S

華中科技大學

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 2107

学　号 M202173757

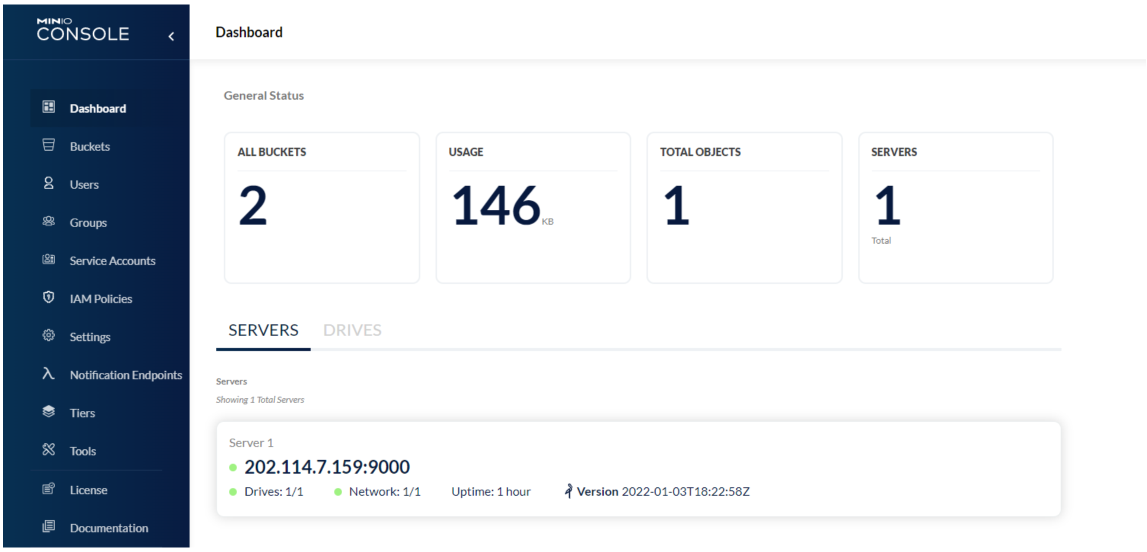
姓　名 王志伟

2022 年 01 月 05 日

基于 Minio 的存储系统性能测试实验

一、 系统搭建

首先安装好 go 环境，下载 Minio 服务端和客户端，进入 minio.exe 所在目录，使用命令“.\minio.exe server D:\minio”运行服务器，使用浏览器在给定的地址中登录，用户名与密码均为默认值 minioadmin。在创建的服务器中新建一个名为“folder1”的桶，完成系统搭建。



