2019 级

《大数据数据存储与管理》课程

实 验 报 告

姓　名 陈昱夫

学　号 U201990056

班　号 CS1903

日　期 2022.04.05

目　　录

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，架设实际应用，示范主要功能。

二、实验背景

目前物联网数据的大爆发，使得物联网存储面临巨大的挑战，数据内容庞大而又复杂。

对象存储系统提供了高可靠性、跨平台性以及安全的数据共享的存储体系结构，可以在一个持久稳固且高度可用的系统中存储任意的对象，且独立于虚拟机实例之外。应用和用户可以在对象存储中使用简单的 API 访问数据。

三、实验环境

表格 1 实验环境

硬件环境	
处理器	I5-8250U
内存	8G
硬盘	256G 固态
软件环境	
Golang 版本	12.5
服务器端	Minio
客户端	Minio Client

四、实验内容

搭建实验环境，包括 python、java、golang 和虚拟机 Ubuntu16.04。安装实验用软件 minio、minio client、s3 bench。

4.1 对象存储技术实践

1.在 windows 环境下配置 minio 服务器端
2.使用下载好的 mc 客户端，创建 bucket，并上传文件
3.使用 s3 bench 测试文件

4.2 对象存储性能分析

1.测试分析性能

五、实验过程

1. 搭建环境

搭建 golang 环境，下载安装 golang，配置好环境变量 gopath。

2. 配置使用 minio

下载 minio.exe，进入对应文件夹使用命令行打开，输入 minio.exe server d:\ms 在 d:\ms 打开 minio 服务器。成功后在 dos 界面现实 endpoint、accessKey 和 secretKey，如图 1 所示。Key 内容可在 D:\ms\.minio.sys\config\config.json 中修改。

```
D:\minio>minio.exe server d:\stuart
API: http://10.21.206.183:9000  http://192.168.18.1:9000  http://192.168.91.1:9000  http://127.0.0.1:9000
RootUser: minioadmin
RootPass: minioadmin

Console: http://10.21.206.183:57530 http://192.168.18.1:57530 http://192.168.91.1:57530 http://127.0.0.1:57530
RootUser: minioadmin
RootPass: minioadmin

Command-line: https://docs.min.io/docs/minio-client-quickstart-guide
   $ mc.exe alias set myminio http://10.21.206.183:9000 minioadmin minioadmin

Documentation: https://docs.min.io
```

