SockShop测试报告

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2020-05/10 | 1.0 | 对SockShop进行性能测试，从吞吐量、并发行及响应时间等角度进行测试分析 | 王笑然、褚宇轩、王志远、吕艺 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 范围 4

1.3 参考资料 4

1.4 概述 4

2. 测试概要 4

3. 测试环境 4

4. 测试过程及结果 5

4.1 参数化测试用例 5

4.2 关联测试用例 6

5. 测试结果及分析 6

5.1 性能测试结果 6

5.1.1 场景一测试结果 7

**a.** **吞吐量** 7

**b.** **响应时间** 7

**c.** **并发性** 7

5.1.2 场景二测试结果 8

5.1.3 场景三测试结果 9

5.2 SLA服务水平协议定义 10

6. 测试结论与建议 10

6.1 瓶颈分析 10

6.2 优化建议 11

测试报告

# 简介

## 目的

该测试报告将项目的测试过程及结果写成文档，对发现的问题进行分析，为纠正软件存在的质量问题提供依据，对SockShop应用进行质量评估。

## 范围

此报告覆盖的是定义、首字母缩写词和缩略语

无

## 参考资料

《测试计划》

《Software Testing􀀅A Craftsman’s Approach Fourth Edition》Paul C. Jorgensen􀀅2014

《软件测试：第3版》，人民邮电出版社，2011

## 概述

SockShop系统是在的weaveworks已有的SockShop Demo的基础上进行改造的。SockShop 系统需支持用户完成注册、登录，并支持商品的信息进行浏览、加入商品到购物车，以及结算、支付和订单查询的等功能。我们选取其中的登陆以及选取商品进行测试，

# 测试概要

测试时间：2020-07-02

测试方法：性能测试(包括参数化测试和性能测试)

测试人员：王笑然，褚宇轩，王志远,吕艺

测试内容：使用Load runner套件进行测试

# 测试环境

软件：Load Runner,Microsoft IE Explorer

硬件：笔记本电脑

# 测试过程及结果

## 参数化测试用例

这里我们利用登录信息，商品数量，用户地址以及支付卡号信息作为参数进行参数化测试。

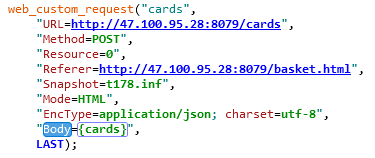
在这里我们通过输入用户名和密码后点击登陆按钮登录系统。这里我们使用不同的用户地址，卡号信息以及商品数量。

相关代码截图如下所示：

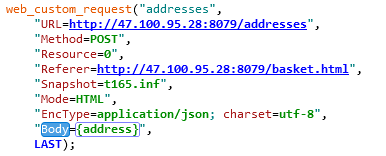
a. 商品数量设置



b. 支付卡号设置



c. 用户地址设置



d. 用户登录

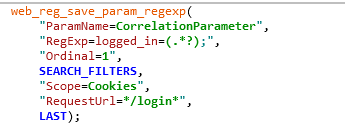


## 关联测试用例

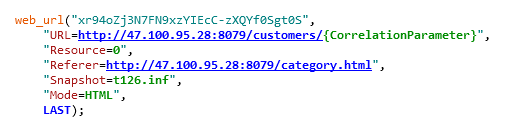
在网站中我们使用了Cookie。然而每一次的cookie值中的login信息是服务器动态生成的，我们需要动态保存服务器生成的值，并且在代码中使用这个保存的参数值进行测试。

在这里我们使用了自动关联的方法，首先录制一遍操作流程，之后replay，在开启自动关联的功能后就可以自动监测到需要关联的对象，创建关联规则，生成保存动态参数的程序。

