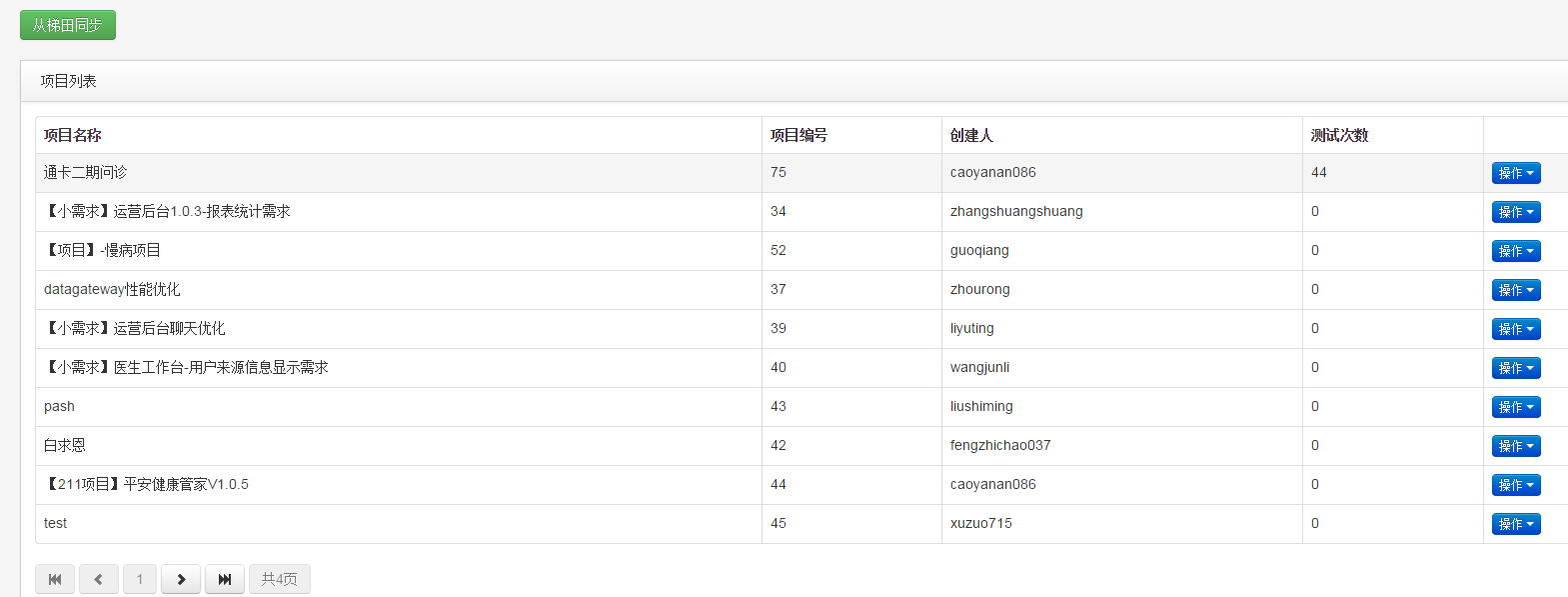
# 1 mozart功能介绍

## 1.1 项目列表

项目列表位于“项目管理”菜单下面，目前主要是从龙猫同步，不支持自定义添加。发现找不到对应的项目，请联系路遥。



## 1.2 脚本

添加脚本中脚本名称支持中英文，但是工程名称和class name必须要根据java工程的名称和包结构填写。



添加好的脚本会在脚本列表展示。

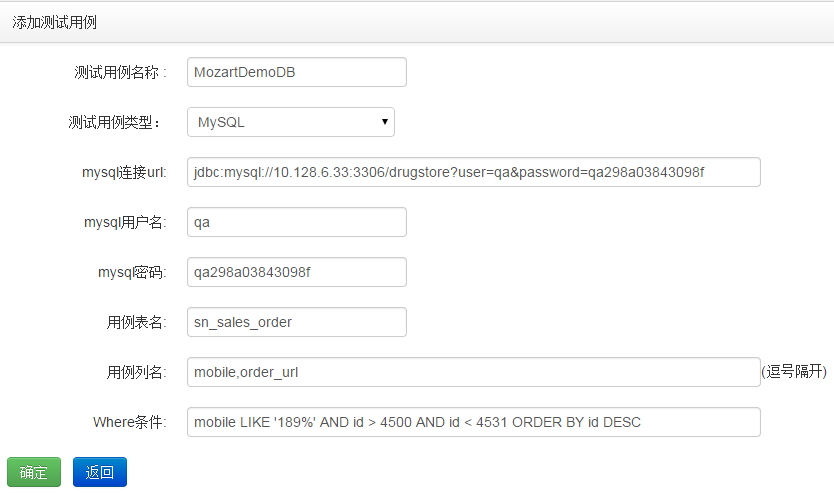
## 1.3 数据列表

测试数据提供上传文件和直接查询数据库2种方式获得。

### 1.3.1 查询数据库

首先需要从数据库里面获取一些数据，用在测试脚本中，可使用Mysql类型的测试数据方式。

首先添加测试数据



点击确定，返回到测试数据列表



进入任务列表，给相应场景增加测试数据



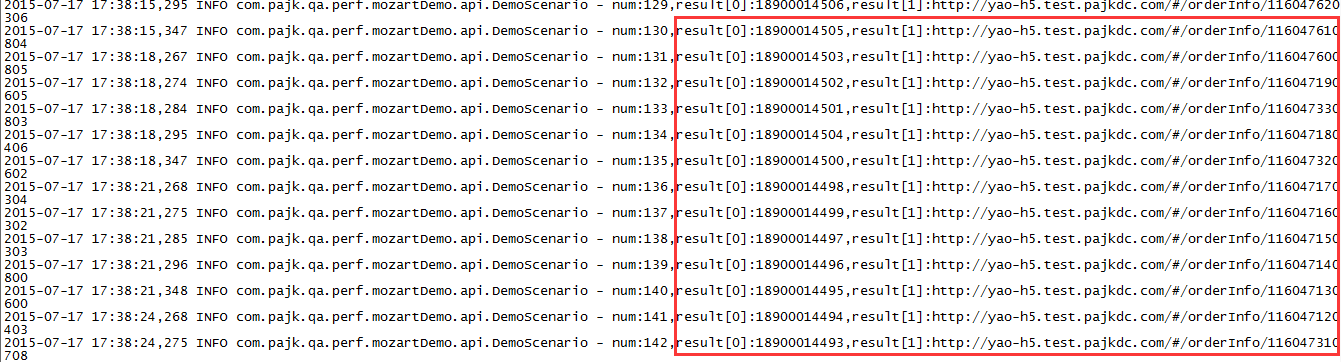
再测试脚本使用数据，增加以下代码即可：

String[] resultStrings = PerfTestCase.getTestCase("MozartDemoDB");

log.info("num:{},result[0]:{},result[1]:{}", num.get(), resultStrings[0], resultStrings[1]);



