2021 级

《数据中心》课程

实 验 报 告

姓　名　姜志鹏

学　号　M202173812

班　号　电子信息 2109 班

日　期　2022.01.03

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

对象存储服务（Object Storage Service，OSS）是一种海量、安全、低成本、高可靠的云存储服务，适合存放任意类型的文件。容量和处理能力弹性扩展，多种存储类型供选择，全面优化存储成本。对象存储是面向企业级应用和互联网服务推出的非结构化数据存储服务，提供安全、可靠、稳定的海量数据存储解决方案。随着大数据、云计算和移动互联网等的发展，很多以前存在技术瓶颈的业务，都可以在对象存储这里得到完美的解决。

本实验建立对象存储服务端和客户端，两者进行数据通信，通关观察尾时延的变化来了解对象存储技术以及对尾时延相关技术进行探索。

三、实验环境

高级语言	Python
对象存储服务端	Minio
对象存储客户端	Minio-Client
吞吐量基准测试工具	s3bench

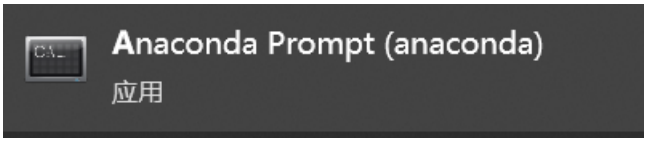
四、实验内容与过程

本实验在搭建好对象存储技术环境后，利用 python 脚本向对象存储服务端发起请求，利用 S3 工具进行性能评估，并观察在不同数据量的情况下尾时延的变化情况，最后对尾时延优化技术进行探索。

4.1 Preparation

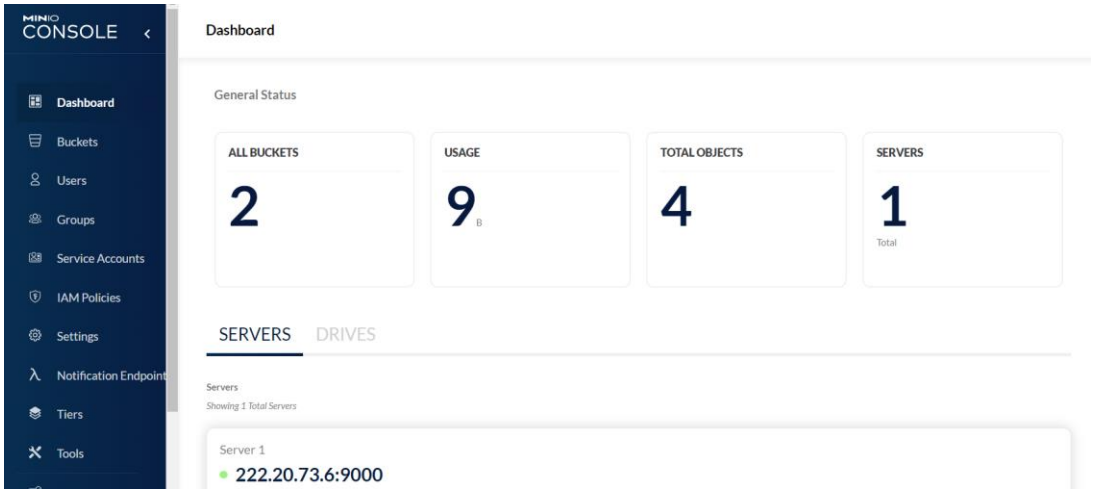
4.1.1 python 环境准备

本实验采用开源的 Python 发行版本 Anaconda



4.1.2 对象存储服务端

下载最新的 Minio，按照说明配置好后，可进入 Web 前端 Dashboard 观察当前服务端的情况，如下图，可直接利用前端界面创建 Bucket 或者上传数据等操作。Minio 包含了 PUT/POST、GET、PUT/POST/PATCH 和 DELETE 等 CRUD 操作。



4.1.3 对象存储客户端

在搭建好 Minio 服务端后，利用 cmd 命令添加 minio-client，如下图所示，成功添加 minio 客户端：

```
D:\datacenter>mc.exe alias set myminio http://222.20.73.6:9000 hust hust_obs
Added `myminio` successfully.
```

4.1.4 s3bench 工具

在服务端和客户端搭建完之后，利用 s3bench 工具测试系统的吞吐量、时延等指标，如下图所示：

```
D:\datacenter>s3bench.exe -accessKey=hust -accessSecret=hust_obs -bucket=jzp -endpoint=http://
222.20.73.6:9000 -numClients=8 -numSamples=256 -objectNamePrefix=loadgen -objectSize=1024
```

