華中科技大學

数据中心技术实验报告

院　系　计算机科学与技术学院

班　级　计算机硕2105班

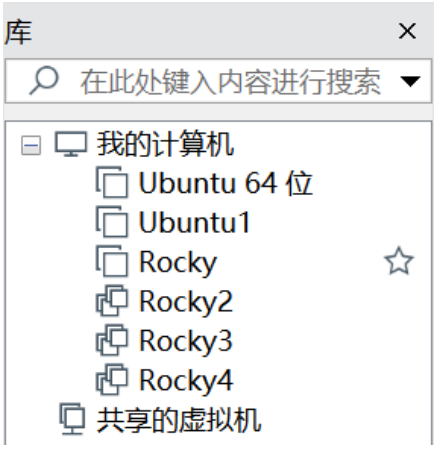
学　号　M202173670

姓　名　万兴宇

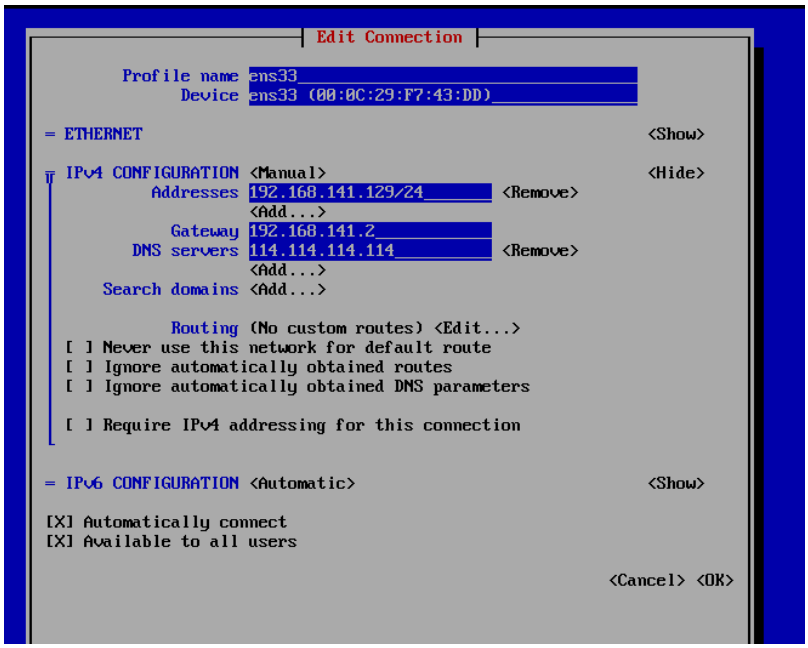
2021年 12 月 26 日

实验一：系统搭建

本实验系统搭建通过VMWareStation创建Rocky虚拟机，安装yum，python等基础使用环境



如上图所示，创建完主机Rocky后，进行克隆得到Rocky2、Rocky3、Rocky4，总体配置为一台master(radosgw)，三台worker(osds)构建一个集群，每个机器配置不同的ip地址，通过#nmtui打开NetworkManager，配置相应机器网关以及ip地址。



上图中为master的NM配置，上台worker分别选取192.168.141.130/131/132。

Plain Text

```
#vi /etc/hostname
#ssh-keygen
#ssh-copy-id -i 192.168.141.130
```

通过命令行修改hostname，worker上host也需要修改，然后通过ssh-keygen命令生成ssh密钥，并用ssh-copy-id命令将拷贝本机公钥到远程主机上面。

接着安装ceph-ansible，安装完后修改/ect/ansible/hosts文件，配置相应的hosts

```
[mgrs]
master
[monitoring]
master
[rgws]
master
[mons]
master
worker1
worker2
worker3
[osds]
worker1
worker2
worker3
```