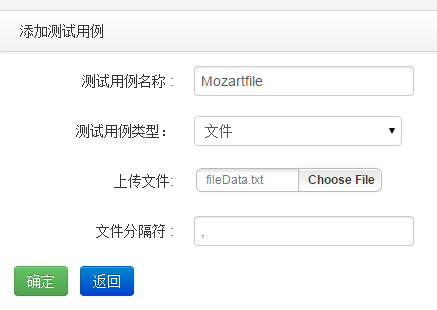
查看后台framework.log日志，可以看出已经拿到数据库的数据



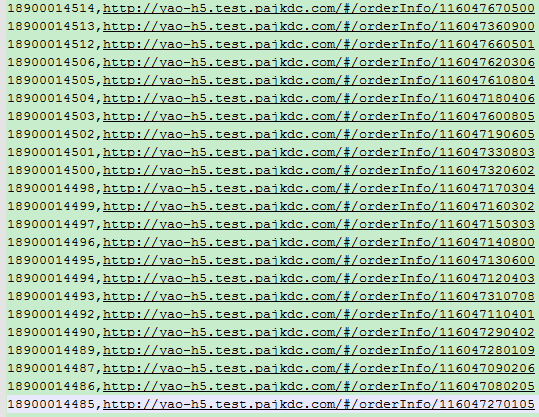
### 1.3.2 上传测试数据文件

另外一种方式是直接上传文件的方式

首先添加测试数据，上传文件



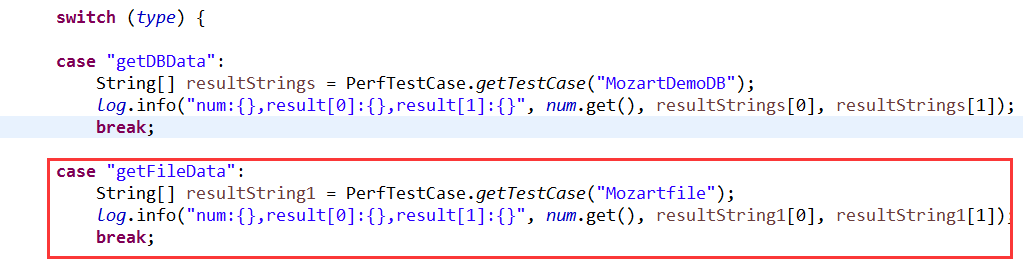
文件内容如下



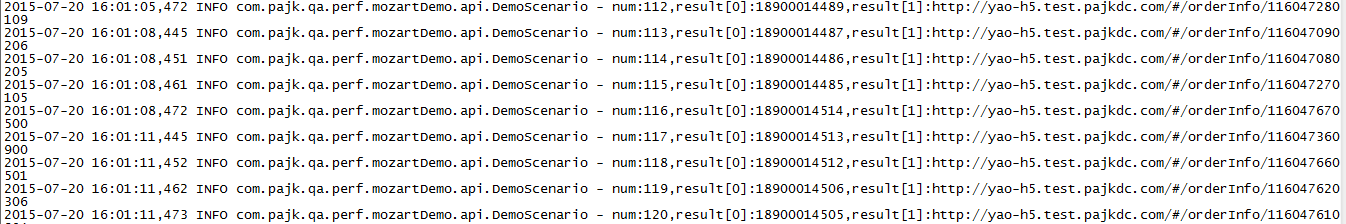
在任务的场景中增加此测试数据调用



测试代码中增加调用

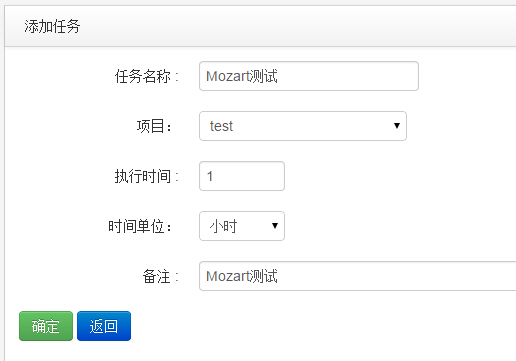


启动任务，并查看后台framework.log日志，可以看出已经得到上传文件的数据



## 1.4 任务管理

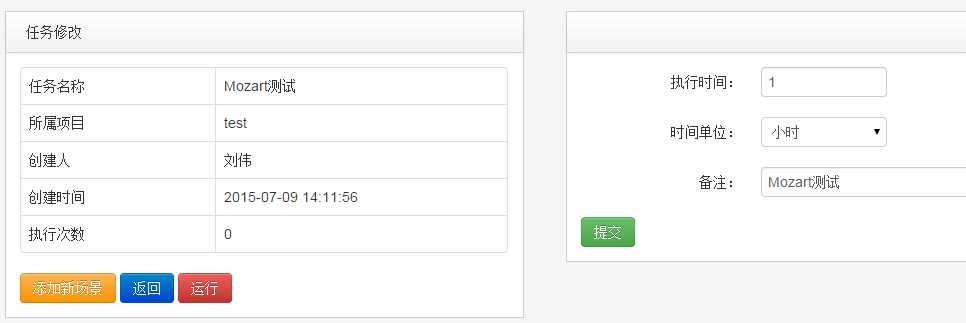
首先需要添加任务，自定义任务名称，选择项目名称，执行时间等。



任务添加完成后会在任务列表展现



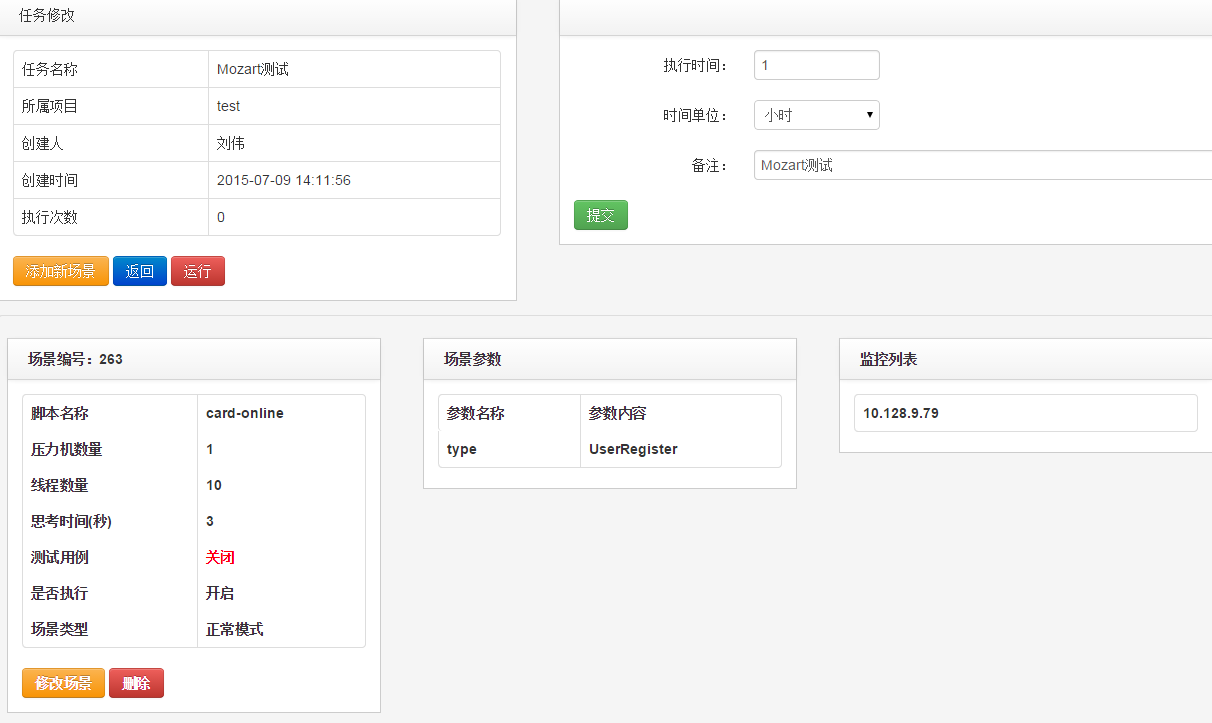
点击操作按钮的修改功能，进入任务修改页面



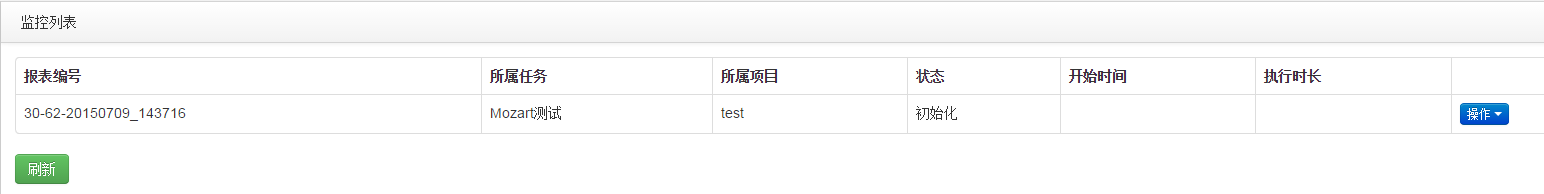
添加新场景



在添加场景页面，选择执行脚本，压力机数量，思考时间，需要监控的服务器ip，确定提交。



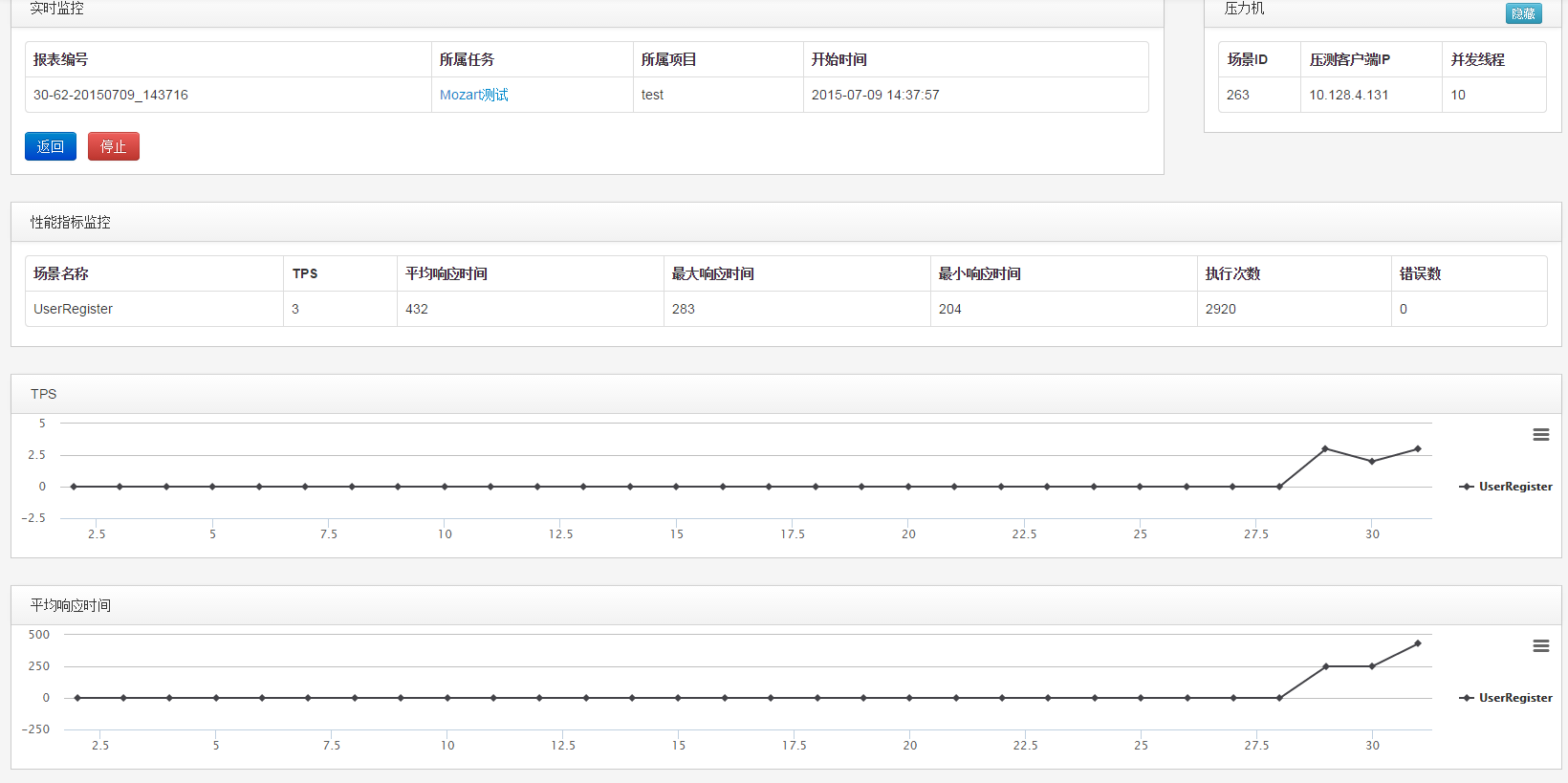
之后回到任务页面，点击运行，进入实时监控页面



监控页面会每3秒刷新状态

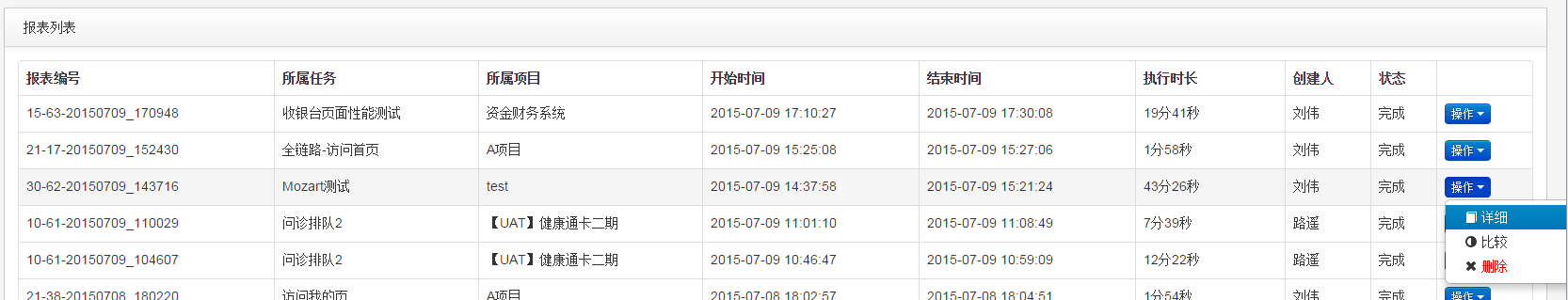


点击进入实时监控

