山东大学 计算机科学与技术 学院

云计算技术 课程实验报告

学号：202200130048	姓名： 陈静雯	班级： 6
实验题目：负载均衡		
实验学时：2	实验日期： 5.21	

实验目的：使用 Docker 部署 Nginx 和 Tomcat 实现负载均衡
具体包括：
1、要求部署一台 Nginx 和四台 Tomcat 服务器
2、Ngnix 需要实现四种策略：
1）轮询法；
2）随机法；
3）加权轮询法，权重为 4，2，1，5；
4）IP Hash。
3、需要设计一个前端网页，通过该网页来显示负载均衡的效果。例如，每次访问页面时，页面可以显示当前处理请求信息，以体现 Nginx 的负载均衡策略。

硬件环境：
PC

软件环境：
Linux 或 Windows

相关知识：

一、Docker 基础
1. 镜像（Image）：只读模板，用于创建容器（如 tomcat:9.0）。
2. 容器（Container）：镜像的运行实例，提供隔离的进程环境。
3. 仓库（Registry）：存储镜像的平台（如 Docker Hub、阿里云镜像仓库）。
4. Dockerfile：定义镜像构建步骤的脚本文件。

二、Docker Compose
1. 服务（Service）：一个容器化的应用组件（如 Nginx、Tomcat）。
2. 编排文件（docker-compose.yml）：定义多容器应用的配置（网络、卷、依赖关系）。

三、Nginx 基础
1. 反向代理：将客户端请求转发到后端服务器。
2. 负载均衡：通过 upstream 模块分配请求到多个后端节点。
3. 静态文件服务：直接托管 HTML/CSS/JS 文件。

四、Tomcat 基础
1. Web 应用部署：将 WAR 包或静态文件放入 webapps/ROOT 目录。
2. 默认端口：8080（可通过 server.xml 修改）。

实验步骤：

1、要求部署一台 Nginx 和四台 Tomcat 服务器
2、Ngnix 需要实现四种策略：

1）轮询法；
2）随机法；
3）加权轮询法，权重为 4，2，1，5；
4）IP Hash。
3、需要设计一个前端网页，通过该网页来显示负载均衡的效果。例如，每次访问页面时，页面可以显示当前处理请求信息，以体现 Nginx 的负载均衡策略。

实验内容：

1. 准备工作

确保已安装 Docker 和 Docker Compose。创建项目目录结构如下

```
nginx-tomcat-lb/
├── docker-compose.yml
├── nginx/
│   └── nginx.conf
└── tomcat/
    ├── tomcat1/
    │   └── index.html
    ├── tomcat2/
    │   └── index.html
    ├── tomcat3/
    │   └── index.html
    └── tomcat4/
        └── index.html
```

2. 配置 Tomcat 实例

为每个 Tomcat 实例创建 **index.html** 文件，内容如下（以 tomcat1 为例）

nginx.conf　　index.html

```
<html>
<body>
  <h1>Tomcat Instance 1</h1>
  <p>Port: 8080</p>
</body>
</html>
```

3. Docker Compose 配置

编写 docker-compose.yml 文件：

nginx.conf | index.html | index.html | docker-compose.yml

```yaml
version: '3'

services:
  # Tomcat 服务
  tomcat1:
    image: tomcat:9.0
    volumes:
      - ./tomcat/tomcat1:/usr/local/tomcat/webapps/ROOT
    networks:
      - app_network

  tomcat2:
    image: tomcat:9.0
    volumes:
      - ./tomcat/tomcat2:/usr/local/tomcat/webapps/ROOT
    networks:
      - app_network

  tomcat3:
    image: tomcat:9.0
    volumes:
      - ./tomcat/tomcat3:/usr/local/tomcat/webapps/ROOT
    networks:
      - app_network

  tomcat4:
    image: tomcat:9.0
    volumes:
      - ./tomcat/tomcat4:/usr/local/tomcat/webapps/ROOT
```

4. 配置 Nginx 负载均衡策略

编辑 nginx/nginx.conf 文件：

nginx.conf × index.html index.

```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    access_log off;
    error_log /dev/null;

    # 定义四个不同的upstream策略
    upstream backend_round_robin {
        server tomcat1:8080;
        server tomcat2:8080;
        server tomcat3:8080;
        server tomcat4:8080;
    }

    upstream backend_weighted {
        server tomcat1:8080 weight=4;
        server tomcat2:8080 weight=2;
        server tomcat3:8080 weight=1;
        server tomcat4:8080 weight=5;
    }

    upstream backend_ip_hash {
        ip_hash;
        server tomcat1:8080;
        server tomcat2:8080;
```

nginx.conf × index.html

```
            end
        }
    }

    server {
        listen 80;

        # 轮询策略
        location /round_robin {
            proxy_pass http://backend_round_robin;
        }

        # 加权轮询
        location /weighted {
            proxy_pass http://backend_weighted;
        }

        # IP Hash
        location /ip_hash {
            proxy_pass http://backend_ip_hash;
        }

        # 随机策略
        location /random {
            proxy_pass http://backend_random;
        }
    }
}
```

