Redis详解（七）------ AOF 持久化

目录

上一篇文章我们介绍了Redis的RDB持久化，RDB 持久化存在一个缺点是一定时间内做一次备份，如果redis意外down掉的话，就会丢失最后一次快照后的所有修改(数据有丢失)。对于数据完整性要求很严格的需求，怎么解决呢?

本篇博客接着来介绍Redis的另一种持久化方式——AOF。

回到顶部

1、AOF简介

Redis的持久化方式之一RDB是通过保存数据库中的键值对来记录数据库的状态。而另一种持久化方式 AOF 则是通过保存Redis服务器所执行的写命令来记录数据库状态。

比如对于如下命令：

```
127.0.0.1:6379> set str1 "123"
OK
127.0.0.1:6379> sadd str2 "1" "2" "3"
(integer) 3
127.0.0.1:6379> lpush str3 "1" "2" "3"
(integer) 3
127.0.0.1:6379>
```

RDB 持久化方式就是将 str1,str2,str3 这三个键值对保存到 RDB文件中，而 AOF 持久化则是将执行的 set,sadd,lpush 三个命令保存到 AOF 文件中。

回到顶部

2、AOF 配置

在 redis.conf 配置文件的 APPEND ONLY MODE 下：

昵称：YSOcean
园龄：2年1个月
粉丝：2404
关注：13
+加关注

< 2019年4月 >

日	一	二	三	四	五	六
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

我的标签

Linux系列教程(25)
深入理解计算机系统(24)
Java数据结构和算法(16)
MyBatis详解系列(11)
JDK源码解析(11)
Maven系列教程(8)
Redis详解(8)
Spring入门系列(8)
Java IO详解系列(7)
Java高并发设计(7)
更多

随笔分类

Java SE(22)
JavaWeb(34)
Java高并发
Java关键字(6)
Java数据结构和算法(15)

```
652 ############################## APPEND ONLY MODE ###############
653
654 # By default Redis asynchronously dumps the dataset on disk. This m
655 # good enough in many applications, but an issue with the Redis pro
656 # a power outage may result into a few minutes of writes lost (depe
657 # the configured save points).
658 #
659 # The Append Only File is an alternative persistence mode that prov
660 # much better durability. For instance using the default data fsync
661 # (see later in the config file) Redis can lose just one second of
662 # dramatic event like a server power outage, or a single write if s
663 # wrong with the Redis process itself happens, but the operating sy
664 # still running correctly.
665 #
666 # AOF and RDB persistence can be enabled at the same time without p
667 # If the AOF is enabled on startup Redis will load the AOF, that is
668 # with the better durability guarantees.
669 #
670 # Please check http://redis.io/topics/persistence for more informat
671
672 appendonly no
673
674 # The name of the append only file (default: "appendonly.aof")
675
676 appendfilename "appendonly.aof"
677
678 # The fsync() call tells the Operating System to actually write dat
679 # instead of waiting for more data in the output buffer. Some OS wi
680 # data on disk, some other OS will just try to do it ASAP.
681 #
```

①、**appendonly**：默认值为no，也就是说redis 默认使用的是rdb方式持久化，如果想要开启 AOF 持久化方式，需要将 appendonly 修改为 yes。

②、**appendfilename** ： aof文件名，默认是"appendonly.aof"

③、**appendfsync:** aof持久化策略的配置；

no表示不执行fsync，由操作系统保证数据同步到磁盘，速度最快，但是不太安全；

always表示每次写入都执行fsync，以保证数据同步到磁盘，效率很低；

everysec表示每秒执行一次fsync，可能会导致丢失这1s数据。通常选择 everysec ，兼顾安全性和效率。

④、no-appendfsync-on-rewrite：在aof重写或者写入rdb文件的时候，会执行大量IO，此时对于everysec和always的aof模式来说，执行fsync会造成阻塞过长时间，no-appendfsync-on-rewrite字段设置为默认设置为no。如果对延迟要求很高的应用，这个字段可以设置为yes，否则还是设置为no，这样对持久化特性来说这是更安全的选择。 设置为yes表示rewrite期间对新写操作不fsync,暂时存在内存中,等rewrite完成后再写入，默认为no，建议yes。Linux的默认fsync策略是30秒。可能丢失30秒数据。默认值为no。

