大话缓存之redis

一、缓存

什么叫缓存，在我们日常项目中有很多缓存的方式，比如，jvm的堆内存，redis缓存，MongoDB非结构化数据库等。用内存当缓存，内存显然不是无限大的。内存是宝贵而且有效的IT资源，磁盘反而是廉价大量的。可能一台机器有几十个G的内存，但是可以有几个T的硬盘空间，然鹅，redis是基于内存来进行的高性能、高并发的读写操作。

二、Redis的存储结构

大概了解一下，redis的存储结构，有string（字符串），list（列表），set（set集合），hash（map集合），zset（有序集合）。在项目中如何使用，具体可以看我的另一个文档。

三、过期策略

redis的过期策略有两种：定期删除和惰性删除。

定期删除，就是redis默认的是每隔100ms就随机抽取一些设置了过期时间的key，检查其是否过期，如果过期就删除。假设redis里放了10w个key，都设置了过期时间，你每隔几百毫秒，就检查这10w个key是否过期，无疑是对cpu有较高的负载和消耗，这样基本上你的redis就崩掉。所以，redis是每隔100ms随机抽取一些key来检查和删除的。

但是，这样问题来了，定期删除过期的key，会导致本来很多已经过期的key到了时间并没有被删除，那redis要怎么办呢？

惰性删除来帮忙解决这个问题，当你获取某个key的时候，redis会检查一下，如果这个key设置了过期时间，就要判断这个key是否超过了过期时间，如果超过了此时就会删除该key，并且不会返回任何东西给你。

定期删除+惰性删除是否就能完全没问题了吗？想想看，如果定期删除一直没有删除过期的key，同时你也一直没有查询，走惰性删除，那么就会有很多过期的key堆积在内存中，导致redis内存块耗尽，此时要怎么办呢？

答案是：内存淘汰机制

四、内存淘汰机制

首先，了解一下内存淘汰机制有几种方式：

1. noeviction: 当内存不足以容纳新写入数据时，新写入操作会报错，这个一般没人会用的吧，这个机制着实让人捉鸡。
2. allkeys-lru: 当内存不足以容纳新写入数据时，在键空间中，移除最近最不常用的key。（这个机制最常用）
3. allkey-random: 当内存不足以容纳新写入数据时，在键空间中，随机移除某个key。（这个一般也不会使用的吧，为什么要随机呢，肯定是最不常用的要移除才对）
4. volatile-lru: 当内存不足以容纳新写入数据时，在设置了过期时间的key空间中，移除最近最不常用的key。
5. volatile-random: 当内存不足以容纳新写入数据时，在设置了过期时间的key空间中，随机移除某一个key。
6. volatile-ttl: 当内存不足以容纳新写入数据时，在设置了过期时间的key空间中，有更早的过期时间的key优先移除。

这样在有内存堆积的问题，redis就知道如何处理了，既然LRU最常用，那我们简单的手写一个LRU算法咯。

五、简单的LRU算法

我们不求纯手工从底层开始打造自己的LRU算法，我们至少知道利用已有的JDK数据结构实现一个java的LRU。实现的原理：通过双向链表，新加入的元素放到链表的头，最近使用的元素放到链表的头，在此过程中如果内存出现不足就将链表中的尾部数据进行移除。

```
class LRUCache<K, V> extends LinkedHashMap<K, V> {
    private final int CACHE_SIZE;

    /**
     *传递进来最多能缓存数据的大小
     *@param cacheSize 缓存大小
     */
    public LRUCache(int cacheSize) {
        // true 表示让linkedHashMap按照访问顺序进行排序
        super((int) Math.ceil(cacheSize/0.75) + 1, 0.75f, true);
        CACHE_SIZE = cacheSize;
    }

    @Override
    protected boolean removeEldestEntry(Map.Entry<K, V> eldst) {
        //当map中的数据量大于指定的缓存个数的时候，就会自动删除最老的数据。
        return size() > CACHE_SIZE;
    }
}
```

给一个测试的demo：

```
package com.darryl.sun.LRU;

/**
 * @Auther: Darryl
 * @Description: test for lru cache
 * @Date: created in 2019/7/27 11:40
 */

public class TestLRUCache {

    public static void main(String[] args) {
        LRUCache<String, String> lruCache = new LRUCache<>(4);

        String str1 = "1";
        String str2 = "2";
        String str3 = "3";
        String str4 = "4";
        String str5 = "5";

        lruCache.put("str1", str1);
        lruCache.put("str2", str2);
        lruCache.put("str3", str3);
        lruCache.put("str4", str4);

        System.out.println("LRU CACHE size is " + lruCache.size());
        System.out.println("the fourth of LRU CACHE is " + lruCache.get("str1"));

        lruCache.put("str5", str5);
```

```
        System.out.println("LRU CACHE size is " + lruCache.size());

    }

}
```

六、数据持久化

RDB：保存某个时间点的全量数据的快照。优点：全量快照数据，文件小，恢复快；缺点：无法保存最近一次快照之后的数据，数据量大会有IO性能问题。

AOF：记录所有除了查询以外的变更redis数据的指令，会增量的追加到AOF文件中。优点：能够及时保存怎量数据，数据不易丢失；缺点：文件大，恢复时间长。

两种持久化方式，都可以通过系统fork出来一个子进程，进行同步持久化操作。