

2018 级

《物联网数据存储与管理》课程

实 验 报 告

姓 名 李嘉辉

学 号 U201890060

班 号 物联网 1801 班

日 期 2021.06.28

目 录

一、实验目的.....	1
二、实验背景.....	1
三、实验环境.....	1
四、实验内容.....	1
4.1 对象存储技术实践.....	2
4.2 对象存储性能分析.....	2
五、实验过程.....	2
5.1 安装、配置和启动.....	2
5.2 评测.....	4
六、实验总结.....	7
参考文献.....	7

一、实验目的

1. 熟悉对象存储技术，代表性系统及其特性；
2. 实践对象存储系统，部署实验环境，进行初步测试；
3. 基于对象存储系统，架设实际应用，示范主要功能。

二、实验背景

对象存储，也叫做基于对象的存储，是用来描述解决和处理离散单元的方法的通用术语，这些离散单元被称作为对象。其对应存储和保护大量非结构化的数据所带来的挑战提供了一个直接的响应。

对象存储综合了 NAS 和 SAN 的优点，同时具有 SAN 的高速直接访问和 NAS 的分布式数据共享等优势，提供了具有高性能、高可靠性、跨平台以及安全的数据共享的存储体系结构，非常适用于现在的海量数据的场景。

而在物联网方面，数据则存在着两大特性。一是数据的规模庞大，数据规模持续扩张；二是数据结构复杂，内容丰富，各种物品均携带自己的信息，这些信息上传到网络后，错综复杂的数据也带给管理人员很多管理上的难题。

Mock-S3，是用 Python 重写 fake-S3 实现的，沙盒环境中测试非常有用，无需实际调用 Amazon，其目标是最小化运行时依赖关系，并更像是一个开发工具来测试代码中的 S3 调用。

Object Store Manipulator，对象存储操纵器，用于云存储服务的 curl。osm 可以为 AWS S3、AWS S3 兼容的其他存储服务（即 Minio）、DigitalOcean Spaces、谷歌云存储、Microsoft Azure 存储和 OpenStack Swift 创建和删除存储桶，并从存储桶上载、下载和删除文件。

评测工具 S3 Bench，此工具提供了针对 S3 兼容端点运行非常基本的吞吐量基准测试的能力。它执行一系列的 put 操作，然后执行一系列的 get 操作，并显示相应的统计信息

三、实验环境

实验环境如下表 3-1 所示。

CPU	Inter (R) Core (TM) i7-3632QM CPU@2.20GHz
内存	16GB
操作系统	Windows 7 旗舰版
Python	Python 3.7.7
服务端	Mock-s3
客户端	OSM
评测工具	S3 Bench

表 3-1 实验环境

四、实验内容

1. 熟悉代码管理和仓库；配置 Python、Java 等开发、运行环境
2. 实践对象存储，配置服务端与客户端，并进行基础的操作尝试。
3. 对配置好的对象存储系统进行测试，并分析性能。

4.1 对象存储技术实践

1. 在 windows 下配置 Python, Go 等运行环境。
2. 安装 mock-s3 作为服务器端, 安装 osm 作为客户端。
3. 使用例如新建 bucket、上传删除文件等指令简单测试系统。

4.2 对象存储性能分析

1. 安装测试用工具 s3bench。
2. 改变各种不同的参数, 记录每次的测试结果。
3. 根据测试结果, 分析不同数据的影响。

五、实验过程

5.1 安装、配置和启动

(1) 对象存储服务端 mock_s3 的安装、配置和启动

创建好 python 环境后, 从实验指导仓库下载 mock-s3, 在对应目录下运行 setup.py 程序安装 mock-s3。输入命令设置好其地址为 127.0.0.1 端口为 9000, 然后在浏览器上输入地址端口进行链接, 可以看到结果如下图 5-1 所示, 证明 mock-s3 安装运行成功。

