

**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](#_Toc76102146)

[二、实验背景 1](#_Toc76102147)

[三、实验环境 1](#_Toc76102148)

[四、实验内容 1](#_Toc76102149)

[4.1 对象存储技术实践 2](#_Toc76102150)

[4.2 对象存储性能分析 2](#_Toc76102151)

[五、实验过程 2](#_Toc76102152)

[5.1安装、配置和启动 2](#_Toc76102153)

[5.2 评测 4](#_Toc76102154)

[六、实验总结 7](#_Toc76102155)

[参考文献 7](#_Toc76102156)

# 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。其对应对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。

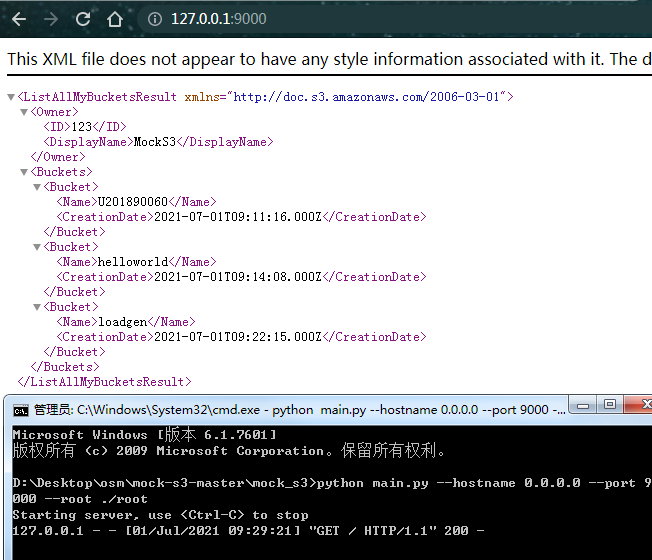


图 5-1 浏览器访问结果与服务端显示信息

1. 对象存储服务端 osm 的安装、配置和启动
2. 从实验指导仓库下载 Windows 版 osm.exe 可执行文件。
3. 在 CMD 中对应目录下执行 osm 配置脚本 config-osm.cmd。
4. 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。

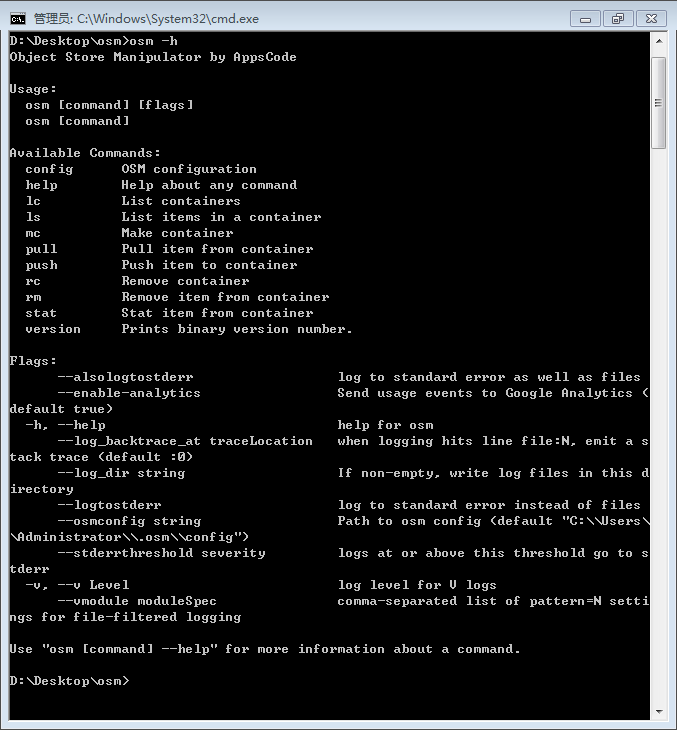


图 5-2 osm -h 命令行输出

1. 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。

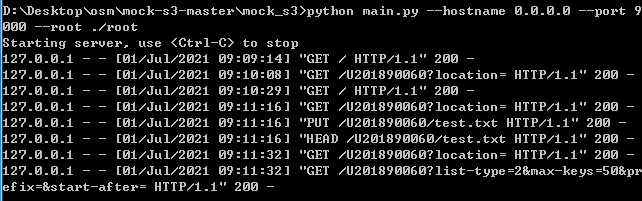


图 5-3 服务端输出

1. 对象存储评测工具 s3bench 的安装、配置和启动
2. 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。

1. 运行脚本 run-s3bench.cmd 启动 s3 bench 进行评测，如图 5-4 所示。

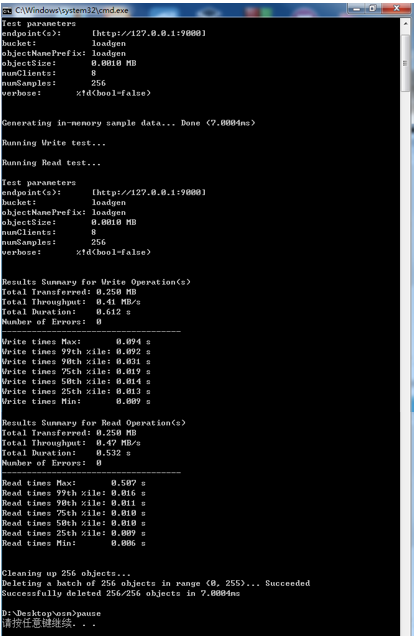


图 5-4 s3 bench 评测结果显示

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，观察它们对读写呑吐率和延迟的影响，数据记录成表格如下表 5-1和表5-2 所示。



表 5-1改变对象大小数据

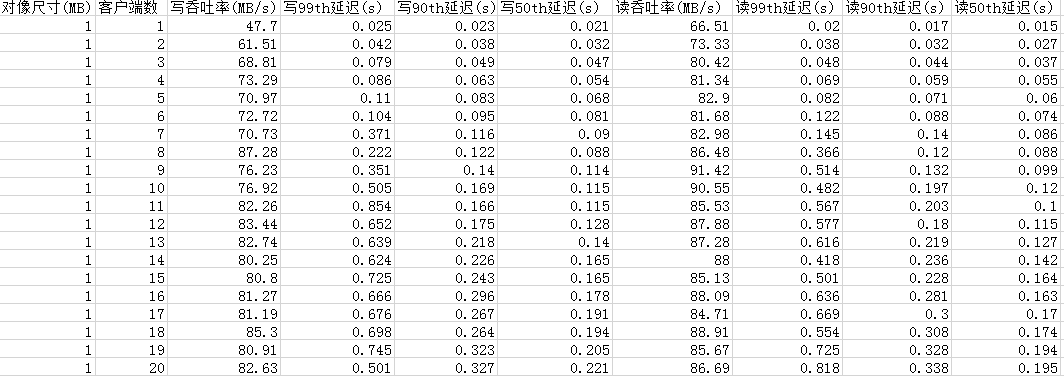


表 5-2 改变客户端数量数据

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，对于读呑吐率，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。而写呑吐率，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，写呑吐率急剧下跌后在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迟延随对象数量增大而增大。

图5-5 对象尺寸对呑吐率的影响

图5-6对像尺寸对读写延迟的影响

1. 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，对于读写呑吐率，在8个以内的客户端随数量增大而增大，超过后呑吐率逐渐平稳。

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客户端数对读写呑吐率的影响

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的𠉕用，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.