备份group_vars下的yml文件，修改备份文件名称为all.yml以及osds.yml并修改，紧接着git checkout stable-6.0

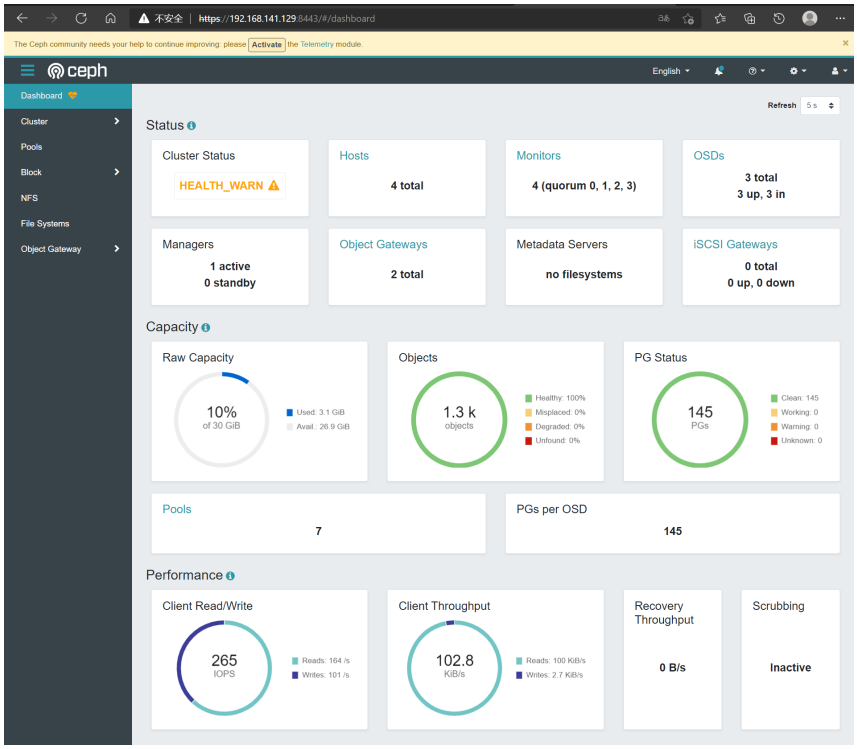
```
monitor_interface: ens33
radosgw_interface: ens33

ceph_origin: repository
ceph_repository: community
ceph_mirror: https://mirror.tuna.tsinghua.edu.cn/ceph/
ceph_stable_release: pacific
ceph_stable_repo: "{{ ceph_mirror }}/rpm-{{ ceph_stable_release }}"
ceph_stable_redhat_distro: el8

journal_size: 1024
generate_fsid: true

dashboard_admin_user: admin
dashboard_admin_password: p@ssw0rd
grafana_admin_user: admin
grafana_admin_password: p@ssw0rd  # this entry seems to be unused;
```

```
osd_auto_discovery: true
```

Plain Text

```
#ansible-playbook -i /etc/ansible/hosts site.yml
```

配置完成后使用上面命令行执行安装，安装成功后可以通过https://192.168.141.129访问ceph dashboard。



实验二：性能观测

首先查询本机S3服务地址，redosgw即为对应的服务地址

```
[root@master ~]# netstat -tnlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      962/sshd
tcp        0      0 192.168.141.129:3000    0.0.0.0:*               LISTEN      1792/grafana-server
tcp        0      0 192.168.141.129:8443    0.0.0.0:*               LISTEN      1561/ceph-mgr
tcp        0      0 192.168.141.129:3300    0.0.0.0:*               LISTEN      1565/ceph-mon
tcp        0      0 192.168.141.129:9093    0.0.0.0:*               LISTEN      1679/alertmanager
tcp        0      0 192.168.141.129:6789    0.0.0.0:*               LISTEN      1565/ceph-mon
tcp        0      0 192.168.141.129:9094    0.0.0.0:*               LISTEN      1679/alertmanager
tcp        0      0 192.168.141.129:8080    0.0.0.0:*               LISTEN      1572/radosgw
tcp        0      0 192.168.141.129:6800    0.0.0.0:*               LISTEN      1561/ceph-mgr
tcp        0      0 192.168.141.129:6801    0.0.0.0:*               LISTEN      1561/ceph-mgr
tcp6       0      0 :::22                   :::*                    LISTEN      962/sshd
tcp6       0      0 :::9283                 :::*                    LISTEN      1561/ceph-mgr
tcp6       0      0 :::9100                 :::*                    LISTEN      1562/node_exporter
[root@master ~]#
```

生成一新用户，在集群当中创建相同的管理用户

Plain Text

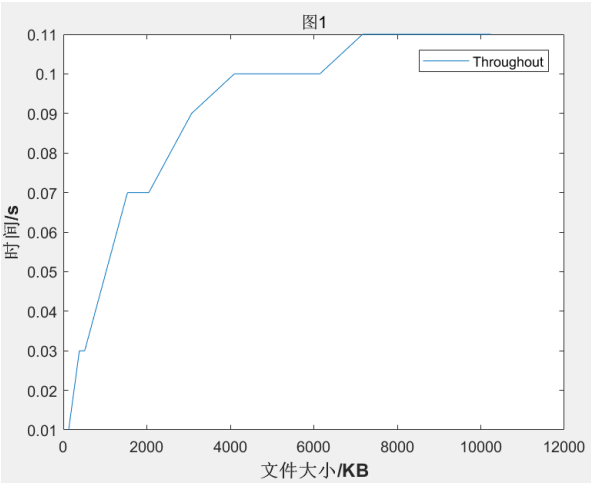
```
radosgw-admin user create --uid=admin --display-name=admin --access_key=admin
 --secret=123456
```