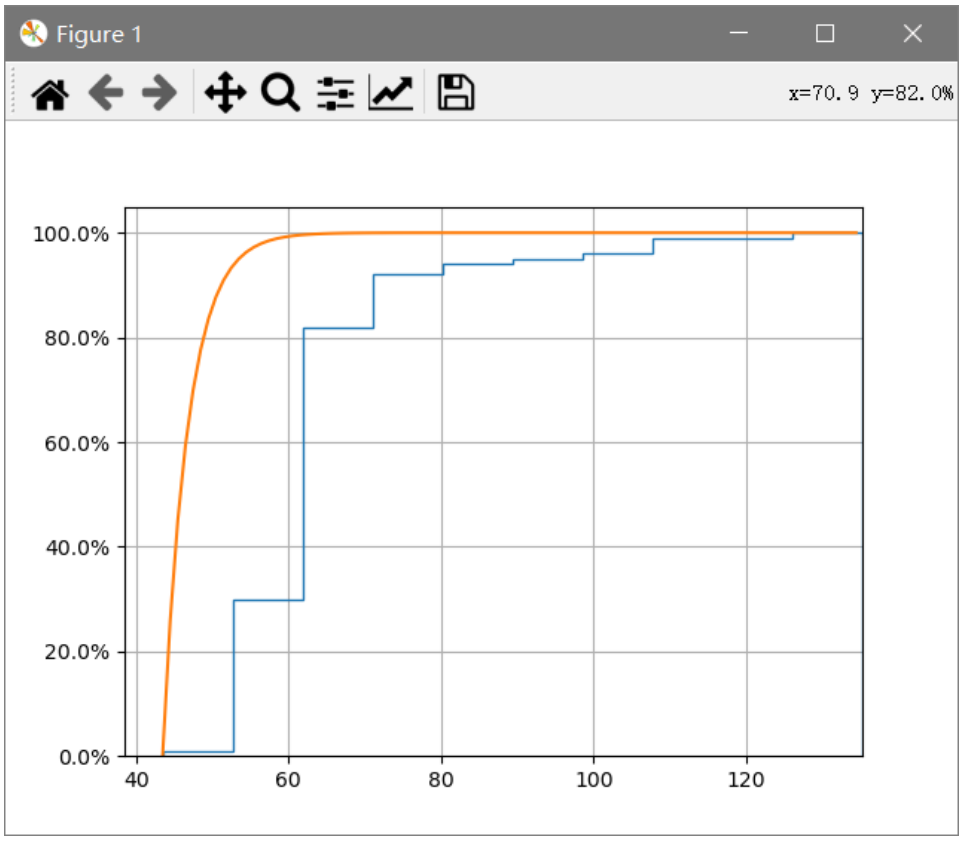
```
Test parameters
endpoint(s):      [http://222.20.73.6:9000]
bucket:           jzp
objectNamePrefix: loadgen
objectSize:       0.0010 MB
numClients:       8
numSamples:       256
verbose:       %!d(bool=false)


Results Summary for Write Operation(s)
Total Transferred: 0.250 MB
Total Throughput:  0.02 MB/s
Total Duration:    16.126 s
Number of Errors:  0
-----------------------------------
Write times Max:       1.142 s
Write times 99th %ile: 1.091 s
Write times 90th %ile: 0.750 s
Write times 75th %ile: 0.574 s
Write times 50th %ile: 0.446 s
Write times 25th %ile: 0.387 s
Write times Min:       0.184 s
```