我们的测试过程中主要涉及到的动态参数是cookie中的login的ID信息。



在使用的时候，我们提取自动保存的变量。



# 测试结果及分析

针对服务器性能测试的三个主要指标：并发性，吞吐量以及响应时间，我们设置了以下几个场景。

**表1 场景设计**

|  |  |  |  |
| --- | --- | --- | --- |
| **场景** | **1** | **2** | **3** |
| **Vuser初始化** | **运行前初始化** | **运行前初始化** | **运行前初始化** |
| **开启Vuser** | **每15s开始一个（共4个）** | **每15s开始一个（共6个）** | **每15s开始一个（共8个）** |
| **持续时间** | 运行直到结束 | 运行直到结束 | 运行直到结束 |

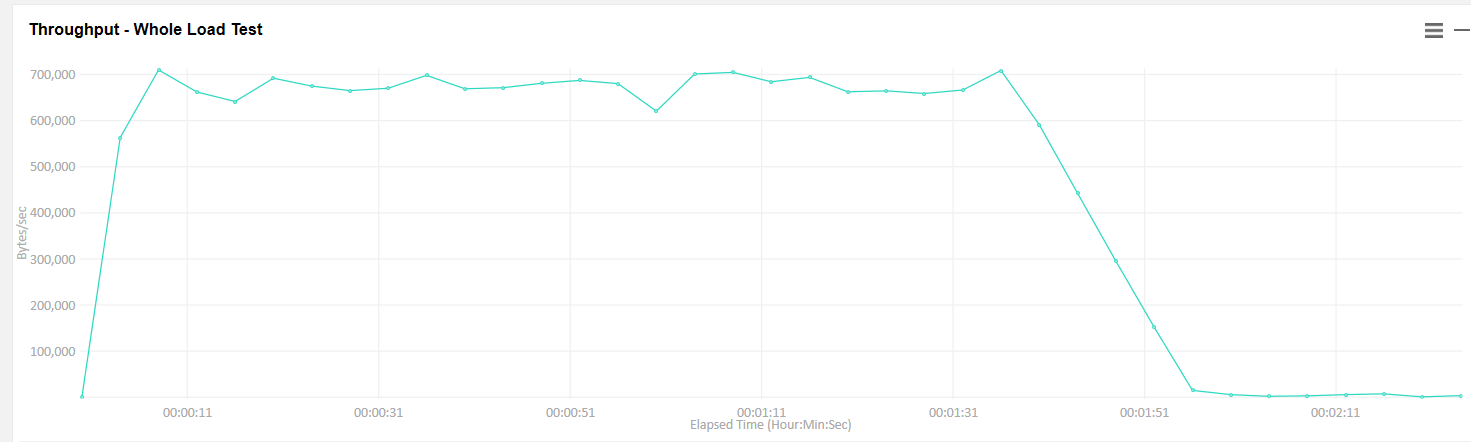
## 性能测试结果

**表2 不同场景下平均性能**

|  |  |  |  |
| --- | --- | --- | --- |
| **场景** | **1** | **2** | **3** |
| **用户数** | **4** | **6** | **8** |
| **并发性** | **2** | **3-4** | **5-6** |
| **吞吐量(Bps)** | 700,000 | 700,000 | 700,000 |
| **响应时间** | 110 | 140 | 170 |

