Handson3

Question 1

Please try to deploy the online shop application according to the dependency graph and the table listed above. You should hand in an executable shell script that contains all your deployment steps and can be used later to deploy the website conveniently.

run.sh中的文件编写如下：

```
sudo docker network create --subnet 10.0.0.1/16 hy_handson3
sudo docker run -d --network hy_handson3 --ip 10.0.0.2 mongo:3.4
sudo docker run -d --network hy_handson3 --ip 10.0.0.3 dplsming/sockshop-cataloguedb:0.1
sudo docker run -d --network hy_handson3 --ip 10.0.0.4 --add-host catalogue-db:10.0.0.3  weaveworksdemos/catalogue:0.3.5
sudo docker run -d --network hy_handson3 --ip 10.0.0.6 --add-host user-db:10.0.0.2  -e MONGO_HOST=user-db:27017  weaveworksdemos/user:0.4.4
sudo docker run -d --network hy_handson3 --ip 10.0.0.7 --add-host carts-db:10.0.0.2 weaveworksdemos/carts:0.4.8
sudo docker run -d --network hy_handson3 --ip 10.0.0.8  weaveworksdemos/shipping:0.4.8
sudo docker run -d --network hy_handson3 --ip 10.0.0.9  weaveworksdemos/payment:0.4.3
sudo docker run -d --network hy_handson3 --ip 10.0.0.5  --add-host orders-db:10.0.0.2  --add-host user:10.0.0.6 --add-host carts:10.0.0.7 --add-host
    payment:10.0.0.9 --add-host shipping:10.0.0.8  weaveworksdemos/orders:0.4.7
sudo docker run -d --network hy_handson3 --ip 10.0.0.10 --add-host shipping:10.0.0.8 --add-host user:10.0.0.6 --add-host orders:10.0.0.5 --add-host
    carts:10.0.0.7 --add-host payment:10.0.0.9 --add-host catalogue:10.0.0.4 -p 8079:8079 dplsming/sockshop-frontend:0.1
```

Question 2

Please try the command by yourself and draw some curves to show the trend of RT and throughput under different loads.

配置环境：

```
sudo snap install docker
sudo apt-get update
sudo apt-get install python2.7
sudo apt update
```

建立ssh连接：

```
ssh -i why.pem.txt ubuntu@ec2-3-226-125-31.compute-1.amazonaws.com
```

Scp传输文件：

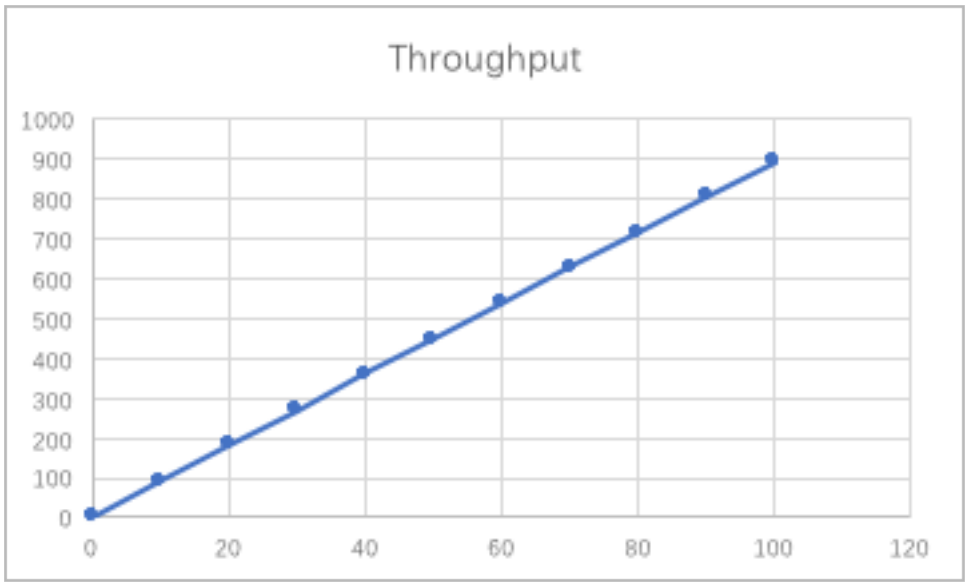
```
scp -i why.pem.txt ubuntu@ec2-3-226-125-31.compute-1.amazonaws.com:~/
hy_handson3/out.csv out.csv
```