得到安装相应的代码的基本配置

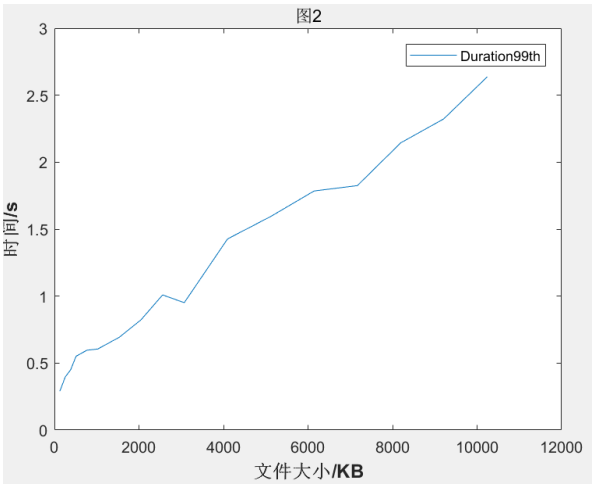
Python

```python
# 本地S3服务地址
local_s3 = "http://192.168.141.129:8080"
# 准备密钥
aws_access_key_id = 'admin'
aws_secret_access_key = '123456'
# filepath
filepath = '/root/test/'
# 新建一个bucket的名称
bucket_name = 'wanxy'
```

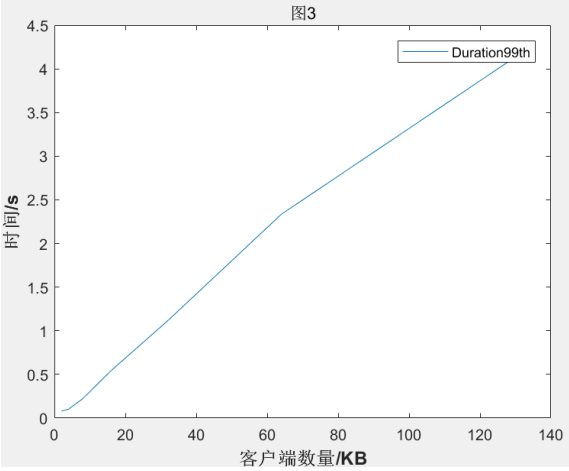
然后安装boto3，进而利用python代码进行实验测试得到如下实验结果：



a. 如图1所示，在客户端数量固定为20的情况下，通过不停调整文件大小进行测试，获得吞吐量随时间变化关系如图，可以看到曲线呈ln型，随着文件大小增加，时间逐渐平滑。



b. 如图2所示，客户端固定情况下，随着文件大小增加，延迟也逐渐增加。



c. 如图3所示，在文件大小固定为1024KB的情况下，延迟时间随客户端数量变化成正比，客户端数量越多，服务器同时需要处理的请求越多，越容易产生更大延迟

```
[root@master test]# python3 test2.py 256 128 1024
Killed
```

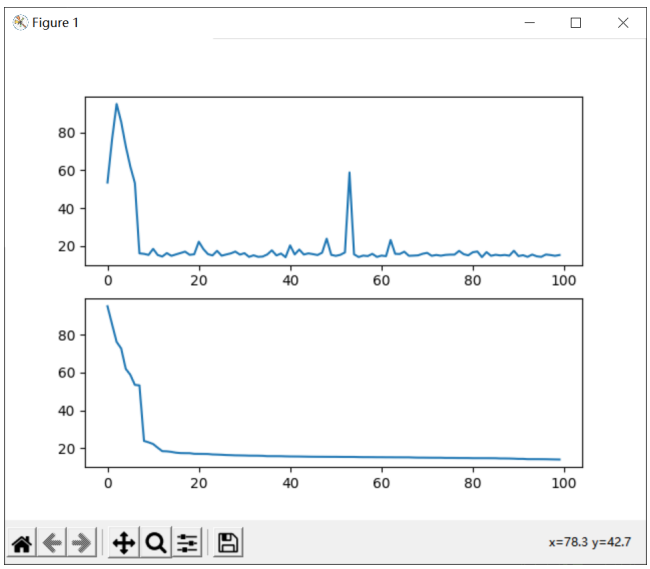
在配置为256个客户端的时候，由于虚拟机配置不够，导致程序中断，因此不做更大数量的测试。