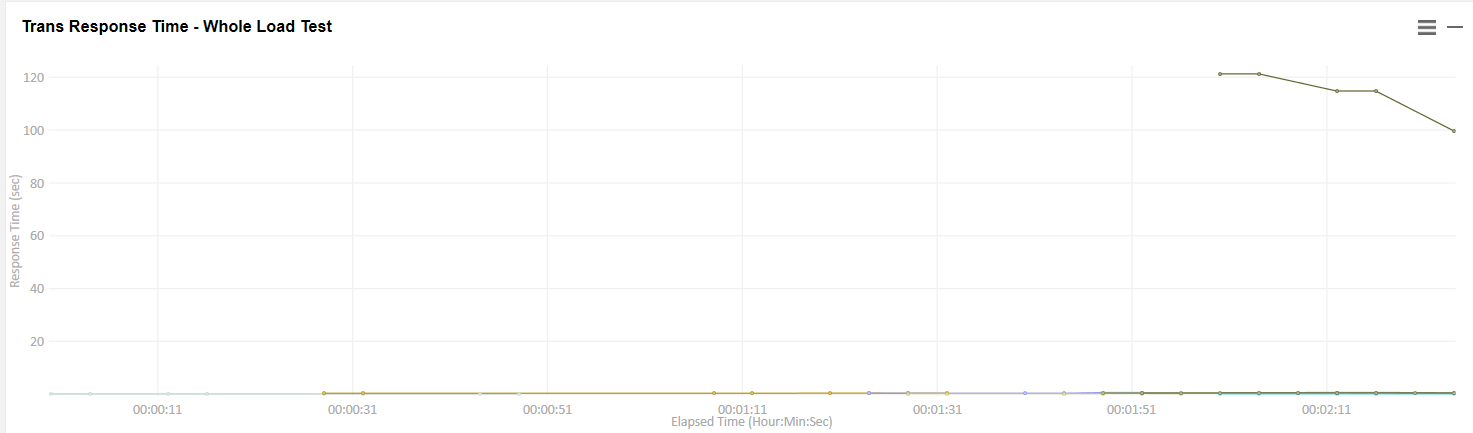
### 场景一测试结果

### **吞吐量**



可以看出当吞吐量到达700,000Bps后就难以上升了，到达了服务器的峰值。

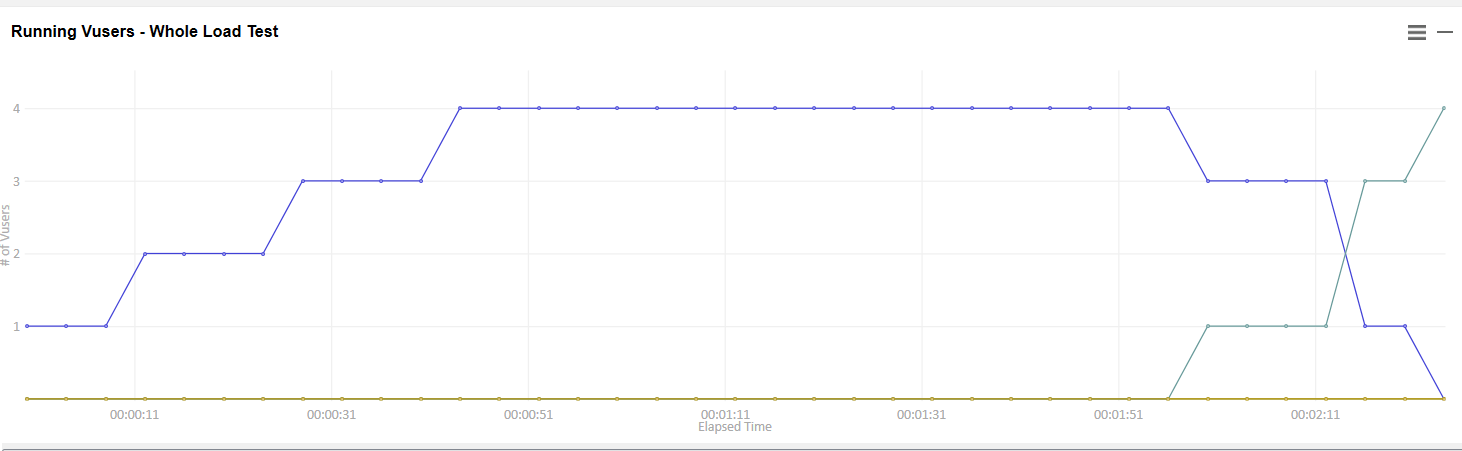
### **响应时间**



首先在这里vuser\_init\_Transaction和vuser\_end\_Transaction的时间是很短的，因此在这里可以看到这两条折线很平缓，基本在0s左右。

