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
二、实验背景 3
三、 实验环境 4
四、实验内容 5
4.1 对象存储技术实践 5
4.2 对象存储性能分析 5
五、实验过程 5
5.1 搭建对象存储服务端 5
5.2 搭建对象存储客户端 6
5.3 测试对象存储功能 7
5.4 对象存储性能分析 7
六、实验总结 10
参考文献 10

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。

三、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。

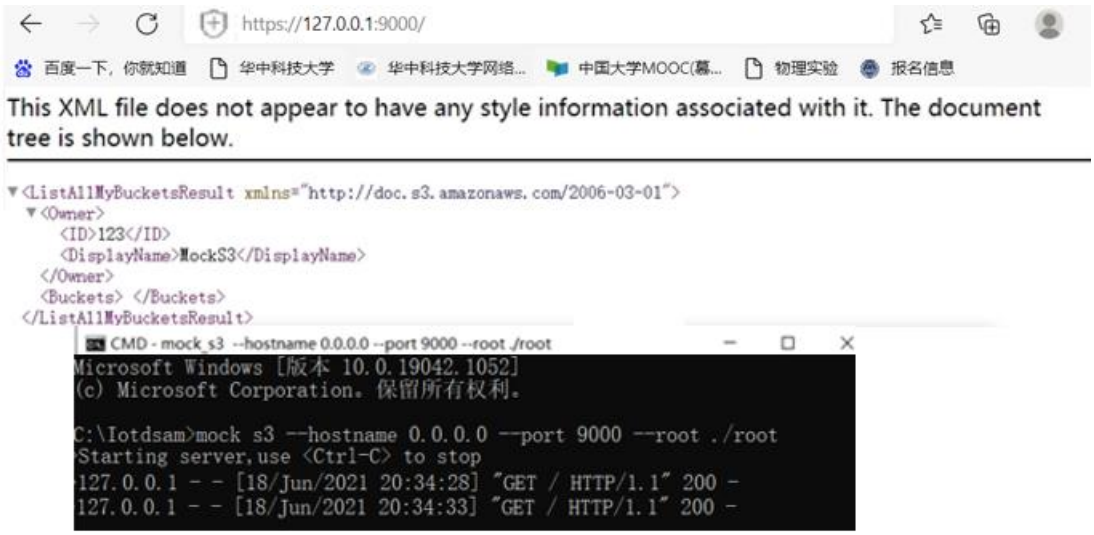

图 5-1 浏览器访问结果与服务端显示信息

5.2 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。

```
C:\Iotdsam\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:
      --alsologtostderr                 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
      --log_dir string                  If non-empty, write log files in this directory
      --logtostderr                     log to standard error instead of files
      --osmconfig string                Path to osm config (default "C:\\Users\\83432\\.osm\\config