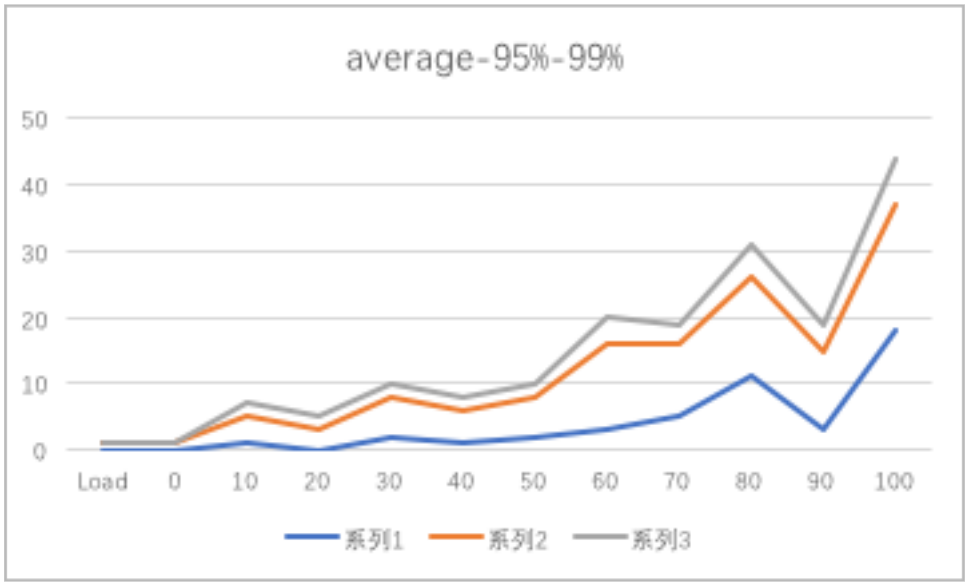
运行native的测试命令：

```
sudo ./load-test.py http://10.0.0.1:8079 case1 > out.csv
```

得到的csv文件：

得到的两张表格





Question 3

Please describe and explain the phenomenon you find in the curve you have just drawn.

图中的折线说明latency随着load的提高逐渐上升，且有约增长越快的趋势，这样服务器的性能压力就比较大。根据hint提示需要进行nginx优化。

Question 4

Try to optimize the static page accessing response time using Nginx. Please briefly describe what you have done and draw the curves to show the effect of your optimization.

- 在实例中右键实例 ➔ 联网 ➔ 更改安全组，选定wizard1或2
- 左侧选项栏选择安全组，点击刚才选定的安全组，在下侧入站的端口号 ->编辑 ➔ 修改为8079-10000
- Ip在实例一页的最下面

```
IPv4 公有 IP
3.226.125.31
```

配置nginx

```
sudo apt install nginx-full
sudo ningx
cd /etc/ningx
cd sites-enabled
touch sock.conf
```

Vim 写入代理命令

```
server{
    listen 8080;
    location ~ \.(html)$ {
        root  /home/ubuntu/hy_handson3/frontend/public;
    }
    location /(js|items|img|fonts|css|/ {
        root  /home/ubuntu/hy_handson3/frontend/public;
    }
    location / {
        proxy_pass http://10.0.0.1:8079;
    }
}
```