图 1 打开 minio 服务器

打开端点 192.168.31.117:9000，浏览器启动后为以下界面。

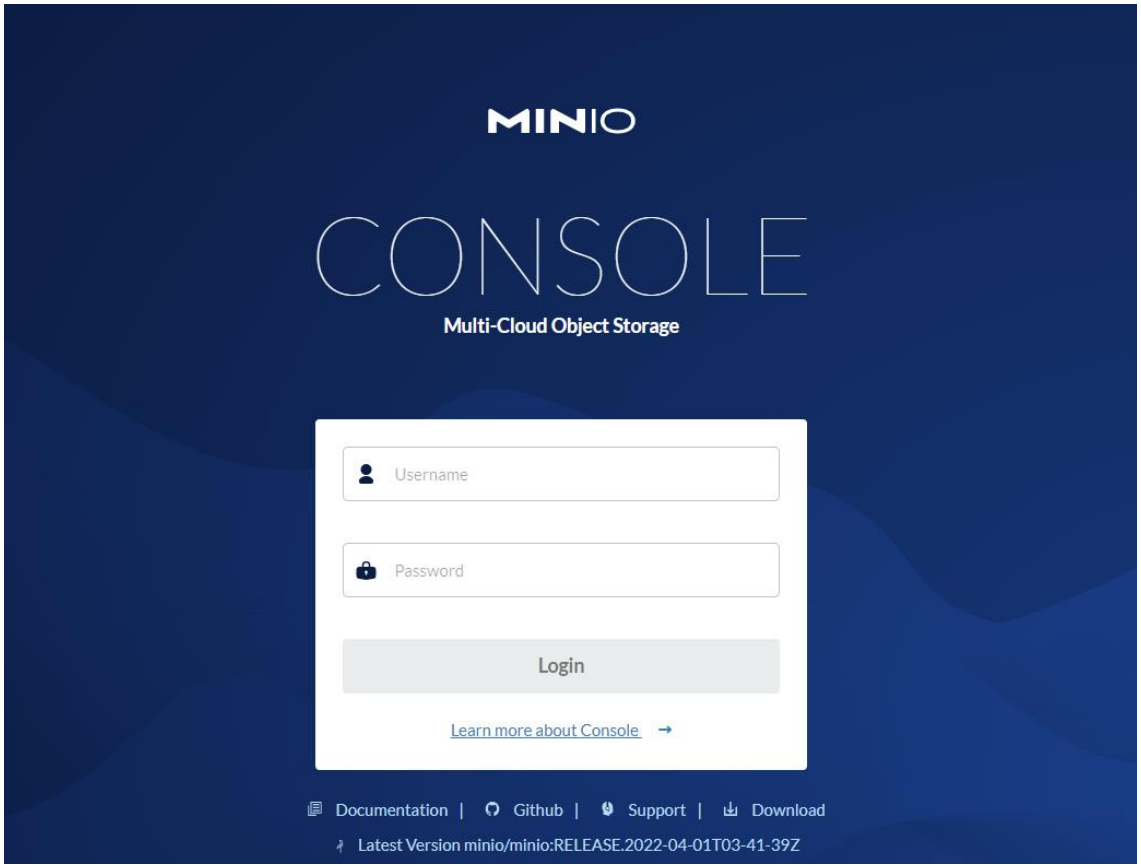


图 2 登陆界面

输入 key 即可登陆，上传至 endpiont 的文件都可直观看见。

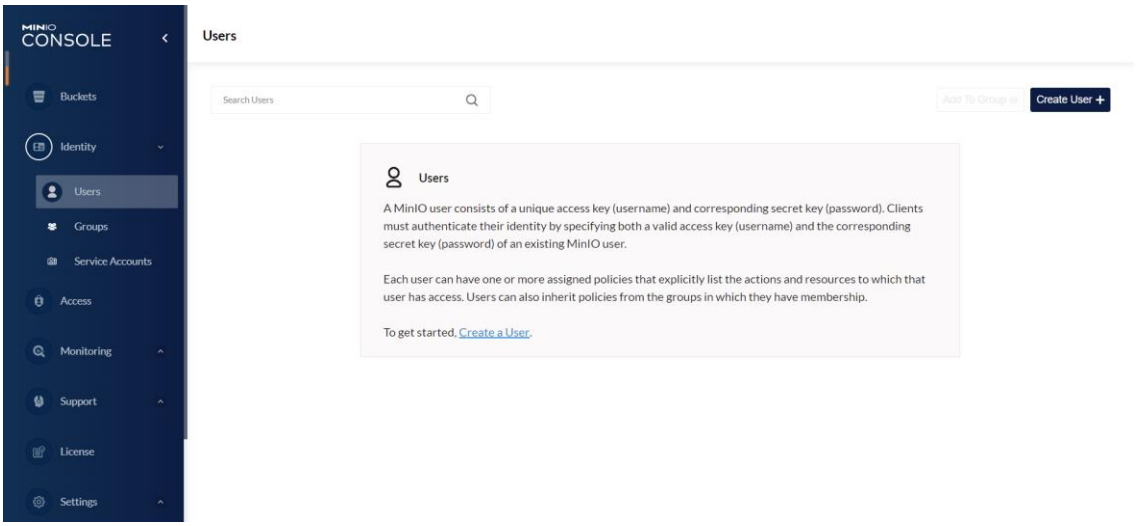


图 3 minio browser

3. 配置使用 minio client

下载 mc.exe 客户端，同理使用命令行打开，输入 mc.exe –help 可见客户端操作提示，如图所示为部分内容。

```
D:\>cd minio

D:\minio>mc.exe --help
NAME:
  mc - MinIO Client for cloud storage and filesystems.

USAGE:
  mc [FLAGS] COMMAND [COMMAND FLAGS | -h] [ARGUMENTS...]

COMMANDS:
  alias      manage server credentials in configuration file
  ls         list buckets and objects
  mb         make a bucket
  rb         remove a bucket
  cp         copy objects
  mv         move objects
  rm         remove object(s)
  mirror     synchronize object(s) to a remote site
  cat        display object contents
  head       display first 'n' lines of an object
  pipe       stream STDIN to an object
  find       search for objects
  sql        run sql queries on objects
  stat       show object metadata
  tree       list buckets and objects in a tree format
  du         summarize disk usage recursively
  retention  set retention for object(s)
  legalhold  manage legal hold for object(s)
  support    support related commands
  share      generate URL for temporary access to an object
  version    manage bucket versioning
  ilm        manage bucket lifecycle
  encrypt    manage bucket encryption config
  event      manage object notifications
  watch      listen for object notification events
  undo       undo PUT/DELETE operations
  anonymous  manage anonymous access to buckets and objects
  tag        manage tags for bucket and object(s)
  diff       list differences in object name, size, and date between two buckets
  replicate  configure server side bucket replication
  admin      manage MinIO servers
  update     update mc to latest release

GLOBAL FLAGS:
```

