root@localhost redis]#
```

终端3

可以看到，AOF文件在不断地增大，可能会有延迟，因为文件已写入，系统还没来得及改文件大小

4、在终端3观察AOF文件最后几行的内容（反复的在使用set命令对键值xxx存随机值）

```
*3
$3
SET
$16
key:__rand_int__
$3
xxx
*3
$3
SET
$16
key:__rand_int__
$3
xxx
*3
$3
[root@localhost redis]#
```

终端3

5、继续查看AOF文件的大小

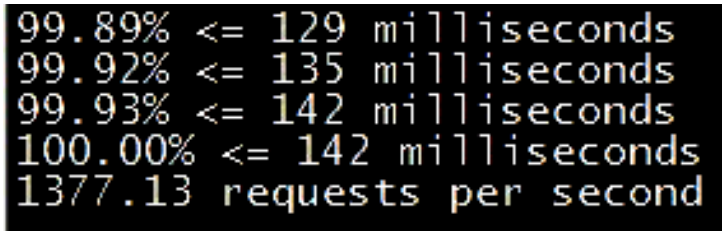
```
-rw-r--r-- 1 root root  19M Oct  3 21:58 appendonly.aof
-rw-r--r-- 1 root root 274K Oct  3 21:58 dump.rdb
[root@localhost redis]# ll /var/rdb/ -h
total 5.0M
-rw-r--r-- 1 root root 4.8M Oct  3 22:05 appendonly.aof
-rw-r--r-- 1 root root 274K Oct  3 22:04 dump.rdb
[root@localhost redis]#
```

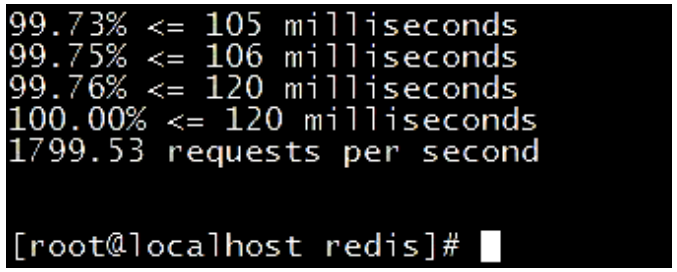
终端3

可以看到了AOF文件从19M，七分钟后降到了4.8M，说明达到了触发条件后AOF文件发生了重写

3.5.2 AOF持久化

在benchmark运行结束后，我们可以看到这个值，左图为刚刚的值，右图为上次课的值，为什么？





redis请求效率下降了，因为AOF每秒都要写磁盘，影响了效率，但这也保证了数据的安全性

3.5.2 AOF持久化

- 手动重写AOF文件

BGREWRITEAOF命令

例：手动触发重写AOF文件

1、在终端3输入BGREWRITEAOF命令手动触发重写

```
redis 127.0.0.1:6379> bgrewriteaof
Background append only file rewriting started
```

2、再次观察AOF文件的大小

```
[root@localhost redis]# ll /var/rdb/ -h
total 924K
-rw-r--r-- 1 root root 646K Oct  3 22:07 appendonly.aof
-rw-r--r-- 1 root root 274K Oct  3 22:05 dump.rdb
[root@localhost redis]#
```

可以看到AOF文件又从原来的4.8M重写后变成了646k