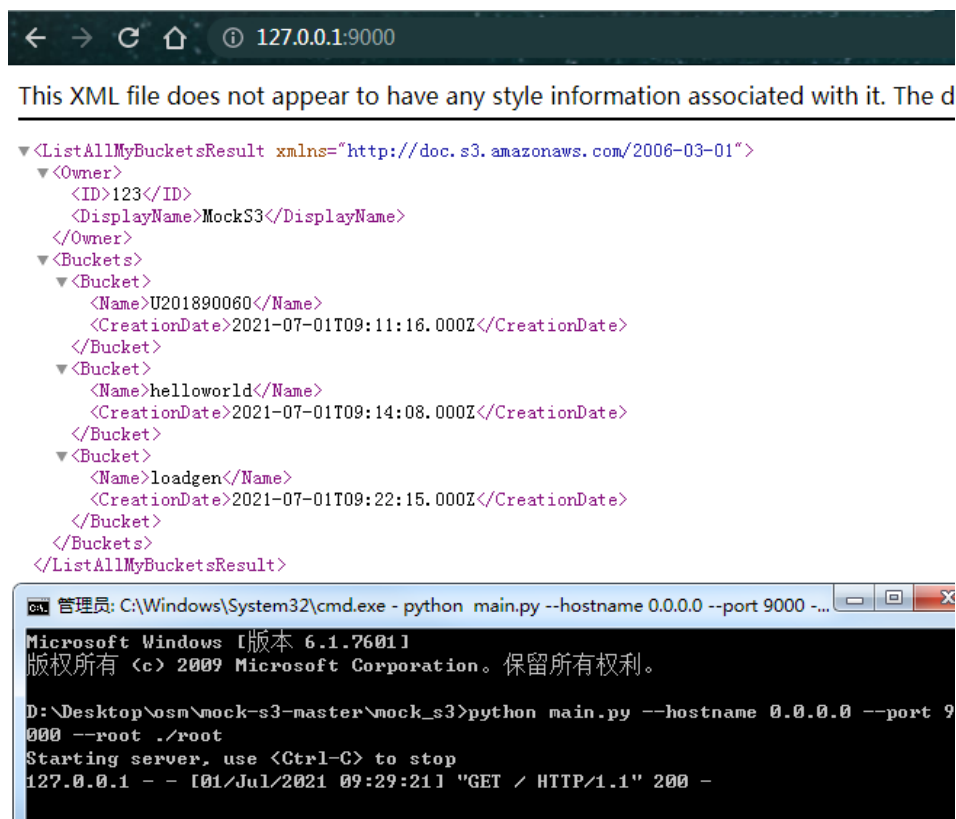


图 5-1 浏览器访问结果与服务端显示信息

(2) 对象存储服务端 osm 的安装、配置和启动

1. 从实验指导仓库下载 Windows 版 osm.exe 可执行文件。
2. 在 CMD 中对应目录下执行 osm 配置脚本 config-osm.cmd。

3. 在 CMD 中执行 `osm -h` 命令如图 5-2 所示，可见相应输出结果，证明 `osm` 安装配置成功。



```
管理员: C:\Windows\System32\cmd.exe
D:\Desktop\osm>osm -h
Object Store Manipulator by AppsCode

Usage:
  osm [command] [flags]
  osm [command]

Available Commands:
  config      OSM configuration
  help        Help about any command
  lc          List containers
  ls          List items in a container
  mc          Make container
  pull        Pull item from container
  push        Push item to container
  rc          Remove container
  rm          Remove item from container
  stat        Stat item from container
  version     Prints binary version number.

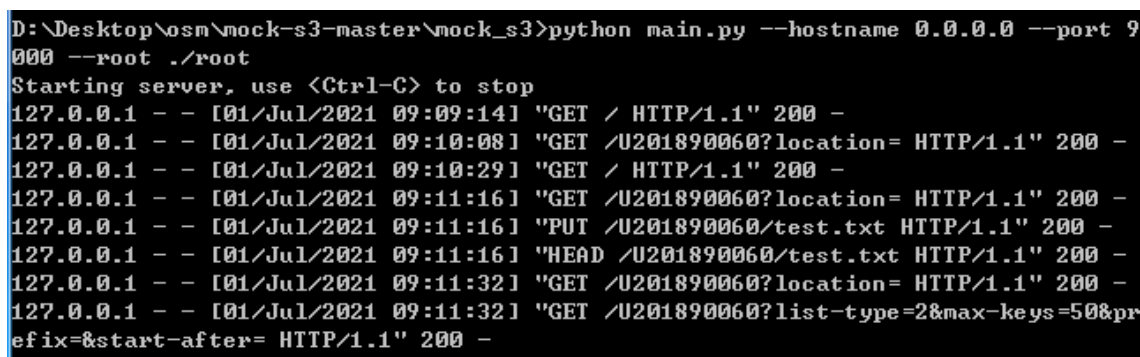
Flags:
  --also-log-to-stderr          log to standard error as well as files
  --enable-analytics           Send usage events to Google Analytics (default true)
  -h, --help                   help for osm
  --log-backtrace-at traceLocation when logging hits line file:N, emit a stack trace (default :0)
  --log-dir string              If non-empty, write log files in this directory
  --log-to-stderr               log to standard error instead of files
  --osm-config string           Path to osm config (default "C:\Users\Administrator\osm\config")
  --stderr-threshold severity   logs at or above this threshold go to stderr
  -v, --v Level                 log level for V logs
  --module moduleSpec           comma-separated list of pattern=N settings for file-filtered logging

Use "osm [command] --help" for more information about a command.

D:\Desktop\osm>
```

