Redis详解（六）------ RDB 持久化

目录

前面我们说过，Redis 相对于 Memcache 等其他的缓存产品，有一个比较明显的优势就是 Redis 不仅仅支持简单的key-value类型的数据，同时还提供list，set，zset，hash等数据结构的存储。这几种丰富的数据类型我们花了两篇文章进行了详细的介绍，接下来我们要介绍 Redis 的另外一大优势——持久化。

由于 Redis 是一个内存数据库，所谓内存数据库，就是将数据库中的内容保存在内存中，这与传统的MySQL，Oracle等关系型数据库直接将内容保存到硬盘中相比，内存数据库的读写效率比传统数据库要快的多（内存的读写效率远远大于硬盘的读写效率）。但是保存在内存中也随之带来了一个缺点，一旦断电或者宕机，那么内存数据库中的数据将会全部丢失。

为了解决这个缺点，Redis提供了将内存数据持久化到硬盘，以及用持久化文件来恢复数据库数据的功能。Redis 支持两种形式的持久化，一种是RDB快照（snapshotting），另外一种是 AOF（append-only-file）。本篇博客先对 RDB 快照进行介绍。

回到顶部

1、RDB 简介

RDB是Redis用来进行持久化的一种方式，是把当前内存中的数据集快照写入磁盘，也就是 Snapshot 快照（数据库中所有键值对数据）。恢复时是将快照文件直接读到内存里。

回到顶部

2、触发方式

RDB 有两种触发方式，分别是自动触发和手动触发。

①、自动触发

在 redis.conf 配置文件中的 SNAPSHOTTING 下，在这篇文章中我们介绍过。

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)

```
196 ################################ SNAPSHOTTING  ################################
197 #
198 # Save the DB on disk:
199 #
200 #   save <seconds> <changes>
201 #
202 #   Will save the DB if both the given number of seconds and the gi
203 #   number of write operations against the DB occurred.
204 #
205 #   In the example below the behaviour will be to save:
206 #   after 900 sec (15 min) if at least 1 key changed
207 #   after 300 sec (5 min) if at least 10 keys changed
208 #   after 60 sec if at least 10000 keys changed
209 #
210 #   Note: you can disable saving completely by commenting out all "
211 #
212 #   It is also possible to remove all the previously configured sav
213 #   points by adding a save directive with a single empty string ar
214 #   like in the following example:
215 #
216 #   save ""
217
218 save 900 1
219 save 300 10
220 save 60 10000
221
222 # By default Redis will stop accepting writes if RDB snapshots are
223 # (at least one save point) and the latest background save failed.
224 # This will make the user aware (in a hard way) that data is not pe
225 # on disk properly, otherwise chances are that no one will notice a
226 # disaster will happen.
227 #
228 # If the background saving process will start working again Redis w
229 # automatically allow writes again.
```

①、**save**：这里是用来配置触发 Redis的 RDB 持久化条件，也就是什么时候将内存中的数据保存到硬盘。比如“save m n”。表示m秒内数据集存在n次修改时，自动触发bgsave（这个命令下面会介绍，手动触发RDB持久化的命令）

默认如下配置：

```
save 900 1：表示900 秒内如果至少有 1 个 key 的值变化，则保存
save 300 10：表示300 秒内如果至少有 10 个 key 的值变化，则保存
save 60 10000：表示60 秒内如果至少有 10000 个 key 的值变化，则保存
```

当然如果你只是用Redis的缓存功能，不需要持久化，那么你可以注释掉所有的 save 行来停用保存功能。可以直接一个空字符串来实现停用：save ""

②、**stop-writes-on-bgsave-error**：默认值为yes。当启用了RDB且最后一次后台保存数据失败，Redis是否停止接收数据。这会让用户意识到数据没有正确持久化到磁盘上，否则没有人会注意到灾难（disaster）发生了。如果Redis重启了，那么又可以重新开始接收数据了

③、**rdbcompression**；默认值是yes。对于存储到磁盘中的快照，可以设置是否进行压缩存储。如果是的话，redis会采用LZF算法进行压缩。如果你不想消耗CPU来进行压缩的话，可以设置为关闭此功能，但是存储在磁盘上的快照会比较大。

④、**rdbchecksum**：默认值是yes。在存储快照后，我们还可以让redis使用CRC64算法来进行数据校验，但是这样做会增加大约10%的性能消耗，如果希望获取到最大的性能提升，可以关闭此功能。

⑤、**dbfilename**：设置快照的文件名，默认是 dump.rdb

⑥、**dir**：设置快照文件的存放路径，这个配置项一定是个目录，而不能是文件名。默认是和当前配置文件保存在同一目录。

也就是说通过在配置文件中配置的 save 方式，当实际操作满足该配置形式时就会进行 RDB 持久化，将当前的内存快照保存在 dir 配置的目录中，文件名由配置的 dbfilename 决定。

