**品优购电商系统开发**

**第20章**

**Nginx负载均衡&MyCat分库分表**

传智播客.黑马程序员

# 课程目标

目标1：运用SpringTask实现任务调度

目标2：了解品优购的整体部署方案

目标3：能配置Nginx的反向代理与负载均衡

目标4：实现配置MyCat分片

# SpringTask任务调度

## 1.1什么是任务调度

在企业级应用中，经常会制定一些“计划任务”，即在某个时间点做某件事情，核心是以时间为关注点，即在一个特定的时间点，系统执行指定的一个操作。常见的任务调度框架有Quartz和SpringTask等。

## 1.2 Cron时间表达式

### 1.2.1 Cron表达式格式

Cron表达式是一个字符串，字符串分6个域，每一个域代表一个含义，Cron语法格式：

* Seconds Minutes Hours DayofMonth Month DayofWeek

【秒】 【分】 【小时】 【日】 【月】 【周】

每一个域可出现的字符如下：

* Seconds： 可出现", - \* /"四个字符，有效范围为0-59的整数
* Minutes： 可出现", - \* /"四个字符，有效范围为0-59的整数
* Hours： 可出现", - \* /"四个字符，有效范围为0-23的整数
* DayofMonth： 可出现", - \* / ? L W C"八个字符，有效范围为1-31的整数
* Month： 可出现", - \* /"四个字符，有效范围为1-12的整数或JAN-DEC
* DayofWeek： 可出现", - \* / ? L C #"八个字符，有效范围为1-7的整数或SUN-SAT两个范围。1表示星期天，2表示星期一， 依次类推

每一个域都使用数字，但还可以出现如下特殊字符，它们的含义是：

* \*：表示匹配该域的任意值，假如在Minutes域使用\*, 即表示每分钟都会触发事件。
* ?：只能用在DayofMonth(日)和DayofWeek(周)两个域。它也匹配域的任意值
* -：表示范围，例如在Minutes域使用5-20，表示从5分到20分钟每分钟触发一次
* /： 表示起始时间开始触发，然后每隔固定时间触发一次，例如在Minutes域使用5/20,则意味着5分钟触发一次，而25，45等分别触发一次。
* ,：表示列出枚举值。例如：在Minutes域使用5,20，则意味着在5和20分每分钟触发一次。
* L: 表示最后，只能出现在DayofWeek(周)和DayofMonth(日)，如果在DayofWeek域使用5L,意味着在最后的一个星期四触发。
* W: 表示有效工作日(周一到周五),只能出现在DayofMonth(日)，系统将在离指定日期的最近的有效工作日触发事件。例如：在 DayofMonth使用5W，如果5日是星期六，则将在最近的工作日：星期五，即4日触发。如果5日是星期天，则在6日(周一)触发；如果5日在星期一 到星期五中的一天，则就在5日触发。另外一点，W的最近寻找不会跨过月份
* LW: 这两个字符可以连用，表示在某个月最后一个工作日，即最后一个星期五。
* #: 用于确定每个月第几个星期几，只能出现在DayofWeek(周)。例如在4#2，表示某月的第二个星期三。

