

**2018** 级

《物联网数据存储与管理》课程

**实 验 报 告**

**姓 名 黄世谱**

**学 号 U201814572**

**班 号 物联网1801班**

**日 期 2021.06.21**

目录

[一、实验目的 3](#_Toc75728603)

[二、实验背景 3](#_Toc75728604)

[三、 实验环境 4](#_Toc75728605)

[四、实验内容 5](#_Toc75728606)

[4.1 对象存储技术实践 5](#_Toc75728607)

[4.2 对象存储性能分析 5](#_Toc75728608)

[五、实验过程 5](#_Toc75728609)

[5.1 搭建对象存储服务端 5](#_Toc75728610)

[5.2 搭建对象存储客户端 6](#_Toc75728611)

[5.3 测试对象存储功能 7](#_Toc75728612)

[5.4 对象存储性能分析 7](#_Toc75728613)

[六、实验总结 10](#_Toc75728614)

[参考文献 10](#_Toc75728615)

# 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

块形式的存储在满足数据可扩展性和数据安全性的增长方面，日益显现出其局限性和不足。国际上主要有两类网络化存储架构，它们是通过命令集来区分的。第一类是SAN（StorageAreaNetwork）结构，它采用SCSI块I/O的命令集，通过在磁盘或FC（FiberChannel）级的数据访问提供高性能的随机I/O和数据吞吐率，它具有高带宽、低延迟的优势，在高性能计算中占有一席之地，但是由于SAN系统的价格较高，且可扩展性较差，已不能满足成千上万个CPU规模的系统。第二类是NAS（NetworkAttachedStorage）结构，它采用NFS或CIFS命令集访问数据，以文件为传输协议，通过TCP/IP实现网络化存储，可扩展性好、价格便宜、用户易管理，如目前在集群计算中应用较多的NFS文件系统，但由于NAS的协议开销高、带宽低、延迟大，不利于在高性能集群中应用。

针对Linux集群对存储系统高性能和数据共享的需求，国际上已开始研究全新的存储架构和新型文件系统，希望能有效结合SAN和NAS系统的优点，支持直接访问磁盘以提高性能，通过共享的文件和元数据以简化管理，目前对象存储系统已成为Linux集群系统高性能存储系统的研究热点，如Panasas公司的ObjectBaseStorageClusterSystem系统和ClusterFileSystems公司的Lustre等。

对象存储（Object-based Storage)是一种新的网络存储架构，基于对象存储技术的设备就是对象存储设备（Object-based Storage Device）简称OSD。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。

对象存储的架构，其核心是将数据通路（数据读或写）和控制通路（元数据）分离，并且基于对象存储设备构建存储系统，每个对象存储设备具有一定的功能，能够自动管理其上的数据分布。对象存储结构由对象、对象存储设备、元数据服务器、对象存储系统的客户端四部分组成。对象包含文件数据以及相关信息，可以进行自我管理；对象存储宿设备，对对象进行存储与管理的设备；元数据服务器，为客户端提供元数据，主要是文件的逻辑视图，包括对象与目录的组织关系，每个对象对应的OSD等。

本次实验使用mock-s3作为对象存储的服务端，采用osm作为对象存储的客户端，评测工具采用S3 Bench。

Mock-s3是一个对Amazon s3的轻量级python克隆版本，运行在python环境上。Amazon Simple Storage Service (Amazon S3) 是一个公开的云存储服务，Web 应用程序开发人员可以使用它存储数字资产，包括图片、视频、音乐和文档。

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。

# 实验环境

实验环境如下表3-1所示。

因为之前的学习，已经在Windows下把git、python环境还有Java环境安装完成，接下来再安装Go环境（客户端osm与评测工具S3 Bench依赖的环境）后即可进行实验，所以为了方便操作，本次实验都在Windows下进行。

|  |  |
| --- | --- |
| 操作系统 | Windows 10 |
| 内存 | 8GB |
| Go版本 | go version go1.14.3 windows/amd64 |
| Python版本 | Python 3.7.10 |
| 客户端 | Osm |
| 服务端 | Mock-s3 |
| 评测工具 | S3 Bench |