经过一段较长的时间，服务器端才开始进行处理，从图中可以看出在1:40s的时候服务器开始进行相应，我们的Action响应时间在110s左右，响应时间较长。

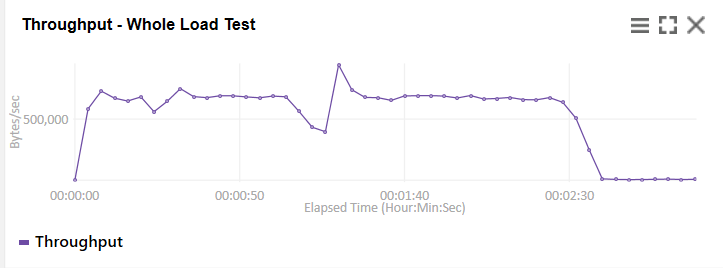
### **并发性**



考虑到服务器的性能，当我们选取过多的用户时，服务器可能会宕机。因此经过我们的一系列试验，用户数为4的时候是一个比较合适的系统上限。可以看到上图中的浅绿色折线表示在不同时刻的用户数量。可以看出平均情况下系统的并发用户数为2-3个。

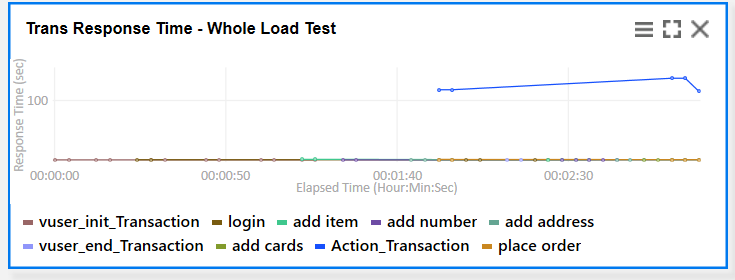
### 场景二测试结果

**a. 吞吐量**



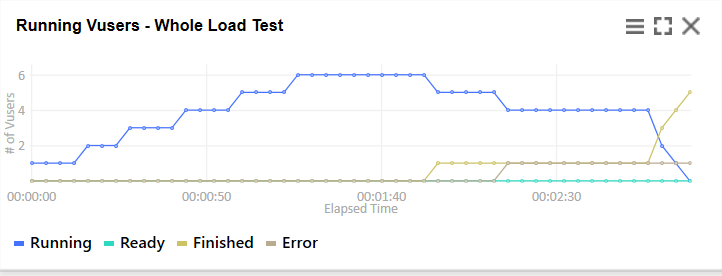
可以看出这里的吞吐量与场景一中相似，为700,000Bps左右。

**b. 响应时间**



这里主要的信息为蓝色折线所代表的Action\_Tranaction, 可以粗略的看出平均响应时间在135s左右。随着并发用户数量的增加，平均响应时间也相应增加。