### 1.2.2 Cron表达式例子

"0 0 10,14,16 \* \* ?"： 每天上午10点，下午2点，4点

"0 0/30 9-17 \* \* ?"： 朝九晚五工作时间内每半小时

"0 0 12 ? \* WED"：表示每个星期三中午12点

"0 0 12 \* \* ?"：每天中午12点触发

"0 15 10 ? \* \*"：每天上午10:15触发

"0 15 10 \* \* ?"：每天上午10:15触发

"0 \* 14 \* \* ?"：在每天下午2点到下午2:59期间的每1分钟触发

"0 0/5 14 \* \* ?"： 在每天下午2点到下午2:55期间的每5分钟触发

"0 0/5 14,18 \* \* ?"： 在每天下午2点到2:55期间和下午6点到6:55期间的每5分钟触发

"0 0-5 14 \* \* ?"： 在每天下午2点到下午2:05期间的每1分钟触发

"0 10,44 14 ? 3 WED"： 每年三月的星期三的下午2:10和2:44触发

"0 15 10 ? \* MON-FRI"： 周一至周五的上午10:15触发

"0 15 10 15 \* ?"： 每月15日上午10:15触发

"0 15 10 L \* ?"： 每月最后一日的上午10:15触发

"0 15 10 ? \* 6L"： 每月的最后一个星期五上午10:15触发

"0 15 10 ? \* 6#3"： 每月的第三个星期五上午10:15触发

# **2.秒杀订单超时处理**

**需求：当用户下单后5分钟尚未付款应该释放订单，增加库存。**

## 2.1集成SpringTask

### 2.1.1配置文件

pinyougou-seckill-web/src/main/resources/springmvc.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="utf-8"***?>* <**beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xmlns:task="http://www.springframework.org/schema/task"  xmlns:dubbo="http://code.alibabatech.com/schema/dubbo"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc.xsd  http://www.springframework.org/schema/task  http://www.springframework.org/schema/task/spring-task.xsd  http://code.alibabatech.com/schema/dubbo  http://code.alibabatech.com/schema/dubbo/dubbo.xsd"**>  *<!-- 开启任务注解驱动 -->* <**task:annotation-driven**/>  *<!-- 配置任务调度方式 -->* <**task:scheduler id="scheduler"**/>  ***<!-- 开启组件扫描 -->* <context:component-scan base-package="com.pinyougou.seckill"/>** *<!-- 开启MVC注解驱动 --> ......*  *<!-- ############## 配置dubbo服务消费者 ############## -->  <!-- 配置当前应用的名称 -->* <**dubbo:application name="pinyougou-seckill-web"**/>  *<!-- 配置zookeeper作为注册中心，发现服务地址 -->* <**dubbo:registry address="zookeeper://192.168.12.131:2181"**/>  *<!-- 配置采用包扫描来引用服务，产生服务接口的代理对象 -->* <**dubbo:annotation package="com.pinyougou.seckill"**/> </**beans**> |

### 2.1.2任务服务类

pinyougou-seckill-web\src\main\java\com\pinyougou\seckill\task\包下，创建SeckillOrderTask.java秒杀订单定时任务类

|  |
| --- |
| */\*\* 秒杀订单任务 \*/* @Component **public class** SeckillOrderTask {  @Reference(timeout = 10000)  **private** SeckillOrderService **seckillOrderService**;  @Reference(timeout = 10000)  **private** WeixinPayService **weixinPayService**;  */\*\* 定时关闭超时未支付定单(每隔3秒调度一次) \*/* @Scheduled(cron = **"0/3 \* \* \* \* ?"**)  **public void** closeOrderTask(){  System.***out***.println(**"==毫秒=="** + System.*currentTimeMillis*());  *// 查询所有超时未支付的订单* List<SeckillOrder> seckillOrderList = **seckillOrderService** .findOrderByTimeout();  *// 迭代未支付的订单* **for** (SeckillOrder seckillOrder : seckillOrderList){  *// 关闭微信未支付订单* Map<String,String> map = **weixinPayService** .closePayTimeout(seckillOrder.getId().toString());  *// 如果正常关闭* **if** (**"SUCCESS"**.equals(map.get(**"result\_code"**))){  System.***out***.println(**"===超时，删除订单==="**);  *// 删除超时未支付的订单* **seckillOrderService**.deleteOrderFromRedis(seckillOrder);  }  }  } } |

## 2.2查询超时未支付订单

### 2.2.1服务接口层

pinyougou-interface\src\main\java\com\pinyougou\service\SeckillOrderService.java：

|  |
| --- |
| */\*\* 查询超时未支付订单( \*/* List<SeckillOrder> findOrderByTimeout(); |

### 2.2.2服务实现层

pinyougou-seckill-service\src\main\java\com\pinyougou\seckill\service\impl\SeckillOrderServiceImpl.java：

|  |
| --- |
| */\*\* 查询超时未支付订单( \*/* **public** List<SeckillOrder> findOrderByTimeout(){  **try**{  *// 定义List集合封装超时5分钟未支付的订单* List<SeckillOrder> seckillOrders = **new** ArrayList<>();  *// 查询Redis中所有未支付的订单* List<Object> seckillOrderList = **redisTemplate**.  boundHashOps(**"seckillOrderList"**).values();  *// 判断集合* **if** (seckillOrderList != **null** && seckillOrderList.size() > 0){  *// 迭代所有未支付的订单* **for** (Object obj : seckillOrderList){  SeckillOrder seckillOrder = (SeckillOrder)obj;  *// 当前系统毫秒数 - 5分钟* **long** endTime = **new** Date().getTime() - (5 \* 60 \* 1000);  *// 判断创建订单时间是否超出5分钟* **if** (seckillOrder.getCreateTime().getTime() < endTime){  *// 把超时的订单添加到集合* seckillOrders.add(seckillOrder);  }  }  }  **return** seckillOrders;  }**catch** (Exception ex){  **throw new** RuntimeException(ex);  } } |

