華中科技大学

数据中心技术
课程实验报告

院　　系 计算机科学与技术学院

班　　级 2106 班

学　　号 M202173728

姓　　名 文苏洋

2022 年　1 月 6 日

一、 系统搭建

1. 实验环境

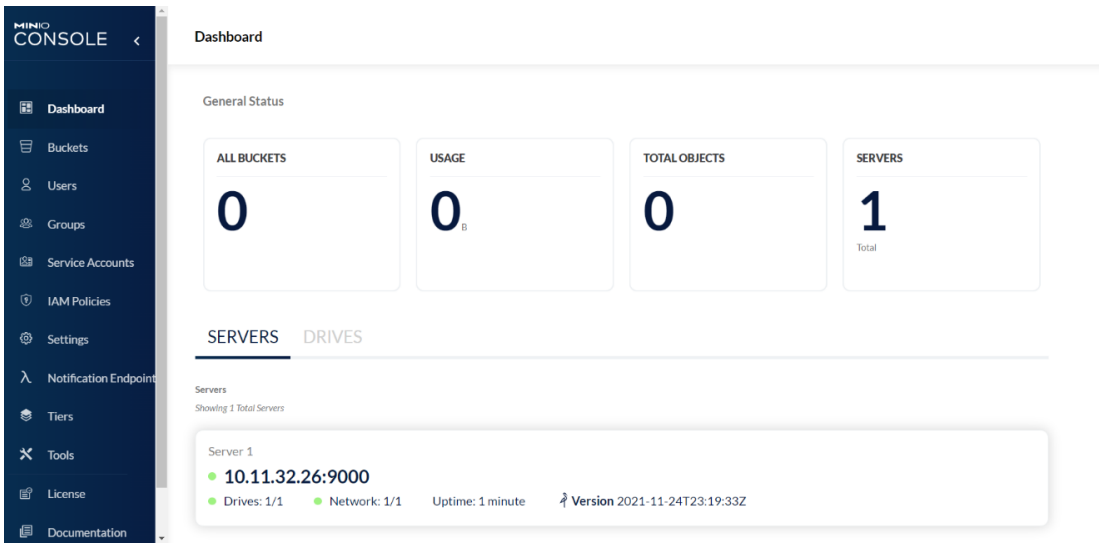
   CPU：11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz 3.11 GHz

   内存：16GB

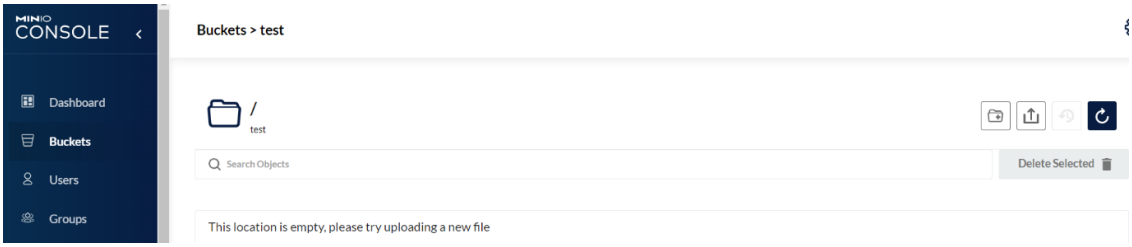
   系统：Windows10 64 位

2. 下载 MinIO，在 bin 目录下创建 data 文件夹作为数据存储的文件夹，以管理员身份打开 cmd，运行 run-minio.cmd 程序，搭建 MinIO 服务器。服务器成功启动后会在 cmd 输出默认的用户名和密码，为 hust、hust_obs。

3. 在浏览器中访问 cmd 输出的网址 http://127.0.0.1:9090，进入 MinIO 服务器端的图形管理界面。



4. 在 buckets 面板上创建一个名为“test”的 bucket 桶，用于性能测试。



二、 性能测试

1. 运行 run-s3bench.cmd 程序开始 s3bench 基准测试，文件内容如下所示：

run-s3bench.cmd - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
@rem -accessKey       Access Key
@rem -accessSecret    Secret Key
@rem -bucket=loadgen  Bucket for holding all test objects.
@rem -endpoint=http://127.0.0.1:9000 Endpoint URL of object storage service being tested.
@rem -numClients=8    Simulate 8 clients running concurrently.
@rem -numSamples=256  Test with 256 objects.
@rem -objectNamePrefix=loadgen Name prefix of test objects.
@rem -objectSize=1024        Size of test objects.
@rem -verbose         Print latency for every request.