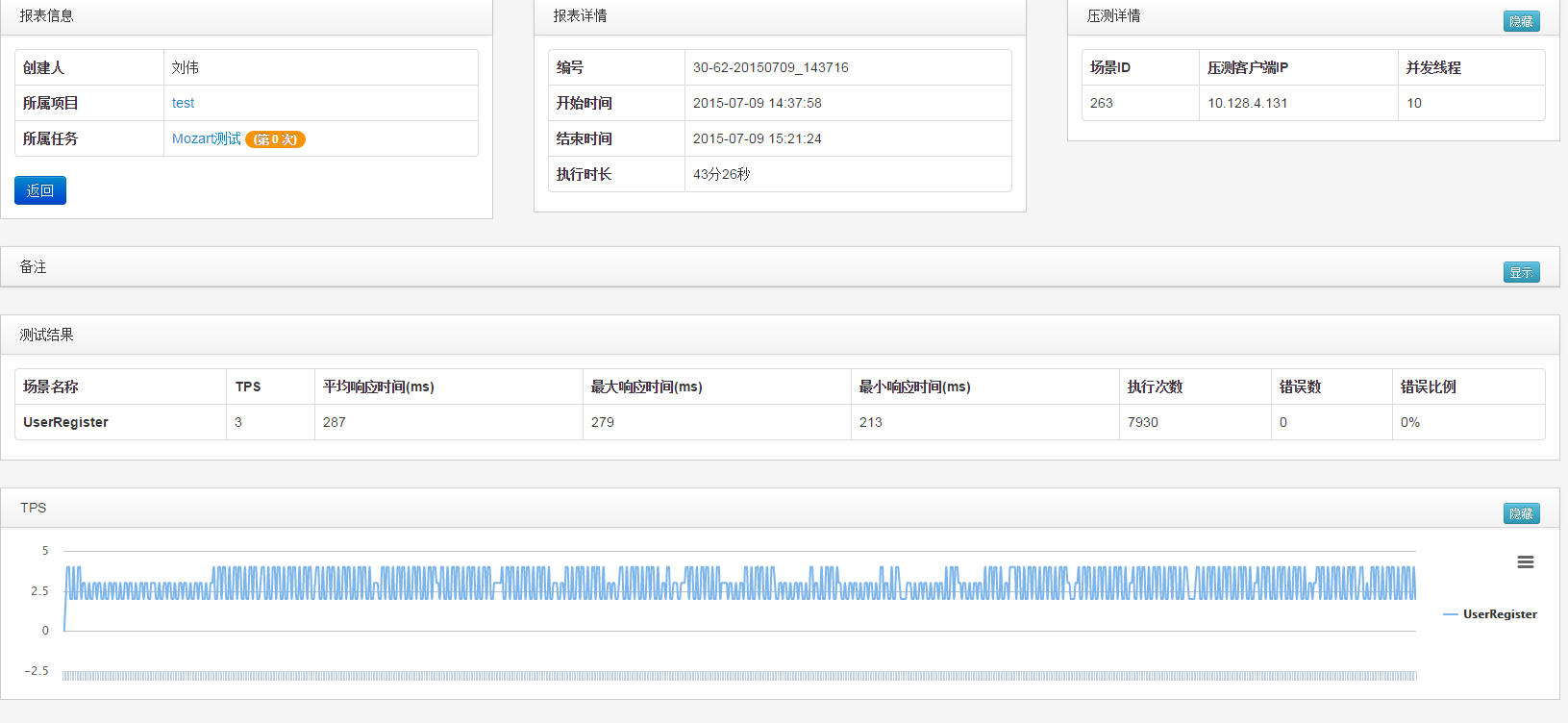


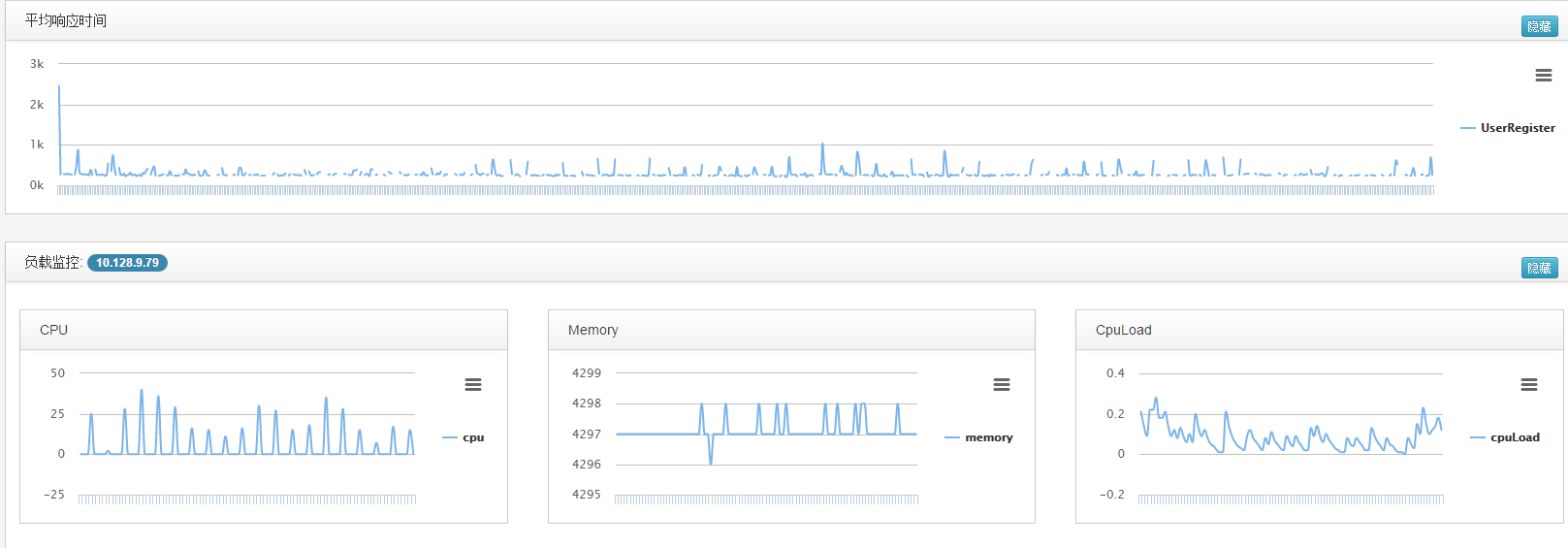
## 1.5 报表管理

停止测试程序后，进入报表管理



点击详细，即可展示详细的测试结果以及指标

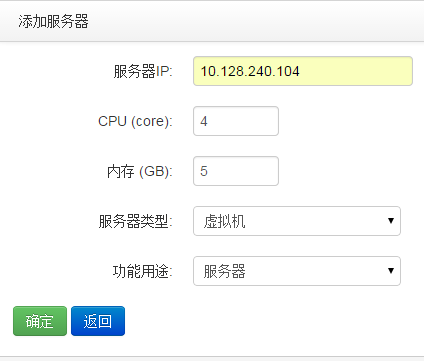




## 1.6 服务器管理

服务器管理分为服务器列表和添加服务器，位于“系统管理”菜单下面。

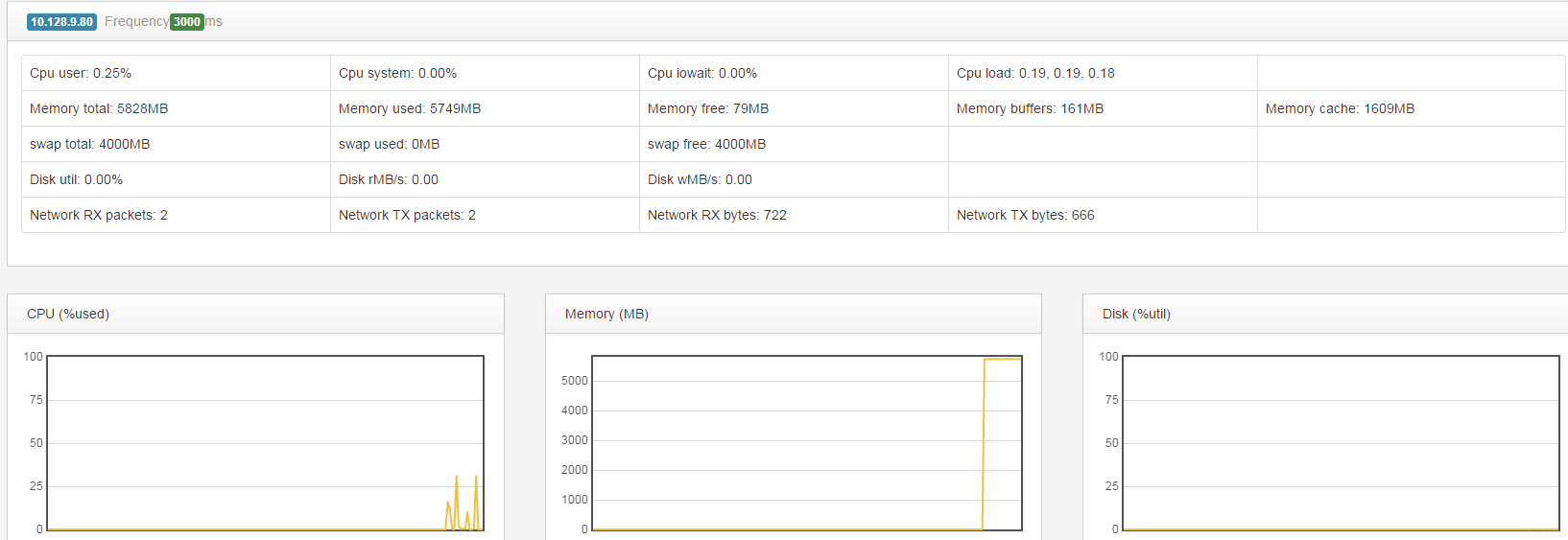
性能测试中如果要监控服务器的性能指标，必须先在服务器部署监控程序，并把相应的服务器地址添加进来，点击添加服务器，输入相应的服务器IP，以及功能用途。



功能用途分为服务器和压力机，开始性能测试的时候，mozart会自动分配空闲状态的压力机，作为压力发生器，对服务器产生压力。

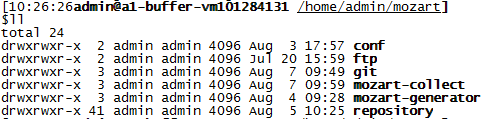


在服务器列表，点击实时监控，即可实时监控服务器状态。



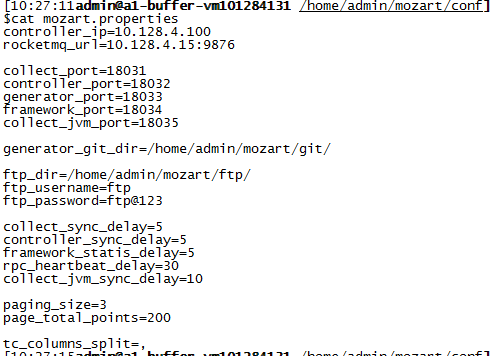
## 1.7 压测客户端目录架构

进入压测客户端mozart目录



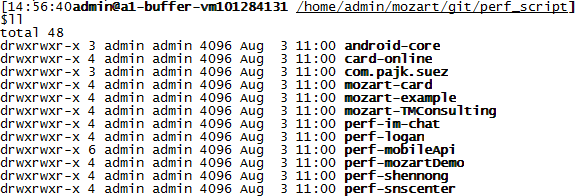
可以发现主要由conf，ftp，git，respository组成。

Conf：目录下存放的是配置文件。



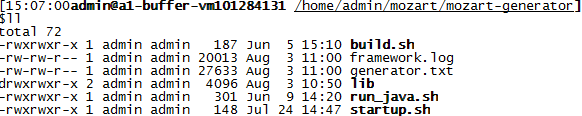
ftp：目录下存放的是上传的测试数据，用户在controller上传测试数据，压测开始后controller会把数据文件传到压测客户端，保存在ftp目录下。