图 5-2 `osm -h` 命令行输出

4. 执行 `osm` 的 Bucket 管理命令与存储管理操作：创建 bucket、上传测试文件。服务端输出如图 5-3 所示，证明对象存储系统功能正常。



```
D:\Desktop\osm\mock-s3-master\mock_s3>python main.py --hostname 0.0.0.0 --port 9000 --root ./root
Starting server, use <Ctrl-C> to stop
127.0.0.1 - - [01/Jul/2021 09:09:14] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [01/Jul/2021 09:10:08] "GET /U201890060?location= HTTP/1.1" 200 -
127.0.0.1 - - [01/Jul/2021 09:10:29] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [01/Jul/2021 09:11:16] "GET /U201890060?location= HTTP/1.1" 200 -
127.0.0.1 - - [01/Jul/2021 09:11:16] "PUT /U201890060/test.txt HTTP/1.1" 200 -
127.0.0.1 - - [01/Jul/2021 09:11:16] "HEAD /U201890060/test.txt HTTP/1.1" 200 -
127.0.0.1 - - [01/Jul/2021 09:11:32] "GET /U201890060?location= HTTP/1.1" 200 -
127.0.0.1 - - [01/Jul/2021 09:11:32] "GET /U201890060?list-type=2&max-keys=50&prefix=&start-after= HTTP/1.1" 200 -
```

图 5-3 服务端输出

(3) 对象存储评测工具 `s3bench` 的安装、配置和启动

1. 从仓库下载 `s3 bench` 的 Windows 预编译版本，下载可执行文件到本地克隆的资料库里。
2. 运行脚本 `run-s3bench.cmd` 启动 `s3 bench` 进行评测，如图 5-4 所示。

```
C:\Windows\system32\cmd.exe
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)

Generating in-memory sample data... Done (7.0004ms)

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: 0.41 MB/s
Total Duration: 0.612 s
Number of Errors: 0
-----
Write times Max: 0.094 s
Write times 99th xile: 0.092 s
Write times 90th xile: 0.031 s
Write times 75th xile: 0.019 s
Write times 50th xile: 0.014 s
Write times 25th xile: 0.013 s
Write times Min: 0.009 s

Results Summary for Read Operation(s)
Total Transferred: 0.250 MB
Total Throughput: 0.47 MB/s
Total Duration: 0.532 s
Number of Errors: 0
-----
Read times Max: 0.507 s
Read times 99th xile: 0.016 s
Read times 90th xile: 0.011 s
Read times 75th xile: 0.010 s
Read times 50th xile: 0.010 s
Read times 25th xile: 0.009 s
Read times Min: 0.006 s

Cleaning up 256 objects...
Deleting a batch of 256 objects in range (0, 255)... Succeeded
Successfully deleted 256/256 objects in 7.0004ms

D:\Desktop\asn>pause
请按任意键继续. . .
```

图 5-4 s3 bench 评测结果显示

上图可见 S3 Bench 成功安装与并配置，能正常完成对当前对象存储系统的评测。

5.2 评测

修改范例脚本 run-s3bench.cmd，保持其余值不变的情况下，分别改变当中对象大小 **objectSize** 和并发客户端数量 **numClients** 两个数值，观察它们对读写吞吐率和延迟的影响，数据记录成表格如下表 5-1 和表 5-2 所示。