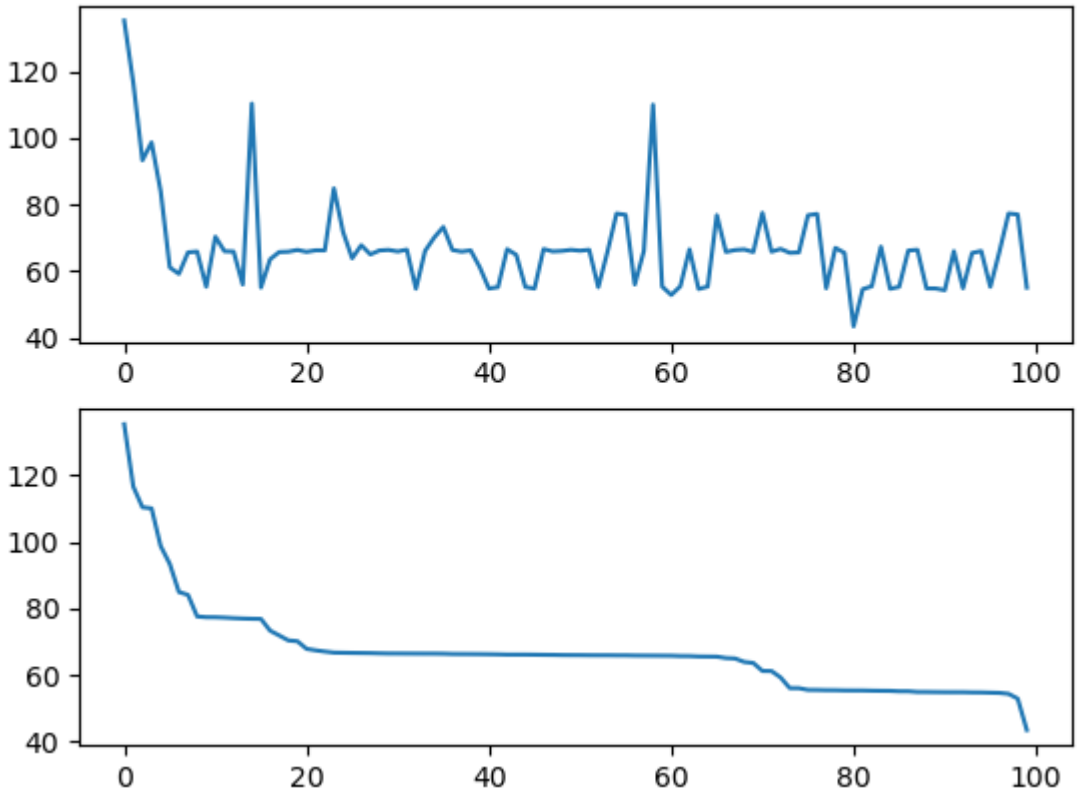
```
Results Summary for Read Operation(s)
Total Transferred: 0.250 MB
Total Throughput:  1.73 MB/s
Total Duration:    0.144 s
Number of Errors:  0
-----------------------------------
Read times Max:       0.031 s
Read times 99th %ile: 0.031 s
Read times 90th %ile: 0.006 s
Read times 75th %ile: 0.004 s
Read times 50th %ile: 0.003 s
Read times 25th %ile: 0.002 s
Read times Min:       0.001 s


Cleaning up 256 objects...
Deleting a batch of 256 objects in range {0, 255}... Succeeded
Successfully deleted 256/256 objects in 436.6999ms
```

4.2 Performance Evaluation

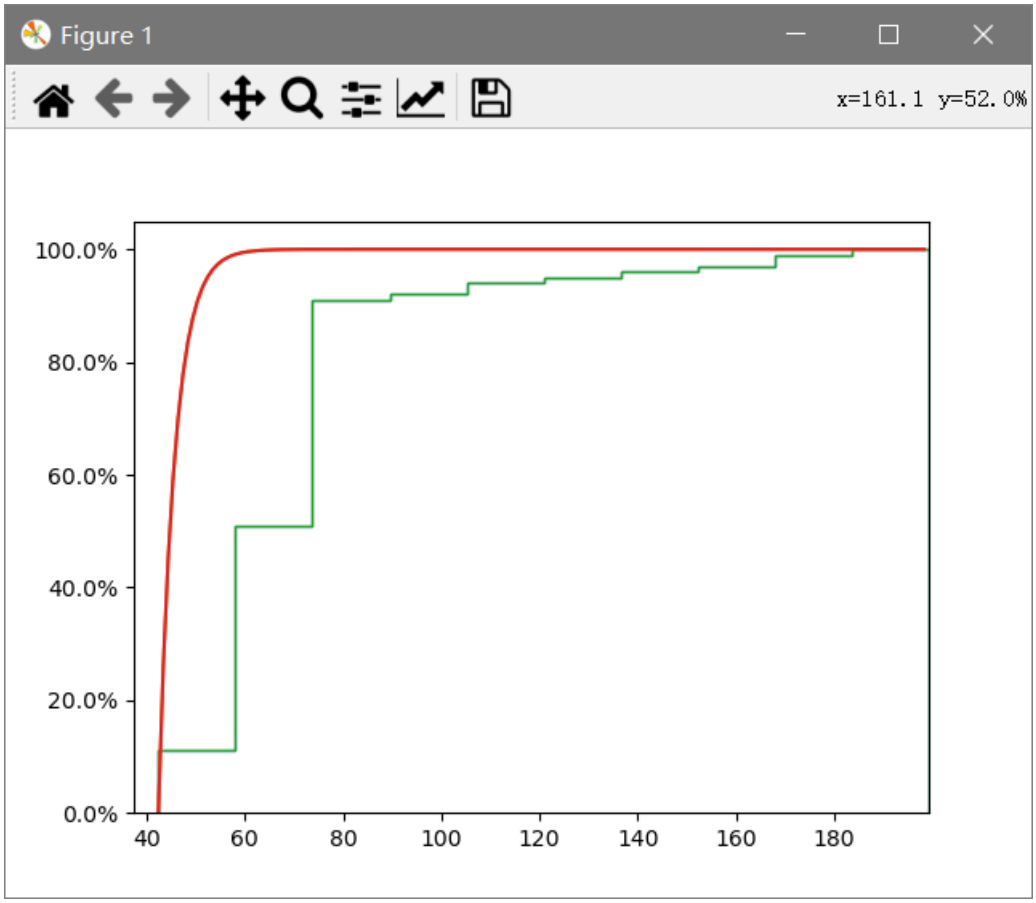
本实验利用 python 程序向服务端发送数据，收集尾延时大小并将其绘制成统计图。