图 4 客户端使用指南

使用 mc.exe config host add minio http:// 192.168.31.117:9000 ballball 19980607 S3v4 添加主机。Minio 为主机名称，之后为 endpoint、accessKey、secretKey 和 API 签名默认为 S3v4。

```
D:\minio>mc.exe alias set myminio http://10.21.206.183:9000 minioadmin minioadmin
Added `myminio` successfully.
```

图 5 添加主机 minio

为服务器添加 bucket：使用 mc.exe mb minio/newbucket 添加。Mb 为 makebucket 添加命令，newbucket 为 bucket 名称。

```
D:\minio>mc.exe mb myminio/newbucket
Bucket created successfully `myminio/newbucket`.
```

图 6 添加 bucket

创建成功后可在 minio browser 中看见新的 bucket 如图所示

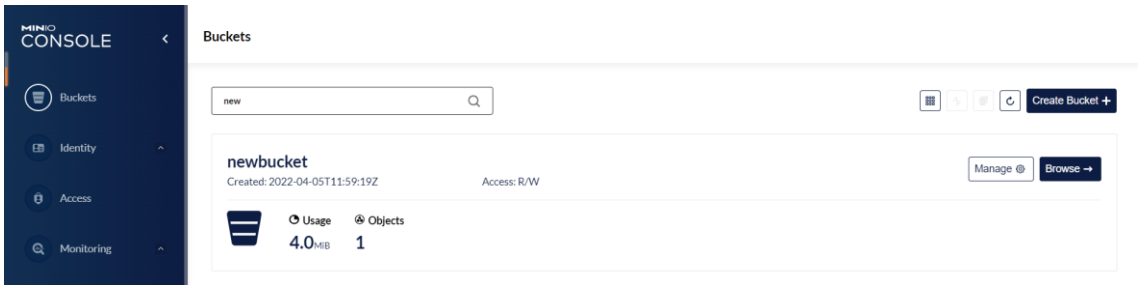


图 7 添加结果

上传文件，使用 mc.exe cp d:\mino\test.txt minio/newbucket，上传至目标 bucket 如图所示。

```
D:\minio>mc.exe cp d:\Stuart\miniotest\test.JPG myminio/newbucket
...\test.JPG:  4.02 MiB / 4.02 MiB [============================] 56.05 MiB/s 0s
```

图 8 上传文件

刷新页面可观察到上传的文件。

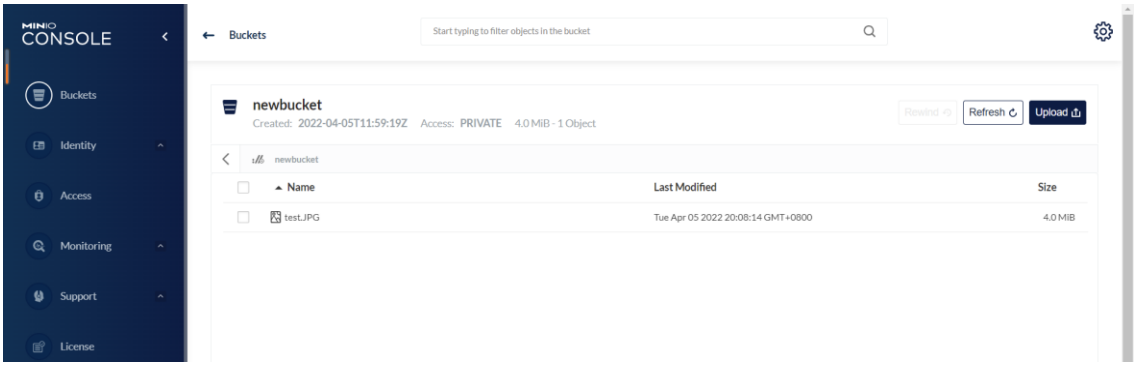


图 9 上传成功

4. 配置使用 s3 bench

使用 go get -u github.com/igneous-systems/s3bench 下载安装好 s3 bench，在 $GOPATH/bin/s3bench 文件中，可移动至使用文件夹。之后使用命令 s3bench.exe -accessKey=KEY -accessSecret=SECRET -bucket=loadgen -endpoint=http://endpoint1:80,http://endpoint2:80 -numClients=2 -numSamples=10 -objectNamePrefix=loadgen -objectSize=1024 测试。

```
D:\>cd minio

D:\minio>d:./s3bench -accessKey=minioadmin -accessSecret=minioadmin -bucket=newbucket -endpoint
=http://10.21.206.183:9000,http://127.0.0.1:9000 -numClients=2 -numSamples=10 -objectNamePrefix
=newbucket -objectSize=1024
```