git：目录下存放的是压测代码。

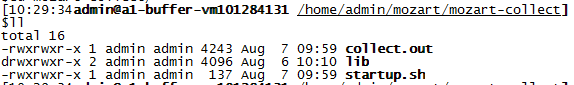


Respository：目录下存放的是maven库代码。

mozart-generator：目录下存放的是generator的启动命令，启动所需要的jar包，压测运行中的一些日志和generator运行日志。



mozart-collect：目录下存放的是服务器server的监控jar包。



## 1.8 应用数据分析

进入任务列表页面，点击数据分析功能

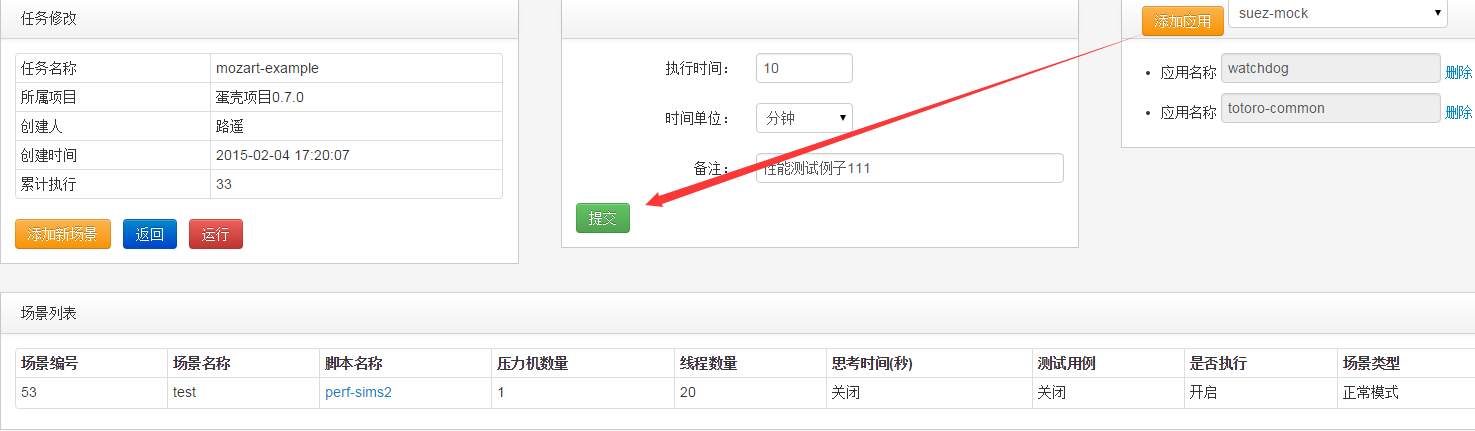




数据为空的原因是此任务暂时还没有添加相关的应用数据。

### 1.8.1 添加应用数据

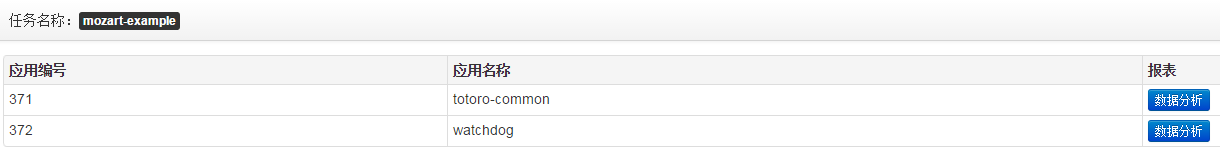
点击修改，进入任务修改页面。



需要添加的应用增加后，点击提交即可。

### 1.8.2 任务入口

再次回到任务列表页面，点击数据分析。



跳转页面就会显示此任务所绑定的应用，在对应的应用后面点击数据分析。



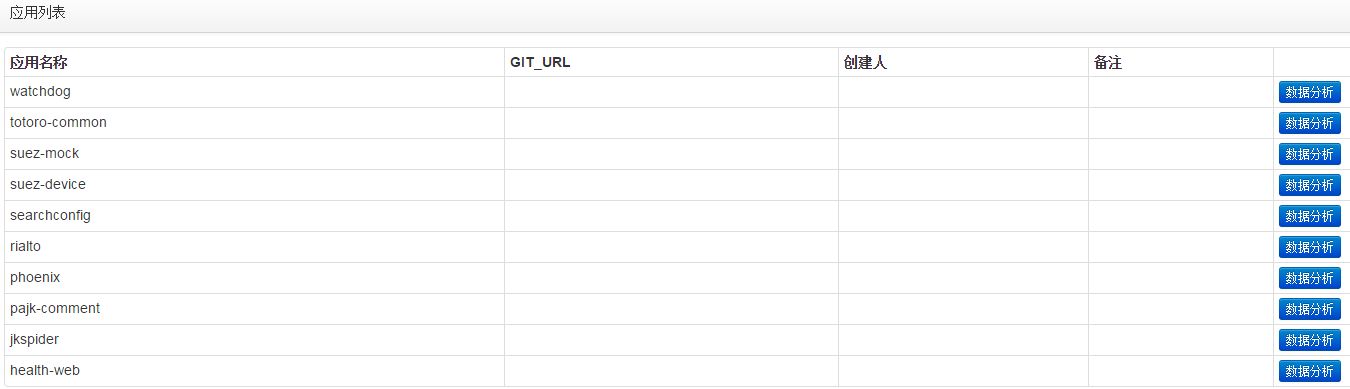
此时会跳转到涉及此应用的所有的任务页面，点击后面的测试报表。



此时会跳转到，任务所涉及到的所有测试报告页面。

### 1.8.3 应用页面入口

点击进入项目管理下面的应用列表。



点击需要查看的应用后面的数据分析

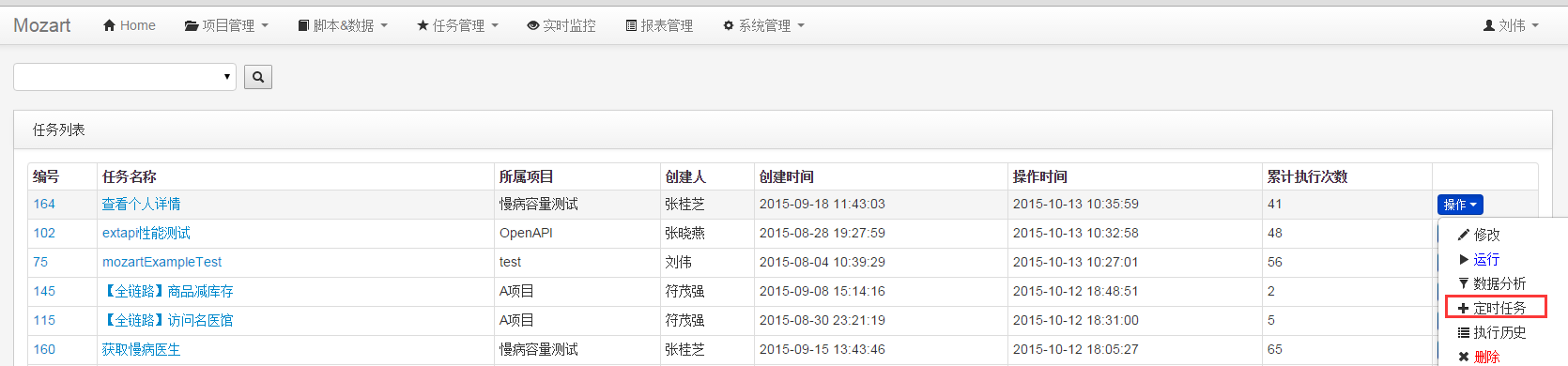


此时会跳转到涉及此应用的所有的任务页面，点击后面的测试报表。

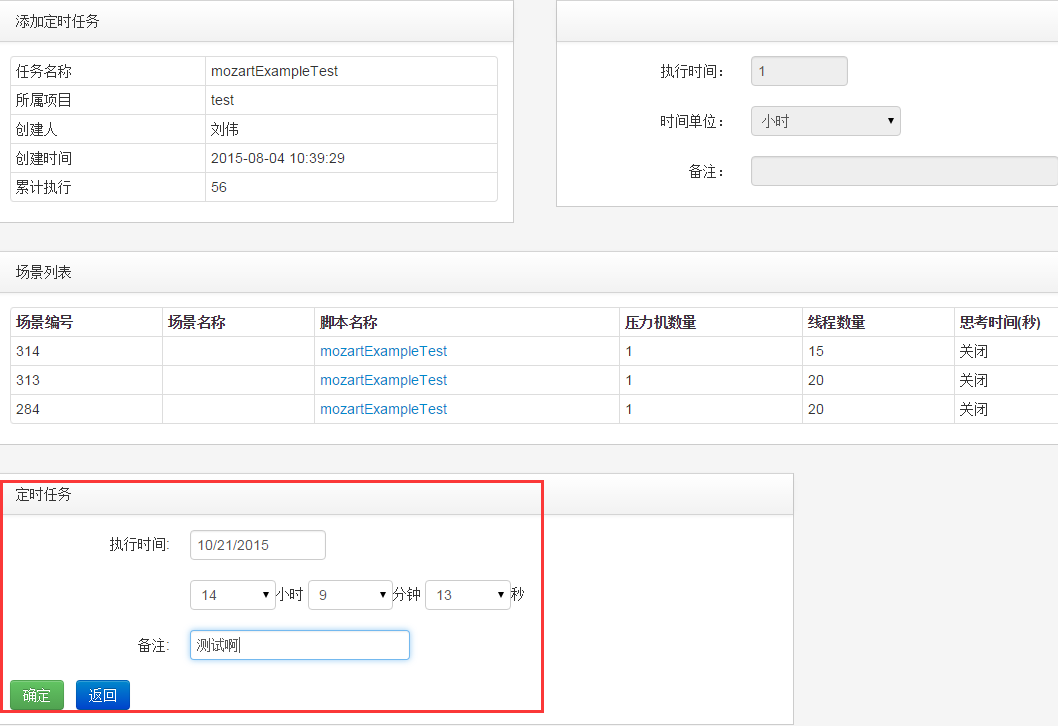


## 1.9 定时任务

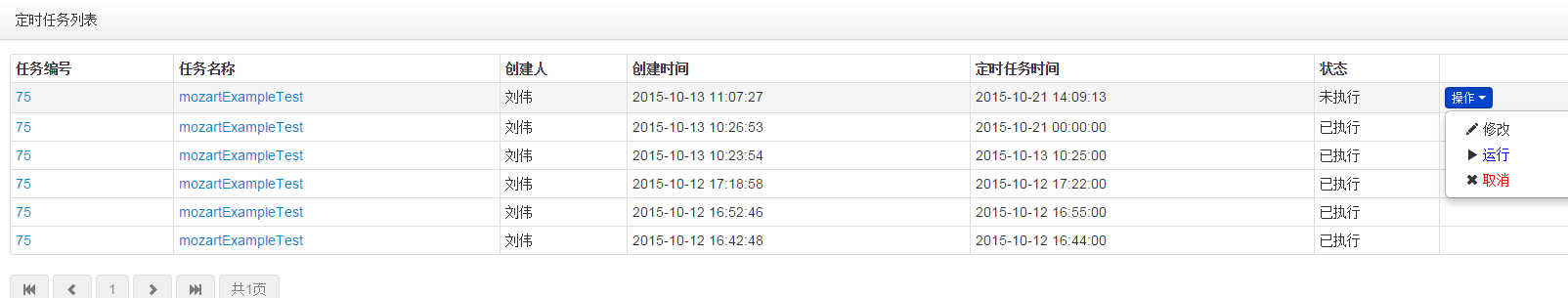