对象尺寸(MB)	客户端数	写吞吐量(MB/s)	写99th延迟(s)	写90th延迟(s)	写50th延迟(s)	读吞吐量(MB/s)	读99th延迟(s)	读90th延迟(s)	读50th延迟(s)
0.001	8	0.6	0.22	0.16	0.13	0.37	0.14	0.12	0.1
0.0039	8	2.28	0.028	0.016	0.013	1.38	0.015	0.012	0.01
0.0156	8	8.18	0.026	0.017	0.015	4.79	0.016	0.013	0.011
0.0625	8	31.13	0.094	0.019	0.015	42.55	0.016	0.013	0.012
0.25	8	59.37	0.121	0.04	0.03	68.08	0.052	0.041	0.035
1	8	72.21	0.297	0.14	0.108	85.39	0.232	0.16	0.098
2	8	13.47	1.254	1.231	1.18	91.42	0.482	0.234	0.174
4	8	22.42	1.695	1.529	1.419	94.8	1.4	0.586	0.267
6	8	29.2	1.857	1.778	1.674	95.92	1.172	0.67	0.537
8	8	33.42	2.247	2.141	1.947	95.45	2.012	1.044	0.6
10	8	36.19	2.551	2.424	2.313	96.03	2.064	1.217	0.719
12	8	35.49	3.033	2.91	2.692	92.61	1.918	1.286	0.902
14	8	36.59	3.381	3.298	3.058	95.61	2.039	1.53	1.025
16	8	42.17	4.003	3.831	3.25	92.37	2.947	1.841	1.32

表 5-1 改变对象大小数据

对象尺寸(MB)	客户端数	写吞吐量(MB/s)	写99th延迟(s)	写90th延迟(s)	写50th延迟(s)	读吞吐量(MB/s)	读99th延迟(s)	读90th延迟(s)	读50th延迟(s)
1	1	47.7	0.025	0.023	0.021	66.51	0.02	0.017	0.015
1	2	61.51	0.042	0.038	0.032	73.33	0.038	0.032	0.027
1	3	68.81	0.079	0.049	0.047	80.42	0.048	0.044	0.037
1	4	73.29	0.086	0.063	0.054	81.34	0.069	0.059	0.055
1	5	70.97	0.11	0.083	0.068	82.9	0.082	0.071	0.06
1	6	72.72	0.104	0.095	0.081	81.68	0.122	0.088	0.074
1	7	70.73	0.371	0.116	0.09	82.98	0.145	0.14	0.086
1	8	87.28	0.222	0.122	0.088	86.48	0.366	0.12	0.088
1	9	76.23	0.351	0.14	0.114	91.42	0.514	0.132	0.099
1	10	76.92	0.505	0.169	0.115	90.55	0.482	0.197	0.12
1	11	82.26	0.854	0.166	0.115	85.53	0.567	0.203	0.1
1	12	83.44	0.652	0.175	0.128	87.88	0.577	0.18	0.115
1	13	82.74	0.639	0.218	0.14	87.28	0.616	0.219	0.127
1	14	80.25	0.624	0.226	0.165	88	0.418	0.236	0.142
1	15	80.8	0.725	0.243	0.165	85.13	0.501	0.228	0.164
1	16	81.27	0.666	0.296	0.178	88.09	0.636	0.281	0.163
1	17	81.19	0.676	0.267	0.191	84.71	0.669	0.3	0.17
1	18	85.3	0.698	0.264	0.194	88.91	0.554	0.308	0.174
1	19	80.91	0.745	0.323	0.205	85.67	0.725	0.328	0.194
1	20	82.63	0.501	0.327	0.221	86.69	0.818	0.338	0.195

表 5-2 改变客户端数量数据

将实验所得到的数据分别组合绘制成曲线图，从而对存储系统性能进行分析：

(1) 对象大小对存储系统性能影响

客户端数量固定为 8，修改对象大小。

如图 5-5 所见，对于读吞吐量，在 1MB 以内基本符合随对象数量增大而增大的规律，超过 1MB 后变化不大。而写吞吐量，在 1MB 以内基本符合随对象数量增大而增大的规律，但测验发现对象在 1~2MB 之间发生较大起伏，写吞吐量急剧下跌后在 2MB 后缓慢上升，超过 10MB 后渐趋平稳。