图 10 测试服务器

测试结果如下所示。

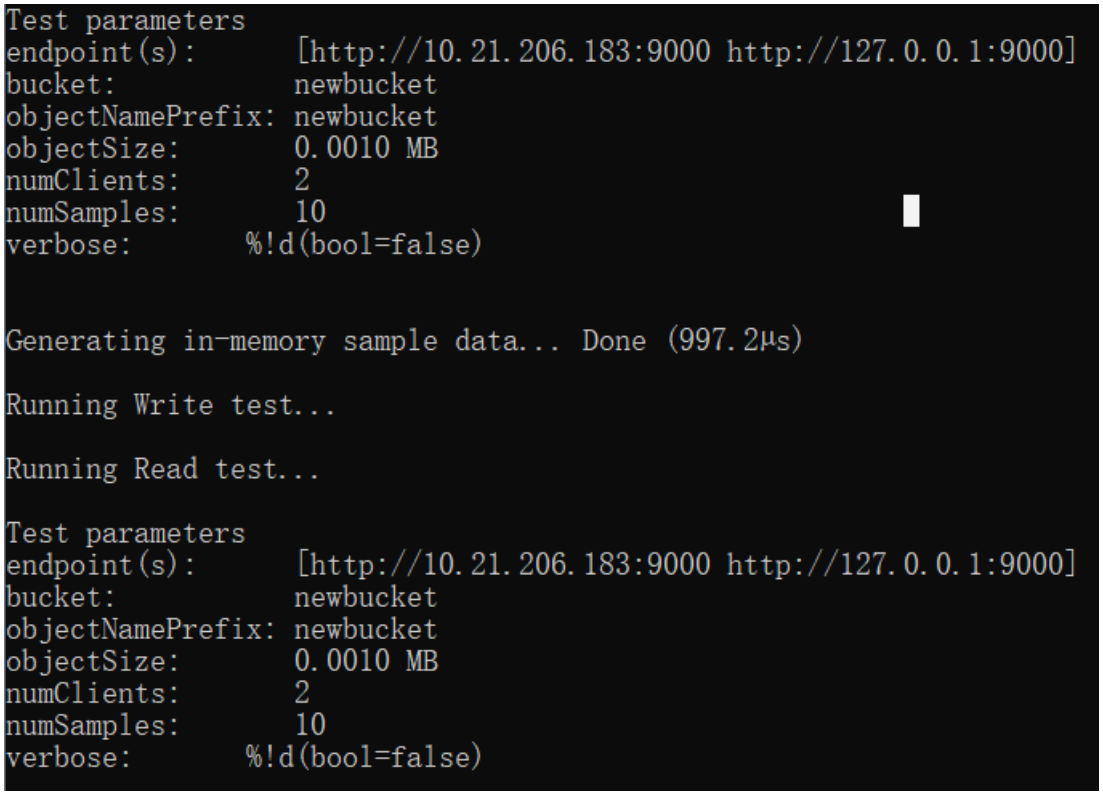

Test parameters
endpoint(s): [http://10.21.206.183:9000 http://127.0.0.1:9000]
bucket: newbucket
objectNamePrefix: newbucket
objectSize: 0.0010 MB
numClients: 2
numSamples: 10
verbose: %!d(bool=false)
Generating in-memory sample data... Done (997.2µs)
Running Write test...
Running Read test...
Test parameters
endpoint(s): [http://10.21.206.183:9000 http://127.0.0.1:9000]
bucket: newbucket
objectNamePrefix: newbucket
objectSize: 0.0010 MB
numClients: 2
numSamples: 10
verbose: %!d(bool=false)