性能平台支持定时任务功能，进入任务列表，点击操作按钮下的定时任务的功能



进入定时任务页面后，配置需要添加的日期，时分秒点击确定提交即可



提交后进入到定时任务列表



未执行状态的定时任务可以修改定时时间，取消，直接运行等功能

直接点击任务管理下面的定时任务菜单也可以进入此页面。

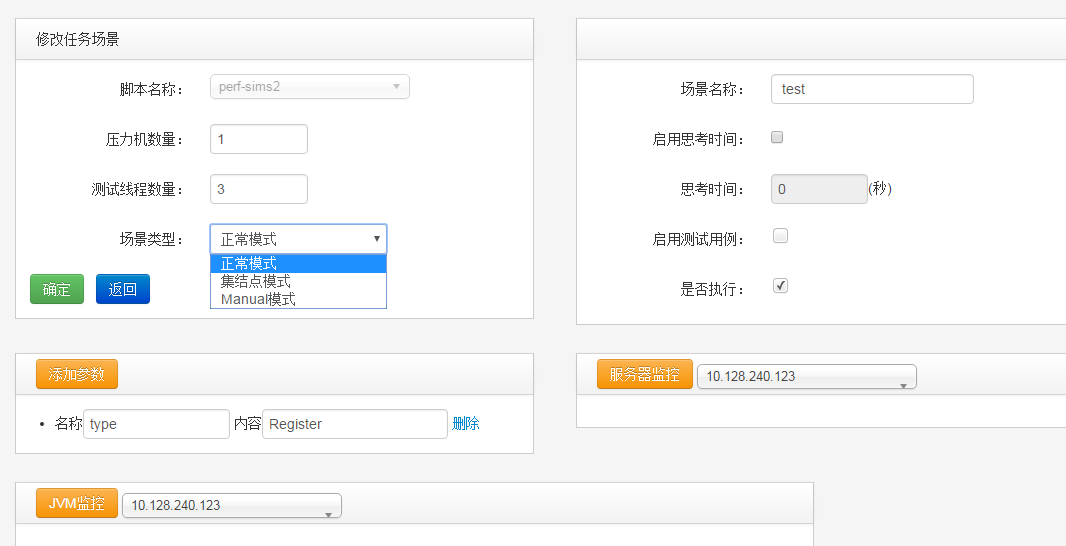


## 1.10 任务场景类型

### 1.10.1 自定义场景-逐渐增加/减少线程

当增加或者修改任务场景的时候，可以根据需要设置线程逐渐的增减。

在增加/修改任务场景的时候，在场景类型下拉框中选择manual模式



在配置场景的页面选择manual模式

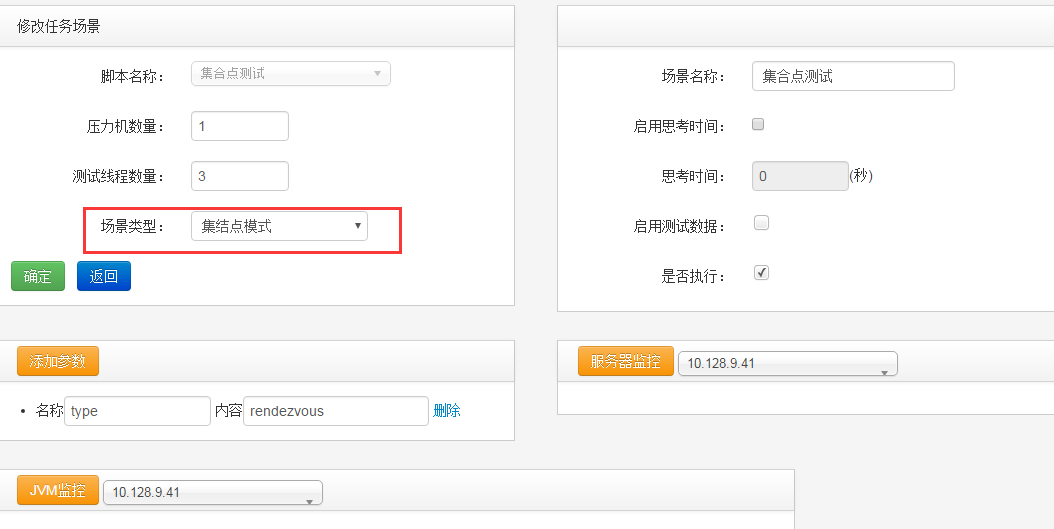


例如上面的设置含义是：

每台机器起始线程3个，之后会每间隔8秒增加5个线程，增加6次，最终是3+5\*6=33个线程。

### 1.10.2 集合点场景

首先在修改任务场景的地方配置场景类型为集结点模式



场景里面配置完成后，在代码里面的setupTest方法里面写需要预先设置的一些方法，之后真正集中并发的场景写在runTest里面。

例如下面的截图，先在setupTest里面注册用户，注册之后在runTest的“rendezvous”里面进行真正的场景并发。

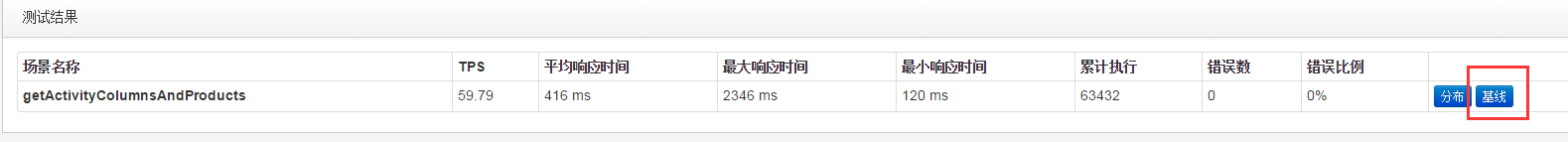


## 1.11 性能基线

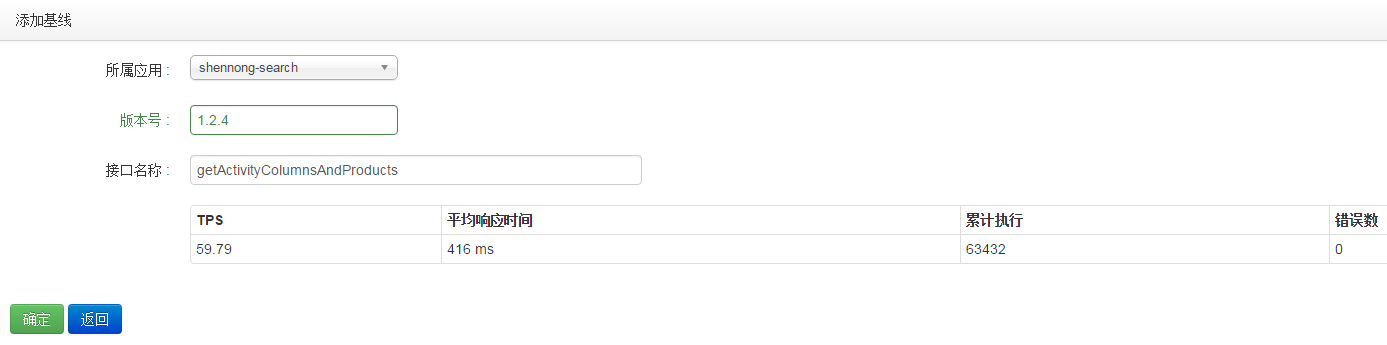
性能基线是以接口为单位的，之后和应用连接起来，可以按照应用名称来查找包含的接口基线测试数据，也可以单独查找接口名称。