## 2.3微信关闭订单接口

参考文档:<https://pay.weixin.qq.com/wiki/doc/api/index.html>

### 2.3.1服务接口层

pinyougou-interface\src\main\java\com\pinyougou\service\WeixinPayService.java：

|  |
| --- |
| */\*\* 关闭超时未支付订单 \*/* Map<String, String> closePayTimeout(String outTradeNo); |

### 2.3.2服务实现层

pinyougou-pay-service\src\main\java\com\pinyougou\pay\service\impl\WeixinPayServiceImpl.java：

|  |
| --- |
| */\*\* 关闭订单请求地址 \*/* @Value(**"${closeorder}"**) **private** String **closeorder**;  */\*\* 关闭超时未支付订单 \*/* **public** Map<String, String> closePayTimeout(String outTradeNo){  */\*\* 创建Map集合封装请求参数 \*/* Map<String, String> params = **new** HashMap<>();  */\*\* 公众账号 \*/* params.put(**"appid"**, **appid**);  */\*\* 商户账号 \*/* params.put(**"mch\_id"**, **partner**);  */\*\* 订单交易号 \*/* params.put(**"out\_trade\_no"**, outTradeNo);  */\*\* 随机字符串 \*/* params.put(**"nonce\_str"**, WXPayUtil.*generateNonceStr*());  **try** {  */\*\* 生成签名的xml参数 \*/* String xmlParam = WXPayUtil.*generateSignedXml*(params, **partnerkey**);  System.***out***.println(**"请求参数："** + xmlParam);  */\*\* 创建HttpClientUtils对象 \*/* HttpClientUtils client = **new** HttpClientUtils(**true**);  */\*\* 发送post请求，得到响应数据 \*/* String result = client.sendPost(**closeorder**, xmlParam);  System.***out***.println(**"响应数据："** + result);  */\*\* 将xml响应数据转化成Map \*/* **return** WXPayUtil.*xmlToMap*(result);  } **catch** (Exception ex) {  **throw new** RuntimeException(ex);  } } |

pinyougou-pay-service\src\main\resources\props\weixinpay.properties

|  |
| --- |
| *# 关闭订单请求地址：* **closeorder**=**https://api.mch.weixin.qq.com/pay/closeorder** |

## 2.4删除超时未支付订单

### 2.4.1服务接口层

pinyougou-interface\src\main\java\com\pinyougou\service\SeckillOrderService.java

|  |
| --- |
| */\*\* 从Redis中删除超时未支付订单 \*/* **void** deleteOrderFromRedis(SeckillOrder seckillOrder); |

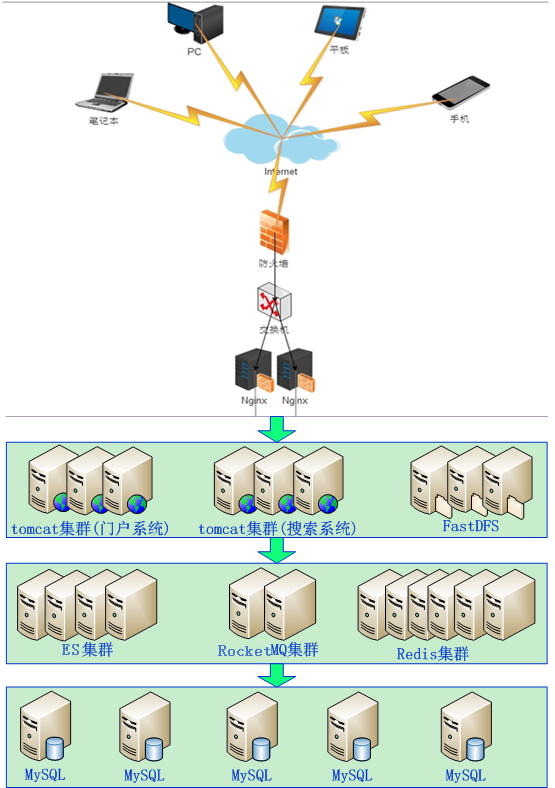
### 2.4.2服务实现层

pinyougou-seckill-service\src\main\java\com\pinyougou\seckill\service\impl\SeckillOrderServiceImpl.java

