Lab4: 性能测试 测试计划

1. 被测对象

1.1 介绍

我们利用开源软件搭建了一个电商网站的Demo —— weavesocks，作为被测对象。该网站的主要功能如下：

- 注册、登录
- 浏览商品（主要是各种袜子）
- 加入购物车
- 填写地址、卡号，下单购买

1.2 部署

1.2.1 部署环境

为了满足该网站的基本运行需要，我们将其部署在一个云服务器上，服务器的基本参数如下：

参数	名称
操作系统	Ubuntu 18.04 LTS
CPU	4核
Memory	15G

1.2.2 部署方式

我们使用了 docker 容器部署被测网站。被测网站是微服务架构的，我们将每一个服务装载在一个容器内部，通过暴露容器端口的方式，进行容器间的相互通信。

网站的基本架构如图1所示，*Frontend* 作为静态资源和动态资源的响应者，是整个网站的入口，客户端通过请求 *Frontend* 来获取静态资源（图片、html、css、js……）和动态资源（xhr等）。各个组件之间的依赖关系如图所示。

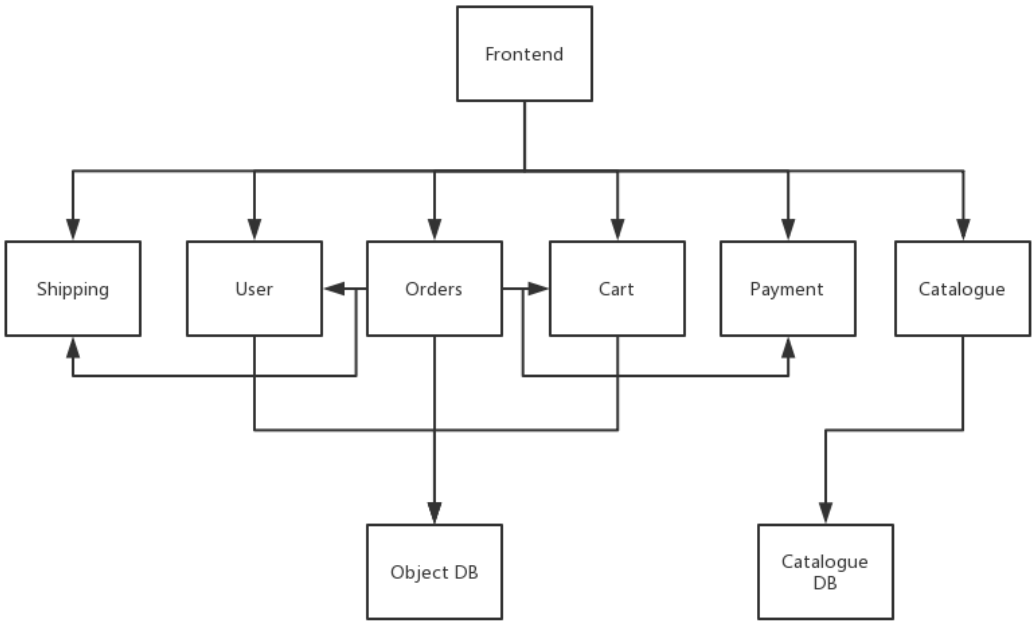

图1：被测网站基本架构

部署成功后，访问 weavesocks，将会看到 图2 的页面。

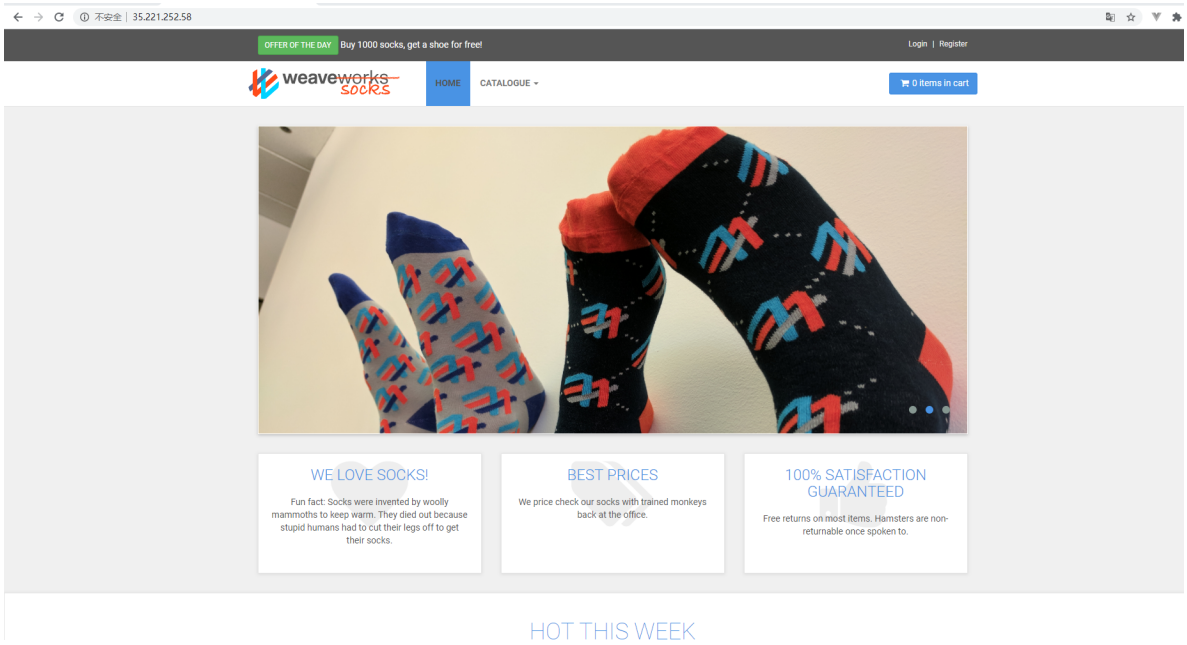

图2：网站主页

2. 测试流程

本节主要说明本次性能测试的几个关键部分，测试指标、测试工具及方法、测试用例设计及描述。下面将一一介绍这几个关键部分。

2.1 测试指标

本次性能测试主要关注 3 个性能指标，并发性、吞吐量、响应时间。

- 并发性

  并发性描述了系统在较短时间内可响应的请求数。并发性好的系统，在短时间内可以处理大量的请求，可以容忍大量的用户同时进行操作，相应的也会对系统的吞吐量造成一定的影响。并发性差的系统，在短时间内接收到大量的请求，就会造成请求的响应时间延长，甚至可能丢失请求，造成难以想象的后果。

- 吞吐量

  吞吐量关注的是每单位时间内，系统的流量大小，这个流量可以是以字节数为单位的网络流量，也可以是以事务数来衡量的事务吞吐量。

  在本次性能测试中，吞吐量更多的是关注 事务数/秒 ，简称 TPS。

- 响应时间

