

**2019** 级

《物联网数据存储与管理》课程

**实 验 报 告**

**姓 名 黄欣宇**

**学 号 U201915014**

**班 号 计算机1905班**

**日 期 2022.04.14**

**目 录**

[一、实验目的 1](#_Toc509412095)

[二、实验背景 1](#_Toc509412096)

[三、实验环境 1](#_Toc509412097)

[四、实验内容 1](#_Toc509412098)

[4.1 对象存储技术实践 1](#_Toc509412099)

[4.2 对象存储性能分析 1](#_Toc509412100)

[五、实验过程 1](#_Toc509412101)

[六、实验总结 1](#_Toc509412102)

[参考文献 1](#_Toc509412103)

# 一、实验目的

1. 熟悉对象存储技术，代表性系统及其特性；

2. 实践对象存储系统，部署实验环境，进行初步测试；

3. 基于对象存储系统，分析性能问题，架设应用实践。

# 实验背景

# 三、实验环境

本次实验采用的是Vmware Ubuntu18.04LTS本地化部署。

配置如下：

|  |  |
| --- | --- |
| 处理器 | Intel(R) Core(TM) i7-9750H CPU |
| 机带 RAM | 16.0 GB (15.9 GB 可用) |
| 系统类型 | 64 位操作系统, 基于 x64 的处理器 |

具体环境如下（服务端&客户端）：

|  |  |
| --- | --- |
| 软件名 | 版本号 |
| Gcc | 7.5.0 |
| Python | 3.6 |
| Python-swiftclient | 3.13.1 |
| Openstack-swift | x.x.x |
| Swift-bench | x.x.x |

# 实验内容

实验的主要内容为，在Linux服务器上安装Docker，在Docker上配置并部署单节点的Openstack Swift，部署swift-bench性能测试软件。在本地部署python-swiftclient功能测试客户端，借用了往届学长的一个基于Flask的Web客户端。

## 4.1 对象存储技术实践

(1)在Ubuntu服务器上安装Docker，在Docker上配置并部署一个单节Openstack Swift。

(2)在本地上部署python-swiftclient和CloudBerry两个分别是命令行和图形化的客户端，进行功能性的测试。

(3)使用往届学长基于Flask，使用Python语言调用python-swiftclient提供的API开发的一个轻量级的Web客户端，进行测试。

## 4.2 对象存储性能分析

使用swift-bench配合脚本进行测试。

# 五、实验过程

5.1.1服务端部署Openstack-swift

首先使用curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyun进行一键化安装docker。



图5.1 docker安装成功

如图5.1所示，当前docker版本为20.10.14, build a224086

然后按照实验中所给的openstack-swift容器开盒即用版地址，clone仓库到本地，依此执行以下指令进行部署单节点。

- docker build -t openstack-swift-docker .

- docker run -v /srv --name SWIFT\_DATA busybox

- docker run -d --name openstack-swift -p 12345:8080 --volumes-from SWIFT\_DATA -t openstack-swift-docker

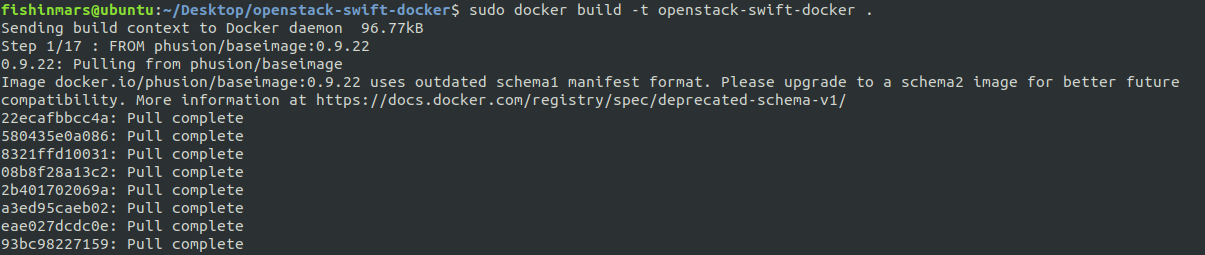


图5.2 .