图 11 测试结果（上）

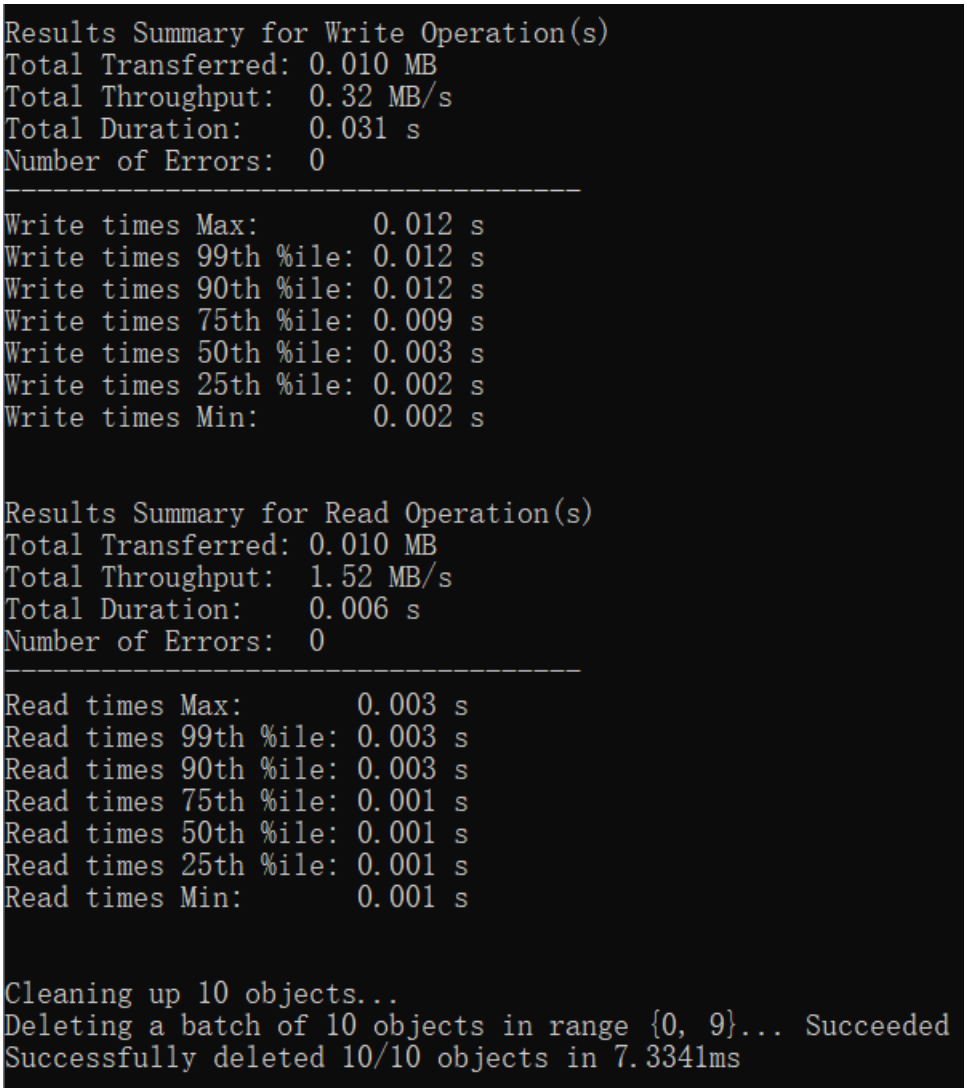
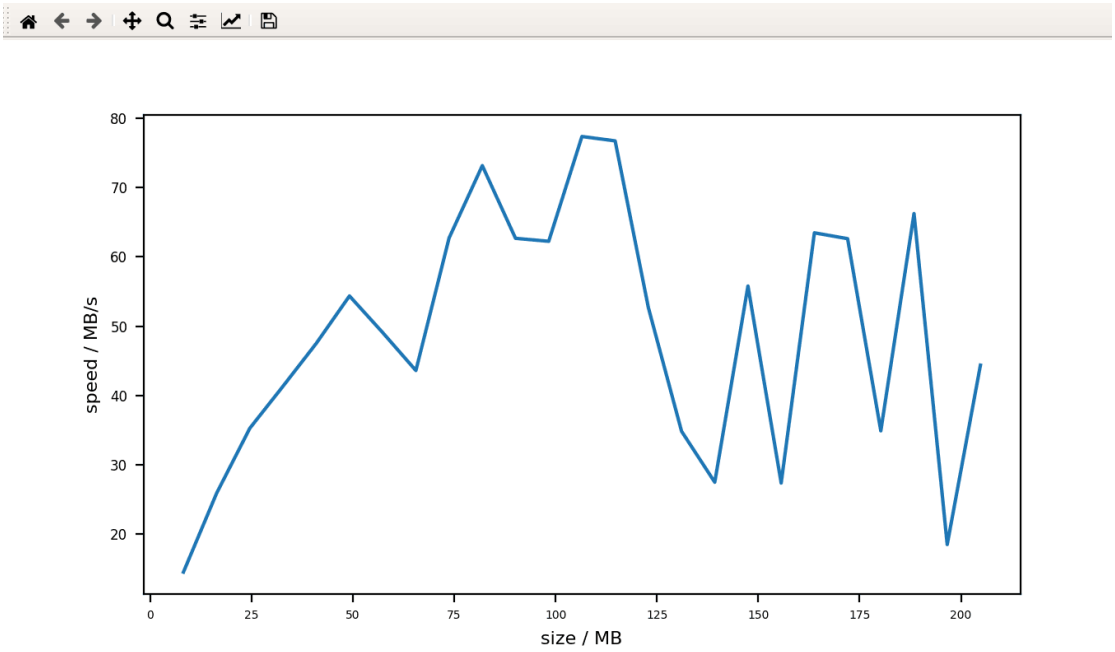


