.NetCore+Jexus代理+Redis模拟秒杀商品活动

2017-07-05 神牛步行3 [dotNET跨平台](https://mp.weixin.qq.com/s?__biz=MzAwNTMxMzg1MA==&mid=2654068835&idx=1&sn=d66fd63d9cd20b2a5d73adec028c3d02&chksm=80dbc436b7ac4d204976acf8ca124acb16bd943bd260d3639eb54cf21527ec56ae4d26e01a0c&mpshare=1&scene=1&srcid=0705wuDz3p8eTsspVzD4OARe&key=0ebc3a09d7ac5a1d5f21366ca3289138f233e169f688ab3af65b2913a2cd12ef33e83973b13b114ab5e849f53fd005eea9c29c7cd9a47310904921b645b78b8c7a14e5b59e829b09eaddd6cf3377741a&ascene=1&uin=Mjg1Mzg5MzU%3D&devicetype=Windows+UnKnow&version=62040525&pass_ticket=Mq1890O9oNEOTT476IbpY5m3EHYRQR0JPGQ4sN%2BNGVk%3D&winzoom=1##)

**开篇叙**

 ，顺手点个推荐也不错；

a. 秒杀流程

b. 封装StackExchange.Redis的使用类

c. Ubuntu16.04上使用Jexus搭建代理完成分布式部署

d. NetCore写实时监控队列服务

**秒杀架构设计图︿(￣︶￣)︿三幅**

**1. 一般业务性架构**

**2. 后端分布式架构**

**3. 整站分布式**

**项目工程结构描述**

a. 该项目git开源地址： https://github.com/shenniubuxing3/SeckillPro ，线上效果地址： http://www.lovexins.com:3333/

b. SeckillPro.Web：面向用户的web站点，主要提供商品展示，秒杀抢购，抢购结果，订单列表等功能；

c. SeckillPro.Api：主要处理秒杀活动的请求，然后加入到秒杀队列中，以及订单状态的查询接口；

d. SeckillPro.Server：处理秒杀队列的服务；根据Redis模糊匹配key的方式，开启多个商品秒杀的任务，并处理秒杀请求和改变订单抢购状态；

e. SeckillPro.Com：集成公共的方法；这里面前有操作Redis的list，hash，string的封装类；

**SeckillPro.Web商品后台管理**

对于商品活动来说，商品维护是必不可少的，由于这里商品维护的信息比较少，并且这里只加入到了RedisDb中，所以就不直接上代码了；一个列表，一个添加仅此而已；这里就不再贴代码了，如果你感兴趣可以去我的git上面看源码： https://github.com/shenniubuxing3/SeckillPro/blob/master/SeckillPro/SeckillPro.Web/Controllers/HomeController.cs

**SeckillPro.Web用户端商品列表+秒杀请求+用户订单列表**

商品列表和订单列表没有可以太多说的，一般订单系统都有这两个列表；关键点在于订单秒杀流程中，咋们来简单分析下面向客户秒杀的流程需要注意的事项：

a. 限制秒杀开始时间和结束时间(测试未限制)

b. 未开始活动限制提交按钮不可点(测试未限制)

c. 获取真实剩余库存限制秒杀提交(获取redis中商品hash存储的真实剩余量)

d. 把客户的秒杀请求转移到另外的api集群，以此提高面向客户端的web站点并发承载率(测试项目中我直接指定4545端口的api测试)

这里就不再贴代码了，如果你感兴趣可以去我的git上面看看这部分源码： https://github.com/shenniubuxing3/SeckillPro/blob/master/SeckillPro/SeckillPro.Web/Controllers/HomeController.cs

**.NetCore写处理秒杀活动队列的服务**

这个处理队列服务处理流程：模糊匹配Redis中每种商品的队列key-》开启不同商品的处理队列任务-》处理秒杀订单-》更新库存和秒杀订单状态；

a. 模糊匹配Redis中每种商品的队列key：这里采用的是StackExchange.Redis中指定redis原生命令的方法来获取匹配队列key，设计的代码如下：

/// <summary>

        /// 模糊匹配redis中的key

        /// </summary>

        /// <param name="paramArr"></param>

        /// <returns></returns>

        public async Task<List<string>> MatchKeys(params string[] paramArr)

        {

            var list = new List<string>();

            try

            {

                var result = await this.ExecuteAsync("keys", paramArr);

                var valArr = ((RedisValue[])result);

                foreach (var item in valArr)

                {

                    list.Add(item);

                }

            }

            catch (Exception ex) { }

            return list;

        }

        /// <summary>

        /// 执行redis原生命令

        /// </summary>

        /// <param name="cmd"></param>

        /// <param name="paramArr"></param>

        /// <returns></returns>

        public async Task<RedisResult> ExecuteAsync(string cmd, params string[] paramArr)

        {

            try

            {

                var db = this.GetDb();

                return await db.ExecuteAsync(cmd, paramArr);

            }

            catch (Exception ex) { }

            return default(RedisResult);

        }

b. 开启不同商品的处理队列任务：通过Task.Factory.StartNew(action,object)方法开启不同商品的处理秒杀订单的任务；

c. 更新库存和秒杀订单状态：由于抢购商品要求库存剩余实时性，所以每处理一个抢购订单，需要对该商品减去相应的库存和修改秒杀订单的状态方便用户查看秒杀结果；

d. 处理队列具体的实现代码可以去**git**看下，个人觉得还是有用的：https://github.com/shenniubuxing3/SeckillPro/blob/master/SeckillPro/SeckillPro.Server/Program.cs