### 1.11.1 添加接口数据基线

首先进入测试结果的报表页面，之后点击操作的详细，进入详细的报表信息。



点击测试结果栏的基线按钮，进入基线的添加页面



例如getActivityColumnsAndProducts接口属于shennong-search的应用，填上相应的测试版本号，点击确定。



基线列表页面会展示出添加的基线数据。

### 1.11.2 基线搜索

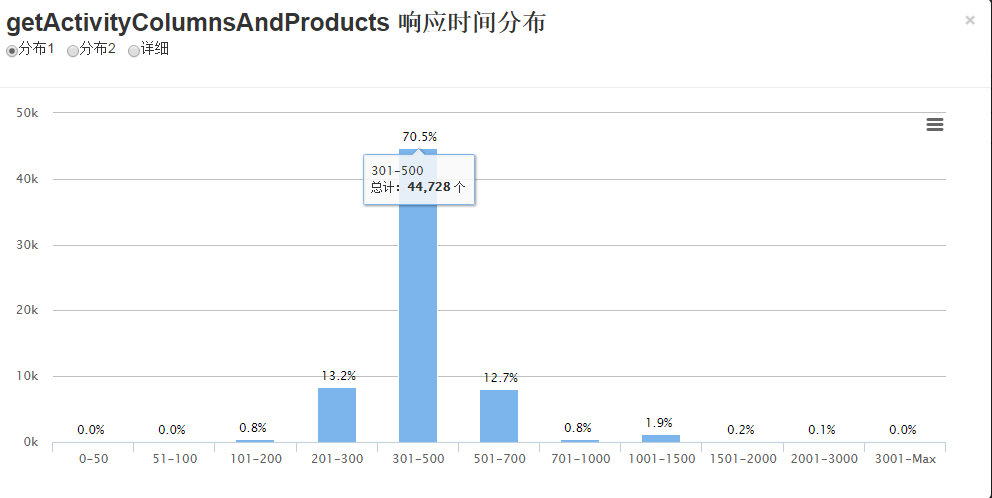
基线列表页面支持按照应用名称和接口名称



### 1.11.3 基线操作

在基线后面的操作按钮中，有数据分布，报表，修改和删除功能。

点击分布按钮，会展示测试结果数据分布，和测试报表的数据分布一致。



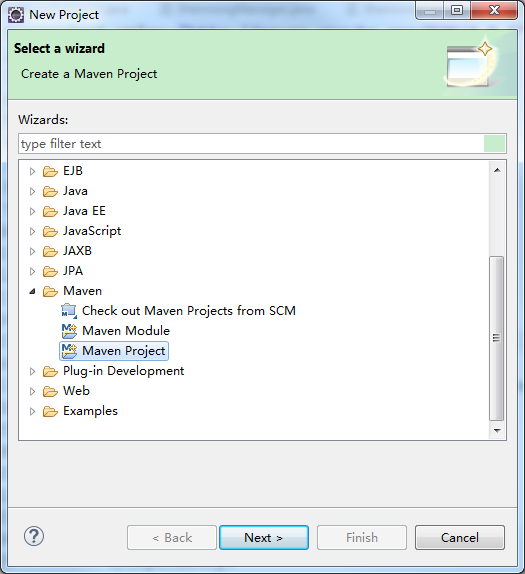
点击报表按钮，会直接跳转到此接口的测试报表页面。

点击修改可以修改应用名，接口名，版本号等。

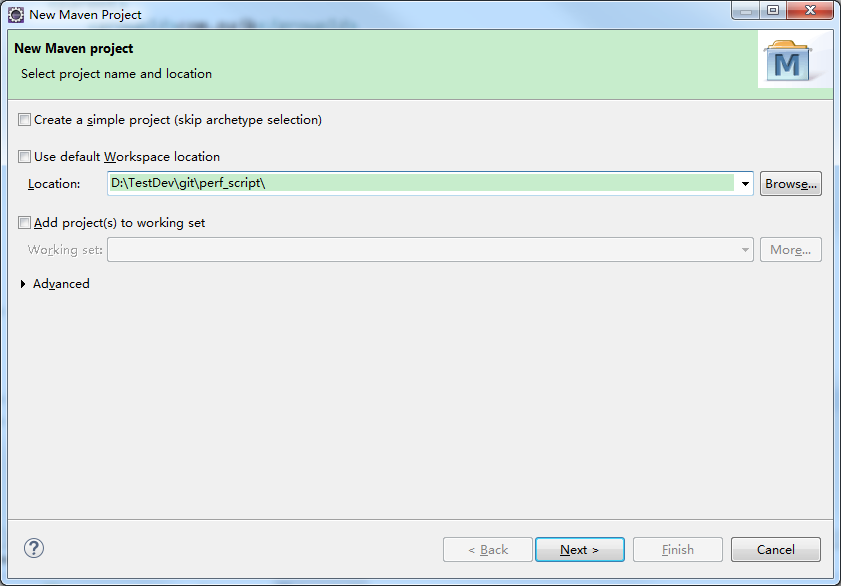
# 2 实例-性能测试流程

## 2.1 maven项目创建

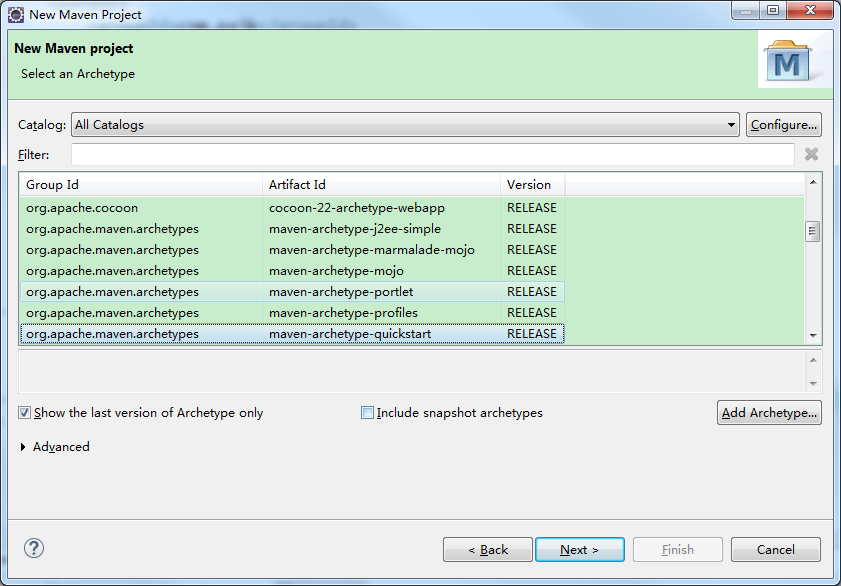
新建maven项目



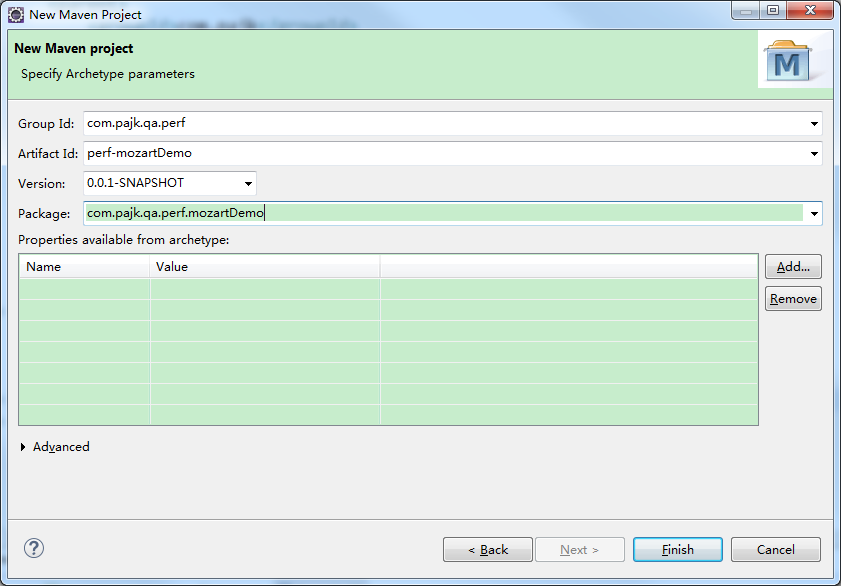
注意新建项目要放在“git@10.0.128.104:perf\_group/perf\_script.git”，因为mozart平台初始化时，会自动从此路径pull最新的项目代码，并且编译



Next



Next，填写相应的pom文件配置信息，finish即可



项目创建完成后，打开pom，增加mozart依赖包

<dependency>

<groupId>com.pajk.qa.mozart</groupId>

<artifactId>mozart-framework</artifactId>

<version>0.0.2-SNAPSHOT</version>

</dependency>

<dependency>

<groupId>com.pajk.qa.mozart</groupId>

<artifactId>mozart-tool</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

之后就可以根据项目增加不同的测试代码了。

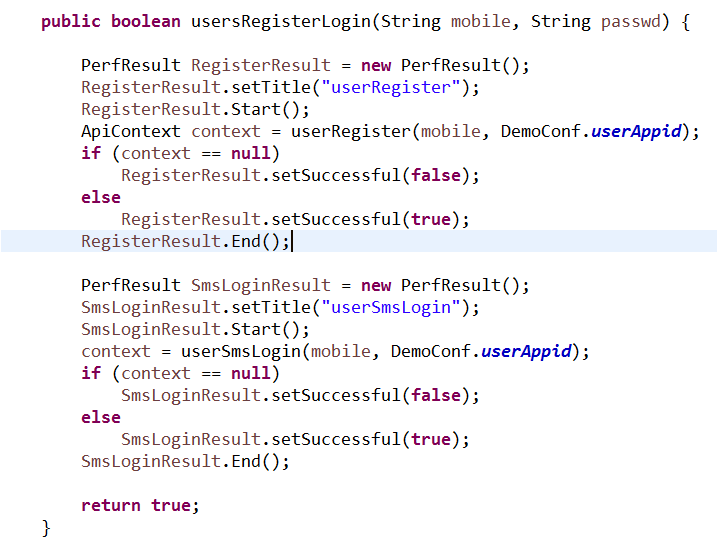
## 2.2 测试代码调用

下面以用户注册登录接口为例

### 2.2.1 实现PerfTestApi

新建class DemoScenario用以实现PerfTestApi，把注册登录的实现接口写在runTest方法里面，mozart平台会自动的循环调用runTest方法。





### 2.2.2 PerfResult测试结果统计

PerfResult RegisterResult = new PerfResult();

RegisterResult.setTitle("userRegister");

RegisterResult.Start();

// 接口调用代码

RegisterResult.setSuccessful(true / false);

RegisterResult.End();

这个是固定用法，使用这个格式就可以把接口消耗的时间计算出来，反馈给mozart controller。

### 2.2.3 PerfError错误信息统计

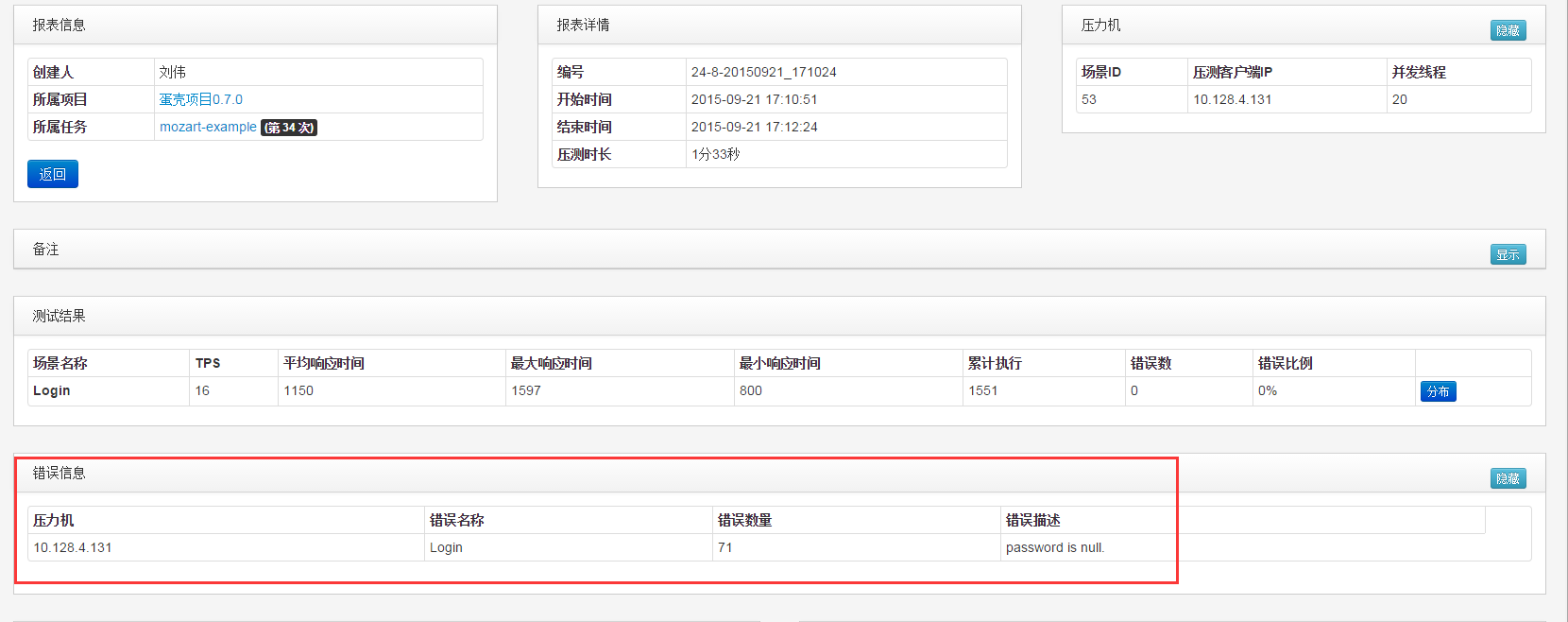
PerfError err = new PerfError();

err.setTitle("Login");

err.setContent("password is null."); // 可选

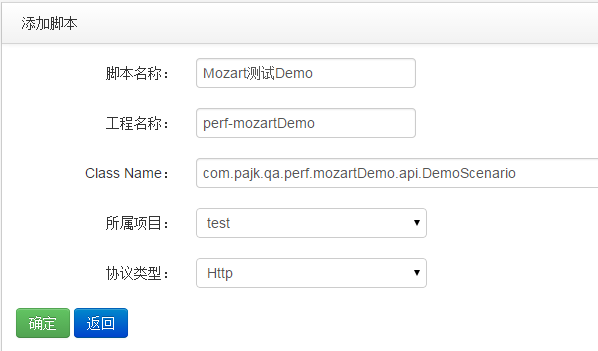
err.Statis();