图 12 测试结果（下）

通过一个脚本将 37 种不同的情况重定向到记事本，并用一个 python 程序将记事本数据提取并画出对应的折线图。

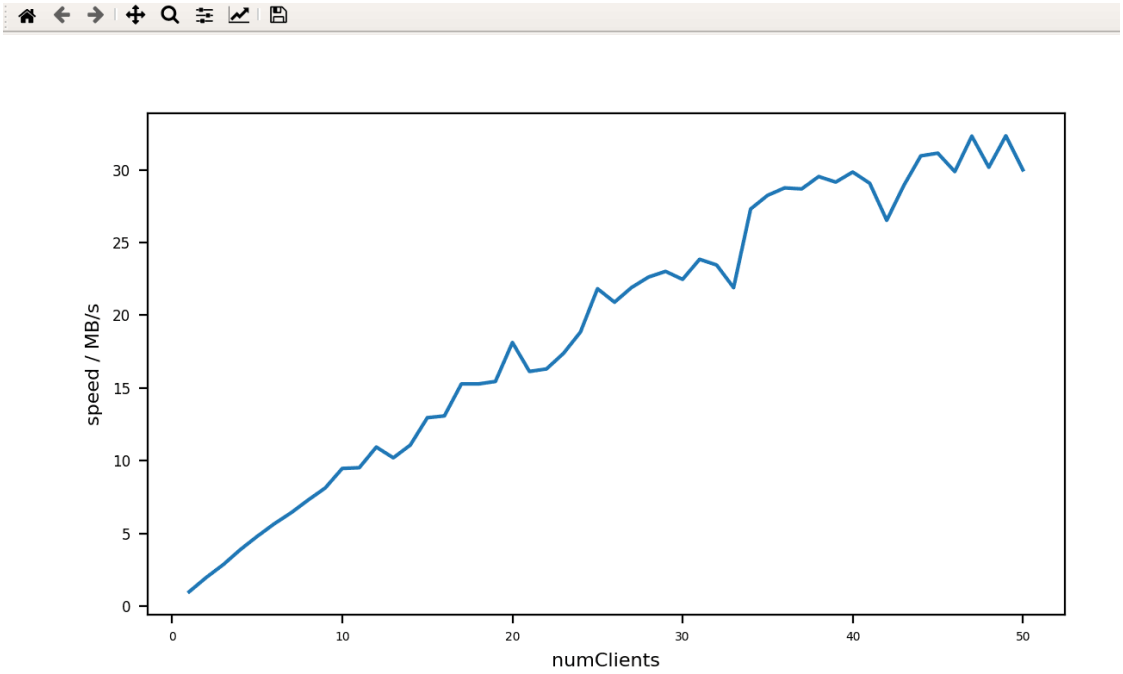
下面来看运行结果并分析。

（1） 对象大小对传输速率的影响，见图 14。对象大小以 8MB 的倍数递增到 200MB。共 25 个点对。客户端数量=2。



根据改图可以看出，当文件大小在 110MB 左右，传输速率达到顶峰，当然由于实验条件有限（VMWare-Ubuntu 性能限制等原因），200MB 以后的数据我们无法考虑。在 110MB 以后，传输速率呈波动式下降。分析波动的原因——文件过大，且 python 程序不成熟。

