

华中科技大学

# 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

学 号 M202173723

姓 名 李贽

2021 年 12 月 31 日

# 基于 MinIO 的存储系统性能测试实验

## 一、 实验目的

熟悉性能指标：吞吐率、带宽、延迟

分析不同负载下的指标、延迟的分布

观测尾延迟现象

尝试对冲请求方案

## 二、 系统搭建

执行 run-minio.cmd 命令搭建了 MinIO 服务器，取得服务器地址。用设定的用户名和密码通过 MinIO 给出的服务器地址登录。在新创建的 MinIO 服务器中新建一个名为“loadgen”的 Bucket，完成系统搭建和预备操作。



```
C:\WINDOWS\system32\cmd.exe
API: http://192.168.0.104:9000 http://192.168.56.1:9000 http://192.168.85.1:9000 http://127.0.0.1:9000
RootUser: hust
RootPass: hust_obs

Console: http://192.168.0.104:9090 http://192.168.56.1:9090 http://192.168.85.1:9090 http://127.0.0.1:9090
RootUser: hust
RootPass: hust_obs

Command-line: https://docs.min.io/docs/minio-client-quickstart-guide
$ mc.exe alias set myminio http://192.168.0.104:9000 hust hust_obs

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

Server 1

- 192.168.0.104:9000
- Drives: 1/1
- Network: 1/1
- Uptime: 1 minute
- Version 2021-11-24T23:19:33Z

## 三、 s3bench 基准测试

执行 run-s3bench.cmd 命令开始 s3bench 基准测试。

```

C:\WINDOWS\system32\cmd.exe
numSamples:      256
verbose:         %!d(bool=false)

Generating in-memory sample data... Done (3.0059ms)

Running Write test...

Running Read test...

Test parameters
endpoint(s):     [http://127.0.0.1:9000]
bucket:          loadgen
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:  0.14 MB/s
Total Duration:    1.749 s
Number of Errors:  0
-----
Write times Max:      0.115 s
Write times 99th %ile: 0.107 s
Write times 90th %ile: 0.083 s
Write times 75th %ile: 0.069 s
Write times 50th %ile: 0.053 s
Write times 25th %ile: 0.042 s
Write times Min:      0.010 s

Results Summary for Read Operation(s)
Total Transferred: 0.250 MB
Total Throughput:  2.37 MB/s
Total Duration:    0.106 s
Number of Errors:  0
-----
Read times Max:       0.012 s
Read times 99th %ile: 0.008 s
Read times 90th %ile: 0.005 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 188.9215ms

D:\minioserver>pause

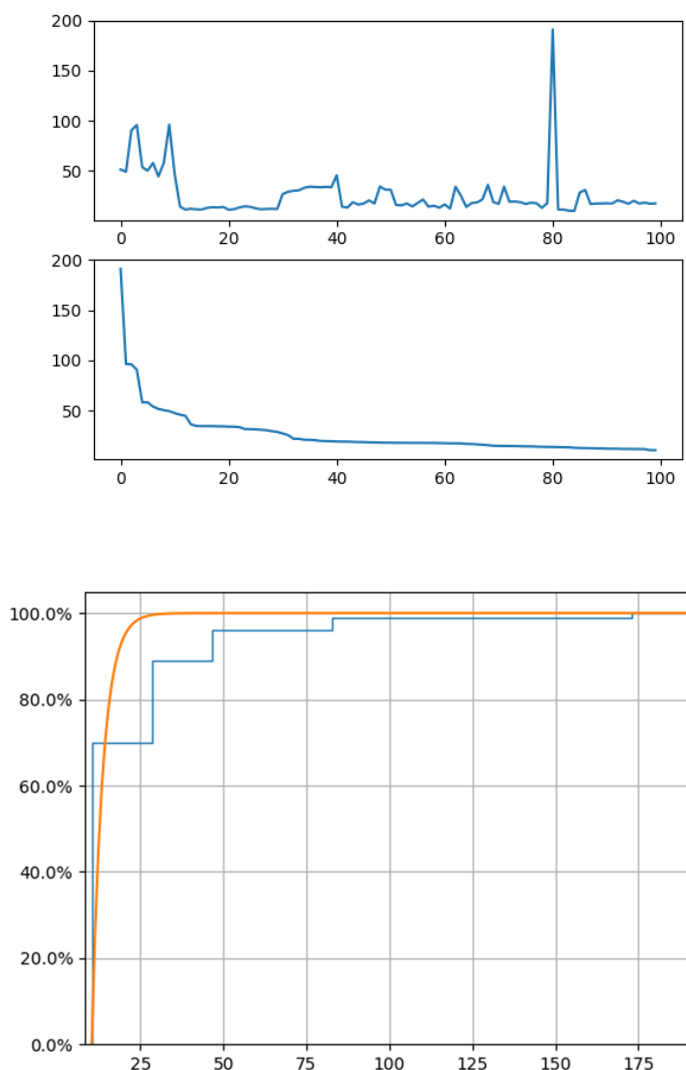
```

从执行结果不难看出，总的文件大小为 0.25MB，共计 256 个。在写操作中：吞吐率为 0.14MB/s，总耗时 1.749 秒。写操作最长耗时 0.115 秒，最短耗时 0.010 秒。99%的写操作在 0.107 秒内完成，90%的操作在 0.083 秒内完成。

对于读操作而言：吞吐率为 2.37MB/s，总耗时 0.106 秒。写操作最长耗时 0.012 秒，最短耗时 0.001 秒。99%的写操作在 0.008 秒内完成，90%的操作在 0.005 秒内完成。

#### 四、 尾延迟观测

对 latency-collect 和 latency-plot 的代码进行修改以适用于自己搭建的 MiniIO 服务器，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。



从第一张图中上面的图像可以看出，在整个实验过程中，一开始延迟偏大且波动剧烈，在基本稳定后突然出现了短时间延迟急剧升高的情况。第一张图中下面的图像将延迟分布进行整理，可以看出，更多时候延迟较低，延迟高的情况相对更少见。

第二张图是用排队论拟合实测数据得到的情况。可以看到，在 60%的情况下，延迟在 8 毫秒以内。如果想覆盖 80%的情况，则需要保证延迟在 15 秒以内。可以看到在本实验的环境中，延迟整体偏低，但尾延迟的现象仍然存在。

## 五、 实验结论

1、观测了系统的吞吐率、带宽、延迟，发现大部分（90%）任务都能在较短时间内完成。

2、尾延迟确实存在，可以观测。尾延迟可能是程序设计本身导致的毛病，但是，即便程序设计完全无误，尾延迟依然可能存在。硬件，操作系统本身，都可能导致尾延迟响应，例如：主机系统其他进程的影响，应用程序里线程调度，CPU 功耗设计等等。