ALL BUCKETS 2

USAGE 146 KB

TOTAL OBJECTS 1

SERVERS 1 Total

Servers

Showing 1 Total Servers

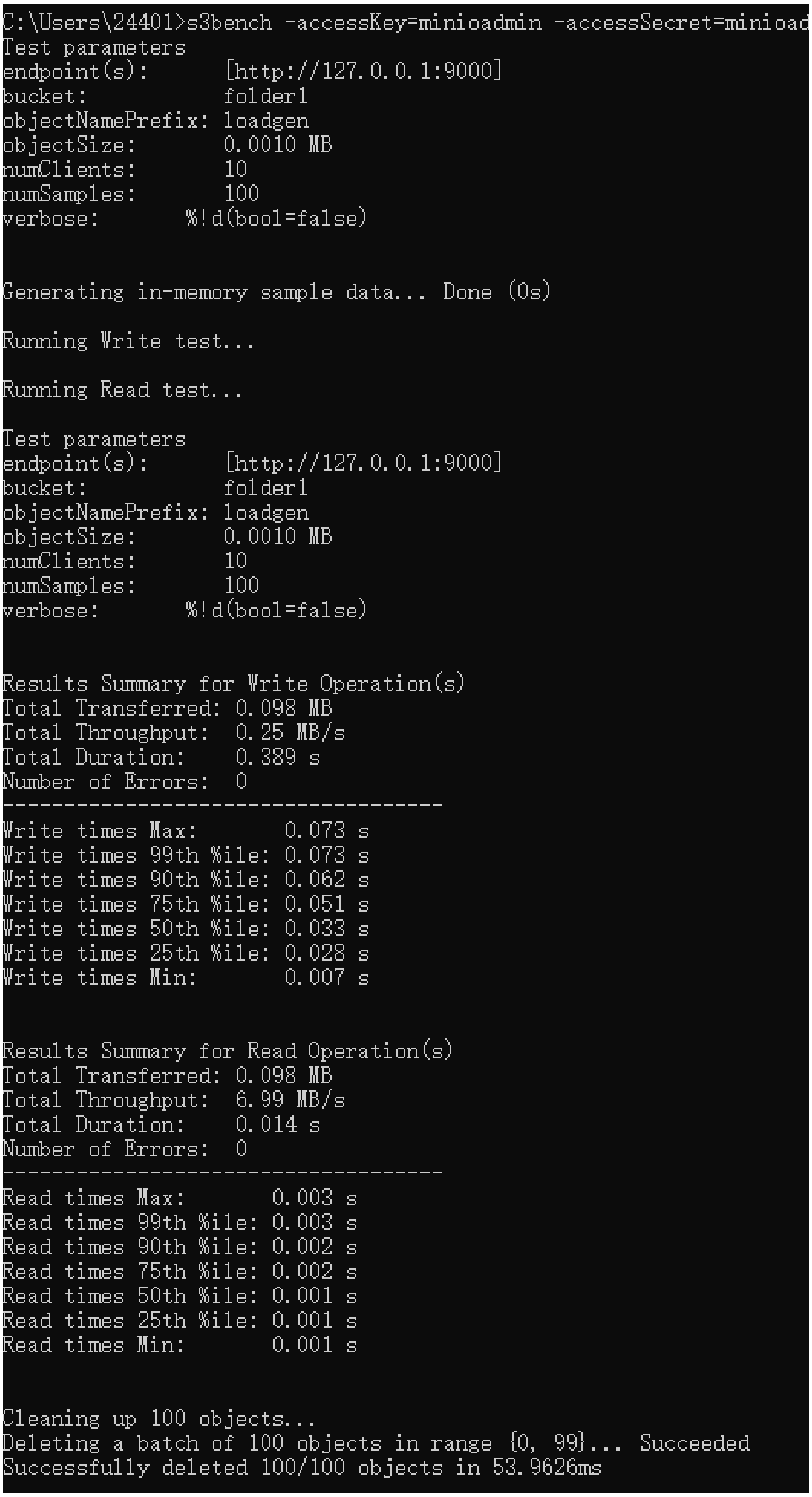
Server 1

202.114.7.159:9000

Drives: 1/1 Network: 1/1 Uptime: 1 hour Version 2022-01-03T18:22:58Z

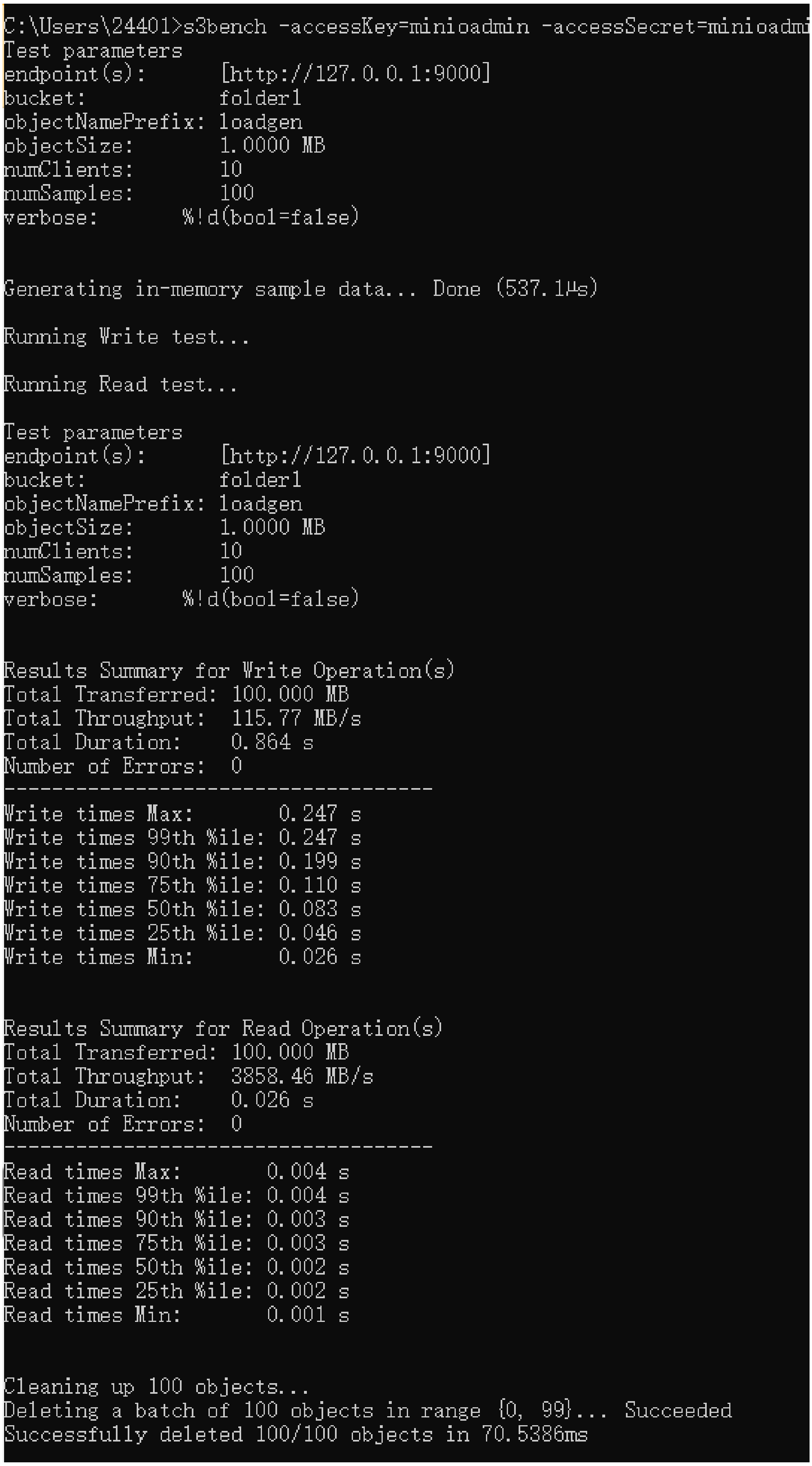
二、 s3bench 基准测试

使用命令“s3bench -accessKey=minioadmin -accessSecret= minioadmin -endpoint= http://127.0.0.1:9000 -bucket=folder1 -objectNamePrefix= loadgen -numClients= 10 -numSamples=100 -objectSize=1024”进行测试，并修改 objectSize 进行多次测试，分别为 1KB，1MB，结果如下图所示。ObjectSize=1024 时，写操作共传输 0.098MB，吞吐率为 0.25MB/s，总耗时 0.389s，99%的文件耗时 0.073s，90%的文件耗时 0.062s。对于读操作，共传输 0。098MB，吞吐率 6.99MB/s，总耗时 0.014s，99%的文件耗时 0.003s，90%的文件耗时 0.002s。



```
C:\Users\24401>s3bench -accessKey=minioadmin -accessSecret=minioad
Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           folder1
objectNamePrefix: loadgen
objectSize:       0.0010 MB
numClients:       10
numSamples:       100
verbose:       %!d(bool=false)


Generating in-memory sample data... Done (0s)

Running Write test...

Running Read test...

Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           folder1
objectNamePrefix: loadgen
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:  0.25 MB/s
Total Duration:    0.389 s
Number of Errors:  0
------------------------------------
Write times Max:       0.073 s
Write times 99th %ile: 0.073 s
Write times 90th %ile: 0.062 s
Write times 75th %ile: 0.051 s
Write times 50th %ile: 0.033 s
Write times 25th %ile: 0.028 s
Write times Min:       0.007 s


Results Summary for Read Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  6.99 MB/s
Total Duration:    0.014 s
Number of Errors:  0