5. 启动服务

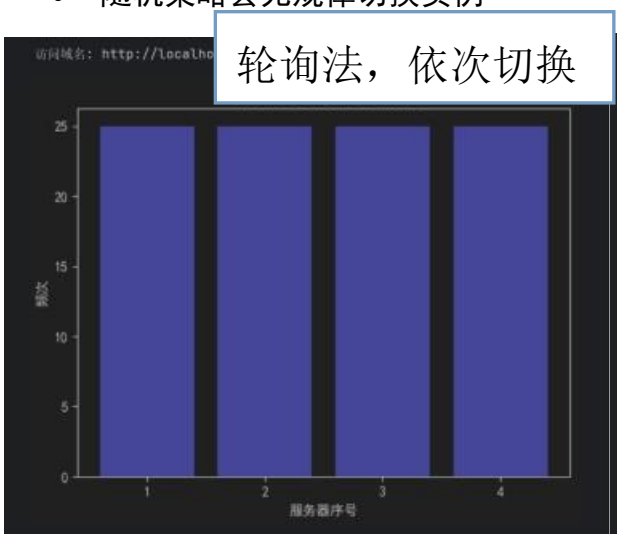
```
orange@orange-VMware-Virtual-Platform:~/nginx-tomcat-lb$ docker-compose up -d
[+] Running 0/5
 ⠿ nginx Pulling
 ⠿ tomcat2 Pulling
 ⠿ tomcat3 Pulling
 ⠿ tomcat4 Pulling
 ⠿ tomcat1 Pulling
```

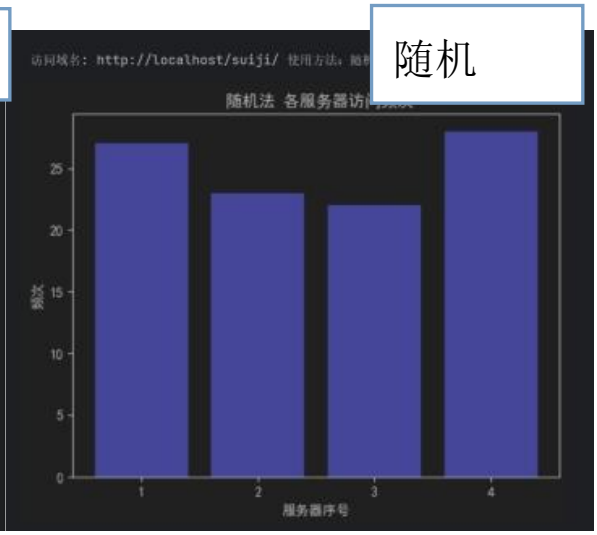
6. 使用 python requests 包模拟连续访问四种策略对应域名各 100 次。

