

华中科技大学

数据中心技术课程实验报告

学 号 M202173861

姓 名 徐 勛

专 业 电子信息

课程指导教师 施展 童薇

院（系、所） 计算机科学与技术

2021 年 1 月 6 日

一. 运行环境搭建

1. 实验环境

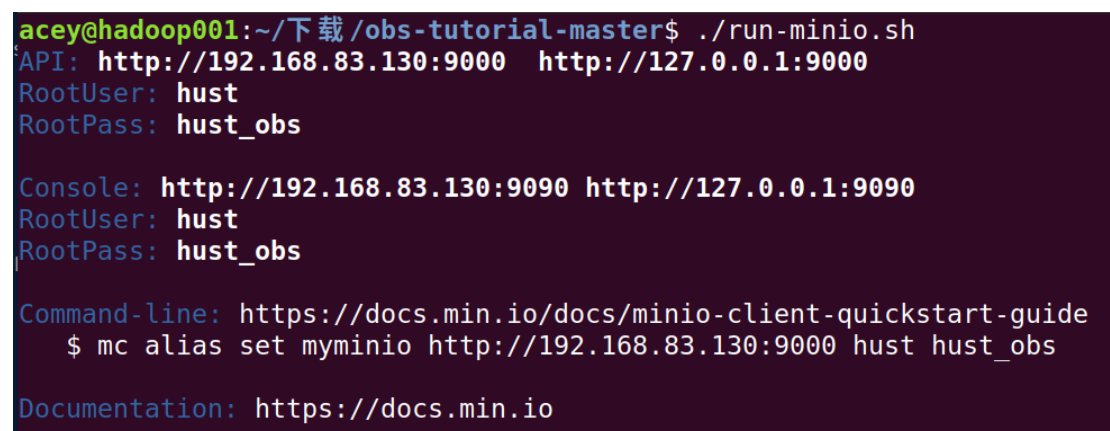
处理器：Intel Core i7-8650U CPU 1.90GHz

内存：3.8GB

操作系统：Ubuntu 18.04

2. 服务端搭建

在官网按照教程在终端下载并安装 minio，在终端执行 `./run-minio.sh` 启动 minio，信息如下图 1 所示，用户名默认为 `hust`，密码默认为 `hust_obs`，web-gui 端口为 9090。



```
acey@hadoop001:~/下载/obs-tutorial-master$ ./run-minio.sh
API: http://192.168.83.130:9000 http://127.0.0.1:9000
RootUser: hust
RootPass: hust_obs

Console: http://192.168.83.130:9090 http://127.0.0.1:9090
RootUser: hust
RootPass: hust_obs

Command-line: https://docs.min.io/docs/minio-client-quickstart-guide
$ mc alias set myminio http://192.168.83.130:9000 hust hust_obs

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

图 1 终端执行 run-minio 脚本启动 minio

3. 实验配置

在浏览器中访问启动信息中的网址 <http://192.168.83.130:9090>，进入 minio 服务器端的 GUI 界面，如图 2 所示。

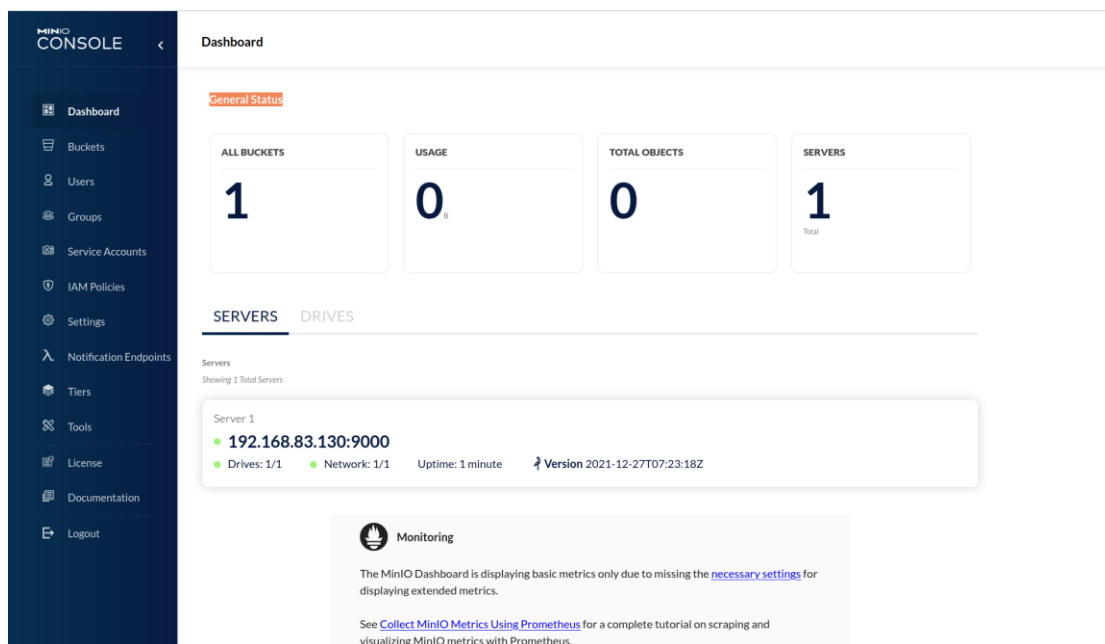


图 2 minio 的 web 界面

在 minio 的 web 界面添加名为 loadgen 的 bucket，后续性能测试是在此基础上进行的。

二. 性能测试

1. 标准脚本测试

修改 run-s3bench.sh 脚本，设置 numClients 为 8，numSamples 为 256，objectSize 为 1024，在终端执行脚本。终端显示如下。