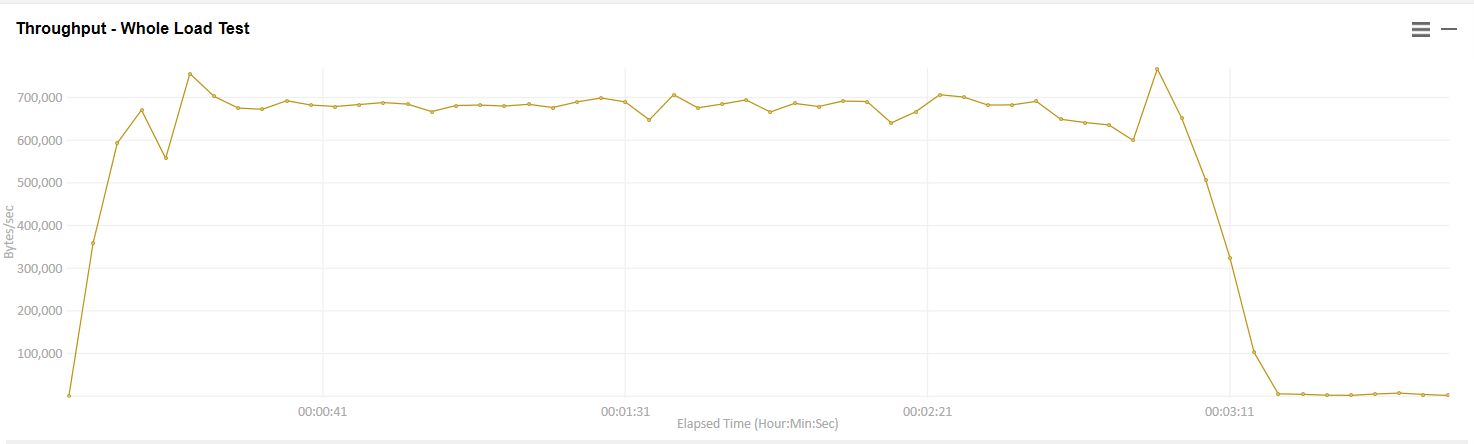
**c. 并发性**



可以看出平均的并发用户数量为3-4个。

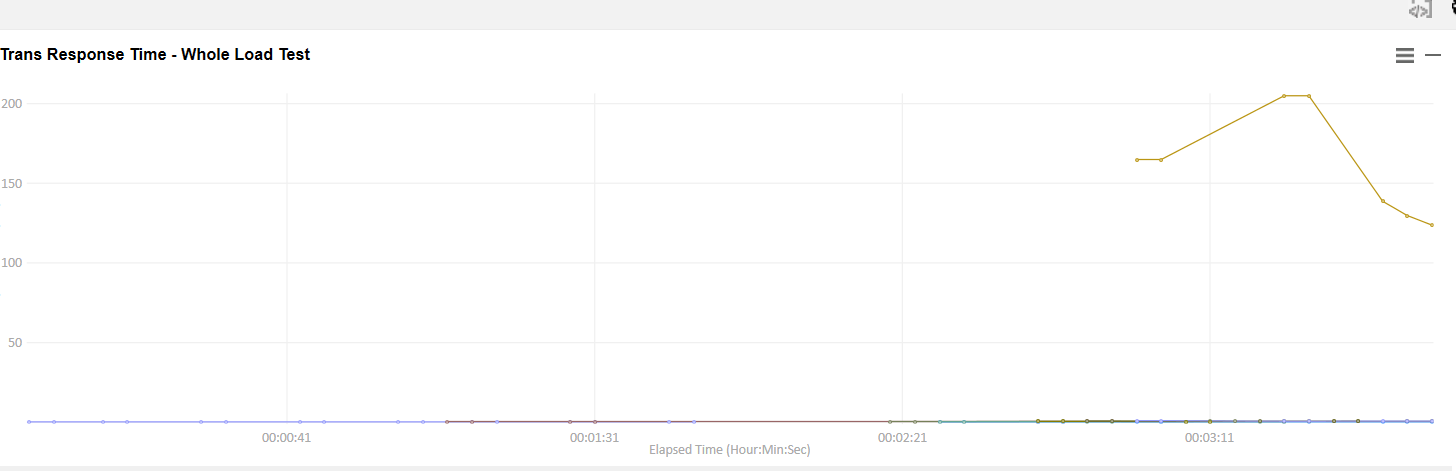
### 场景三测试结果

**a. 吞吐量**



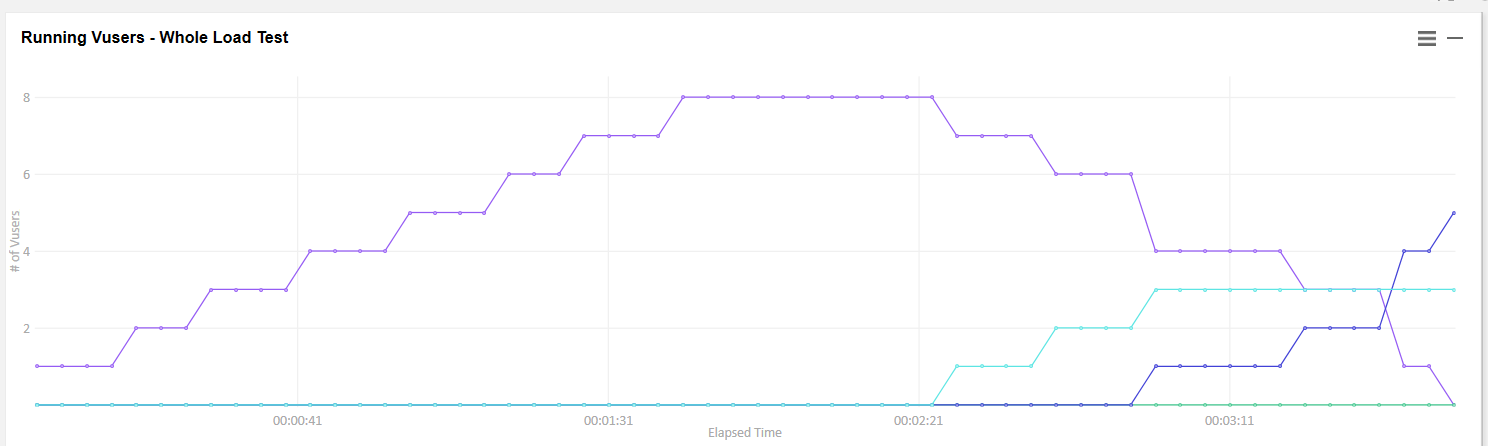
可以看出由于服务器的吞吐量在700,00Bbps左右就将近到达了峰值，即使增加了并发性也无法让吞吐量有很大的提升。

**b. 响应时间**



这张图表中主要信息为黄色的折线，代表Action\_Transaction的响应时间。相较于场景一，场景二的并发度更高，响应时间也更长，是场景一的平均响应时间的1.5倍。

