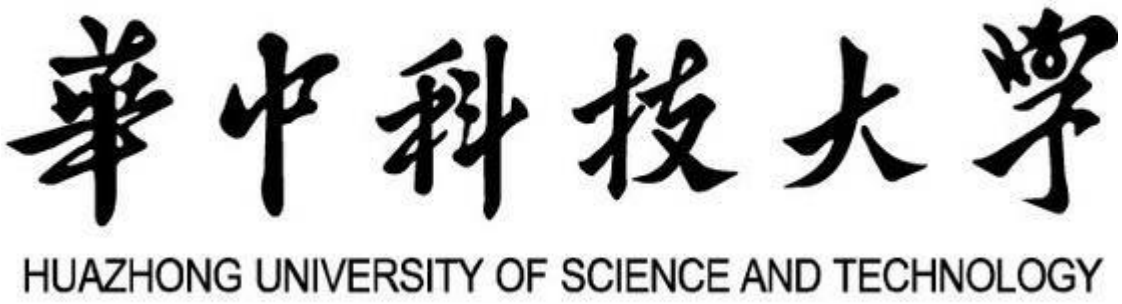


数据中心技术实验报告

学院	计算机科学与技术学院
班级	计算机硕 2021
指导老师	施展 童薇
姓名	李彬弘
学号	M202173755

2021 年 1 月 7 日

实验一、系统搭建

1.服务器搭建：Minio

使用 Homebrew 安装 Minio 软件包

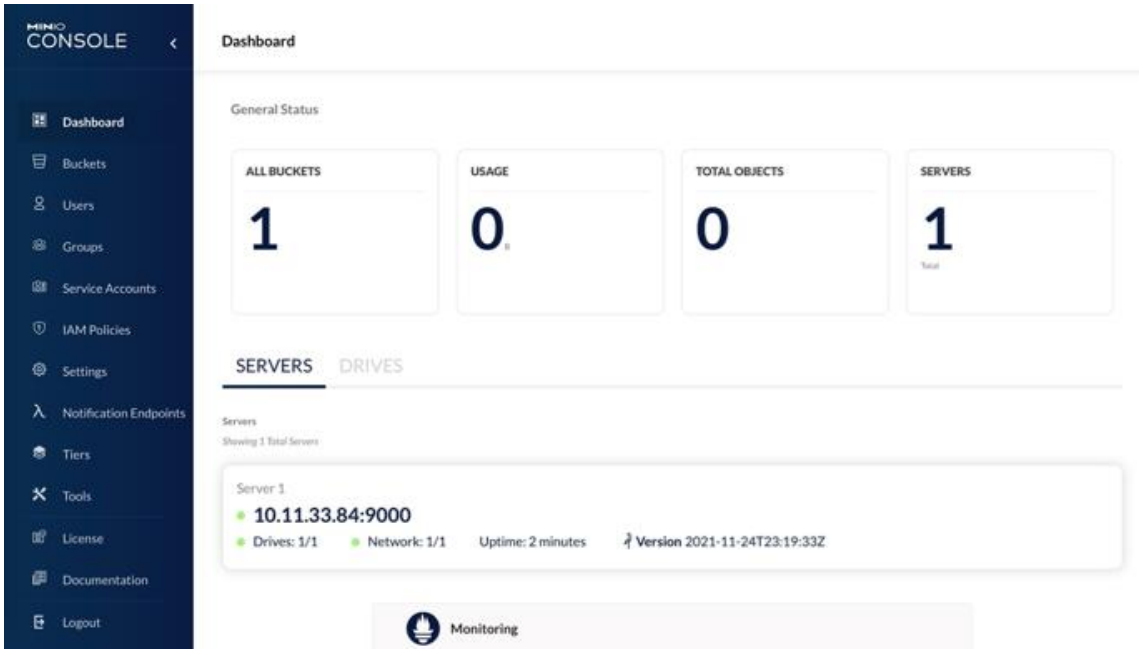
```
brew install minio/stable/minio
minio server /Users/rihiroshi/data
```

```
API: http://10.11.33.84:9000  http://127.0.0.1:9000
RootUser: minioadmin
RootPass: minioadmin

Console: http://10.11.33.84:54867 http://127.0.0.1:54867
RootUser: minioadmin
RootPass: minioadmin

Command-line: https://docs.min.io/docs/minio-client-quickstart-guide
   $ mc alias set myminio http://10.11.33.84:9000 minioadmin minioadmin

Documentation: https://docs.min.io

WARNING: Console endpoint is listening on a dynamic port (54867), please use --c
onsole-address ":PORT" to choose a static port.
WARNING: Detected default credentials 'minioadmin:minioadmin', we recommend that
 you change these values with 'MINIO_ROOT_USER' and 'MINIO_ROOT_PASSWORD' enviro
nment variables
```