图5.3

IMG_256

图5.4

然后运行docker ps，可以看到已经成功部署的节点

图5.5

5.1.2客户端部署python-swiftclient以及往届学长Web端测试

使用python3自带的pip指令即可完成安装，pip3 install python-swiftclient

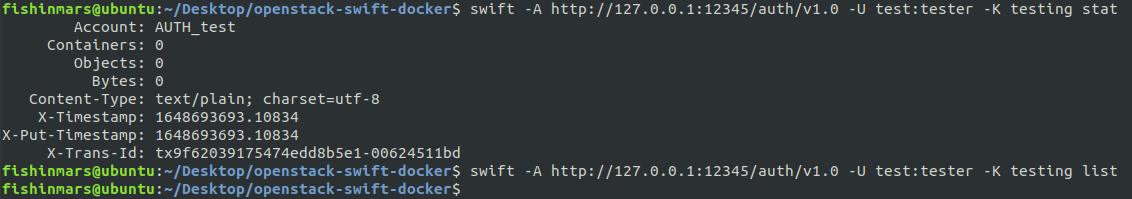
安装完成后可以使用swift -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing stat可以测试成功部署后的节点，通过图5.6可以看到已经成功部署后的服务器stat。  


图5.6

通过clone学长的库，可以获取一个轻量级的Web客户端，这个客户端基于社区内python-swiftclient提供的API，使用Python语言，基于Flask框架作为后端，在Bootstrap 4规范上使用Jinjia模板作为HTML前端的控制，完成一个轻量级的Web端的Swift客户端，支持包括CURD、生成临时下载链接在内的功能。启动后即可使用该Web端进行管理。在填写了相关内容后点击Go按钮即可进入管理界面。

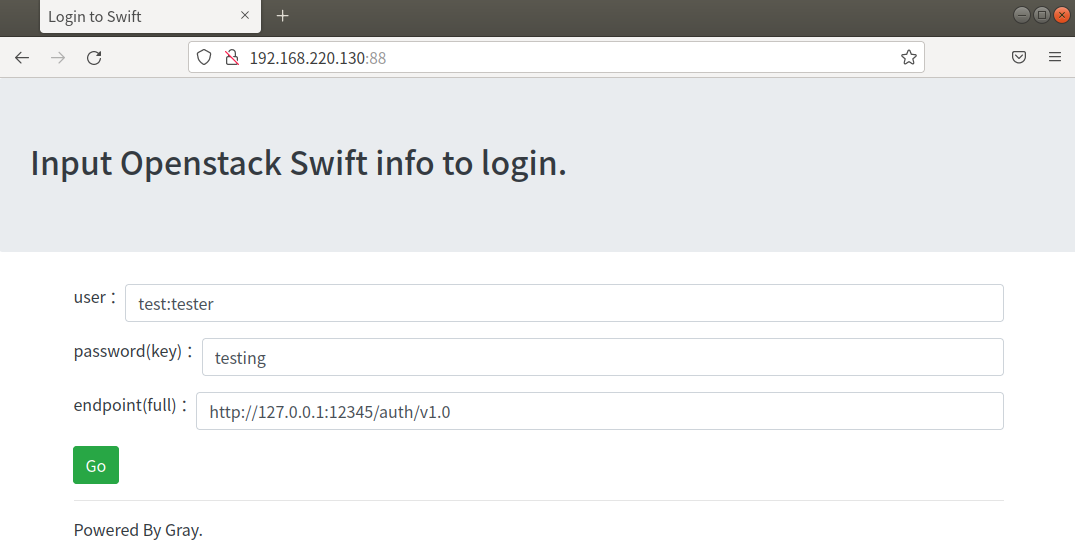


图5.7

登陆之后，会进入到默认的Container上，上方是一个上传文件的界面块，下方是一个卡片列表，遍历展示这个Container内部的所有文件，展示包括其名字，hash值，修改时间，文件大小，文件类型。卡片上有三个按钮，分别对应下载、获得临时下载链接、删除三个功能。