      --stderrthreshold severity        logs at or above this threshold go to stderr
  -v, --v Level                         log level for V logs
      --vmodule moduleSpec              comma-separated list of pattern=N settings for file-filtere

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

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。

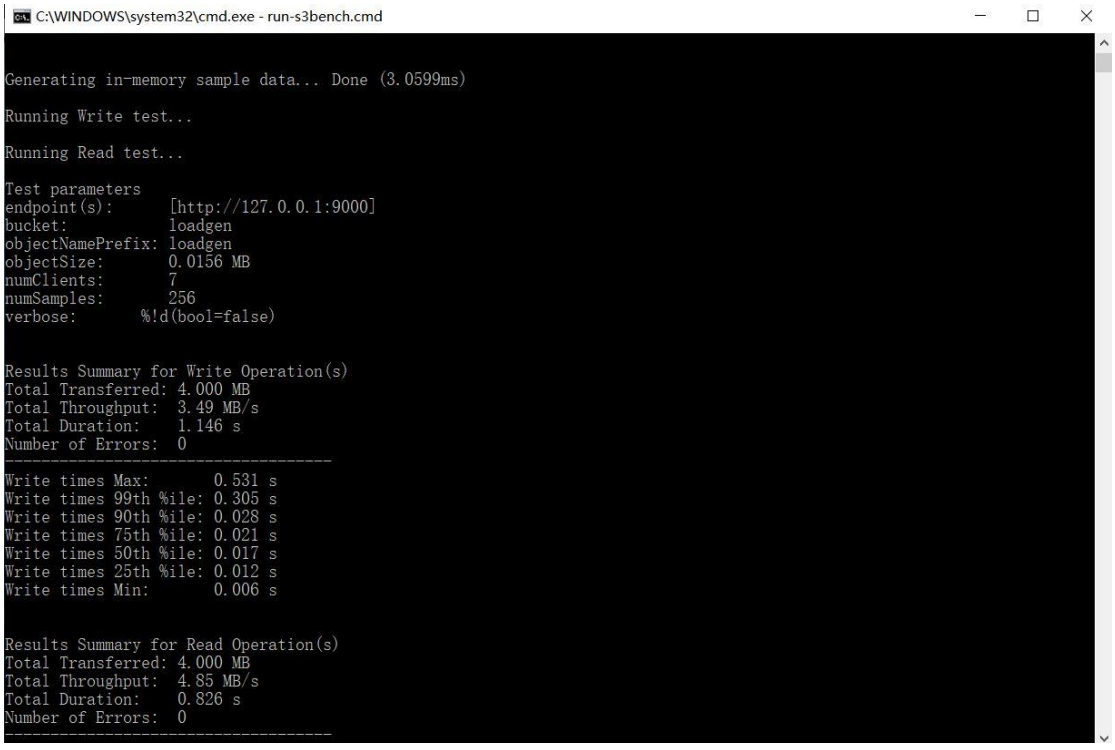

图 5-4 s3 bench 评测结果显示

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。

客户端数量	对象大小(MB)	传输速率(MB/S)	总写延迟(s)	90%ile延迟(s)
1	0.001	0.12	1.846	0.008
2	0.001	0.28	0.938	0.009
3	0.001	0.36	0.742	0.009
4	0.001	0.45	0.615	0.01
5	0.001	0.48	0.563	0.012
6	0.001	0.49	0.556	0.014
7	0.001	0.48	0.527	0.014
8	0.001	0.48	0.519	0.015
9	0.001	0.37	0.825	0.018
10	0.001	0.24	1.036	0.019
11	0.001	0.29	0.924	0.021
12	0.001	0.34	0.745	0.018
13	0.001	0.29	0.827	0.018
14	0.001	0.24	1.045	0.019
15	0.001	0.26	1.038	0.018
16	0.001	0.25	1.012	0.015