调用命令

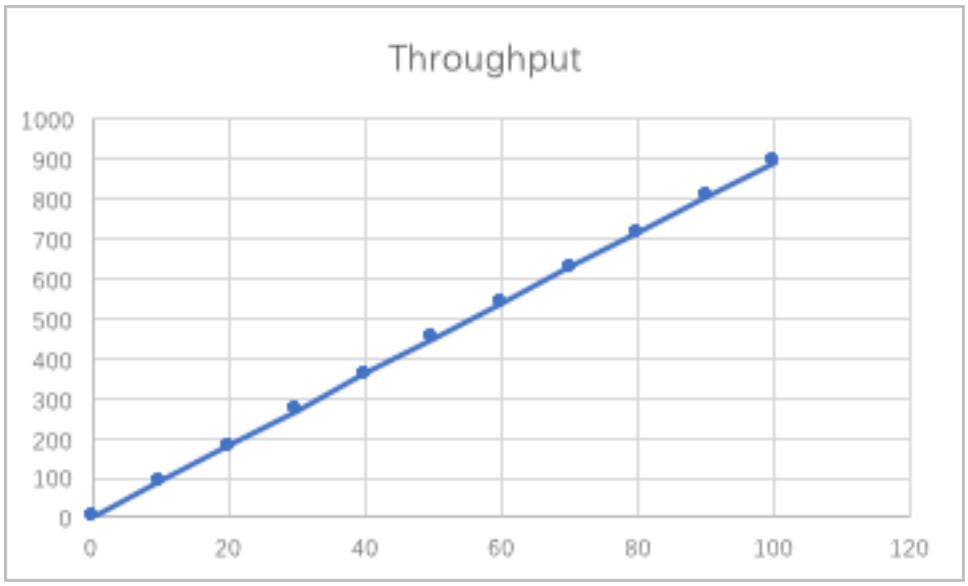
```
ningx -s reload
```

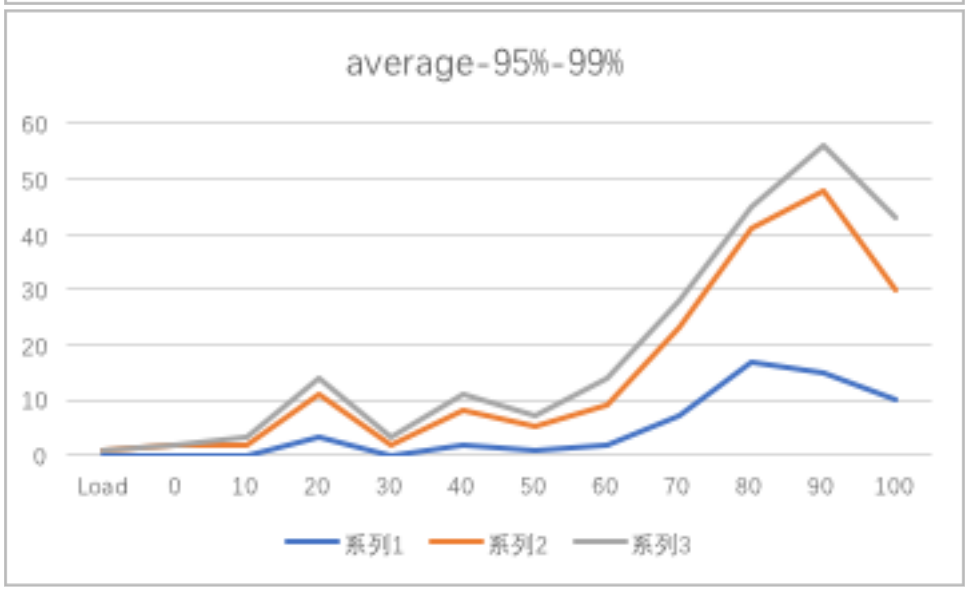
本地调用8080端口，发现代理成功

运行native的测试命令：

```
sudo ./load-test.py http://10.0.0.1:8080 case1 > out_nginx.csv &
```

以下是结果：





看出吞吐量没有明显的变化。
但是在load较少时，反应时间优化不明显。
随着load增大，其反应时间降低的趋势更加明显，优化效果开始体现，尤其是当load达到90时，latency随着load的增大有了下降趋势，有一定的优化效果。

Question 5

Try to optimize the performance of the application under this workload with load-balancing. Please briefly describe your modification. You also need to re-run load-test.sh and generate figures to show how well your optimization works.
在hy_handson文件夹下

```
mkdir conf.d
cd conf.d
sudo vim default.conf
```