**使用Jexus代理部署分布式站点和接口**

这里部署的代理采用的是Jexus代理；作为在linux和unix上部署.net程序实用的工具，真的很感谢jexus作者；首先本篇讲解的部署环境是ubunt16.04x64（至于这么安装jexus可以参考上一篇分享文章），为了更直观的看出来效果我在服务器上拷贝了两份SeckillPro.Web发布的站点，他们代码都是一样的只是分别把\_Layout.cshtml试图模板中加入了端口7777和8888，我就用这两个端口来测试jexus的代理效果；

测试方便直接分别在两个复制站点中执行如下终端命令：dotnet SeckillPro.Web.dll http://ip:端口 ；一个监听7777端口一个监听8888；执行命令效果图：

监听7777和8888端口成功后，我们就可以直接在浏览器输入：http://172.16.9.66:7777 访问，正常情况下能够看到如下图示例：

单个站点访问没问题了，下面开始配置jexus代理；只需要在jexus/siteconf的配置文件中（我这里是default配置文件），增加如下设置：

注意reproxy参数：

a. 第一个/表示根目录，一般不变

b. 多个被代理地址使用‘,’隔开；

c. 被代理地址后面也同样需要加/

此时我们配置完后，只需要启动jexus就行了：./jws start （怎么启动可以参考上一篇文章）；当启动jws成功后，我们就能通过配置的80端口，来访问SeckillPro.Web站点了，效果图：

至于代理分发的策略暂不在本章的讨论范围内，如果可以建议去jexus官网了解下；同样对于Seckill.Api我们也可以这样部署，这里部署了个秒杀线上地址，有兴趣的朋友可以点击试试：http://www.lovexins.com:3333/ （注：这里没有使用代理）

**封装StackExchange.Redis的使用类StackRedis.cs**

其实这个在之前已经分享过了，只不过只有操作string和list的分装；本篇测试涉及到订单查询和商品查询等功能，所以这里我又扩展了对hash的操作方法，可以说更丰富了吧，如果您正打算使用redis或许直接用我这个封装类是个不错的打算；

相关文章：

* [在CentOS上使用Jexus托管运行 ZKEACMS](http://mp.weixin.qq.com/s?__biz=MzAwNTMxMzg1MA==&mid=2654068725&idx=1&sn=84a4267f201c74de0bfe3b60838c7f56&chksm=80dbfba0b7ac72b619d19cf3872bec2d2c92df61ee11f4e396e024299183ff2316357fd8af37&scene=21#wechat_redirect)
* [[.NET跨平台]Jexus独立版本的便利与过程中的一些坑](http://mp.weixin.qq.com/s?__biz=MzAwNTMxMzg1MA==&mid=2654068761&idx=1&sn=bff36f49df09f88bb4360f63db6085b8&chksm=80dbc44cb7ac4d5a763c94ec6b71cfb69127787e32e28889fa8c15edd7eefde92cf0663df1f7&scene=21#wechat_redirect)
* [来腾讯云开发者实验室 学习.NET](http://mp.weixin.qq.com/s?__biz=MzAwNTMxMzg1MA==&mid=2654068793&idx=1&sn=32aaa15a3531fa4d0d4b9f054289d252&chksm=80dbc46cb7ac4d7a901f705b47a384a133da97ff311d997fbfd43d554f07d0093dff60020d1e&scene=21#wechat_redirect)
* [Jexus针对Asp.net core应用程序的六大不可替代的优势](http://mp.weixin.qq.com/s?__biz=MzAwNTMxMzg1MA==&mid=2654068331&idx=1&sn=1ca1dadd5021eb09679ca52c6ca1c0bc&chksm=80dbfa3eb7ac7328be537df1b297a51104373cc2c6dfd86d98b0a48f5a17dc26eeb429f383ca&scene=21#wechat_redirect)
* [Jexus部署.Net Core项目](http://mp.weixin.qq.com/s?__biz=MzAwNTMxMzg1MA==&mid=2654068806&idx=1&sn=853b3768ef2739ecdea3010ad6cb2648&chksm=80dbc413b7ac4d058b7346693091c626be04d1e5072f4fd0a8de1dad90fdedbbaf23c75a6be6&scene=21#wechat_redirect)
* [Linux使用Jexus托管Asp.Net Core应用程序](http://mp.weixin.qq.com/s?__biz=MzAwNTMxMzg1MA==&mid=2654068704&idx=3&sn=b1c00fece1ab8fa3559b1cc9d2e29b98&chksm=80dbfbb5b7ac72a32b429f8fc939a263e7f2e179e69d89b158dd376e83af127669814a9a6493&scene=21#wechat_redirect)
* [[.NET跨平台]Jexus独立版本的便利与过程中的一些坑](http://mp.weixin.qq.com/s?__biz=MzAwNTMxMzg1MA==&mid=2654068761&idx=1&sn=bff36f49df09f88bb4360f63db6085b8&chksm=80dbc44cb7ac4d5a763c94ec6b71cfb69127787e32e28889fa8c15edd7eefde92cf0663df1f7&scene=21#wechat_redirect)

原文地址：http://www.cnblogs.com/wangrudong003/p/7111789.html