⑤、auto-aof-rewrite-percentage：默认值为100。aof自动重写配置，当目前aof文件大小超过上一次重写的aof文件大小的百分之多少进行重写，即当aof文件增长到一定大小的时候，Redis能够调用bgrewriteaof对日志文件进行重写。当前AOF文件大小是上次日志重写得到AOF文件大小的二倍（设置为100）时，自动启动新的日志重写过程。

⑥、auto-aof-rewrite-min-size：64mb。设置允许重写的最小aof文件大小，避免了达到约定百分比但尺寸仍然很小的情况还要重写。

⑦、aof-load-truncated：aof文件可能在尾部是不完整的，当redis启动的时候，aof文件的数据被载入内存。重启可能发生在redis所在的主机操作系统宕机后，尤其在ext4文件系统没有加上data=ordered选项，出现这种现象 redis宕机或者异常终止不会造成尾部不完整现象，可以选择让redis退出，或者导入尽可能多的数据。如果选择的是yes，当截断的aof文件被导入的时候，会自动发布一个log给客户端然后load。如果是no，用户必须手动redis-check-aof修复AOF文件才可以。默认值为 yes。

回到顶部

3、开启 AOF

将 redis.conf 的 appendonly 配置改为 yes 即可。

AOF 保存文件的位置和 RDB 保存文件的位置一样，都是通过 redis.conf 配置文件的 dir 配置：

```
# The working directory.
#
# The DB will be written inside this directory, with the filename specif
# 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 ./
```

可以通过 config get dir 命令获取保存的路径。

回到顶部

4、AOF 文件恢复

重启 Redis 之后就会进行 AOF 文件的载入。

异常修复命令：redis-check-aof --fix 进行修复

回到顶部

5、AOF 重写

由于AOF持久化是Redis不断将写命令记录到 AOF 文件中，随着Redis不断的进行，AOF 的文件会越来越大，文件越大，占用服务器内存越大以及 AOF 恢复要求时间越长。为了解决这个问题，Redis新增了重写机制，当AOF文件的大小超过所设定的阈值时，Redis就会启动AOF 文件的内容压缩，只保留可以恢复数据的最小指令集。可以使用命令 bgrewriteaof 来重新。

比如对于如下命令：

```
127.0.0.1:6379> flushall
OK
127.0.0.1:6379> sadd animals "cat"
(integer) 1
127.0.0.1:6379> sadd animals "dog" "panda" "tiger"
(integer) 3
127.0.0.1:6379> srem animals "cat"
(integer) 1
127.0.0.1:6379> sadd animals "lion" "cat"
(integer) 2
127.0.0.1:6379> sdiff
(error) ERR wrong number of arguments for 'sdiff' command
127.0.0.1:6379> sdiff animals
1) "dog"
2) "tiger"
3) "panda"
4) "lion"
5) "cat"
127.0.0.1:6379>
```

如果不进行 AOF 文件重写，那么 AOF 文件将保存四条 SADD 命令，如果使用AOF 重写，那么AOF 文件中将只会保留下面一条命令：

```
1 sadd animals "dog" "tiger" "panda" "lion" "cat"
```

也就是说 AOF 文件重写并不是对原文件进行重新整理，而是直接读取服务器现有的键值对，然后用一条命令去代替之前记录这个键值对的多条命令，生成一个新的文件后去替换原来的 AOF 文件。

AOF 文件重写触发机制：通过 redis.conf 配置文件中的 auto-aof-rewrite-percentage：默认值为100，以及auto-aof-rewrite-min-size：64mb 配置，也就是说默认Redis会记录上次重写时的AOF大小，**默认配置是当AOF文件大小是上次rewrite后大小的一倍且文件大于64M时触发。**