在该文件中写入

```
upstream sock_server{
    server 10.0.0.11:8079;
    server 10.0.0.12:8079;
    server 10.0.0.13:8079;
}
server{
    listen 8081;
    location ~ \.(html)$ {
        root  /home/ubuntu/hy_handson3/frontend/public;
    }
    location /(js|items|img|fonts|css)/ {
        root  /home/ubuntu/hy_handson3/frontend/public;
    }
    location / {
        proxy_pass http://sock_server;
    }
}
```

Docker跑这三个服务器

```
sudo docker run -d --network hy_handson3 --ip 10.0.0.11 --add-host shipping:
10.0.0.8 --add-host user:10.0.0.6 --add-host orders:10.0.0.5 --add-host carts:
10.0.0.7 --add-host payment:10.0.0.9 --add-host catalogue:10.0.0.4 dplsming/
sockshop-frontend:0.1

sudo docker run -d --network hy_handson3 --ip 10.0.0.12 --add-host shipping:
10.0.0.8 --add-host user:10.0.0.6 --add-host orders:10.0.0.5 --add-host carts:
10.0.0.7 --add-host payment:10.0.0.9 --add-host catalogue:10.0.0.4 dplsming/
sockshop-frontend:0.1

sudo docker run -d --network hy_handson3 --ip 10.0.0.13 --add-host shipping:
10.0.0.8 --add-host user:10.0.0.6 --add-host orders:10.0.0.5 --add-host carts:
10.0.0.7 --add-host payment:10.0.0.9 --add-host catalogue:10.0.0.4 dplsming/
sockshop-frontend:0.1
```

下载ningx的docker

```
```

```
sudo docker run -d --network hy_handson3 --ip 10.0.0.14 -v /home/ubuntu/
hy_handson3/conf.d:/etc/nginx/conf.d  -v  /home/ubuntu/hy_handson3/frontend/
public:/usr/public -p 8081:8081 nginx:latest
```