------------------------------------
Read times Max:       0.003 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: 0.001 s
Read times 25th %ile: 0.001 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 53.9626ms
```

ObjectSize =1KB 实验结果



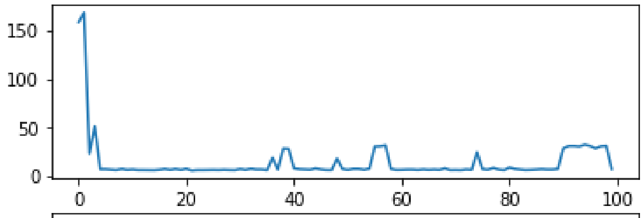
```
C:\Users\24401>s3bench -accessKey=minioadmin -accessSecret=minioadmi
Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           folder1
objectNamePrefix: loadgen
objectSize:       1.0000 MB
numClients:       10
numSamples:       100
verbose:       %!d(bool=false)


Generating in-memory sample data... Done (537.1µs)

Running Write test...

Running Read test...

Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           folder1
objectNamePrefix: loadgen
objectSize:       1.0000 MB
numClients:       10
numSamples:       100
verbose:       %!d(bool=false)


Results Summary for Write Operation(s)
Total Transferred: 100.000 MB
Total Throughput:  115.77 MB/s
Total Duration:    0.864 s
Number of Errors:  0
------------------------------------
Write times Max:       0.247 s
Write times 99th %ile: 0.247 s
Write times 90th %ile: 0.199 s
Write times 75th %ile: 0.110 s
Write times 50th %ile: 0.083 s
Write times 25th %ile: 0.046 s
Write times Min:       0.026 s


Results Summary for Read Operation(s)
Total Transferred: 100.000 MB
Total Throughput:  3858.46 MB/s
Total Duration:    0.026 s
Number of Errors:  0
------------------------------------
Read times Max:       0.004 s
Read times 99th %ile: 0.004 s
Read times 90th %ile: 0.003 s
Read times 75th %ile: 0.003 s
Read times 50th %ile: 0.002 s
Read times 25th %ile: 0.002 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 70.5386ms
```