2.下载资料库

```
git clone https://gitee.com/shi_zhan/obs-tutorial
```

3.打开浏览器进入 https://127.0.0.1:9000 输入账号密码



4.安装 Minio Client

```
brew install minio/stable/mc
```

下图所示为 mc 的命令

```
NAME:
  mc - MinIO Client for cloud storage and filesystems.

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

COMMANDS:
  alias      set, remove and list aliases in configuration file
  ls         list buckets and objects
  mb         make a bucket
  rb         remove a bucket
  cp         copy objects
  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
  share      generate URL for temporary access to an object
  find       search for objects
  sql        run sql queries on objects
  stat       show object metadata
  mv         move objects
  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)
  diff       list differences in object name, size, and date between two buckets
  rm         remove objects
  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)
  replicate  configure server side bucket replication
  admin      manage MinIO servers
  update     update mc to latest release

GLOBAL FLAGS:
  --autocompletion              install auto-completion for your shell
  --config-dir value, -C value  path to configuration folder (default: "/Users/rihiroshi/.mc")
  --quiet, -q                   disable progress bar display
  --no-color                    disable color theme
  --json                        enable JSON lines formatted output
  --debug                       enable debug output
  --insecure                    disable SSL certificate verification
  --help, -h                    show help
  --version, -v                 print the version
```

5.添加服务器

```
rihiroshi@lihongdeMacBook-Pro ~ % mc alias set myminio http://10.11.33.84:9000 minioadmin minioadmin
Added `myminio` successfully.
```

创建一个桶

```
rihiroshi@lihongdeMacBook-Pro ~ % mc mb myminio/mybucket
Bucket created successfully `myminio/mybucket`.
```

显示对象

```
rihiroshi@lihongdeMacBook-Pro ~ % mc ls myminio/
[2022-01-04 11:43:35 CST]     0B mybucket/
```

实验二、性能观测

1.安装 S3 Bench

```
go get -u github.com/igneous-systems/s3bench
```

2.修改配置参数

```
s3bench \
  -accessKey=minioadmin -accessSecret=minioadmin \
  -endpoint=http://10.11.33.84:9000 \
  -bucket=mybucket -objectNamePrefix=mybucket \
  -numClients=10 -numSamples=100 -objectSize=1024
```