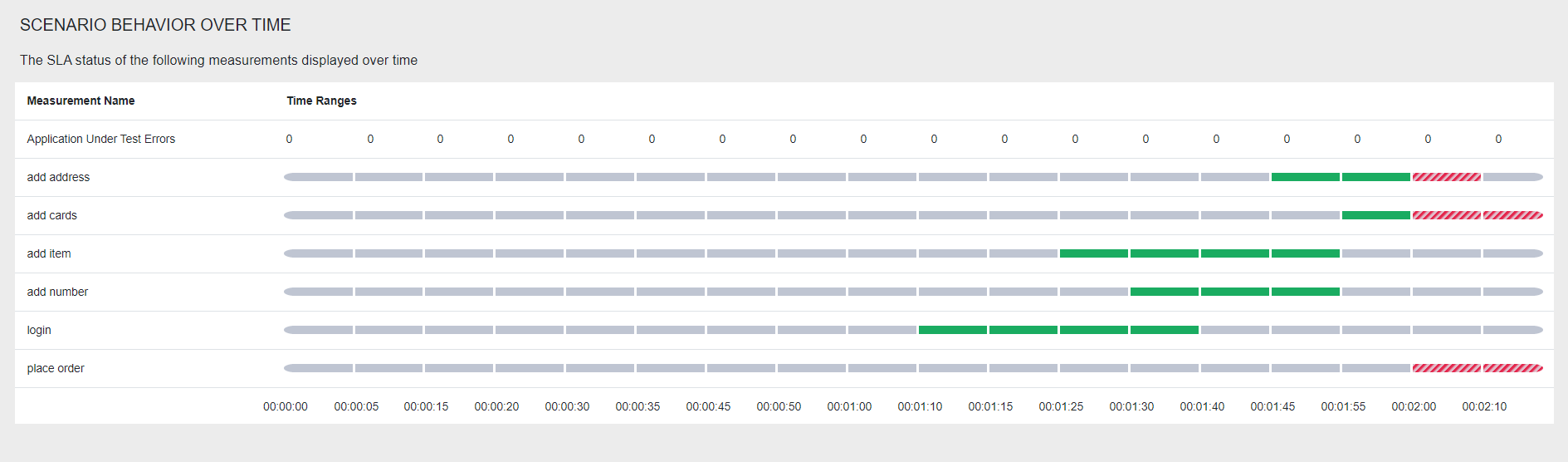
**c. 并发性**



图中的紫色折线代表在不同时间的并发用户数量，可以看出平均并发用户数量在5-6个左右。

## SLA服务水平协议定义

服务水平协议选择平均事务相应时间作为指标，选择login，add item add card,add address,place order作为待测事务，由于服务器资源的原因，选择3个vuser作为分界点，设置低负载下的相应时间为0.1秒 ，高负载下的相应时间为0.5秒。评估报告中的结论如下图

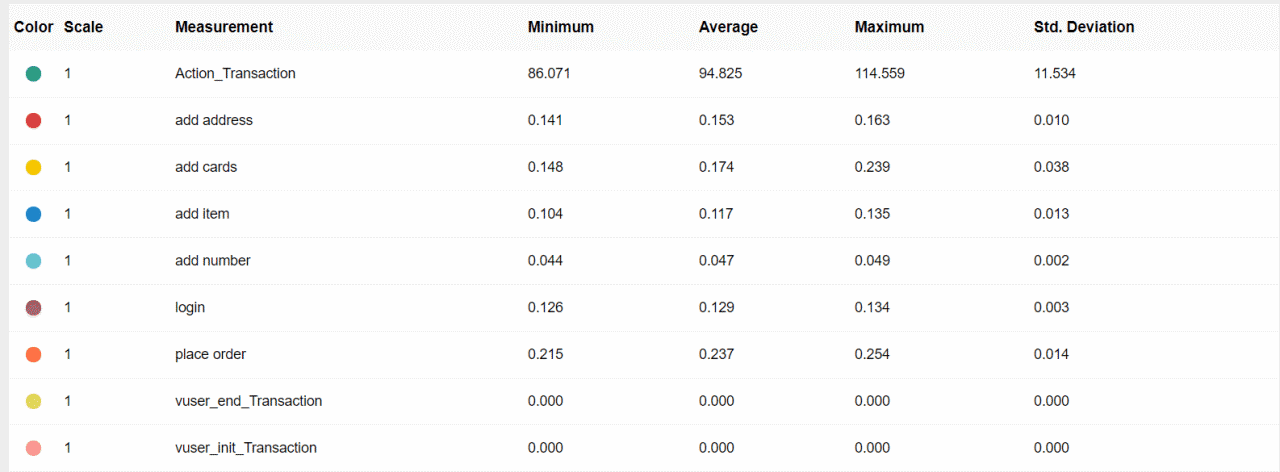


这里服务水平协议选择的0.1秒与0.5秒只是我们对于这个事务时间的一个估计，如果要是对某一个具体事务做定义的话，还是需要参考需求规约文档，使得服务水平可以满足测试文档上用户规定的需求

# 测试结论与建议

## 瓶颈分析

观察具体的事务响应时间，我们发现单个请求的transaction实际响应时间较短，而Action的响应时间很长，探究record生成的代码，我们发现该网页经常性的进行页面刷新而不是ajax动态刷新，这是性能瓶颈所在，说明应用的瓶颈在前端而不是后端，尤其是静态资源。



系统中有大量的静态资源的请求：



## 优化建议

1. 部署多个前端，负载均衡，以便减小单个前端的处理压力，降低响应时间
2. 静态资源、动态资源分离，静态资源时间上占很大的加载比例，宜单独处理
3. 扩大服务器资源，我们测试用的服务器是阿里云的学生机，资源只有1核2G，使用docker部署微服务架构后负载能力有限，基本平均并发在4个用户最为合适，宜扩大资源。