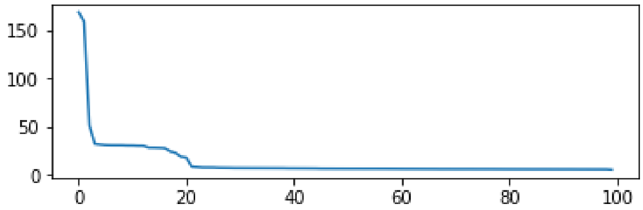
ObjectSize =1MB 实验结果

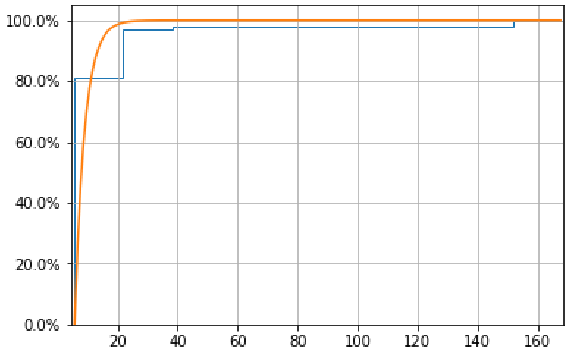
三、 尾延迟观测

对 latency-collect 和 latency-plot 的代码进行修改，生成几种不同大小的数据包对应的数据，分别为 1KB、1MB，通过执行 latency-collect 获取尾延迟分布数据，接着执行 latency-plot 画出延迟分布图像和排队论模型预测。

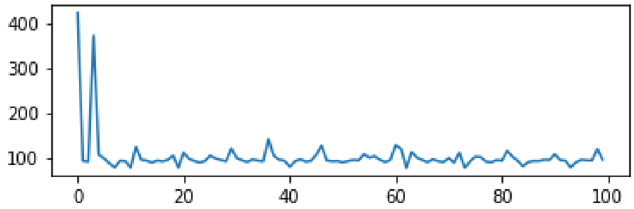
1KB 包:

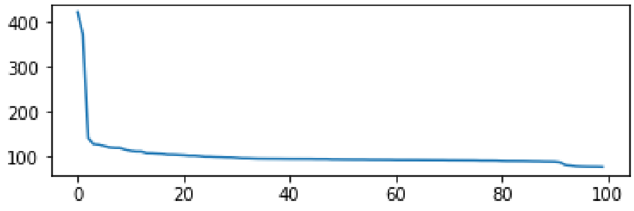


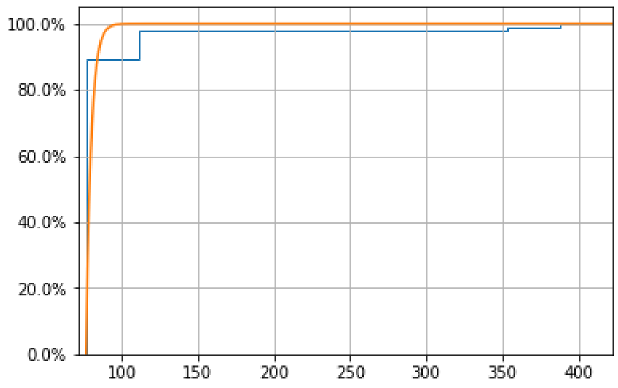




1MB 包:







上图显示的是延迟的波动情况，下图是用排队论拟合实测数据得到的情况。可以看到，对于1KB包，在80%的情况下，延迟在8毫秒以内，60%的情况下，延迟在7毫秒以内；对于1MB包，80%的情况下，延迟在30毫秒以内，60%的情况下，延迟在25毫秒以内。