通过调用PerfError可以对测试过程中返回的错误码等预期错误进行统计，并展示在测试报表中。



## 2.3 mozart创建脚本

为新项目添加脚本



1. 工程名称就是新建的maven项目名称。
2. Class Name是测试代码的路径以及名称。

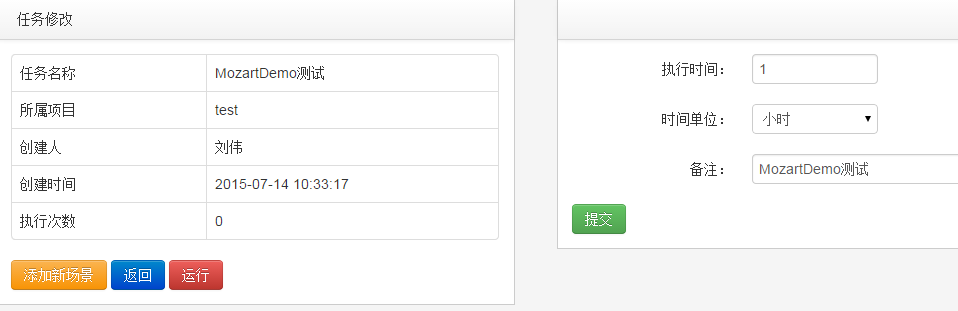
## 2.4 mozart添加任务和场景

首先创建新的任务





点击任务的修改，增加新的场景



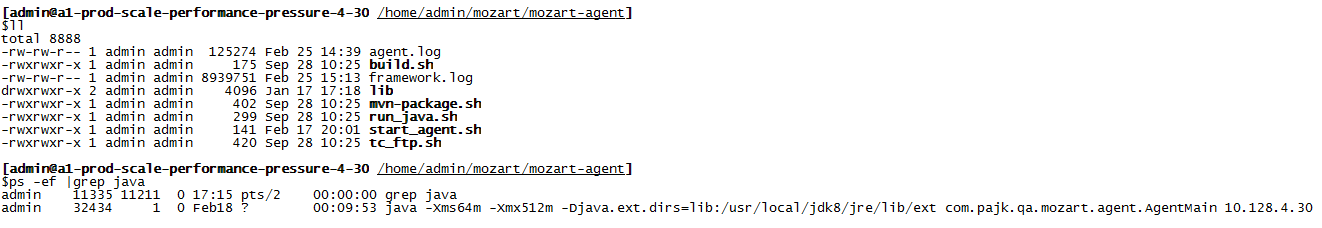


1. 选择刚才新建的脚本。
2. 设置压力机数量1，以及测试线程数量5。修改压力机数量为2，线程数量不变还是5，说明每台机器启动5个线程，共10个线程。
3. 添加参数是在多个场景下，根据type的类型，选择执行哪个场景



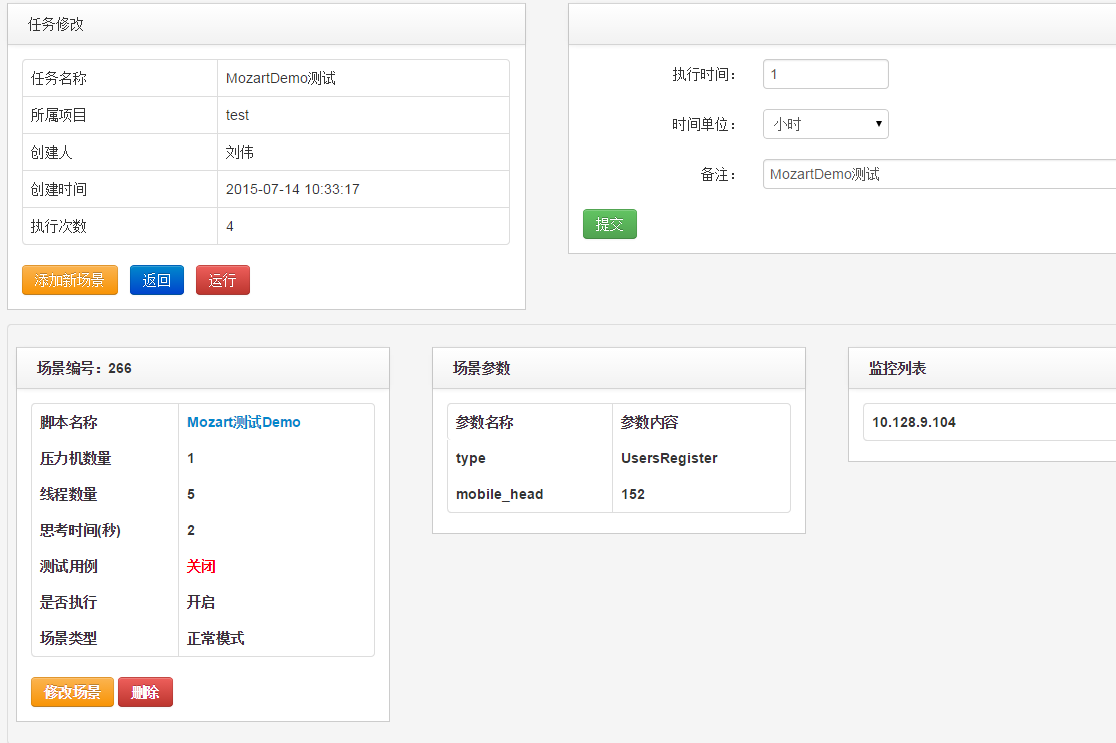
首先是利用private static String type = PerfParameter.getParameter("type")方法从mozart controller获取type类型内容，之后在runTest里面匹配具体的场景代码。