```
[root@localhost redis]# ll /var/rdb/ -h
total 5.0M
-rw-r--r-- 1 root root 4.8M Oct  3 22:05 appendonly.aof
-rw-r--r-- 1 root root 274K Oct  3 22:04 dump.rdb
```

3.5.2 AOF持久化

◆ AOF文件后台重写

为了不妨碍重写AOF时redis继续接收用户其他指令，Redis将AOF文件重写程序放到了一个子进程中执行

存在问题：

- 当子进程进行AOF文件重写的时候，Redis服务器可以继续执行来自客户端的命令请求，新的命令对现有数据库状态进行修改，使得服务器当前的数据库状态与重写的AOF文件所保存的数据库状态不一致

3.5.2 AOF持久化

◆ 解决数据库状态不一致的问题

Redis服务器设置了一个AOF文件重写缓存区

- 当Redis服务器成功执行完一条写命令后，它会将这条写命令发送给AOF文件重写缓存区

如果子进程完成了AOF文件重写的工作

- 将AOF文件重写缓存区中的所有内容写入新的AOF文件中
- 新的AOF文件所保存的数据库状态与服务器当前的数据库状态保持一致

3.5.2 AOF持久化

◆ AOF文件修复

如果在Redis服务器启动加载AOF文件时，发现AOF文件被损坏，那么服务器会拒绝加载这个AOF文件

如果AOF文件被损坏，则可以通过以下方法来修复：

- 及时备份现有AOF文件。
- 利用Redis自带的redis-check-aof程序，对原来的AOF文件进行修复，命令如下：

  redis-check-aof –fix AOFname

- AOFname表示需要修复的AOF文件

3.5.2 AOF持久化

- 回到之前提的问题：

  在/var/rdb目录下存在有上次课留下来的RDB，存有site键。可是服务器启动后，key *命令后显示数据库是空的，为什么没有恢复呢?

- 5、启动客户端，连上服务器，分别查询两个键值是否存在

```
redis 127.0.0.1:6379> get site
"www.zixue.it"
```

上次课的数据库

```
[root@localhost redis]# ls /var/rdb/
appendonly.aof  dump.rdb
[root@localhost redis]# more /var/rdb/appendonly.aof
[root@localhost redis]#
```

```
redis 127.0.0.1:6379> keys *
(empty list or set)
redis 127.0.0.1:6379> set site www.zixue.it
OK
redis 127.0.0.1:6379> set name lily
OK
redis 127.0.0.1:6379>
```

重启后的数据库

3.5.2 AOF持久化

- AOF与RDB的优先级
  - 如果AOF文件与RDB文件同时存在，服务器启动后会优先使用AOF文件进行恢复数据库
  - 上个问题中，由于启动前，AOF文件里什么都没有，因此按照AOF文件来恢复，数据库就是空的
  - 如果要使用RDB恢复，则需要删掉AOF文件，关闭AOF，将RDB功能开启

3.5.2 AOF持久化

- 问：RDB恢复的速度快，还是AOF恢复的速度快?

  RDB快，因为其是数据的内存映射，即快照，直接载入到内存即可，AOF文件记录的是一行一行的命令，需要逐条执行才能得到数据库的最终状态

3.5.2 AOF持久化

- AOF持久化的优点

1、如果将appendfsync设置为always或everysec，则最多丢失1秒的数据甚至不丢失数据，安全性较好

2、高低版本redis的AOF兼容性比较好

3.5.2 AOF持久化

- AOF持久化的缺陷

1、AOF文件的体积会随着时间的推移逐渐变大，导致在加载时速度会比较慢，进而影响数据库状态的恢复速度，性能快速下降

2、AOF文件恢复数据的速度会慢于使用RDB文件恢复数据的速度

3.5.2 AOF持久化

- AOF持久化与RDB持久化抉择
  - 现实应用中存在各种未知风险因素，强烈建议同时使用AOF持久化和RDB持久化，以便最大限度地保证数据的持久化与安全性
  - 如果只使用RDB，能够定时生成RDB快照，便于进行数据库数据备份，同时也能提高服务器的性能，恢复数据的速度要快于AOF恢复数据的速度；但是，必须承受如果服务器出现故障，则会丢失部分数据的风险
  - 如果只使用AOF持久化，未重写的AOF文件体积较大，写入AOF或者重写AOF的过程需要占用大量磁盘；在恢复数据时会比较慢，会影响服务器的性能

总结

- 什么是AOF持久化
- AOF的相关配置项

  appendonly

  appendfsync及三种参数

  no-appendfsync-on-rewrite

  appendfilename
- 触发AOF并查看AOF文件

总结

- AOF重写
  - 什么是AOF重写
  - 三种重写方式
  - 两种触发重写方式
  - 如何解决重写数据不一致
  - AOF文件修复
- AOF的优缺点
- AOF与RDB的抉择