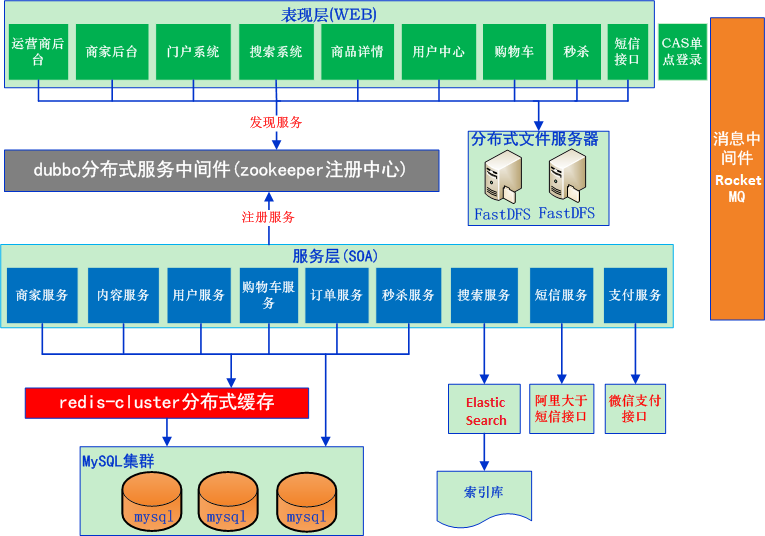
|  |
| --- |
| */\*\* 从Redis中删除超时未支付订单 \*/* **public void** deleteOrderFromRedis(SeckillOrder seckillOrder){  **try**{  *// 删除Redis缓存中的订单* **redisTemplate**.boundHashOps(**"seckillOrderList"**)  .delete(seckillOrder.getUserId());  */\*\* ######## 恢复库存数量 #######\*/  // 从Redis查询秒杀商品* SeckillGoods seckillGoods = (SeckillGoods) **redisTemplate** .boundHashOps(**"seckillGoodsList"**)  .get(seckillOrder.getSeckillId());  *// 判断缓存中是否存在该商品* **if** (seckillGoods != **null**){  *// 修改缓存中秒杀商品的库存* seckillGoods.setStockCount(seckillGoods.getStockCount() + 1);  }**else**{ *// 代表秒光  // 从数据库查询该商品* seckillGoods = **seckillGoodsMapper**.  selectByPrimaryKey(seckillOrder.getSeckillId());  *// 设置秒杀商品库存数量* seckillGoods.setStockCount(1);  }  *// 存入缓存* **redisTemplate**.boundHashOps(**"seckillGoodsList"**)  .put(seckillOrder.getSeckillId(), seckillGoods);  }**catch** (Exception ex){  **throw new** RuntimeException(ex);  } } |

# **3.品优购的整体部署方案**

## 3.1网络拓扑图



## 3.2品优购架构图



## 3.3系统部署

|  |  |
| --- | --- |
| **项目** | **服务器数量** |
| pinyougou-sellergoods-service | 2 |
| pinyougou-manager-web | 2 |
| pinyougou-shop-web | 2 |
| pinyougou-content-service | 2 |
| pinyougou-portal-web | 4 |
| pinyougou-search-service | 2 |
| pinyougou-search-web | 2 |
| pinyougou-item-web | 2 |
| pinyougou-sms-service | 2 |
| pinyougou-sms-web | 2 |
| pinyougou-user-service | 2 |
| pinyougou-user-web | 2 |
| pinyougou-cart-service | 2 |
| pinyougou-cart-web | 2 |
| pinyougou-pay-service | 2 |
| pinyougou-order-service | 2 |
| pinyougou-seckill-service | 2 |
| pinyougou-seckill-web | 2 |
| cas单点登录系统 | 2 |
| 上述小计需要服务器40台 |  |
|  |  |
| MySql | 2 |
| ElasticSearch | 4 |
| Redis | 6 |
| 图片服务器 | 6 |
| Nginx | 2 |
| dubbo注册中心 | 3 |
| RocketMQ | 2 |
| 上述小计需要服务器25台 |  |
| 合计 | 共需要65台应用服务器 |