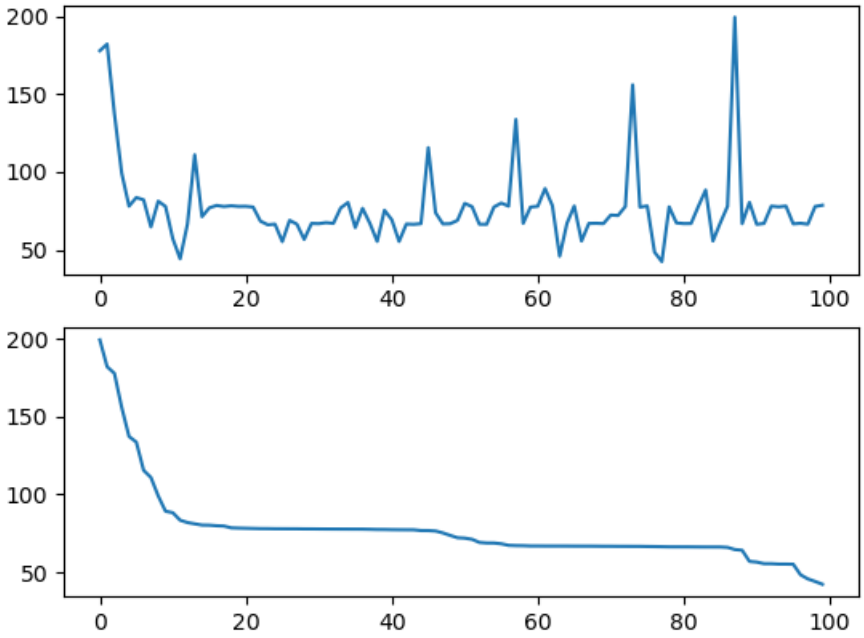
4.2.1 当数据大小设置为 4k 时





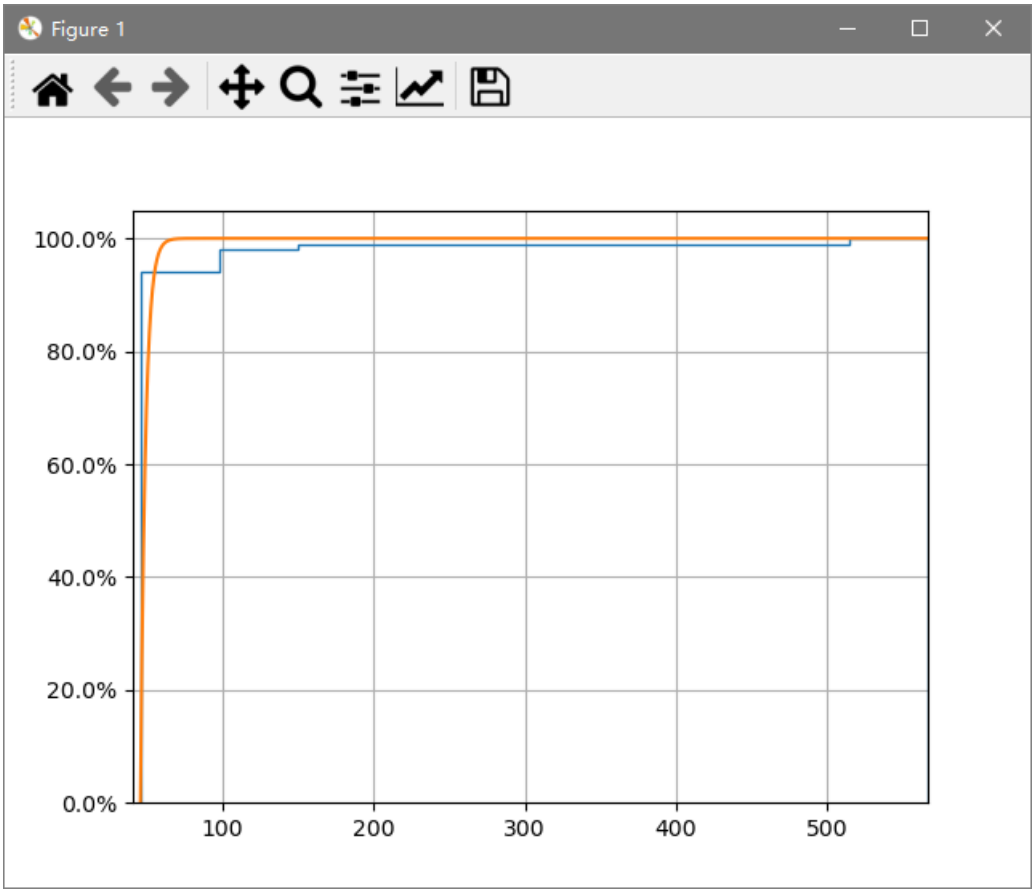
4.2.2 当数据大小设置为 40k 时

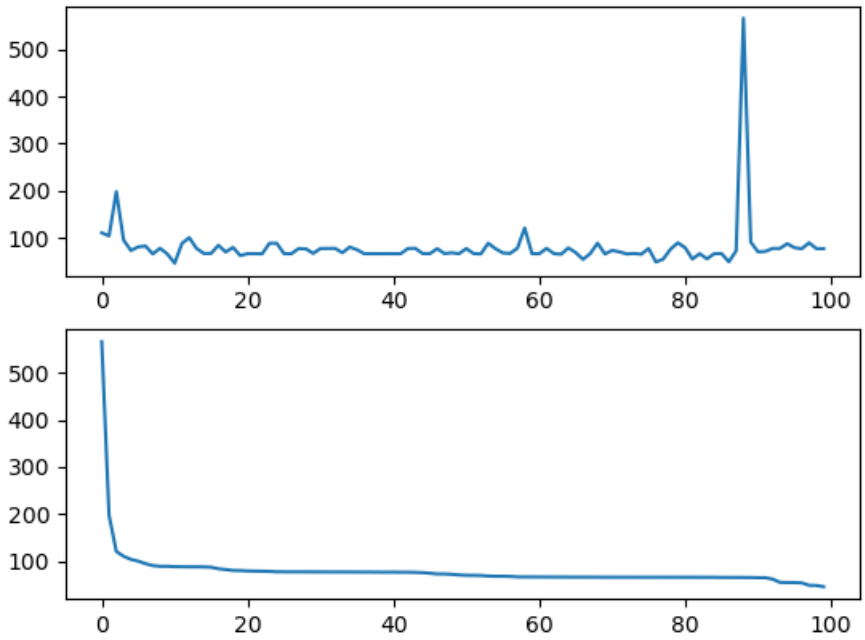




4.3 Tail Latency Challenge

从实验 4.1 和 4.2 可以发现，当数据传输 90%左右时，时延抖动最大，要减小尾时延的影响，算法设计可以在数据传输到 90%时进行打流操作，优化结果如下：





五、实验总结

通过这次实验，弄清楚了对象存储技术的整个技术架构以及原理，并通过程序观测数据请求传输时时延的变化，得出结论：当数据传输在 90%左右时，会出现时延剧烈抖动的情况，因此优化的出发点在于当数据传输 90%时，调用更多的资源去处理请求，以避免尾时延带来的影响。

另外，90%这个数据只是实验观测得来，不一定适用于所有大小的数据，所以本实验的更深一步的研究方向在于能否利用数学模型或者深度学习模型去找到某个特定场景下数据传输打流的最优临界值。