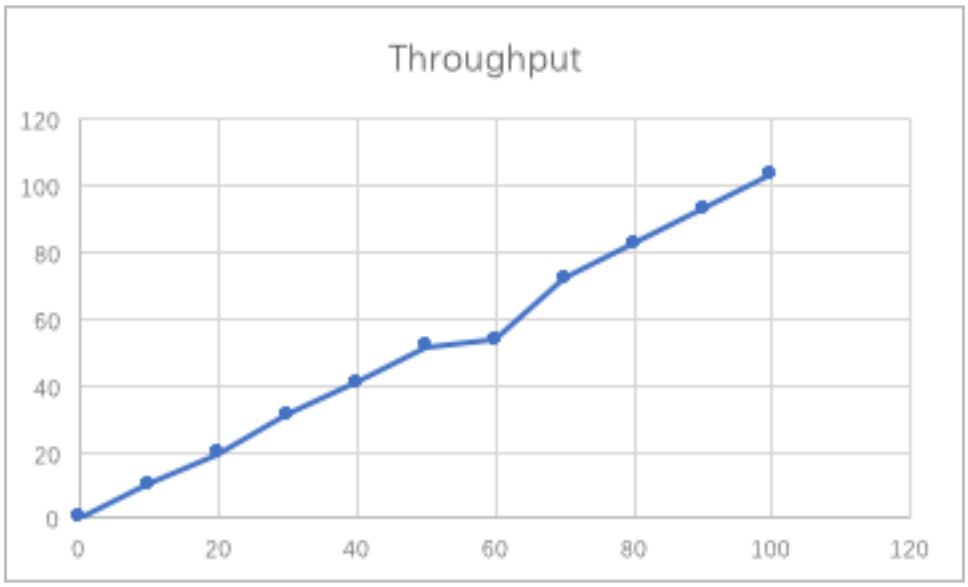
测试8081端口可以访问，接下来进入测试

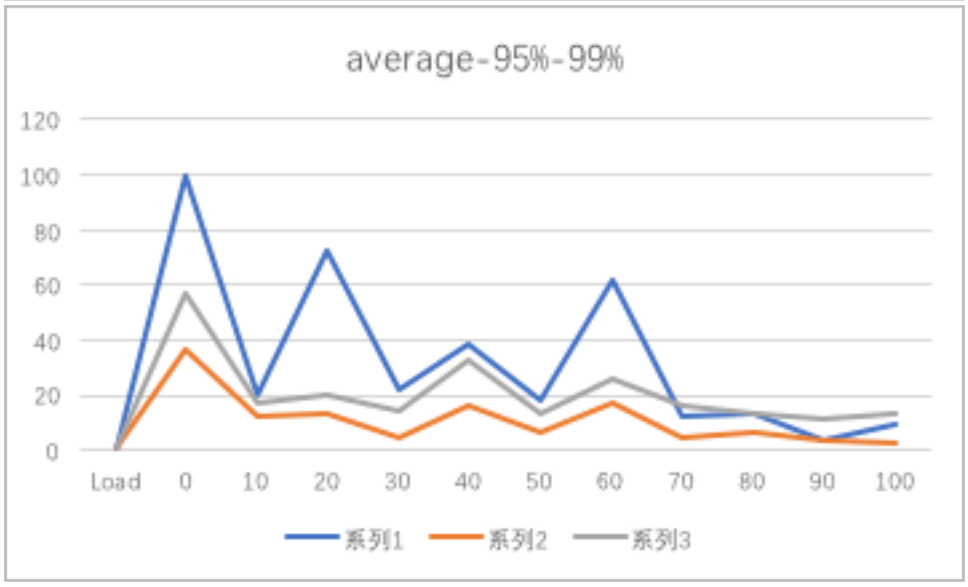
```
sudo ./load-test.py http://10.0.0.1:8081 case2 > out_balance.csv &
```

下拉文件

```
scp -i why.pem.txt ubuntu@ec2-3-226-125-31.compute-1.amazonaws.com:~/
hy_handson3/out_balance.csv out_balance.csv
```

得到图：





负载均衡的确大幅度降低了在load增加的时候的latency。

当随着load的增加，负载均衡使latency降低很多。

Question 6

Try your optimization under the situation where all frontend containers are enforced with a CPU limitation using –cpus 0.5 when created. Is your optimization more effective or less effective? Why?

当cpus=0.5时，

Docker跑这三个服务器

```
sudo docker run -d --network hy_handson3 --ip 10.0.0.11 --add-host shipping:
10.0.0.8 --add-host user:10.0.0.6 --add-host orders:10.0.0.5 --add-host carts:
10.0.0.7 --add-host payment:10.0.0.9 --add-host catalogue:10.0.0.4 --cpus 0.5
dplsming/sockshop-frontend:0.1

sudo docker run -d --network hy_handson3 --ip 10.0.0.12 --add-host shipping:
10.0.0.8 --add-host user:10.0.0.6 --add-host orders:10.0.0.5 --add-host carts:
10.0.0.7 --add-host payment:10.0.0.9 --add-host catalogue:10.0.0.4 --cpus
0.5dplsming/sockshop-frontend:0.1

sudo docker run -d --network hy_handson3 --ip 10.0.0.13 --add-host shipping:
10.0.0.8 --add-host user:10.0.0.6 --add-host orders:10.0.0.5 --add-host carts:
10.0.0.7 --add-host payment:10.0.0.9 --add-host catalogue:10.0.0.4 --cpus 0.5
dplsming/sockshop-frontend:0.1
```

下载ningx的docker

```
sudo docker run -d --network hy_handson3 --ip 10.0.0.14 -v /home/ubuntu/
hy_handson3/conf.d:/etc/nginx/conf.d  -v  /home/ubuntu/hy_handson3/frontend/
public:/usr/public -p 8081:8081 nginx:latest
```

