Redis 实现限流的三种简单方式

Java后端技术 今天

往期热门文章：

1、9个GVP国产Java开源项目！是真滴哇塞

2、网曝字节跳动将实行“1075”工作制！网友：这不是众所周知的嘛！守则的有多少？

3、阿里领导：手下两个应届生，一个踏实喜欢加班，一个技术强挑活，怎么选？

4、这就是最适合程序员的云笔记？

5、人人都写过的5个Bug！

面对越来越多的高并发场景，限流显示的尤为重要。

当然，限流有许多种实现的方式，Redis具有很强大的功能，我用Redis实践了三种的实现方式，可以较为简单的实现其方式。Redis不仅仅是可以做限流，还可以做数据统计，附近的人等功能，这些可能会后续写到。

第一种：基于Redis的setnx的操作

我们在使用Redis的分布式锁的时候，大家都知道是依靠了setnx的指令，在CAS（Compare and swap）的操作的时候，同时给指定的key设置了过期实践（expire），我们在限流的主要目的就是为了在单位时间内，有且仅有N数量的请求能够访问我的代码程序。所以依靠setnx可以很轻松的做到这方面的功能。

比如我们需要在10秒内限定20个请求，那么我们在setnx的时候可以设置过期时间10，当请求的setnx数量达到20时候即达到了限流效果。代码比较简单就不做展示了。

当然这种做法的弊端是很多的，比如当统计1-10秒的时候，无法统计2-11秒之内，如果需要统计N秒内的M个请求，那么我们的Redis中需要保持N个key等等问题

第二种：基于Redis的数据结构zset

其实限流涉及的最主要的就是滑动窗口，上面也提到1-10怎么变成2-11。其实也就是起始值和末端值都各+1即可。

而我们如果用Redis的list数据结构可以轻而易举的实现该功能

我们可以将请求打造成一个zset数组，当每一次请求进来的时候，value保持唯一，可以用UUID生成，而score可以用当前时间戳表示，因为score我们可以用来计算当前时间戳之内有多少的请求数量。而zset数据结构也提供了range方法让我们可以很轻易的获取到2个时间戳内有多少请求

代码如下

```
public Response limitFlow(){
        Long currentTime = new Date().getTime();
        System.out.println(currentTime);
        if(redisTemplate.hasKey("limit")) {
            Integer count = redisTemplate.opsForZSet().rangeByScore("limit", currentTime -  intervalTime, currentTi
            System.out.println(count);
            if (count != null && count > 5) {
                return Response.ok("每分钟最多只能访问5次");
            }
        }
        redisTemplate.opsForZSet().add("limit",UUID.randomUUID().toString(),currentTime);
        return Response.ok("访问成功");
    }
```

通过上述代码可以做到滑动窗口的效果，并且能保证每N秒内至多M个请求，缺点就是zset的数据结构会越来越大。实现方式相对也是比较简单的。

第三种：基于Redis的令牌桶算法

提到限流就不得不提到令牌桶算法了。

令牌桶算法提及到输入速率和输出速率，当输出速率大于输入速率，那么就是超出流量限制了。

也就是说我们每访问一次请求的时候，可以从Redis中获取一个令牌，如果拿到令牌了，那就说明没超出限制，而如果拿不到，则结果相反。

依靠上述的思想，我们可以结合Redis的List数据结构很轻易的做到这样的代码，只是简单实现

依靠List的leftPop来获取令牌

```
//输出令牌
public Response limitFlow2(Long id){
        Object result = redisTemplate.opsForList().leftPop("limit_list");
        if(result == null){
            return Response.ok("当前令牌桶中无令牌");
        }
        return Response.ok(articleDescription2);
    }
```

再依靠Java的定时任务，定时往List中rightPush令牌，当然令牌也需要唯一性，所以我这里还是用UUID进行了生成

```
// 10S的速率往令牌桶中添加UUID，只为保证唯一性
    @Scheduled(fixedDelay = 10_000,initialDelay = 0)
    public void setIntervalTimeTask(){
        redisTemplate.opsForList().rightPush("limit_list",UUID.randomUUID().toString());
    }
```

综上，代码实现起始都不是很难，针对这些限流方式我们可以在AOP或者filter中加入以上代码，用来做到接口的限流，最终保护你的网站。

Redis其实还有很多其他的用处，他的作用不仅仅是缓存，分布式锁的作用。他的数据结构也不仅仅是只有String，Hash，List，Set，Zset。有兴趣的可以后续了解下他的GeoHash算法；BitMap，HLL以及布隆过滤器数据(Redis4.0之后加入，可以用Docker直接安装redislabs/rebloom)结构。

往期热门文章：

1、《历史文章分类导读列表！精选优秀博文都在这里了！》

2、Spring Boot + GraphQL 才是 API 的未来！

3、全员远程办公，半年入 1 亿美元：GitHub 的最大竞争对手上市了！

4、聊一聊Java 泛型通配符 T，E，K，V，？

5、各大公司程序员的工位，你中意哪一款？

6、Stackoverflow 高赞答案，为什么牛逼的程序员都不用“！= null ' 做判空？

7、为什么不建议在MySQL中使用 utf8 ？

8、编写Spring MVC控制器的14个技巧！涨知识了！

9、日志打印的15个建议！血泪啊！

10、List中remove()方法的陷阱，被坑惨了！