17	0.001	0.22	1.028	0.02
18	0.001	0.24	1.036	0.025
19	0.001	0.23	1.042	0.022
20	0.001	0.24	1.034	0.018

表 5-1 改变客户端数量数据

客户端数量	对象大小(MB)	传输速率(MB/S)	总写延迟(s)	90%ile延迟(s)
8	0.001	0.48	0.613	0.014
8	0.002	0.82	0.537	0.015
8	0.003	1.26	0.549	0.014
8	0.004	1.49	0.583	0.013
8	0.005	1.74	0.637	0.015
8	0.006	2.02	0.683	0.016
8	0.007	2.74	0.715	0.014
8	0.008	3.12	0.693	0.015
8	0.009	4.22	0.686	0.016
8	0.01	4.73	0.644	0.016
8	0.011	3.36	0.613	0.014
8	0.012	4.83	0.582	0.013
8	0.013	4.95	0.663	0.015
8	0.014	5.26	0.692	0.016
8	0.015	6.27	0.715	0.016
8	0.016	7.14	0.626	0.016
8	0.017	8.08	0.573	0.014
8	0.018	7.89	0.559	0.015
8	0.019	8.47	0.518	0.015
8	0.02	9.52	0.564	0.015

表 5-1 改变对象大小数据

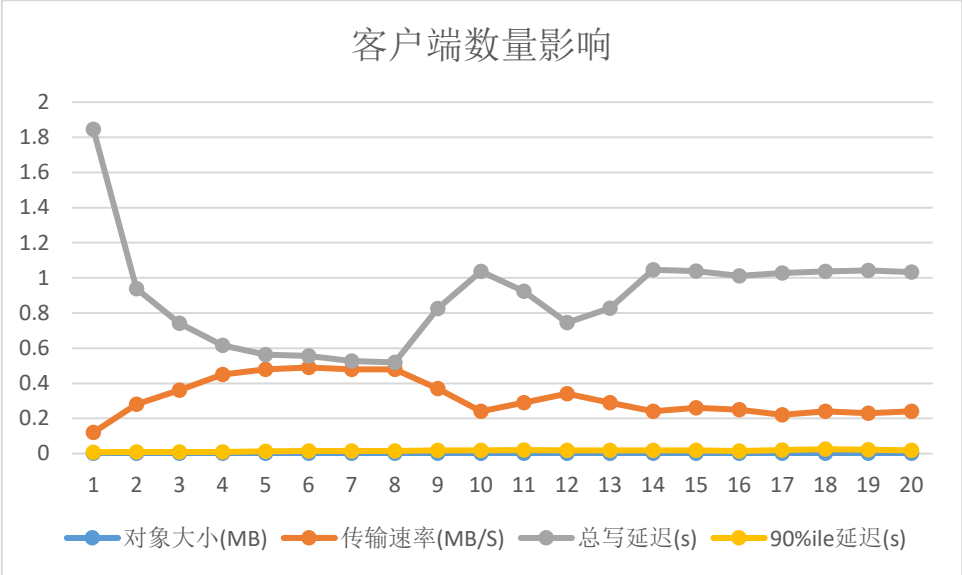

图 5-5 客户端数量影响

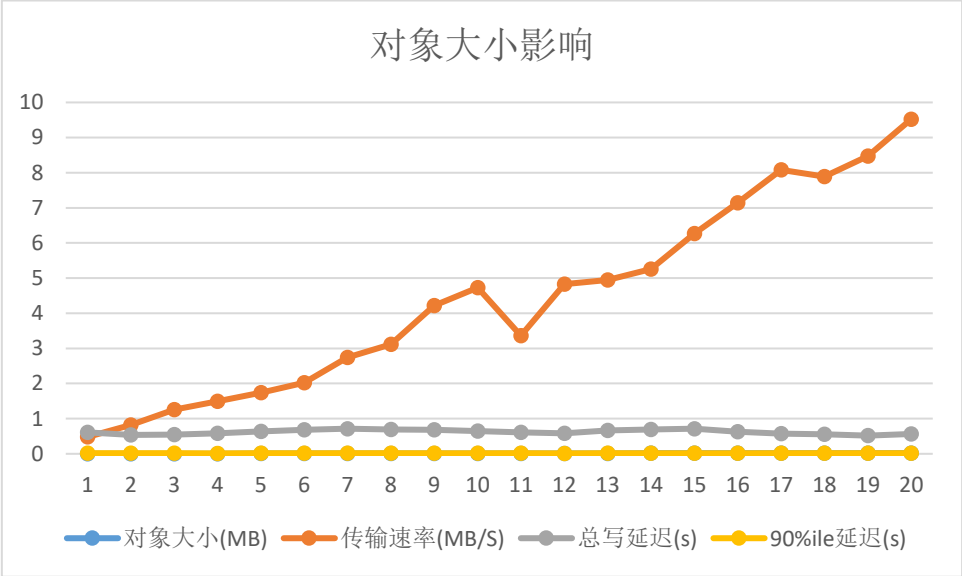

图 5-6 对象大小影响

（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。

（2） 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.