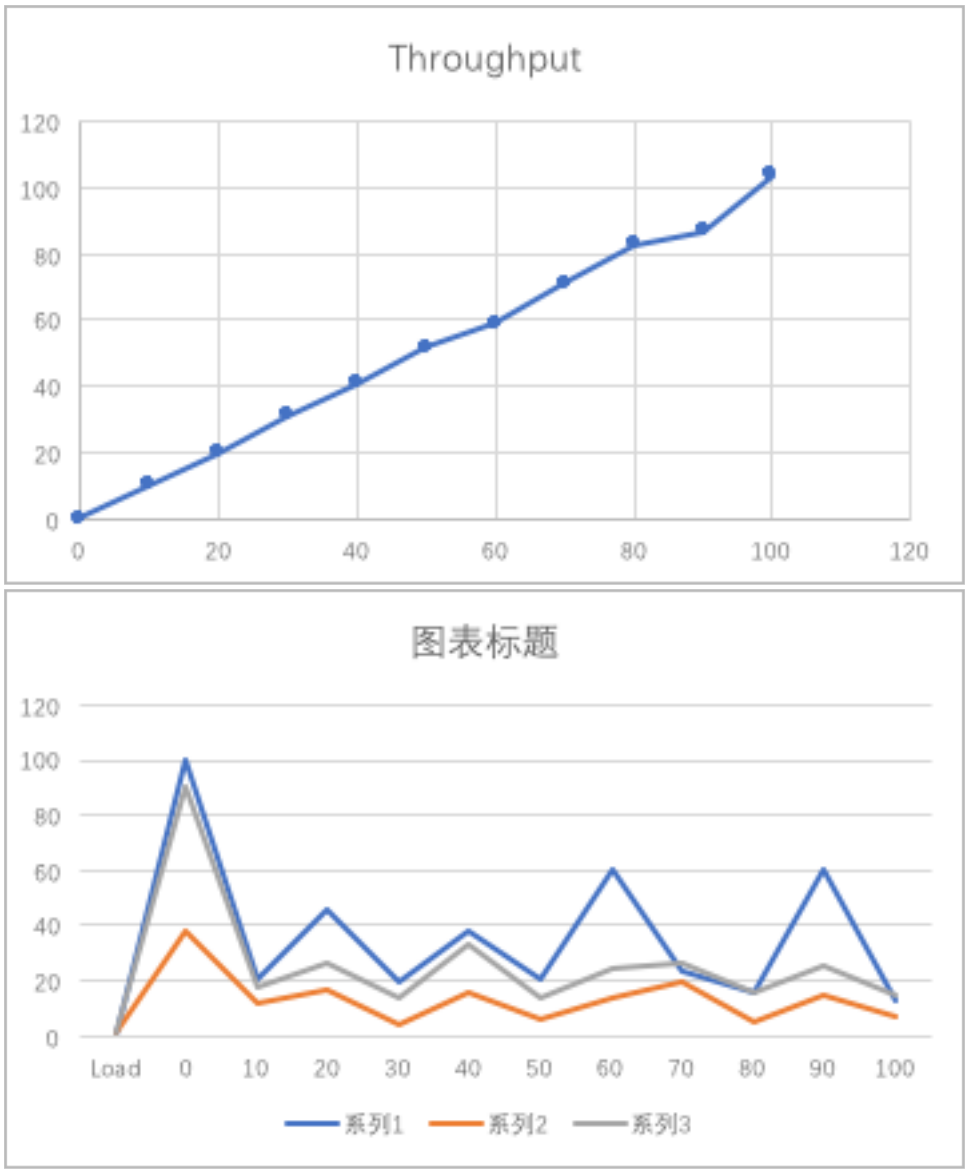
测试8081端口可以访问，接下来进入测试

```
sudo ./load-test.py http://10.0.0.1:8081 case2 > out_balance.csv &
```

下拉文件

```
scp -i why.pem.txt ubuntu@ec2-3-226-125-31.compute-1.amazonaws.com:~/
hy_handson3/out_balance_cpu.csv out_balance_cpu.csv
```

结果如下



由图表可知随着load增大平均的延迟降低没有之前明显。且在高并发

原因是cpu的负载在load达到约80-90的情况下，也许已经达到饱和。

因此cpu 的限制对load比较低的情况影响不大，但在load比较高的情况下也的确会形成bottleneck，形成饱和使得latency提高。