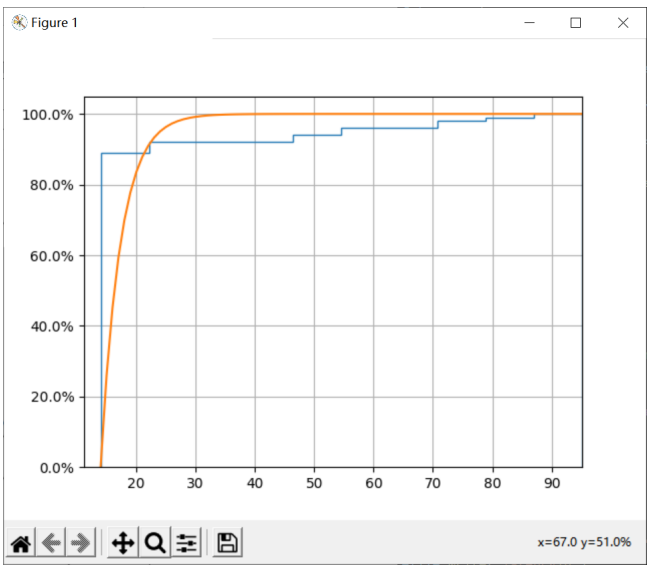
实验三：尾延迟挑战

运行obs-tutorial中的python脚本得到延迟数据

```
[root@master test]# python3 test3.py
bucket name:wanxy
Accessing S3: 100%|                                                                                | 100/100 [00:01<00:00, 57.26it/s]
```

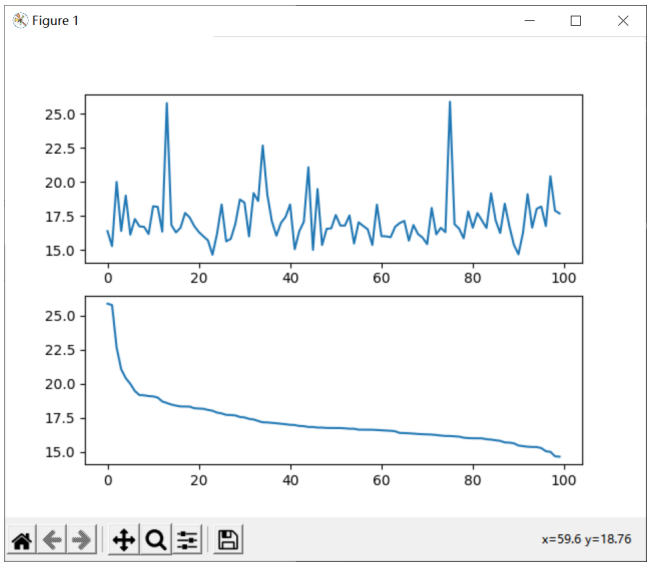
再运行plot脚本转换为图效果如下：

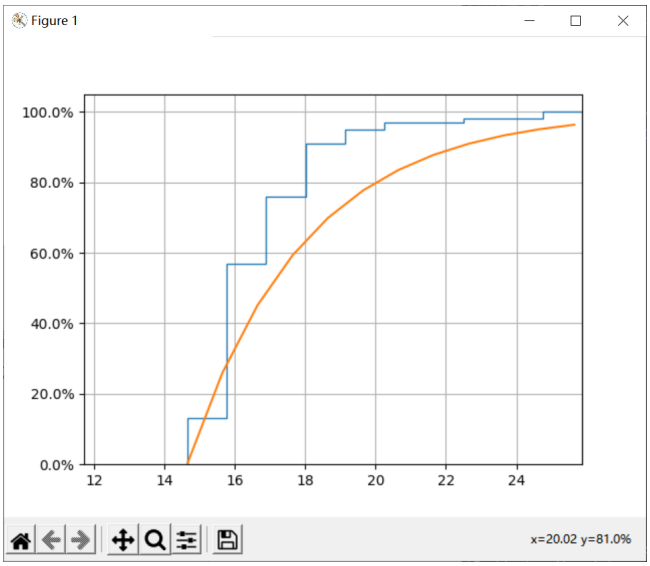




上图中存在一部分写请求的开销远超过其他写请求，就是尾延迟现象的表现。

- **尝试对冲**





由上图可以看到60ms时候有95%的数据请求发送完成，所以通过设置时间阈值为60ms，超时后重发相同请求，得到上图结果，可以看到比之前的结果有明显改善