**域名**：申请一个一级域名www.pinyougou.com、pinyougou.com即可

|  |  |
| --- | --- |
| **项目名称** | **域名** |
| pinyougou-manager-web | manager.pinyougou.com |
| pinyougou-shop-web | shop.pinyougou.com |
| pinyougou-portal-web | [www.pinyougou.com](http://www.pinyougou.com) pinyougou.com |
| pinyougou-search-web | search.pinyougou.com |
| pinyougou-item-web | item.pinyougou.com |
| pinyougou-sms-web | sms.pinyougou.com |
| pinyougou-user-web | user.pinyougou.com |
| CAS单点登录系统 | sso.pinyougou.com |
| pinyougou-cart-web | cart.pinyougou.com |
| pinyougou-seckill-web | seckill.pinyougou.com |

# **4.项目集群&负载均衡**

## 4.1什么是集群

### 4.1.1集群概念

集群是一种计算机系统， 它通过一组松散集成的计算机软件和/或硬件连接起来高度紧密地协作完成计算工作。在某种意义上，他们可以被看作是一台计算机。集群系统中的单个计算机通常称为节点，通常通过局域网连接，但也有其它的可能连接方式。集群计算机通常用来改进单个计算机的计算速度和可靠性。一般情况下集群计算机比单个计算机，比如工作站或超级计算机性能价格比要高得多。



### 4.1.2集群的特点

**可扩展性**：集群的性能不限制于单一的服务实体，新的服务实体可以动态的添加到集群，从而增强集群的性能。

**高可用性**：集群当其中一个节点发生故障时，这台节点上面所运行的应用程序将在另一台节点被自动接管，消除单点故障对于增强数据可用性、可达性和可靠性是非常重要的。

### 4.1.3集群的两大能力

**负载均衡**：负载均衡把任务比较均匀的分布到集群环境下的计算和网络资源，以提高数据吞吐量。

**错误恢复**：如果集群中的某一台服务器由于故障或者维护需要无法使用，资源和应用程序将转移到可用的集群节点上。这种由于某个节点的资源不能工作，另一个可用节点中的资源能够透明的接管并继续完成任务的过程，叫做错误恢复。

负载均衡和错误恢复要求各服务实体中有执行同一任务的资源存在，而且对于同一任务的各个资源来说，执行任务所需的信息视图必须是相同的。

## 4.2集群与分布式的区别

说到集群，可能大家会立刻联想到另一个和它很相近的一个词----“分布式”。那么集群和分布式是一回事吗？有什么联系和区别呢?

相同点：

分布式和集群都是需要有很多节点服务器通过网络协同工作完成整体的任务目标。

不同点：

分布式是指将业务系统进行拆分，即分布式的每一个节点都是实现不同的功能。而集群每个节点做的是同一件事情。

如下图，每个人都有不同的分工，一起协作干一件事，叫做“分布式”



再看下图：每个划桨人干的都是一样的活，叫做集群。



分布式的每一个节点也可以做成集群。其实这个赛龙舟的图，总整体来看属于分布式,包括打鼓和划桨两个分布式节点，而划桨的节点又是集群的形态。

现实生活中例子还有很多，例如，这样古代乐队的图就属于集群：



而现代乐队这样图就是分布式：

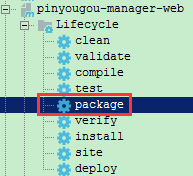


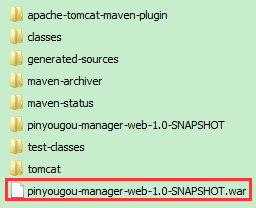
## 4.3安装Nginx

参考【资料\nginx】：Linux安装Nginx.docx

## 4.4部署运营商后台集群

### 4.4.1准备运营商系统war包





### 4.4.2部署2台运营商系统

第一台tomcat：

|  |
| --- |
| mkdir /usr/local/web && cd /usr/local/web  rz 上传apache-tomcat-8.5.28.tar.gz  # 解压 tomcat  tar -xzvf apache-tomcat-8.5.28.tar.gz  # 重命名  mv apache-tomcat-8.5.28 tomcat-pyg-manager-web1  cd tomcat-pyg-manager-web1/webapps/  # 删除所有内容  rm -rf \*  # 创建ROOT目录  mkdir ROOT && cd ROOT  rz 上传pinyougou-manager-web-1.0-SNAPSHOT.war包  # 解压pinyougou-manager-web-1.0-SNAPSHOT.war，访问系统不需要带路径  jar -xvf pinyougou-manager-web-1.0-SNAPSHOT.war  # 删除pinyougou-manager-web-1.0-SNAPSHOT.war  rm -rf \*.war |