3.运行 run-s3bench.sh（参数线程 10，测试 100 个对象，对象大小 1KB）

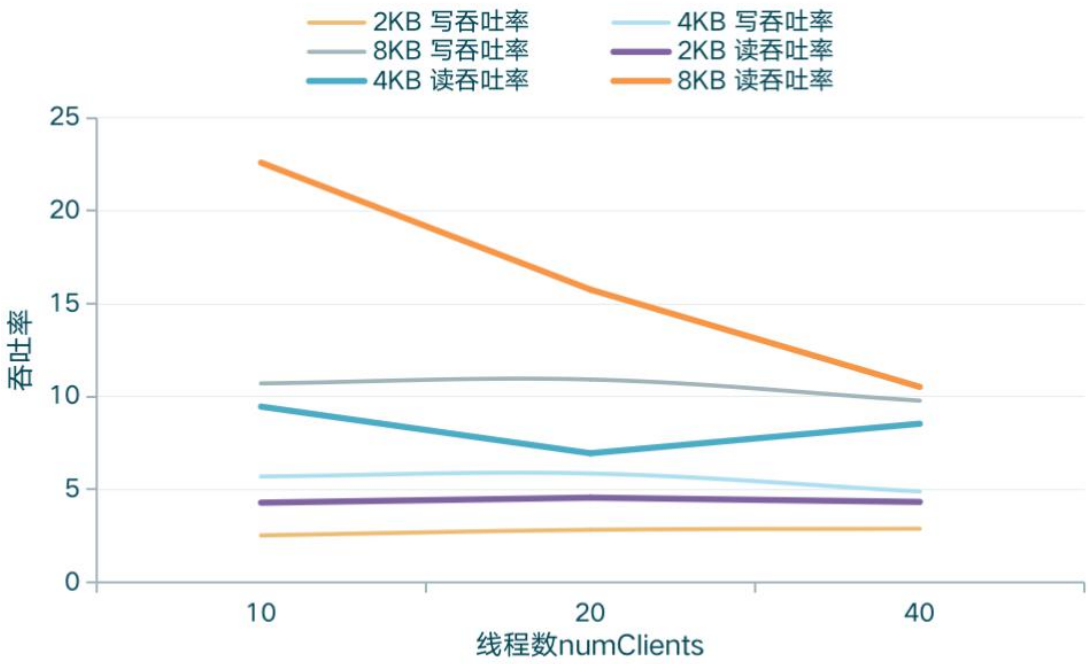
```
rihiroshi@lihongdeMacBook-Pro obs-tutorial % sh run-s3bench.sh
Test parameters
endpoint(s):      [http://10.11.33.84:9000]
bucket:           mybucket
objectNamePrefix: mybucket
objectSize:       0.0010 MB
numClients:       10
numSamples:       100
verbose:       %!d(bool=false)


Generating in-memory sample data... Done (58.142µs)

Running Write test...

Running Read test...

Test parameters
endpoint(s):      [http://10.11.33.84:9000]
bucket:           mybucket
objectNamePrefix: mybucket
objectSize:       0.0010 MB
numClients:       10
numSamples:       100
verbose:       %!d(bool=false)


Results Summary for Write Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  1.19 MB/s
Total Duration:    0.082 s
Number of Errors:  0
------------------------------------
Write times Max:       0.026 s
Write times 99th %ile: 0.026 s
Write times 90th %ile: 0.017 s
Write times 75th %ile: 0.009 s
Write times 50th %ile: 0.006 s
Write times 25th %ile: 0.004 s
Write times Min:       0.002 s


Results Summary for Read Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  2.53 MB/s
Total Duration:    0.039 s
Number of Errors:  0
------------------------------------
Read times Max:       0.008 s
Read times 99th %ile: 0.008 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: 0.003 s
Read times Min:       0.001 s


Cleaning up 100 objects...
Deleting a batch of 100 objects in range {0, 99}... Succeeded
Successfully deleted 100/100 objects in 126.430697ms
```

4.检测参数对指标的影响

参数：对象尺寸 objectSize、线程数 numClients
指标：延迟 latency、吞吐率 Throughput
实验数据:

numClients	objectSize
10	2KB、4KB、8KB
20	2KB、4KB、8KB
40	2KB、4KB、8KB

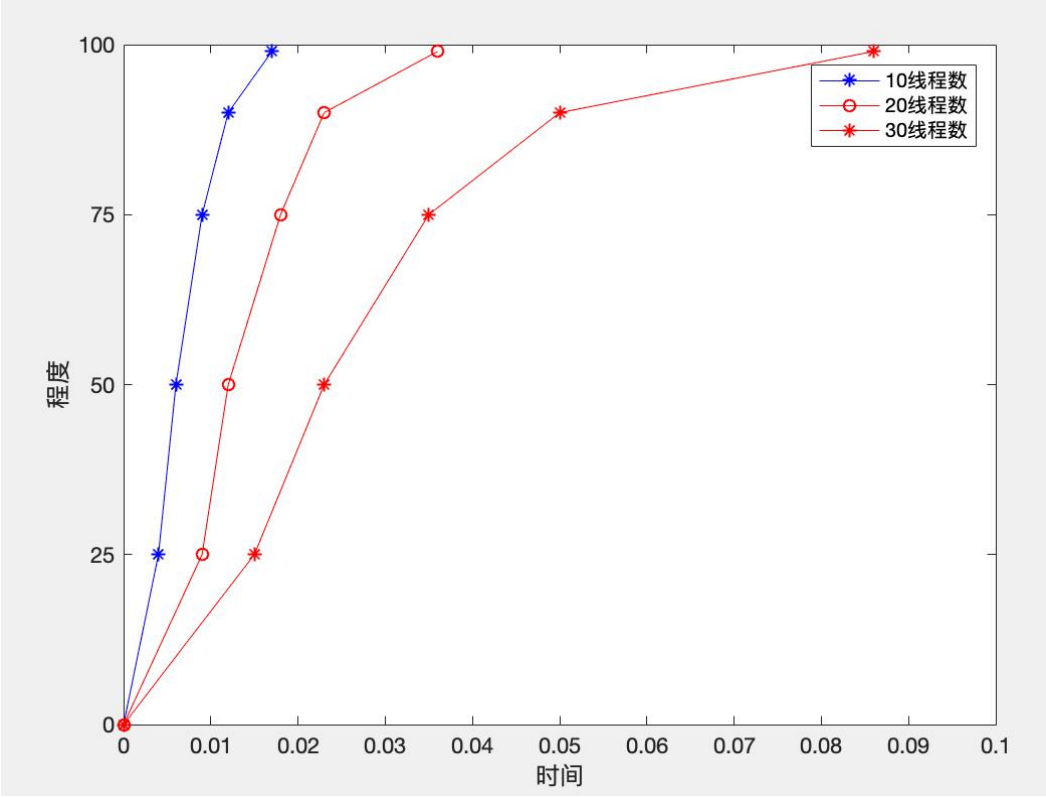
4.1 吞吐率 Throughput



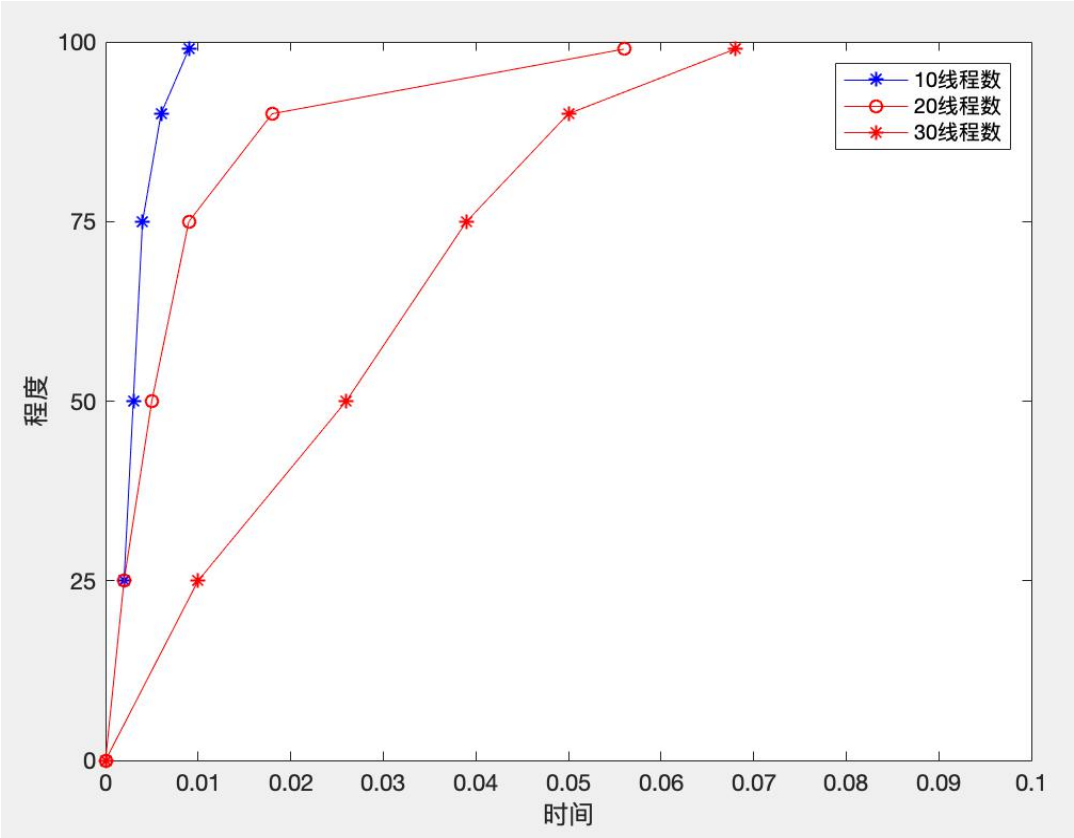
4.2 延迟 latency

4.2.1 相同对象尺寸，不同线程数对延迟的影响

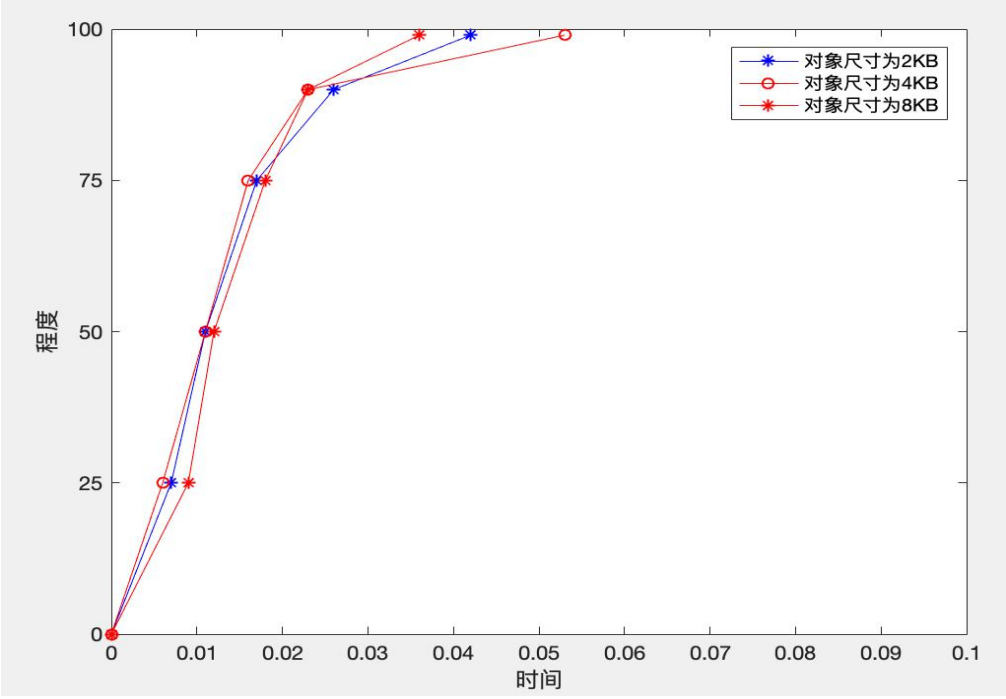