s3bench.exe ^
   -accessKey=hust ^
   -accessSecret=hust_obs ^
   -bucket=test ^
   -endpoint=http://127.0.0.1:9000 ^
   -numClients=8 ^
   -numSamples=256 ^
   -objectNamePrefix=loadgen ^
   -objectSize=1024
pause
```

C:\Windows\system32\cmd.exe

```
D:\Software\Environment\MinIO>s3bench.exe     -accessKey=hust     -accessSecret=hust_obs     -bucket=test     -endpoint=
http://127.0.0.1:9000     -numClients=8     -numSamples=256     -objectNamePrefix=loadgen     -objectSize=1024
Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           test
objectNamePrefix: loadgen
objectSize:       0.0010 MB
numClients:       8
numSamples:       256
verbose:      %!d(bool=false)

Generating in-memory sample data... Done (1.9964ms)

Running Write test...

Running Read test...

Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           test
objectNamePrefix: loadgen
objectSize:       0.0010 MB
numClients:       8
numSamples:       256
verbose:      %!d(bool=false)
```

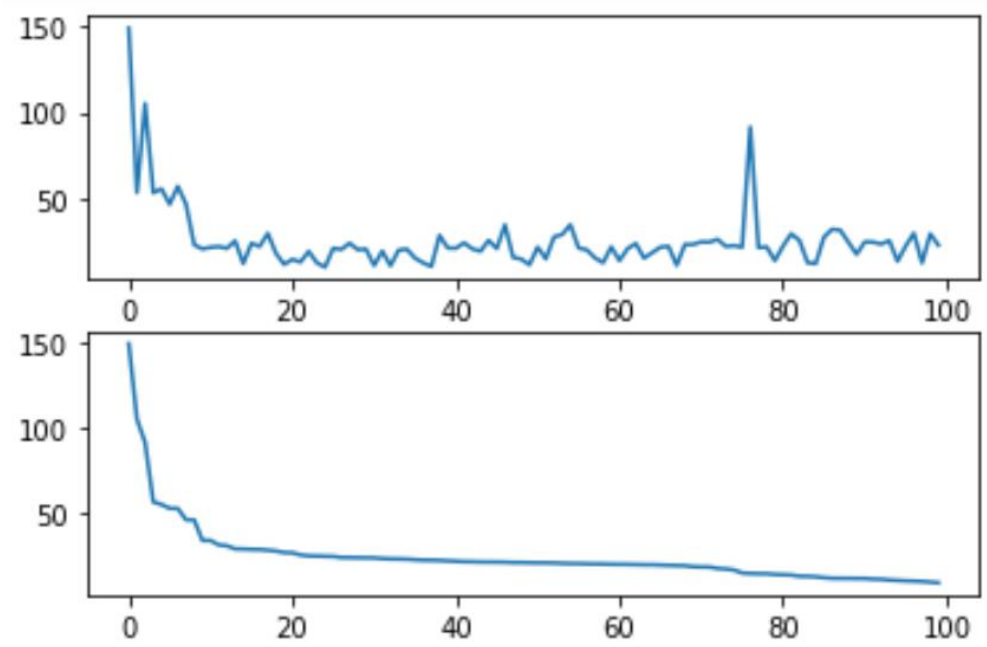
C:\Windows\system32\cmd.exe

```
Results Summary for Write Operation(s)
Total Transferred: 0.250 MB
Total Throughput:  0.27 MB/s
Total Duration:    0.922 s
Number of Errors:  0
------------------------------------
Write times Max:       0.079 s
Write times 99th %ile: 0.064 s
Write times 90th %ile: 0.057 s
Write times 75th %ile: 0.042 s
Write times 50th %ile: 0.027 s
Write times 25th %ile: 0.012 s
Write times Min:       0.007 s