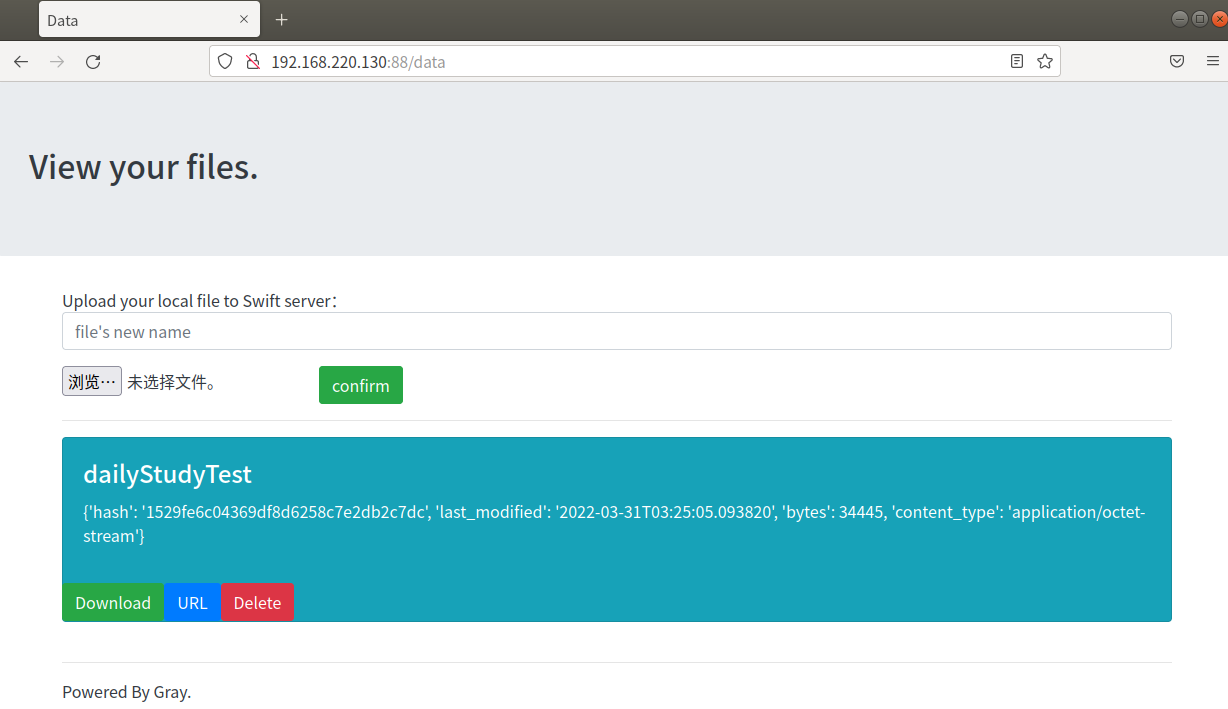


图5.8

接下来对以下功能进行测试，首先是上传文件选取gitPull.bat进行上传，重命名test1后可以看到成功上传。

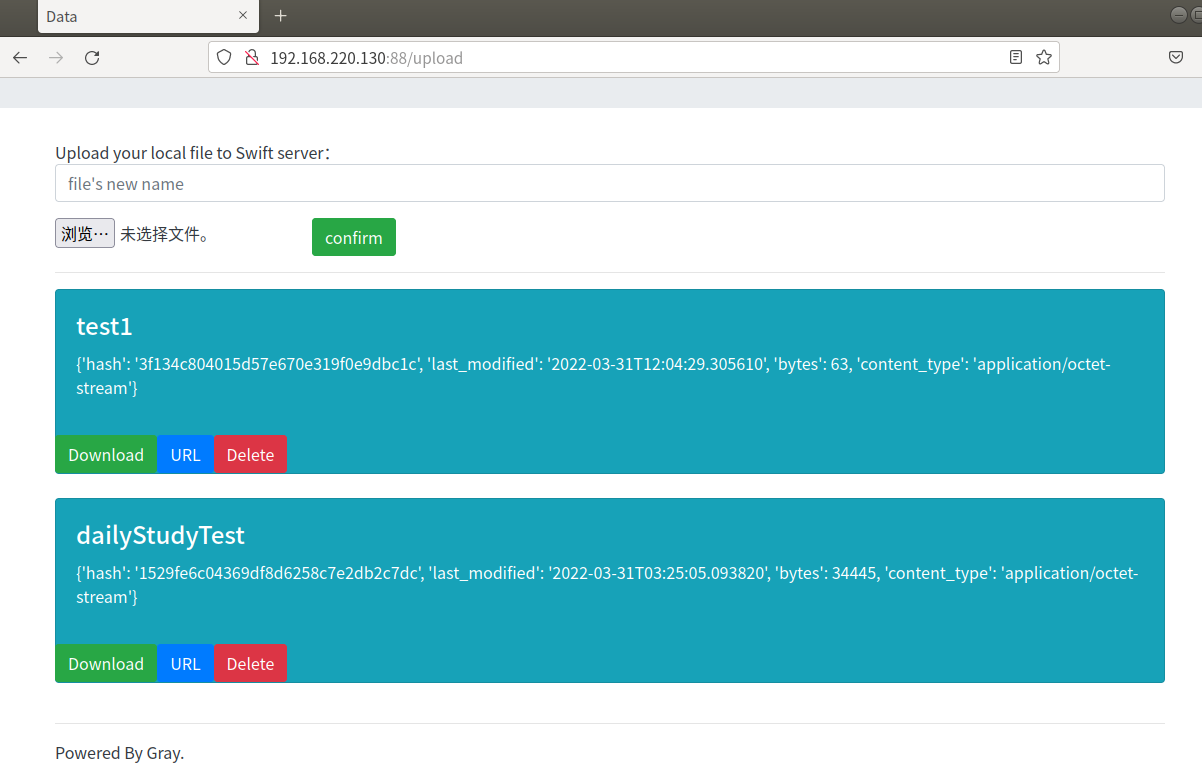


图5.9

接下来测试下载，并使用自带的diff指令对下载文件和源文件进行比对，无异常。



图5.10

接下来测试删除，在后台也是可以看到成功接受到了所有指令并进行相应的操作。

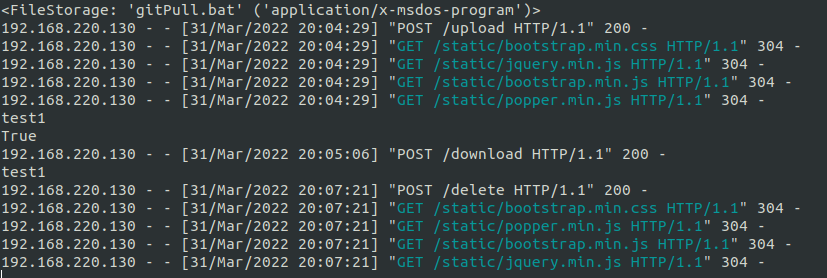


图5.11

## 5.2 对象存储性能分析

5.2.1 Swift-Bench测试

通过pip install swift-bench进行安装，成功安装后也可以看到swiftbench的初始conf为并发量10，文件大小1B，文件数量10000

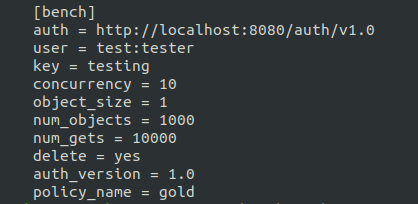


图5.12

运行一次swiftbench，并设置为100并发量，对象大小为1KB，写入次数默认为1000，读取次数默认为10000。运行结果如图5.13所示，可以看到下载速度为340.9/s，上传速度为198.4/s

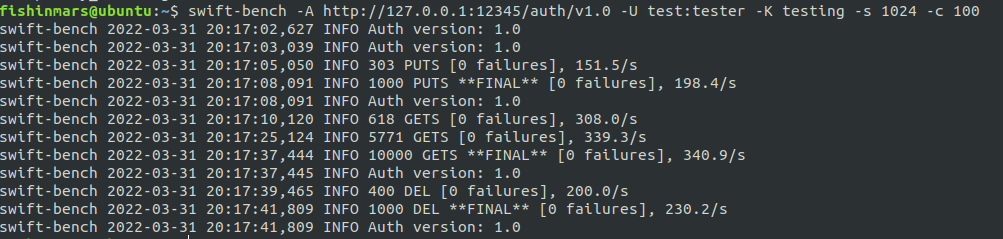


图5.13

# 六、实验总结

（实验结果分析、归纳、体会）

# 参考文献

[1] ZHENG Q, CHEN H, WANG Y等. COSBench: A Benchmark Tool for Cloud Object Storage Services[C]//2012 IEEE Fifth International Conference on Cloud Computing. 2012: 998–999.

[2] ARNOLD J. OpenStack Swift[M]. O’Reilly Media, 2014.

[3] WEIL S A, BRANDT S A, MILLER E L等. Ceph: A Scalable, High-performance Distributed File System[C]//Proceedings of the 7th Symposium on Operating Systems Design and Implementation. Berkeley, CA, USA: USENIX Association, 2006: 307–320.

[4] Dean J, Barroso L A. Association for Computing Machinery, 2013. The Tail at Scale[J]. Commun. ACM, 2013, 56(2): 74–80.

[5] Delimitrou C, Kozyrakis C. Association for Computing Machinery, 2018. Amdahl’s Law for Tail Latency[J]. Commun. ACM, 2018, 61(8): 65–72.

（可以根据实际需要更新调整）