  响应时间，指从请求发出到请求恢复所消耗的时间。对于一个网站来说，响应时间可以从两方面来看待，一是从用户的角度，用户点击按钮，直到页面做出相应的响应，这中间的时间就是响应时间；另一是从网站系统的角度看待，系统收到请求，到系统处理完请求并响应，这段时间也是响应时间。

  在本次测试中，我们将更多的关注从用户的角度出发的响应时间，即从测试工具发出请求，直到测试工具收到请求之后的这段时间。

2.2 测试工具及方法

2.2.1 测试工具

本次性能测试主要使用 *Load Runner v12.60* 作为测试工具。*Load Runner* 主要包含三个工具，也是我们本次性能测试所需要使用到的工具：*Virtual User Generator*、*Controller*、*Analysis* 。三个工具在本次测试中的作用如下：

Virtual User Generator ： 主要用来生成测试脚本

Controller : 主要用来模拟测试场景，可以指定虚拟用户数、实时监控测试指标

Analysis：收集测试数据，并对测试数据进行绘图分析

2.2.2 测试方法

本次性能测试通过使用模拟用户执行场景的方式。

举例来说，比如**大量用户登录网站**这个场景，在本次性能测试中，可以按照以下测试流程方法进行测试：

1. 使用 *Virtual User Generator* 录制单个用户的登录过程，*Virtual User Generator* 会自动记录这一过程发出的请求（Http 协议），并生成一个脚本。根据测试用例场景，定义执行过程中的关键事务。针对该测试用例，比如用户登录，这一动作就可以作为一个 Transaction。
2. 使用 *Controller* 进行大量用户这一场景的模拟。在 *Virtual User Generator* 中定义的是每个用户的动作，在 *Controller* 中定义多个用户之间的交互，比如用户到达速率、同时运行用户数量等等。在该场景模拟过程中，*Controller* 会对执行的数据进行收集，包括在 *Virtual User Generator* 中定义的 Transaction、网络流量等。
3. 使用 *Analysis* 对 *Controller* 中生成的数据进行分析，根据场景定义和测试需求，可以绘制 TPS、Throughput 相关的图，也可以生成相应的总结报告。