```
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: 0.87 MB/s
Total Duration: 0.289 s
Number of Errors: 0
-----
Write times Max: 0.025 s
Write times 99th %ile: 0.020 s
Write times 90th %ile: 0.015 s
Write times 75th %ile: 0.011 s
Write times 50th %ile: 0.008 s
Write times 25th %ile: 0.006 s
Write times Min: 0.002 s

Results Summary for Read Operation(s)
Total Transferred: 0.250 MB
Total Throughput: 1.70 MB/s
Total Duration: 0.147 s
Number of Errors: 0
-----
Read times Max: 0.025 s
Read times 99th %ile: 0.025 s
Read times 90th %ile: 0.010 s
Read times 75th %ile: 0.006 s
Read times 50th %ile: 0.003 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 61.931662ms
```

图 3 numClients=8, numSamples=256, objectSize=1024

分析图中数据可知，最大时长往往比最小时长高出许多倍，同时也大大超出 90%基准线的标准，这些占比很小的数据传输很多时候会造成额外的时间成本。

2. 不同负载下指标、延迟的分布

使用控制变量法，更改负载，修改 numClients, numSamples 和 objectSize。执行 ./run-s3bench.sh，通过 s3bench 向 minio 发送请求，观测延迟分布情况。部分截图如图 4 图 5 所示。全部测试结果如表 1

所示。

```
Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           loadgen
objectNamePrefix: loadgen
objectSize:       0.0010 MB
numClients:       8
numSamples:       512
verbose:          %!d(bool=false)

Results Summary for Write Operation(s)
Total Transferred: 0.500 MB
Total Throughput:  0.96 MB/s
Total Duration:    0.522 s
Number of Errors:  0
-----
Write times Max:      0.032 s
Write times 99th %ile: 0.027 s
Write times 90th %ile: 0.013 s
Write times 75th %ile: 0.010 s
Write times 50th %ile: 0.007 s
Write times 25th %ile: 0.005 s
Write times Min:      0.002 s