表3-1 实验环境

# 四、实验内容

## 4.1 对象存储技术实践

1.熟悉基础环境：安装Python、Go等开发、运行环境。

2.实践对象存储：根据系统开发环境选择并安装合适的对象存储客户端和服务端。本次实验中采用mock\_s3作为服务端，osm作为客户端。

3.评测系统：选择并下载评测工具，对本机上的对象存储系统的性能进行评测。，本次实验采用s3bench来进行评测。

## 4.2 对象存储性能分析

1.选择合适的对象存储评测工具，本次实验选用S3 Bench。

2.调整对象存储评测参数，包括客户端数量、对象数量、对象大小，观察数据存储性能。

3.编写cmd脚本，设计循环结构实现批量测试将终端输出结果重定向到文本文件，以便进一步分析。

4.整理测试数据，分析不同参数条件下的存储性能。延迟的评测参数采用总的写延迟和90 percentile的延迟

# 五、实验过程

## 5.1 搭建对象存储服务端

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。

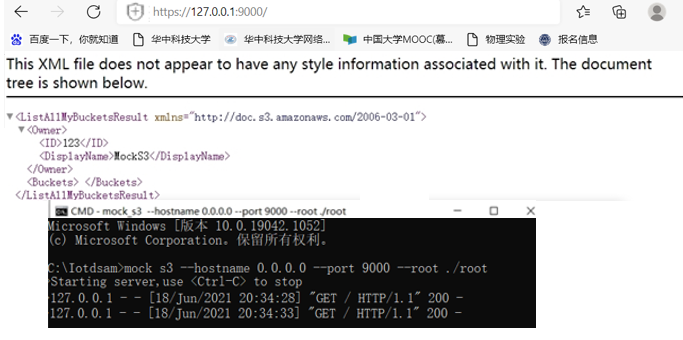


图5-1 浏览器访问结果与服务端显示信息

## 搭建对象存储客户端

从仓库上下载osm.exe文件，将其放置到先前本地克隆的资料库里，然后运行脚本config-osm.cmd对osm进行配置，即可对osm进行操作，在CMD中执行osm -h命令，可见结果如图5-2所示，证明osm安装配置成功。