Results Summary for Read Operation(s)
Total Transferred: 0.250 MB
Total Throughput:  4.54 MB/s
Total Duration:    0.055 s
Number of Errors:  0
------------------------------------
Read times Max:       0.005 s
Read times 99th %ile: 0.003 s
Read times 90th %ile: 0.002 s
Read times 75th %ile: 0.002 s
Read times 50th %ile: 0.002 s
Read times 25th %ile: 0.001 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 444.6202ms
```

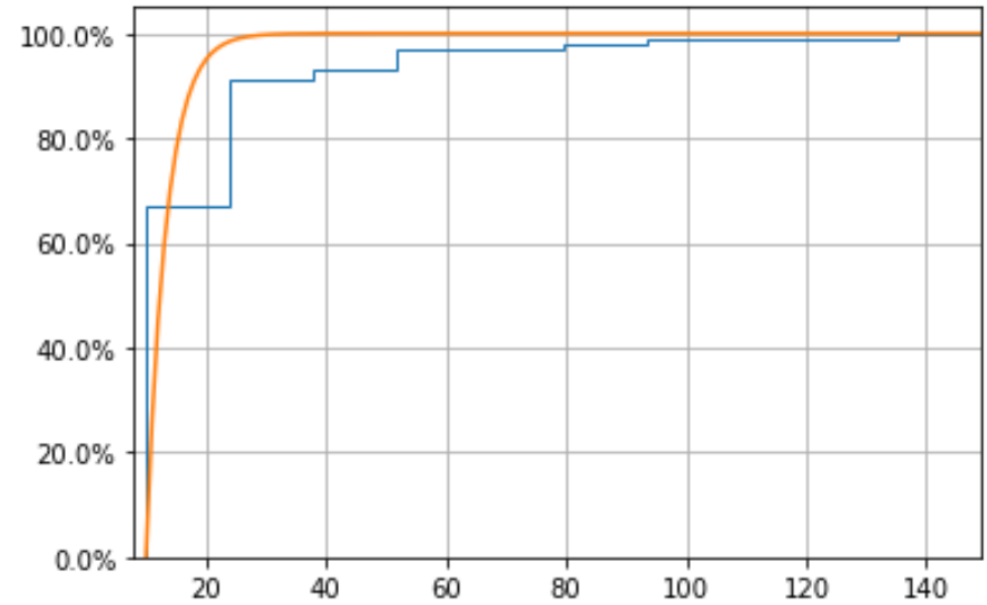