Results Summary for Read Operation(s)
Total Transferred: 0.500 MB
Total Throughput:  1.88 MB/s
Total Duration:    0.265 s
Number of Errors:  0
-----
Read times Max:       0.027 s
Read times 99th %ile: 0.018 s
Read times 90th %ile: 0.009 s
Read times 75th %ile: 0.005 s
Read times 50th %ile: 0.003 s
Read times 25th %ile: 0.001 s
Read times Min:       0.001 s

Cleaning up 512 objects...
Deleting a batch of 512 objects in range {0, 511}... Succeeded
Successfully deleted 512/512 objects in 123.828952ms
```

图 4 numClients=8, numSamples=512, objectSize=1024

```
acey@hadoop001:~/下载/obs-tutorial-master$ bash run-s3bench.sh
Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           loadgen
objectNamePrefix: loadgen
objectSize:       0.0312 MB
numClients:       8
numSamples:       256
verbose:          %!d(bool=false)
```

```

Results Summary for Write Operation(s)
Total Transferred: 8.000 MB
Total Throughput: 10.47 MB/s
Total Duration: 0.764 s
Number of Errors: 0
-----
Write times Max: 0.104 s
Write times 99th %ile: 0.056 s
Write times 90th %ile: 0.037 s
Write times 75th %ile: 0.029 s
Write times 50th %ile: 0.022 s
Write times 25th %ile: 0.016 s
Write times Min: 0.005 s

Results Summary for Read Operation(s)
Total Transferred: 8.000 MB
Total Throughput: 19.27 MB/s
Total Duration: 0.415 s
Number of Errors: 0
-----
Read times Max: 0.050 s
Read times 99th %ile: 0.043 s
Read times 90th %ile: 0.026 s
Read times 75th %ile: 0.017 s
Read times 50th %ile: 0.010 s
Read times 25th %ile: 0.005 s
Read times Min: 0.002 s