第二台tomcat：

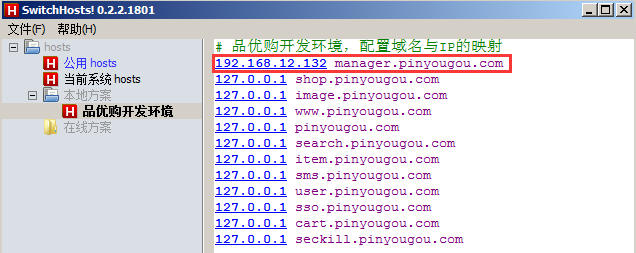
|  |
| --- |
| # 复制第一台服务器的所有内容  cd /usr/local/web  cp -r tomcat-pyg-manager-web1/ tomcat-pyg-manager-web2  # 修改第2台tomcat配置的conf/server.xml中的三个端口：  # server port 8006 (22行)  # http port 8081 (69行)  # connector port 8010 (116行)  vi tomcat-pyg-manager-web2/conf/server.xml |

启动：

【注意】如果pinyougou-sellergoods-service与pinyougou-content-service服务是在windows上启动的，需要将**windows**的对专用和对外的**防火墙关闭**。

|  |
| --- |
| # 启动tomcat  cd /usr/local/web  tomcat-pyg-manager-web1/bin/startup.sh  tomcat-pyg-manager-web2/bin/startup.sh  **#注意要在windows中启动pinyougou-sellergoods与pinyougou-content服务和开启注册中心** |

修改hosts:



测试：

[http://manager.pinyougou.com:8080/](http://192.168.12.169:8080/rest/page/index)

[http://manager.pinyougou.com:8081/](http://192.168.12.169:8081/rest/page/index)

### 4.4.3配置运营商系统nginx集群

修改nginx的nginx.conf：

|  |
| --- |
| cd /usr/local/nginx/conf  vi nginx.conf  # 在http节点内添加：  # 配置集群节点  upstream pinyougou-manager-web {  ip\_hash;  server 192.168.12.132:8080;  server 192.168.12.132:8081;  }  server {  listen 80;  server\_name manager.pinyougou.com;  proxy\_set\_header X-Forwarded-Host $host;  proxy\_set\_header X-Forwarded-Server $host;  proxy\_set\_header Host $host;  location / {  proxy\_pass http://pinyougou-manager-web;  proxy\_connect\_timeout 600;  proxy\_read\_timeout 600;  }  } |

重启nginx：

|  |
| --- |
| cd ../sbin  ./nginx -s reload |

测试：可以在132上关闭对应的两台tomcat来验证系统的集群。只要还有一台服务器在运行系统就是可使用的；除非两台服务器全部关闭才不能访问系统。

Nginx负载均衡调度算法：

**1、轮询(默认)**

每个请求按时间顺序逐一分配到不同的服务器，如果当前服务器down掉，则会跳转到下一台服务器。

upstream pinyougou-manager-web {

server 192.168.12.132:8080;

server 192.168.12.132:8081;

}

1. **weight(权重)**

指定在轮询的基础上加上权重，weight和访问比率成正比，即用于表明服务器的性能好坏，若服务器性能较好则可将大部分请求分配给它，已实现其力所能及。

upstream pinyougou-manager-web {

server 192.168.12.132:8080 weight=2;

server 192.168.12.132:8081 weight=1;

}

**3、ip\_hash(ip哈希值)**

每个请求按访问ip的hash结果分配，当新的请求到达时，先将其客户端IP通过哈希算法进行哈希出一个值，在随后的请求客户端IP的哈希值只要相同，就会被分配至同一个后端服务器，该调度算法可以解决session的问题。

upstream pinyougou-manager-web {

**ip\_hash;**

server 192.168.12.132:8080;

server 192.168.12.132:8081;

}

# **5.MyCat分库分表**

参考【**资料\mycat\Mycat课程讲义.docx**】