从执行结果不难看出，ObjectSize=1024 时，写操作吞吐率为 0.27MB/s，总耗时 0.922 秒。写操作最长耗时 0.079 秒，最短耗时 0.007 秒。99%的写操作在 0.064 秒内完成，90%的操作在 0.057 秒内完成。

三、 尾延迟挑战

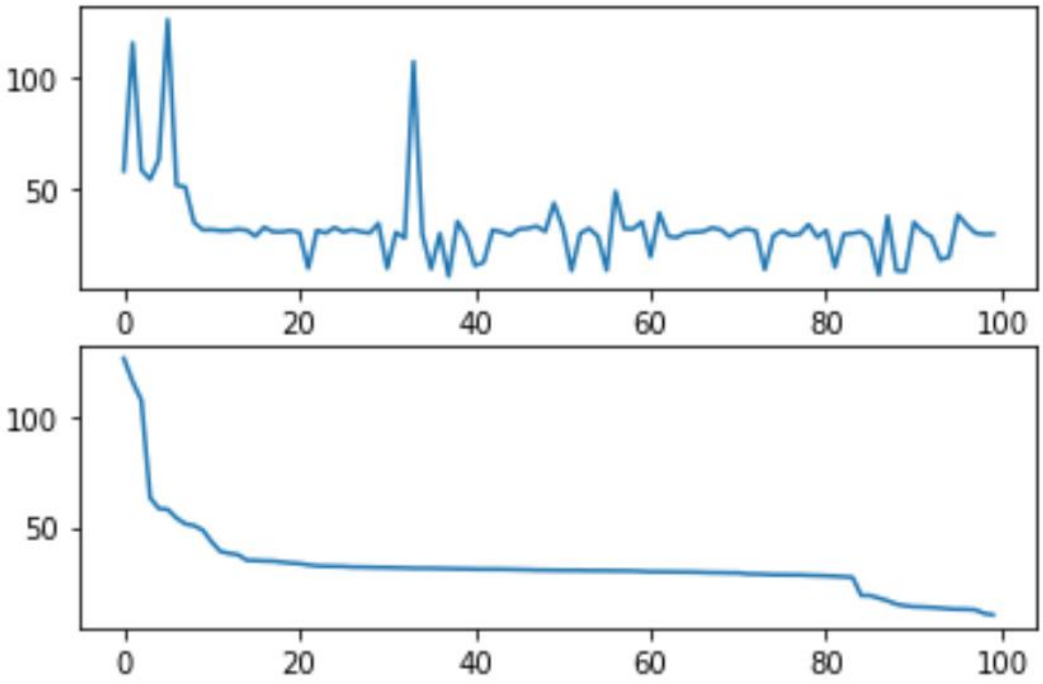
对 latency-collect.ipynb 文件和 latency-plot.ipynb 文件的代码进行修改，当有 100 项上传任务时，观察到的尾延迟分布数据如图所示：



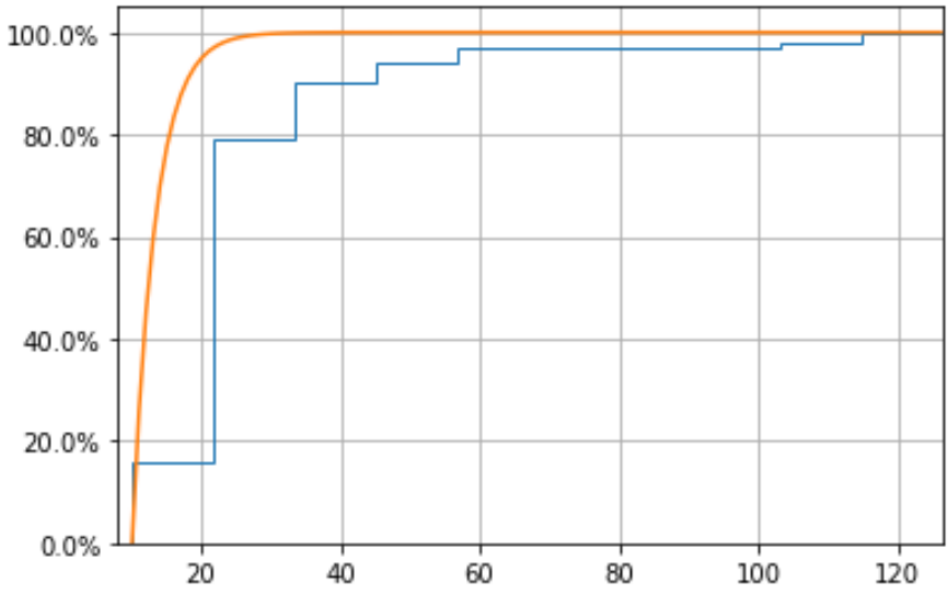
用排队论模型来拟合实际数据：



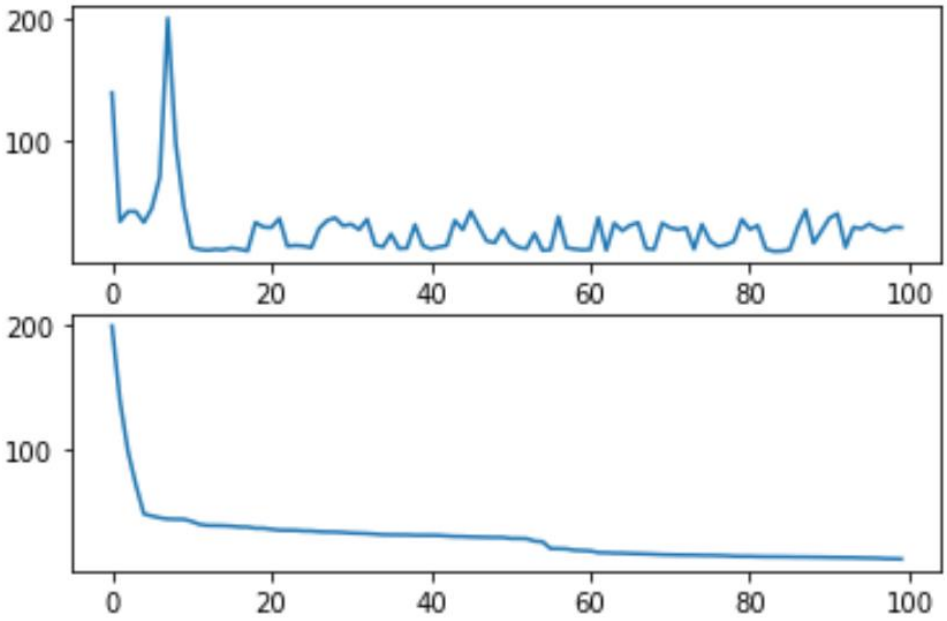
当有 500 项上传任务时，观察到的尾延迟分布数据如图所示：



用排队论模型来拟合实际数据：



当有 1000 项上传任务时，观察到的尾延迟分布数据如图所示：



用排队论模型来拟合实际数据：