②、手动触发

手动触发Redis进行RDB持久化的命令有两种：

1、save

该命令会阻塞当前Redis服务器，执行save命令期间，Redis不能处理其他命令，直到RDB过程完成为止。

显然该命令对于内存比较大的实例会造成长时间阻塞，这是致命的缺陷，为了解决此问题，Redis提供了第二种方式。

2、bgsave

执行该命令时，Redis会在后台异步进行快照操作，快照同时还可以响应客户端请求。具体操作是Redis进程执行fork操作创建子进程，RDB持久化过程由子进程负责，完成后自动结束。阻塞只发生在fork阶段，一般时间很短。

基本上 Redis 内部所有的RDB操作都是采用 bgsave 命令。

ps:执行执行 flushall 命令，也会产生dump.rdb文件，但里面是空的，无意义

回到顶部

3、恢复数据

将备份文件 (dump.rdb) 移动到 redis 安装目录并启动服务即可，redis就会自动加载文件数据至内存了。Redis 服务器在载入 RDB 文件期间，会一直处于阻塞状态，直到载入工作完成为止。

获取 redis 的安装目录可以使用 config get dir 命令

```
127.0.0.1:6379> config get dir
1) "dir"
2) "/usr/local/redis/bin"
127.0.0.1:6379>
```

回到顶部

4、停止 RDB 持久化

有些情况下，我们只想利用Redis的缓存功能，并不像使用 Redis 的持久化功能，那么这时候我们最好停掉 RDB 持久化。可以通过上面讲的在配置文件 redis.conf 中，可以注释掉所有的 save 行来停用保存功能或者直接一个空字符串来实现停用：save ""

也可以通过命令：

```
redis-cli config set save " "
```

回到顶部

5、RDB 的优势和劣势

①、优势

1.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和灾难恢复。

2.生成RDB文件的时候，redis主进程会fork()一个子进程来处理所有保存工作，主进程不需要进行任何磁盘IO操作。

3.RDB 在恢复大数据集时的速度比 AOF 的恢复速度要快。

②、劣势

1、RDB方式数据没办法做到实时持久化/秒级持久化。因为bgsave每次运行都要执行fork操作创建子进程，属于重量级操作(内存中的数据被克隆了一份，大致2倍的膨胀性需要考虑)，频繁执行成本过高(影响性能)

2、RDB文件使用特定二进制格式保存，Redis版本演进过程中有多个格式的RDB版本，存在老版本Redis服务无法兼容新版RDB格式的问题(版本不兼容)

3、在一定间隔时间做一次备份，所以如果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)

回到顶部

6、RDB 自动保存的原理

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)

Redis有个服务器状态结构：

```
struct redisService{
    //1、记录保存save条件的数组
    struct saveparam *saveparams;
    //2、修改计数器
    long long dirty;
    //3、上一次执行保存的时间
    time_t lastsave;

}
```

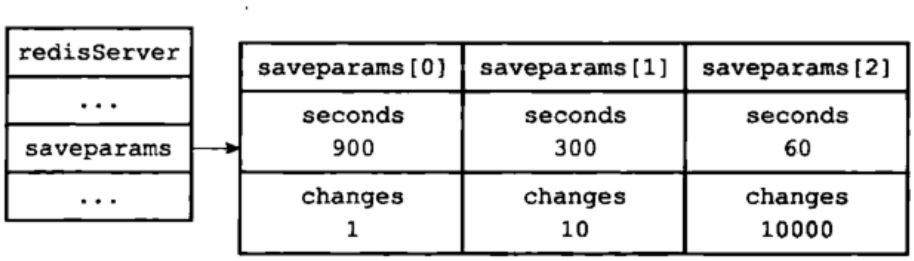
①、首先看记录保存save条件的数组 saveparam，里面每个元素都是一个 saveparams 结构：

```
struct saveparam{
    //秒数
    time_t seconds;
    //修改数
    int changes;
};
```

前面我们在 redis.conf 配置文件中进行了关于save 的配置：

```
save 900 1：表示900 秒内如果至少有 1 个 key 的值变化，则保存
save 300 10：表示300 秒内如果至少有 10 个 key 的值变化，则保存
save 60 10000：表示60 秒内如果至少有 10000 个 key 的值变化，则保存
```

那么服务器状态中的saveparam 数组将会是如下的样子：



②、dirty 计数器和lastsave 属性

dirty 计数器记录距离上一次成功执行 save 命令或者 bgsave 命令之后，Redis服务器进行了多少次修改（包括写入、删除、更新等操作）。

lastsave 属性是一个时间戳，记录上一次成功执行 save 命令或者 bgsave 命令的时间。

通过这两个命令，当服务器成功执行一次修改操作，那么dirty 计数器就会加 1，而lastsave 属性记录上一次执行save或bgsave的时间，Redis 服务器还有一个周期性操作函数 severCron，默认每隔 100 毫秒就会执行一次，该函数会遍历并检查 saveparams 数组中的所有保存条件，只要有一个条件被满足，那么就会执行 bgsave 命令。

执行完成之后，dirty 计数器更新为 0 ，lastsave 也更新为执行命令的完成时间。

分类：Redis详解

标签： Redis详解

3

« 上一篇：Redis详解（五）------ redis的五大数据类型实现原理
» 下一篇：Redis详解（七）------ AOF 持久化

posted @ 2018-06-07 23:29 YSOcean 阅读(7940) 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。