Cleaning up 256 objects...
Deleting a batch of 256 objects in range {0, 255}... Succeeded
Successfully deleted 256/256 objects in 154.185789ms

```

图 5 numClients=8, numSamples=256, objectSize=1024*32

表 1 不同负载下指标、延迟的分布

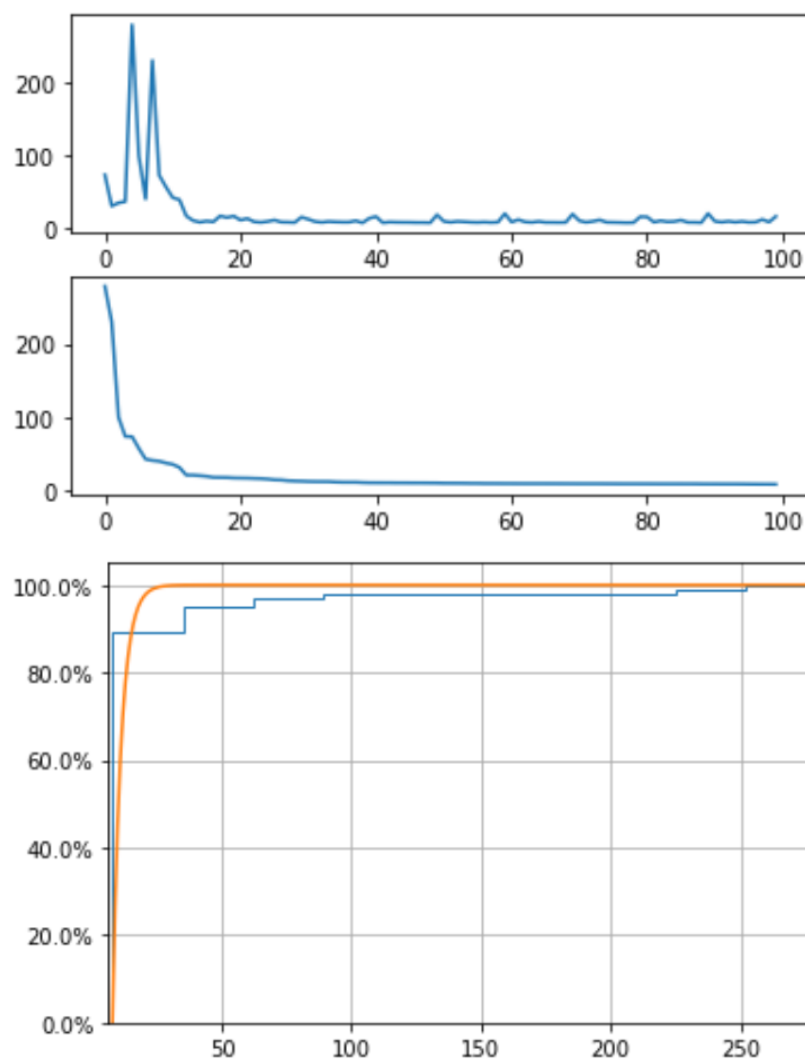
numClients	numSamples	objectSize	Write					Read				
			Throughput/MBps	Latency/s	50%/s	90%/s	99%/s	Throughput/MBps	Latency/s	50%/s	90%/s	99%/s
8	256	1024	0.87	0.289	0.008	0.015	0.020	1.70	0.147	0.003	0.010	0.025
		1024*2	1.83	0.273	0.007	0.014	0.031	3.73	0.134	0.002	0.009	0.021
		1024*4	3.32	0.301	0.008	0.015	0.030	6.37	0.157	0.004	0.010	0.018
		1024*8	7.72	0.259	0.007	0.014	0.021	11.84	0.169	0.004	0.010	0.027
	512	1024	0.96	0.522	0.007	0.013	0.027	1.88	0.265	0.003	0.009	0.018
		1024*2	2.11	0.473	0.006	0.013	0.020	3.18	0.315	0.003	0.010	0.028
		1024*4	3.91	0.512	0.007	0.014	0.024	7.66	0.261	0.003	0.009	0.016
		1024*8	7.97	0.502	0.007	0.014	0.022	15.79	0.253	0.003	0.009	0.018
	1024	1024	0.96	1.044	0.007	0.011	0.021	1.88	0.532	0.002	0.010	0.022
		1024*2	1.92	1.040	0.007	0.014	0.024	4.25	0.470	0.002	0.008	0.016
		1024*4	3.42	1.168	0.009	0.015	0.023	7.27	0.550	0.003	0.009	0.017
		1024*8	6.95	1.151	0.008	0.015	0.026	15.41	0.519	0.003	0.009	0.019
16	256	1024	0.95	0.263	0.015	0.029	0.042	1.67	0.150	0.007	0.021	0.034