5. Java数据结构和算法（七）——链表(58754)

评论排行榜

1. 深入理解计算机系统（1.1）------Hello World 是如何运行的(28)

2. SpringMVC详解（四）----------SSM三大框架整合之登录功能实现(23)

3. Java数据结构和算法（十）——二叉树(21)

4. 深入理解计算机系统（序章）------谈程序员为什么要懂底层计算机结构(20)

5. Java 集合详解(20)

这里再提一下，我们知道 Redis 是单线程工作，如果 重写 AOF 需要比较长的时间，那么在重写 AOF 期间，Redis将长时间无法处理其他的命令，这显然是不能忍受的。Redis为了克服这个问题，解决办法是将 AOF 重写程序放到子程序中进行，这样有两个好处：

①、子进程进行 AOF 重写期间，服务器进程（父进程）可以继续处理其他命令。

②、子进程带有父进程的数据副本，使用子进程而不是线程，可以在避免使用锁的情况下，保证数据的安全性。

使用子进程解决了上面的问题，但是新问题也产生了：因为子进程在进行 AOF 重写期间，服务器进程依然在处理其它命令，这新的命令有可能也对数据库进行了修改操作，使得当前数据库状态和重写后的 AOF 文件状态不一致。

为了解决这个数据状态不一致的问题，Redis 服务器设置了一个 AOF 重写缓冲区，这个缓冲区是在创建子进程后开始使用，当Redis服务器执行一个写命令之后，就会将这个写命令也发送到 AOF 重写缓冲区。当子进程完成 AOF 重写之后，就会给父进程发送一个信号，父进程接收此信号后，就会调用函数将 AOF 重写缓冲区的内容都写到新的 AOF 文件中。

这样将 AOF 重写对服务器造成的影响降到了最低。

回到顶部

6、AOF的优缺点

优点：

①、AOF 持久化的方法提供了多种的同步频率，即使使用默认的同步频率每秒同步一次，Redis 最多也就丢失 1 秒的数据而已。

②、AOF 文件使用 Redis 命令追加的形式来构造，因此，即使 Redis 只能向 AOF 文件写入命令的片断，使用 redis-check-aof 工具也很容易修正 AOF 文件。

③、AOF 文件的格式可读性较强，这也为使用者提供了更灵活的处理方式。例如，如果我们不小心错用了 FLUSHALL 命令，在重写还没进行时，我们可以手工将最后的 FLUSHALL 命令去掉，然后再使用 AOF 来恢复数据。

缺点：

①、对于具有相同数据的的 Redis，AOF 文件通常会比 RDF 文件体积更大。

②、虽然 AOF 提供了多种同步的频率，默认情况下，每秒同步一次的频率也具有较高的性能。但在 Redis 的负载较高时，RDB 比 AOF 具好更好的性能保证。

③、RDB 使用快照的形式来持久化整个 Redis 数据，而 AOF 只是将每次执行的命令追加到 AOF 文件中，因此从理论上说，RDB 比 AOF 方式更健壮。官方文档也指出，AOF 的确也存在一些 BUG，这些 BUG 在 RDB 没有存在。

那么对于 AOF 和 RDB 两种持久化方式，我们应该如何选择呢?

如果可以忍受一小段时间内数据的丢失，毫无疑问使用 RDB 是最好的，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，而且使用 RDB 还可以避免 AOF 一些隐藏的 bug；否则就使用 AOF 重写。但是一般情况下建议不要单独使用某一种持久化机制，而是应该两种一起用，在这种情况下,当redis重启的时候会优先载入AOF文件来恢复原始的数据，因为在通常情况下AOF文件保存的数据集要比RDB文件保存的数据集要完整。Redis后期官方可能都有将两种持久化方式整合为一种持久化模型。

分类：Redis详解

标签： Redis详解

« 上一篇：Redis详解（六）------ RDB 持久化
» 下一篇：Redis详解（八）------ 主从复制

posted @ 2018-06-09 10:25 YSOcean 阅读(1969) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。