对象尺寸均为 8KB，线程数分别为 10，20，40 时对写延迟的影响如图所示



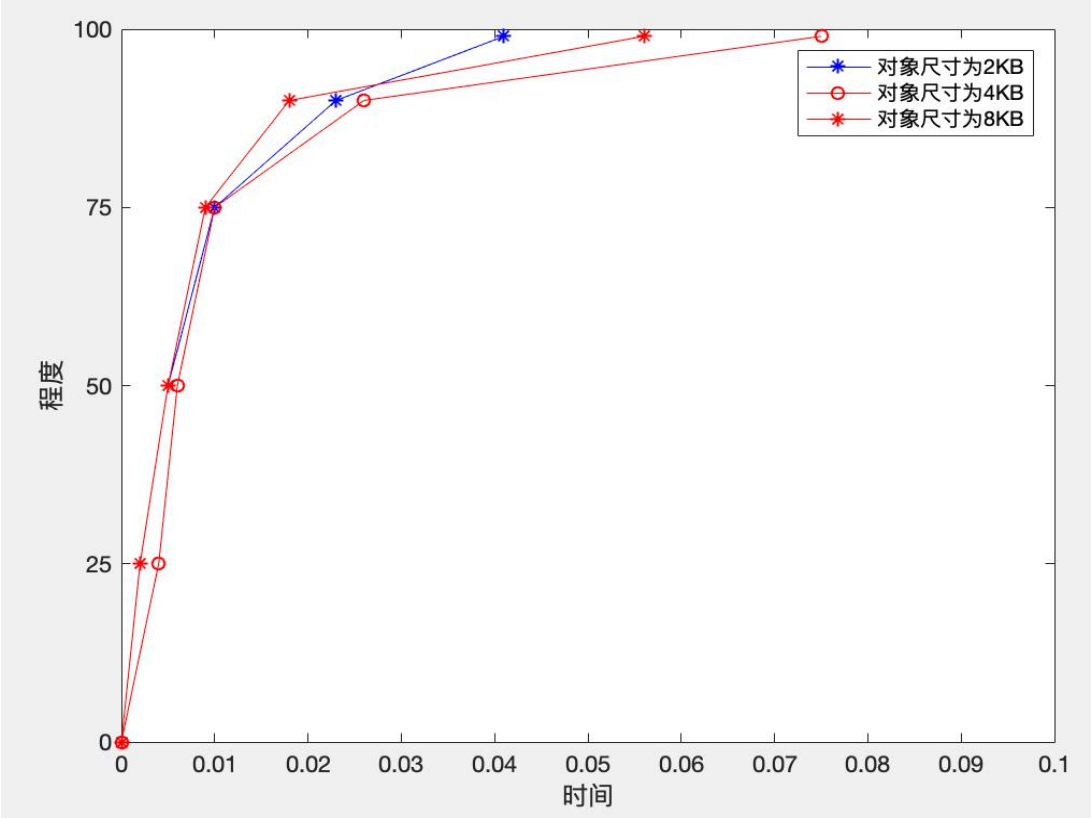
对象尺寸均为 8KB，线程数分别为 10，20，40 时对读延迟的影响如图所示



线程数均为 20，对象尺寸为 2，4，8KB 时对写延迟的影响如图所示



线程数均为 20，对象尺寸为 2，4，8KB 时对读延迟的影响如图所示



4.3 总结

观察可得:

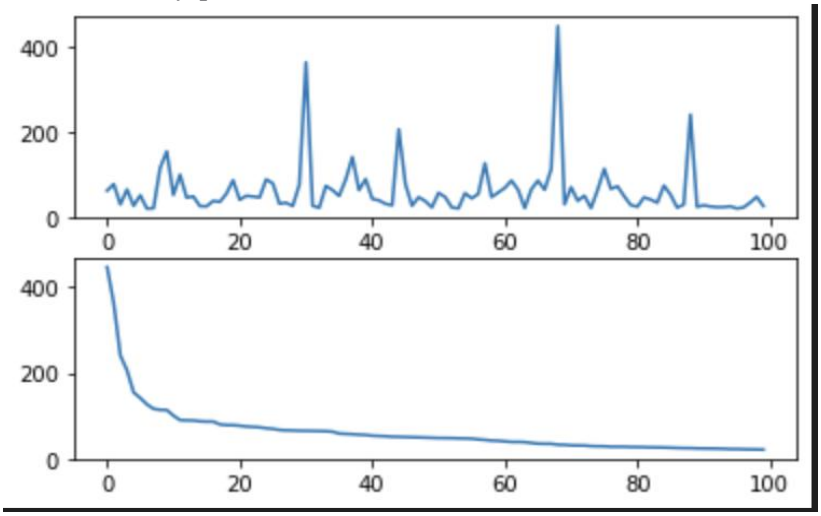
对象尺寸 objectSize 越大，吞吐量 Throughput 越大;

并发数 numClients 越大，尾延迟 latency 越大。

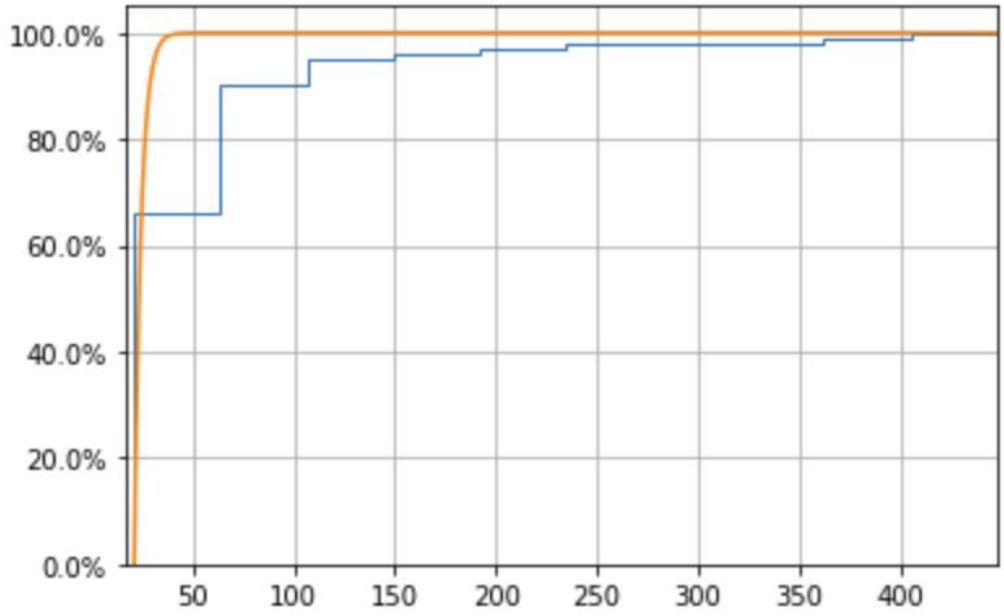
实验三、尾延迟挑战

运行 latency-collect 程序进行实验，并收集数据至 latency.csv 中

再运行 latency-plot 程序，下图为请求延迟分布情况



用排队论来拟合实测模型



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，超时后重新发送请求，得到如图所示结果，尾延迟有了明显的改进