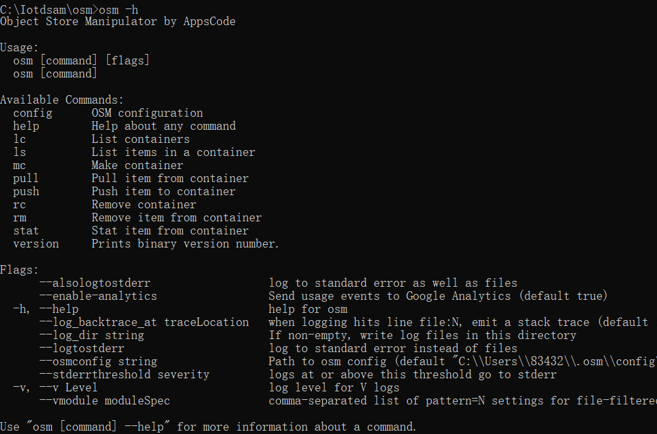


图5-2 对客户端osm进行操作

## 5.3 测试对象存储功能

服务端保存运行状态。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。

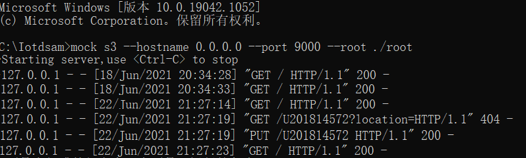


图5-3 服务端输出

## 5.4 对象存储性能分析

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

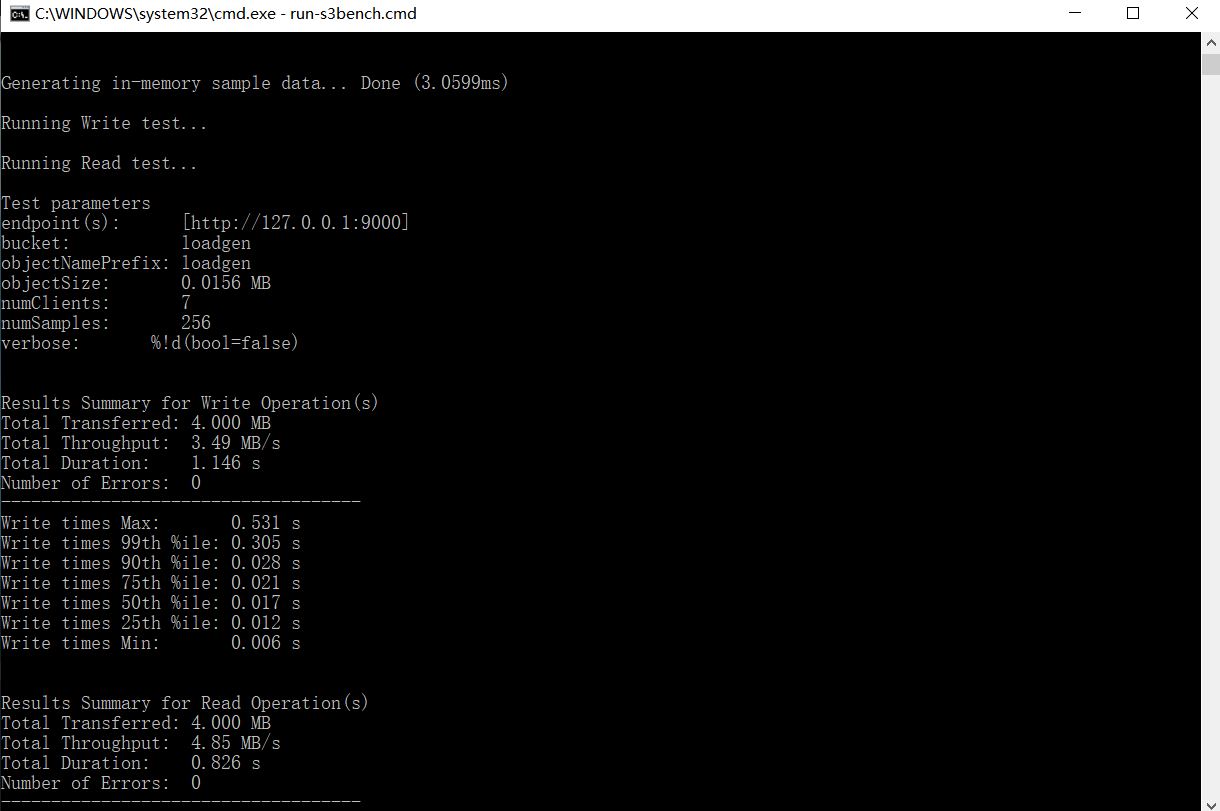


图5-4 s3 bench评测结果显示

分别改变客户端数量与对象大小，观察它们对传输速率和延迟的影响，数据记录成表格如下表5-1，5-2所示，将实验所得到的数据绘制成曲线图如下图5-5和图5-6所示。

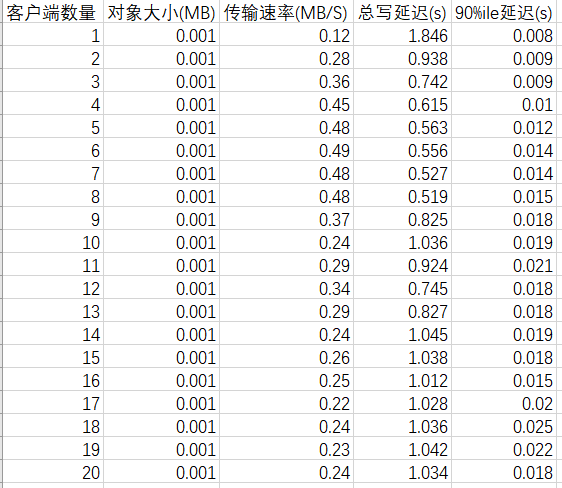


表5-1 改变客户端数量数据

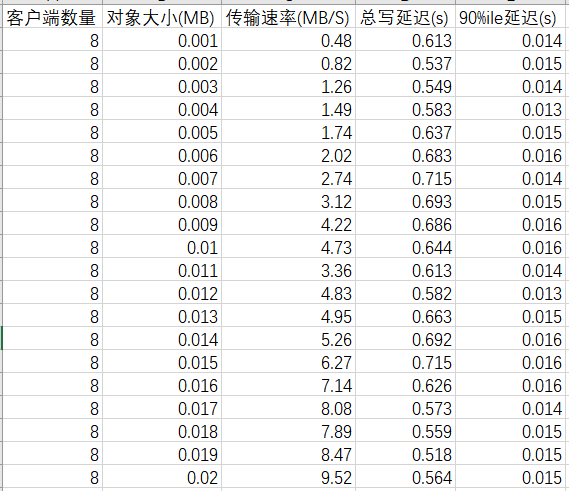


表5-1 改变对象大小数据

图5-5 客户端数量影响

图5-6 对象大小影响

1. 并发客户端数量numClients对存储系统性能影响

对象的大小固定为0.01M，修改客户端数量。

观察曲线图分析测试结果，可以看到客户端数量在1~8之间时传输速率越来越大，稍微减小一点后基本保持不变。可以得出随着客户端数量的增加，传输速率先随着增大，然后趋近于饱和，不再受并发数的影响。

客户端数量在1~8之间时总延迟随着客户端数量的增加而减少，然后趋近平缓。8之后，总延迟随客户端数量的增加而增加，然后区域平缓。

客户端数量变化时，90%ile的写延迟基本保持不变。

1. 对象大小objectSize对存储系统性能影响

客户端数量固定为8，修改对象大小。

对于传输速率，基本符合随对象数量增大而增大的规律。

对于总写延迟和90%ile延迟，对象大小变化两者都基本保持不变。可以推测出对象的大小不对延迟产生影响。

# 六、实验总结

本次实验首先搭建运行环境，然后采用mock\_s3作为服务端，osm作为客户端，并用s3bench来进行评测，搭建了一个对象存储系统。实验中进行了对象存储系统的评测，通过上面的分析，可以把得出的结论当作针对不同的环境下的应对策略。

在这次实验中，最大的收获是对git的使用。在之前对git了解较少，这次实验要求在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，这些之前都没有用过。通过这次实验，学习和熟悉了Git常用命令与特性，对对象存储系统也有了更多的了解。

# 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.