2.3 测试用例设计与描述

2.3.1 测试用例设计

2.3.1.1 用例1：登录 + 主页浏览详情

模拟场景：

大量用户同时在线，并在网站的主页上进行了浏览。与此同时又有一群用户，登录，加载进入主页。

性能指标：

- Throughput 吞吐量
- Transaction Response Time：事务的响应时间，相当于Latency

用例设计:

- 用户数
    - Vuser从5逐渐增加，每次增加5，直到出现性能瓶颈
- 事务
    - 登录 + 浏览作为一整个事务
- 预期行为
    - 用户数小于 20 时，响应时间在 3s 以内；用户数介于 20 - 40 之间时，响应时间在 8s 以内；用户数大于 40时，响应时间在 20s 以内。
    - 网络流量吞吐量大于 200KB/s
- 参数化
    - 使用登录的用户名作为参数，在多个Vuser的情况下模拟多用户登录。

2.3.1.2 用例2：注册 + 登录

模拟场景

用户首先注册，之后登出后再进行登录。

性能指标

- Transaction Response Time：事务的响应时间，相当于Latency

用例设计

- 用户数
    - 因为该场景不需要高并发，所以 Vuser 数为 5。
- 参数化设计
    - 本测试可以以用户的用户名作为参数，在多个Vuser的情况下模拟多用户进行注册。
- 预期行为
    - 用户数小于5时，登录响应时间在 500ms 以下，注册响应时间在 2s 以下。

2.3.1.3 用例3：登录 + 按类别查找商品

模拟场景

用户首先登录，之后从目录页浏览商品详情界面，并可以对浏览商品做适当筛选。

性能指标

- Throughput 吞吐量
- Transaction Response Time：事务的响应时间，相当于Latency

用例设计

- 用户数
    - Vuser从5逐渐增加，每次增加5，直到出现性能瓶颈。
- 参数化设计：
    - 本测试可以以浏览商品的种类作为参数，在多个Vuser的情况下模拟多用户访问目录页。
- 预期行为：
    - 用户数小于5时，响应时间在20s以下；用户数小于10大于5时，响应时间在50s以下
    - 网络流量吞吐量大于200KB/s

2.3.1.4 用例4：登录 + 加入购物车 + 下单

模拟场景：

大量用户登录网站之后，点击想要购买的商品，加入购物车。添加收货地址、付款方式后，下单。

性能指标：

- TPS Transaction Per Second，关注单位时间内的事务数
- Transaction Response Time：事务的响应时间，主要考察下单的响应时间

用例设计：

- 用户数
    - vuser 每 15 s增加一个，直到 50 个 vuser 停止增加。每个 vuser 运行 5 分钟，运行过程中不断将商品添加购物车，添加下单，查看订单。运行一段时间后，vuser每隔10s退出。
- 事务
    - 用例4 的动作比较多，因此可以定义不同的事务。
        - 登录 —— tx_login ，登录还是需要作为一个事务，可以评价用户在下订单时对登录的延时影响。
        - 加入购物车 —— tx_add_cart，从用户的角度加入购物车的响应时间是比较严格的，响应时间长了，用户会怀疑没有添加到购物车，就会重复操作，对用户造成不良印象。
        - 添加收货地址 —— tx_add_address
        - 添加付款方式 —— tx_add_card
        - 下单 —— tx_order，下单是用例4最重要的一个事务，也是用例4最关注的一个事务。这个事务时被测网站中最复杂的一个事务之一，需要更新较多状态，微服务之间的相互通信也比较复杂。
- 预期行为
    - 用户数小于 20 时，每个事务的响应时间在 0.5s 以内；用户数介于 20 - 40 之间时，每个事务的响应时间在2s 以内；用户数大于40时，响应时间在 8s 以内。
    - 用户数小于 20 时，TPS 大于 1；用户数介于 20 - 40 之间时，TPS 大于2；用户数大于40时，TPS大于1。
- 关联设计
    - 用户登录后的登录凭证（Cookie）和用户关联，在下单、购物车等请求中都需要使用到这个登陆凭证。
- 参数化设计
    - 用户名可以进行参数化，加入购物车的商品也可以进行参数化。