（2） 客户端数量对传输速率的影响，见图 15。该实验做了多次，取了其中三次图片输出。



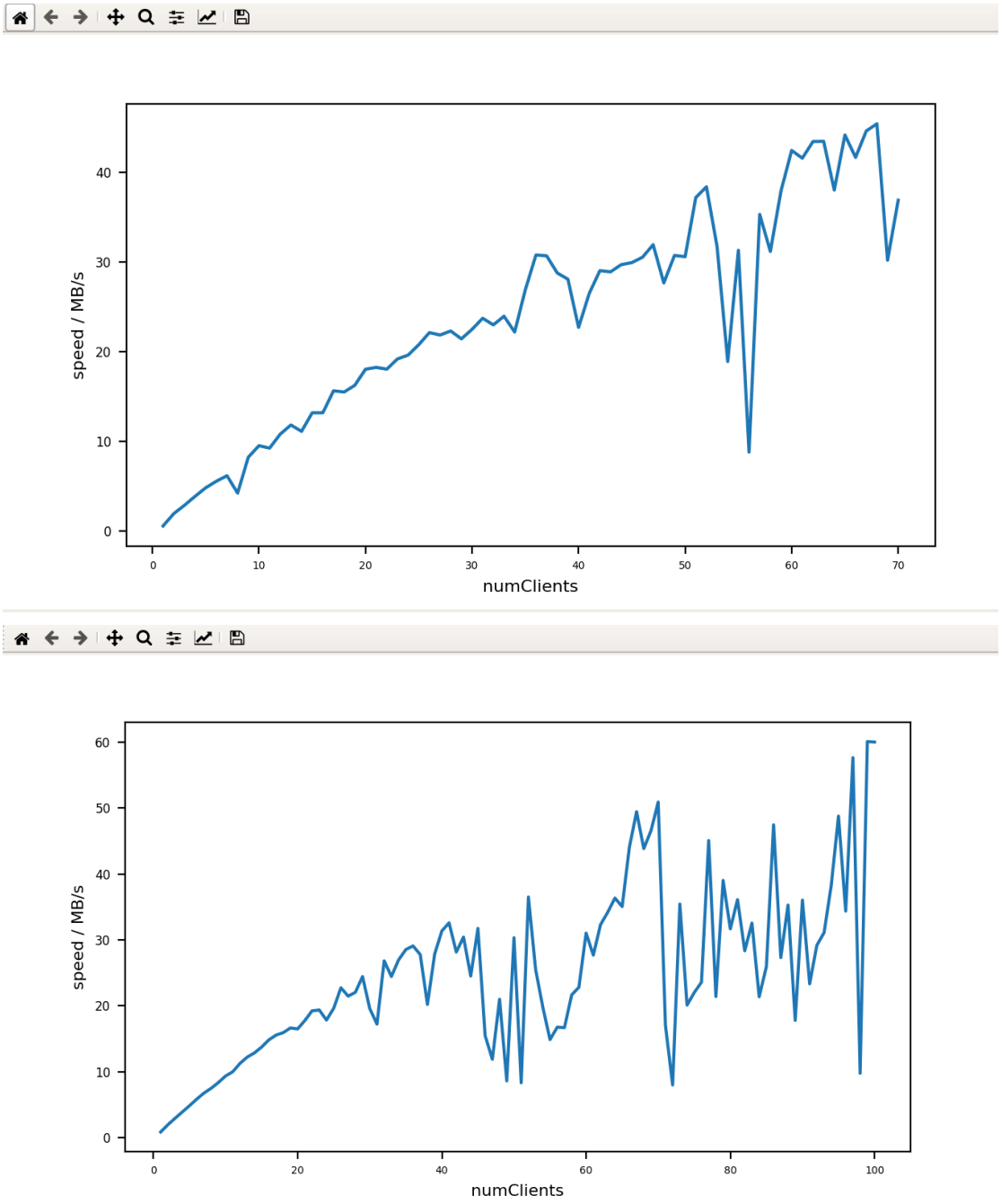
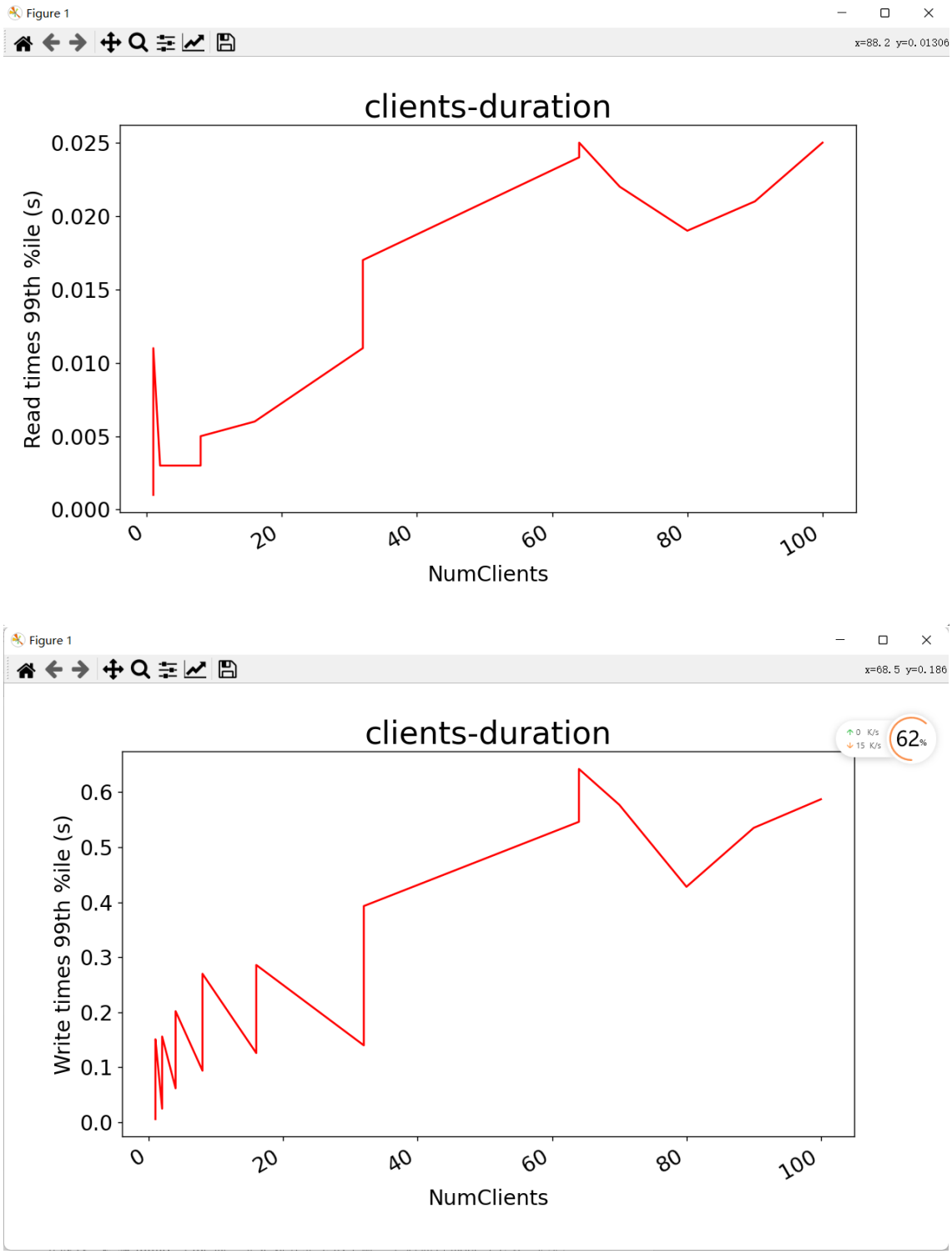