1. 添加服务器，服务器上面必须已经部署mozart-collect包并启动，



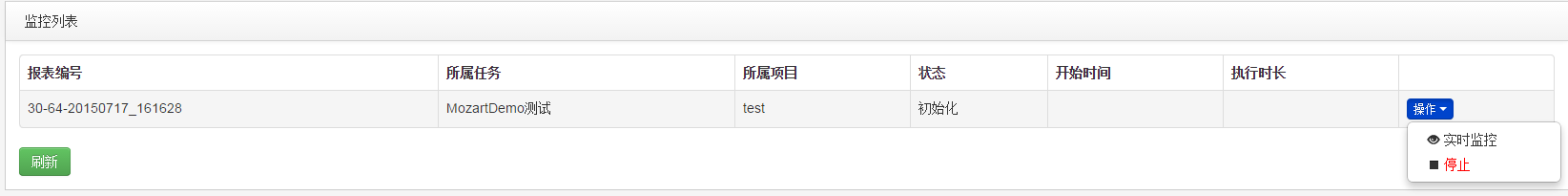
Mozart启动之后会自动监控信息。

添加场景之后，详细信息如下

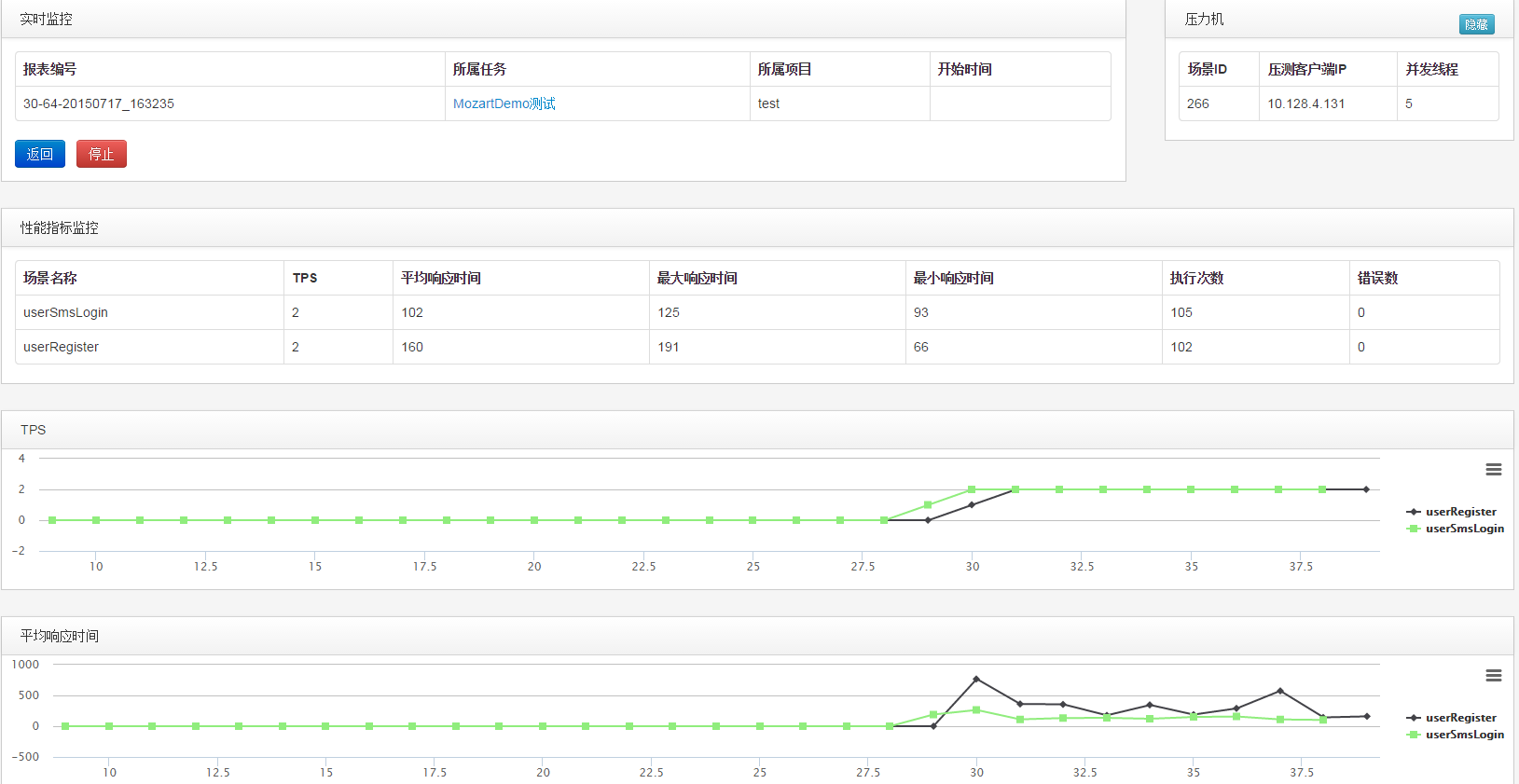


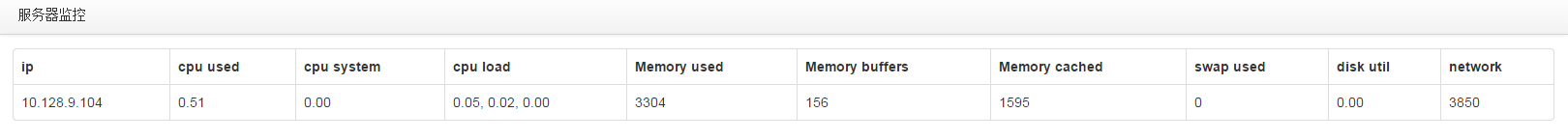
## 2.5 运行测试

点击运行，会跳转到实时监控页面

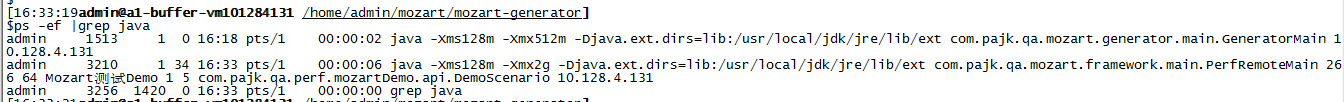


点击实时监控链接，可以实时监控TPS，响应时间，错误数，服务器CPU，内存，CPU Load等指标。





登陆压测机131可以发现PerfRemoteMain线程已经启动



打开服务器列表，可以发现131压力机已经处于使用中状态

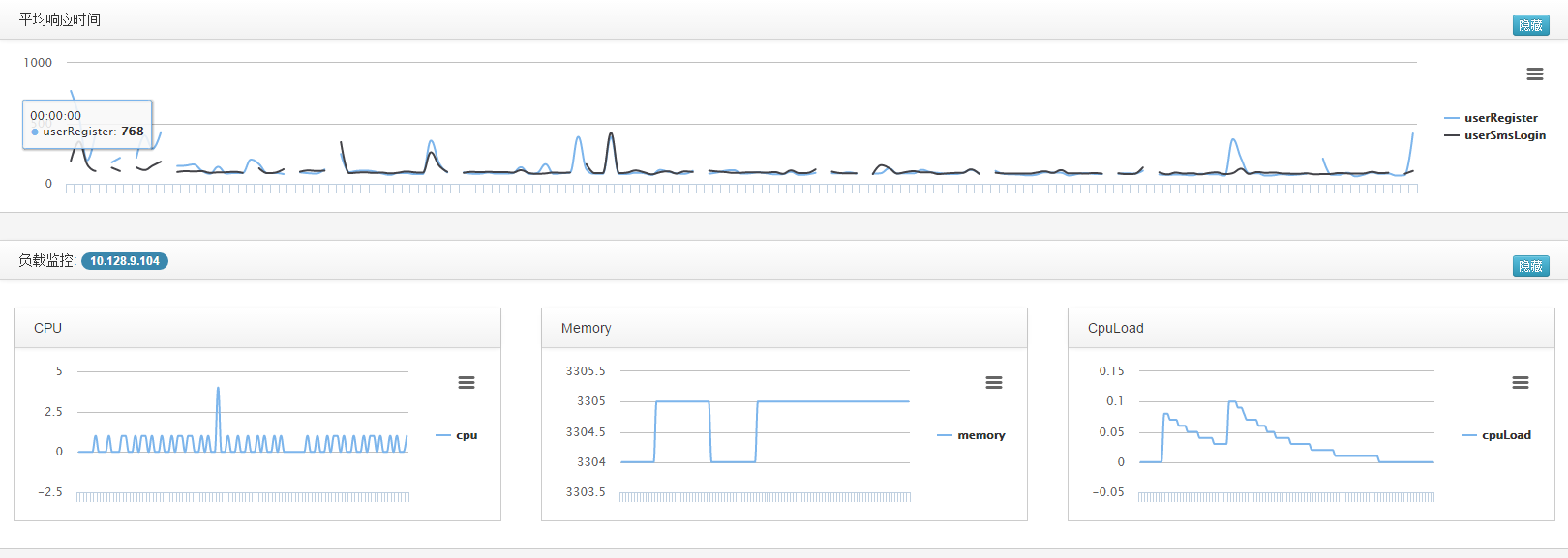


测试停止后，进入报表管理页面



可以发现刚才停止的测试报告，点击详细进入，可以查看整个测试过程中详细的数据指标。





## 2.6 报表比较

测试结束后，相同的任务场景经过优化，跑了很多次，可以选择任意的俩次进行报表对比。对比结果可以比较明显的展示测试结果的优化与否。

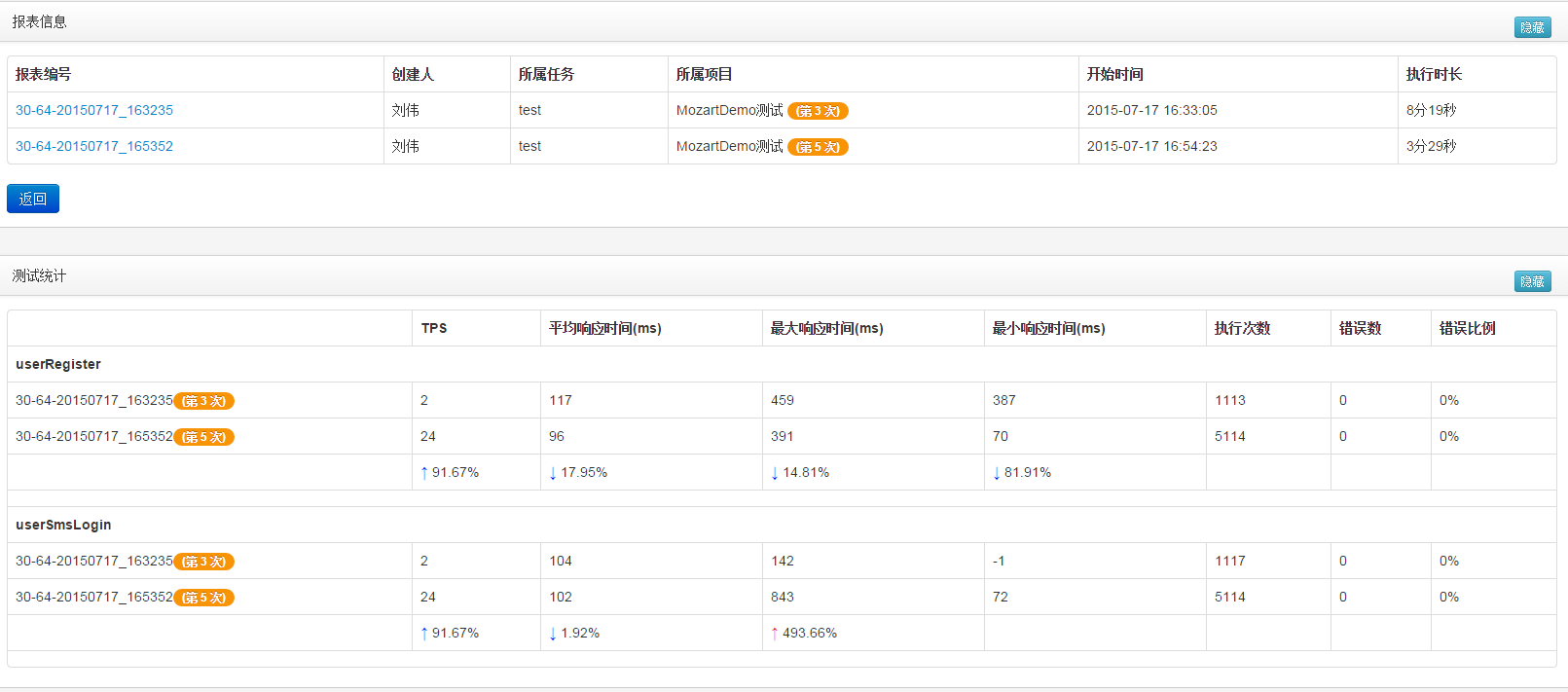
选择想要对比的报表，点击比较

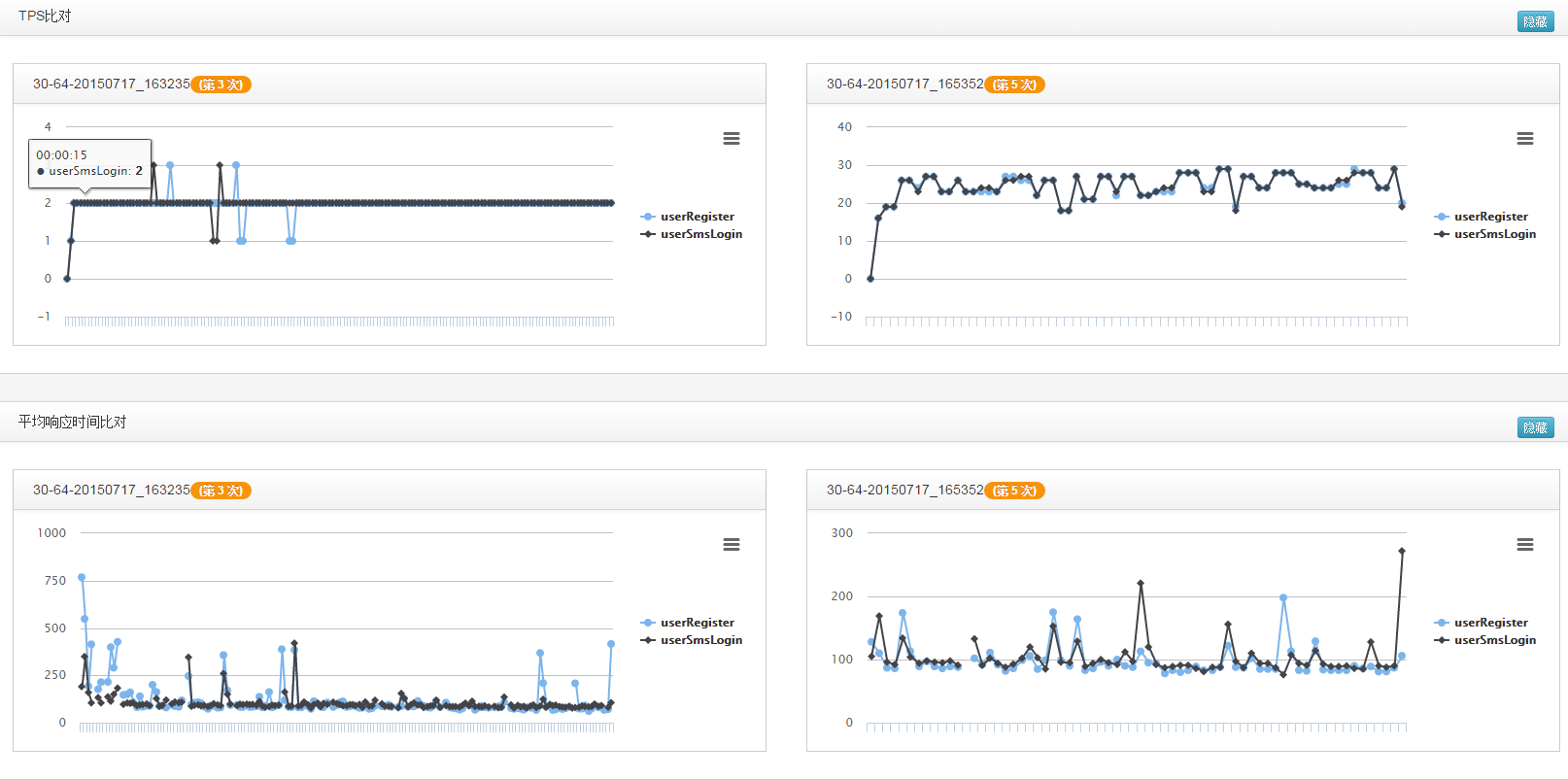


在选择第二个报表页面会自动筛选出所属任务相同的报表



点击比较





# 常见问题

## 部署监控客户端

Mozart的监控客户端功能含有服务器监控和jvm监控，目前已经优化集合成一个包mozart-agent，所以只需要部署一个包就可以实现监控的需要。

例如10.128.4.120部署监控客户端，首先需要登录此服务器，切换至/home/admin/目录下(固定路径，请不要更改)，到<http://doc.pajk-ent.com/pages/viewpage.action?pageId=24117389> 下载脚本并执行。