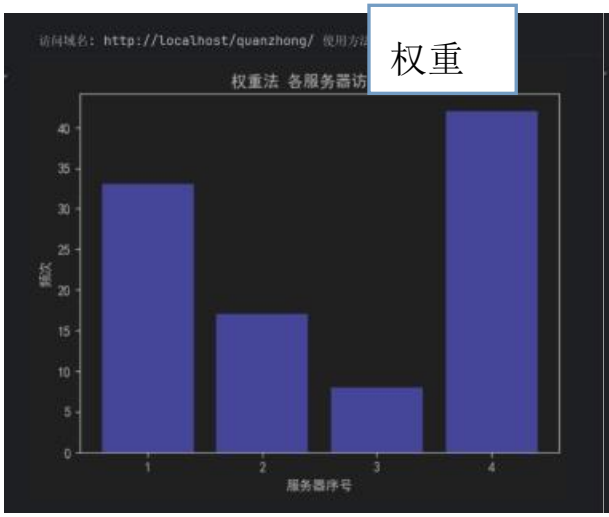
实验结果如下：

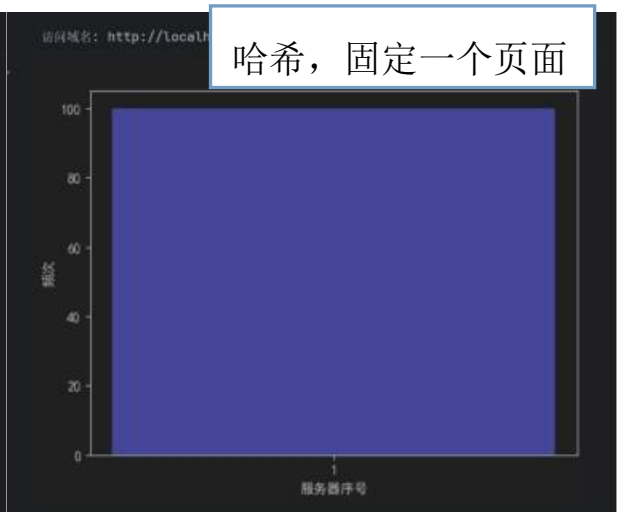
- 轮询策略会依次切换不同实例
- 加权轮询根据权重分布请求
- IP Hash 会保持同一客户端的访问固定

- 随机策略会无规律切换实例

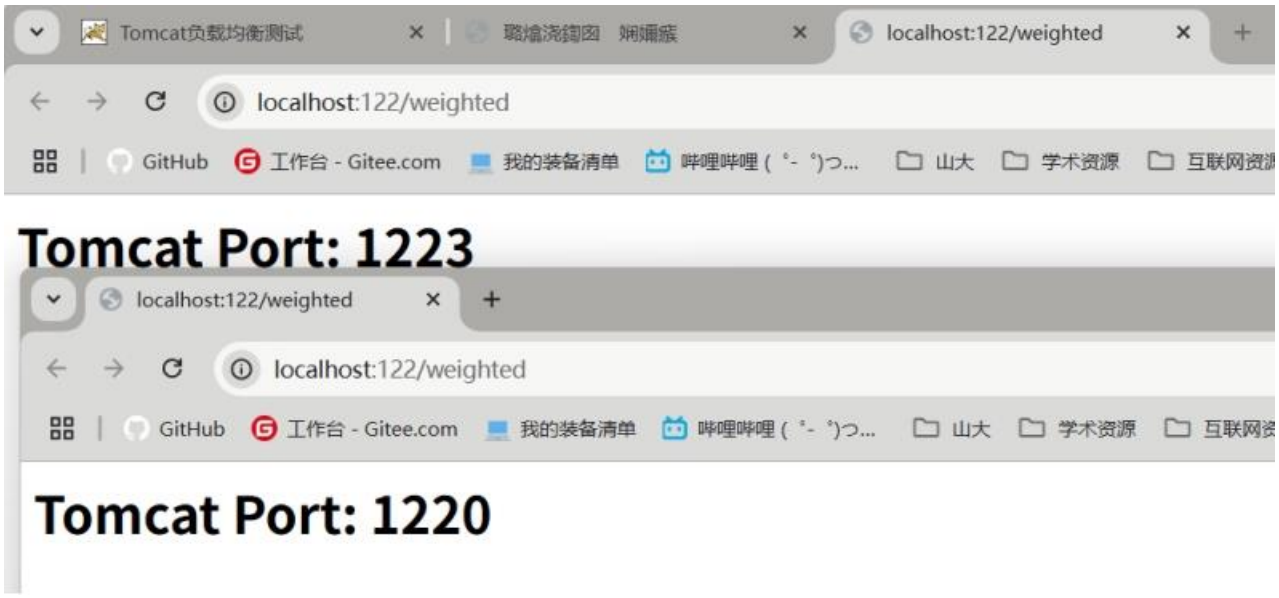








比如按权重中访问页面如下，



结论分析与体会：

通过本次实验，成功使用 **Docker** 和 **Docker Compose** 部署了 **1 台 Nginx** 和 **4 台 Tomcat** 服务器，并实现了以下负载均衡策略：

- 轮询（Round Robin）：请求按顺序分配到各 Tomcat 实例。
- 加权轮询（Weighted Round Robin）：根据权重（4, 2, 1, 5）分配请求比例。
- IP 哈希（IP Hash）：固定客户端 IP 与后端服务器的映射。
- 随机（Random）：无规则随机选择后端节点。

通过前端页面验证，各策略均能正确体现预期的负载分配效果。

本次实验不仅巩固了容器化技术和负载均衡的理论知识，更通过实践培养了系统性调试和复杂问题解决的能力。负载均衡作为分布式系统的核心组件，其灵活配置与优化对高并发场景下的稳定性至关重要。未来将进一步探索 Kubernetes 等进阶编排工具，构建更健壮的云原生架构。