如图 5-6 所见 1MB 以内读写延迟基本不变，超过 1MB 延迟随对象数量增大而增大。

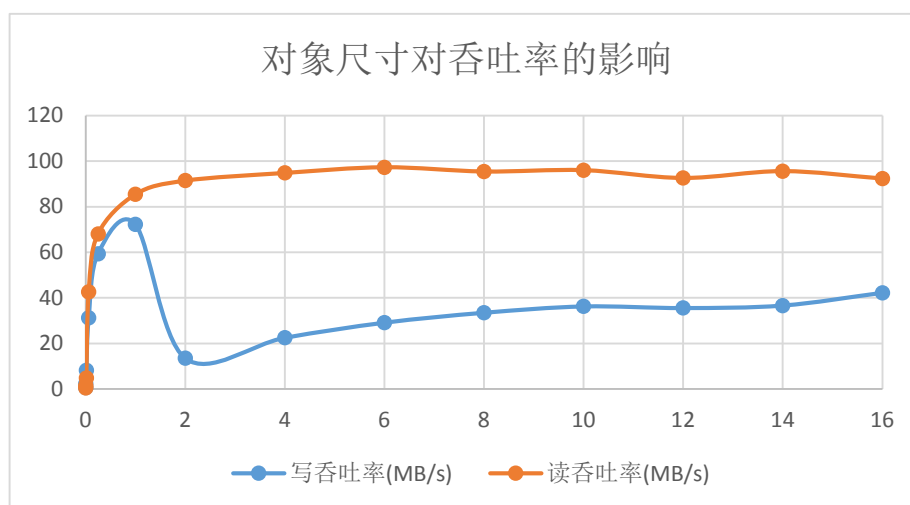


图 5-5 对象尺寸对吞吐率的影响

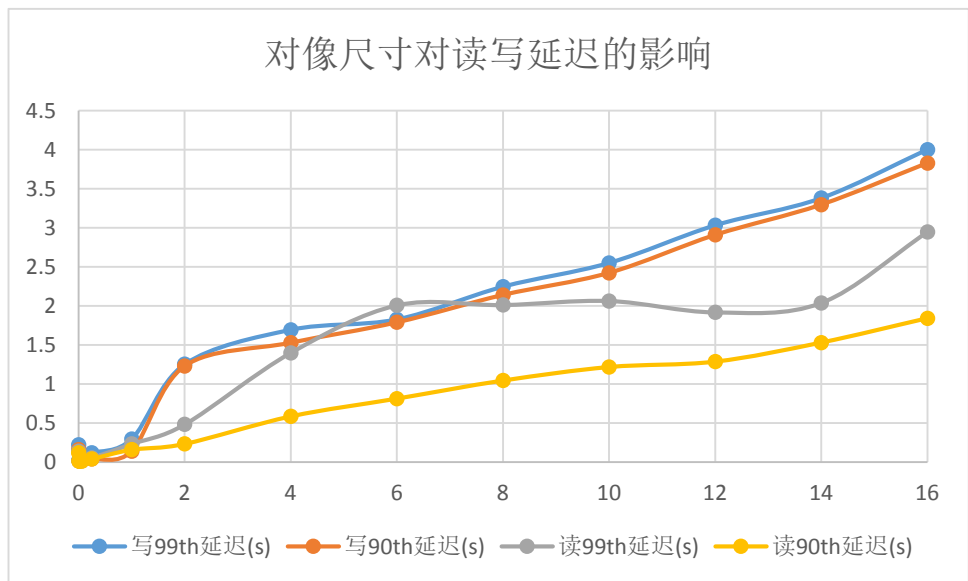


图 5-6 对像尺寸对读写延迟的影响

(2) 并发客户端数量对存储系统性能影响

对象的大小固定为 1M，修改客户端数量。

如图 5-7 所见，对于读写吞吐率，在 8 个以内的客户端随数量增大而增大，超过后吞吐率逐渐平稳。

如图 5-8 所见 90th%ile 的读写延迟随客户端数量增大而增大，而 99th%ile 的读写延迟在 6 个以内的客户端随数量增大而增大，超过后延迟开始变得不稳定。