图 15 客户端数量对传输速率的影响

这三次图片输出的客户端数量上限分别为 50、70、100。可以看出，客户端数=50 时，传输速率增长开始变慢；客户端数=70 时，传输速率增长变慢，且波动性增大；客户端数=100 时，已经呈现水平值上线波动的特性。可知，当客户端数=100 时，传输速率的增长放慢甚至不再增长。从理论分析，客户端数再继续增大时，传输速率会减小。

（3） 客户端数量对 99th%ile 延迟的影响，见下图。



客户端数量对 99th%ile Duration 的影响

从本图可以看出，客户端越多，基本上呈现延迟越大的规律，且延迟向上波动性增长。从理论分析也不难得出这个结论。客户数量越多，服务端需要同时处理的请求越多，越难调度，越容易产生更大延迟。

六、实验总结

通过本次实验，我学习到了许多关于对象存储技术的知识，并实践运用了部分内容。搭建 minio 与 mc 时，先是对其完全不了解，之后通过询问与学习，慢慢安装好文件运行，之后慢慢才真正了解到其功能与实践方法，觉得挺有趣。之后使用 s3 bench 测试时，开始不懂测试的各项命令与内容，后来经过讨论与观察，也渐渐明白数据的各项意义,学长留下的将控制台输出重定向到文本文件的脚本比较友好，我也另外用 python 读取这些文本数据，并绘制出了相关图表，学习了一下 python，算是不错的收获。结束实验后，从中能学到许多不同于课本内容的东西。

参考文献

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