		1024*2	1.88	0.266	0.014	0.028	0.048	3.36	0.149	0.007	0.020	0.029
		1024*4	3.88	0.258	0.014	0.028	0.042	5.65	0.177	0.009	0.022	0.039
		1024*8	7.78	0.257	0.014	0.028	0.042	12.31	0.162	0.007	0.024	0.040
	512	1024	0.89	0.562	0.016	0.031	0.045	1.95	0.257	0.006	0.017	0.034
		1024*2	1.43	0.699	0.019	0.038	0.066	3.67	0.273	0.006	0.019	0.031
		1024*4	3.87	0.516	0.015	0.029	0.039	7.23	0.277	0.006	0.020	0.038
		1024*8	7.18	0.557	0.016	0.029	0.047	13.88	0.288	0.007	0.019	0.031
	1024	1024	0.99	1.012	0.015	0.028	0.042	1.87	0.536	0.006	0.019	0.040
		1024*2	1.75	1.141	0.016	0.033	0.054	4.14	0.483	0.006	0.017	0.028
		1024*4	3.61	1.109	0.016	0.030	0.048	8.02	0.499	0.005	0.018	0.030
		1024*8	5.30	1.510	0.020	0.044	0.082	15.40	0.519	0.006	0.018	0.031

不同环境参数下得到的数据也能得到与第二部分第 1 节中一样的结论。

三. 尾延迟挑战

1. 数据采集

修改 latency-collect.ipynb 和 latency-plot.ipynb 文件中的部分参数（请求数=100，对象尺寸=1024*4）以连接到搭建的 minio 服务器，执行测试获得以下尾延迟分布图。

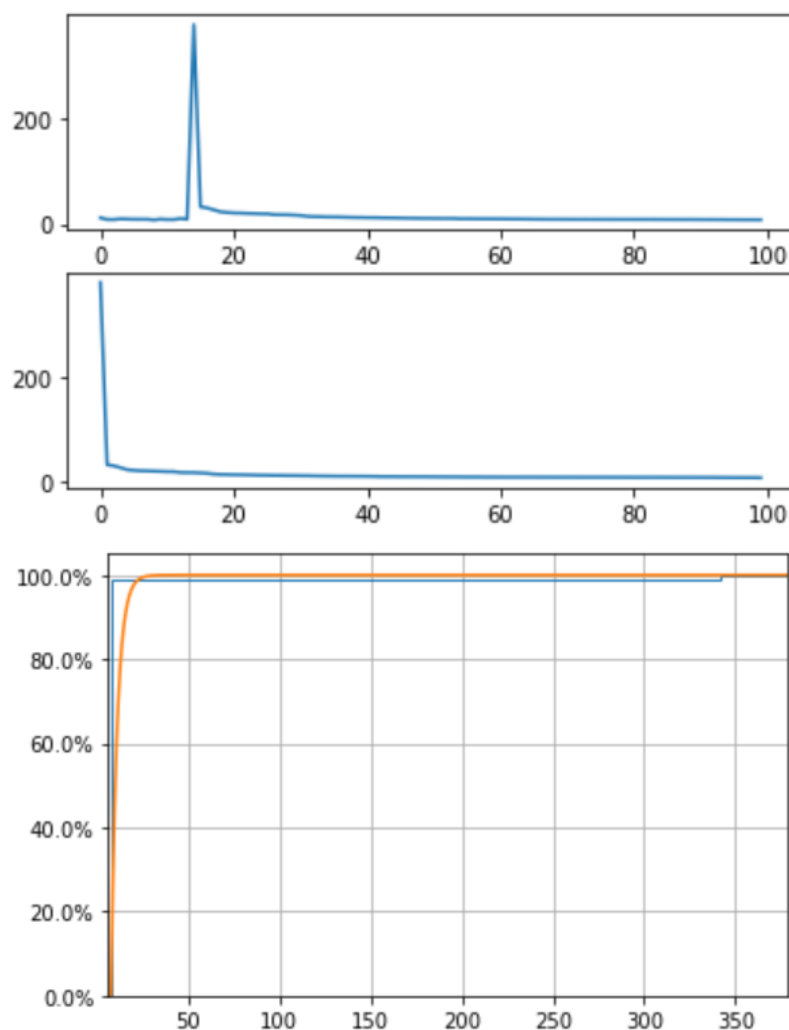


由图像可知，存在少部分的写请求耗时远远大于其他请求，即观测到尾延迟现象。

2. 对冲请求

从上图运行数据可视化结果来看,大约 90%的数据都可以在 40ms 内完成。

因此将请求数翻倍为 200 (每两个请求负责同一个对象), 设置 40ms 作为界限, 如果请求的时间超过 40ms, 那么就认定为失败, 再发送一个相同的请求, 实验结果如下。



可以看到几乎全部请求都可以在 50ms 内完成, 有效地解决了尾延迟问题。

四. 总结

通过这三个实验，我初步了解了对象存储系统的搭建过程，了解对象存储的优缺点，同时切实、直观地感受尾延迟问题，并设计实验用对冲请求的方式来试图解决尾延迟问题。