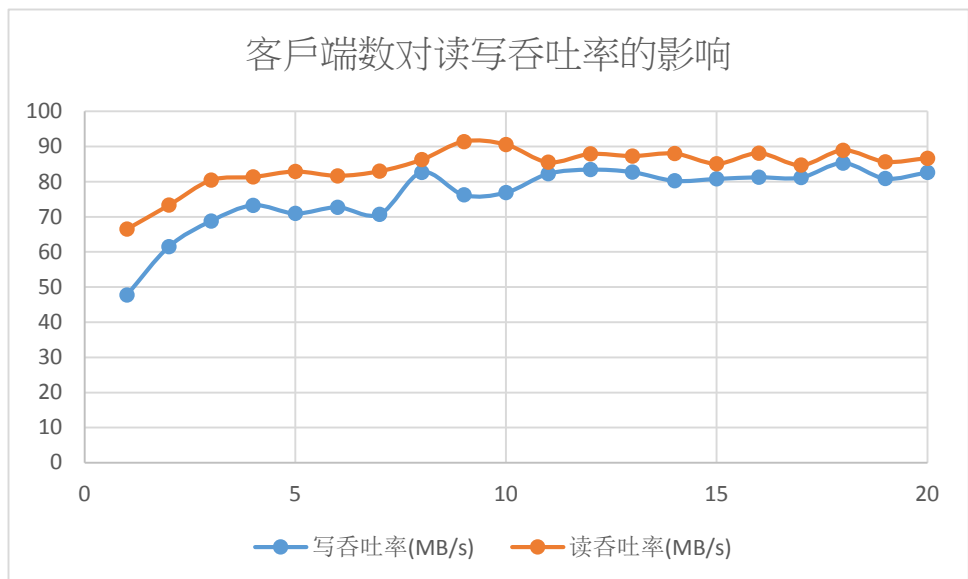


图 5-7 客户端数对读写吞吐率的影响

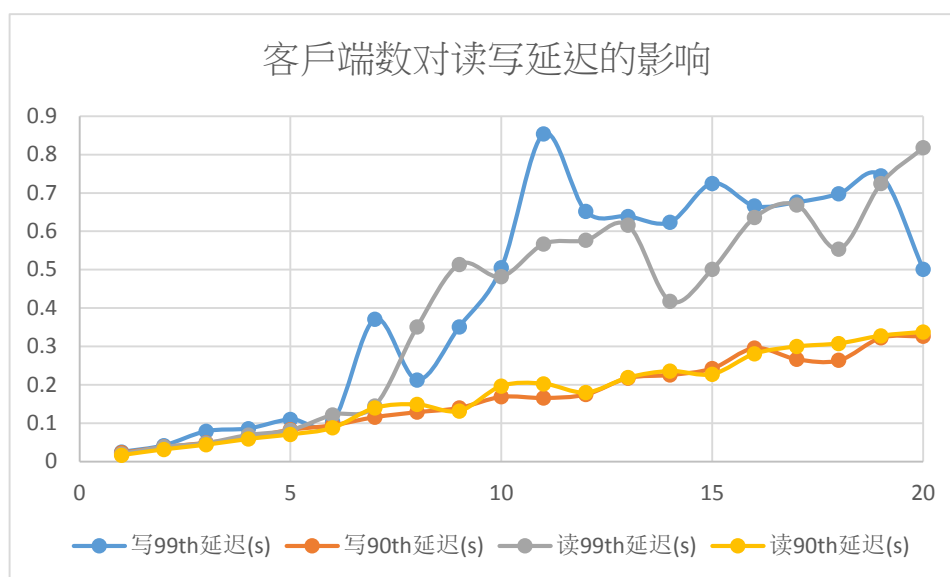


图 5-8 客户端数对读写延迟的影响

六、实验总结

本次实验采用 `mock_s3` 作为服务端，`osm` 作为客户端，搭建出一个简易的对象存储系统并利用工具 `s3bench` 对其存储性能进行针对性的评测。在此过程中，对于对象存储的概念有了基本的认识，常用的对象存储工具有了初步的掌握，培养了动手能力，也丰富了理论知识。

同时在这次实验中也常会了 `git` 的使用，实验需要在 `Github` 仓库中获取实验资源和提交实验成果。对项目库进行 `fork`，克隆到本地进行编写，推送到远程库，最后 `pull request` 请求，这对日后团结分工编写项目有很大的帮助。

参考文献

- [1] ARNOLD J. OpenStack Swift[M]. O' Reilly Media, 2014.
- [2] ZHENG Q, CHEN H, WANG Y 等. COSBench: A Benchmark Tool for Cloud Object Storage Services[C]//2012 IEEE Fifth International Conference on Cloud Computing. 2012: 998 - 999.
- [3] WEIL S A, BRANDT S A, MILLER E L 等. Ceph: A Scalable, High-performance Distributed File System[C]//Proceedings of the 7th Symposium on Operating Systems Design and Implementation. Berkeley, CA, USA: USENIX Association, 2006: 307 - 320.