《大数据存储与管理》课程

实 验 报 告

姓　名　程丁丁

学　号　U201813729

班　号　CS1807 班

日　期　2022.04.19

目　　录

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（Object Storage Service，OSS），也叫基于对象的存储，是一种解决和处理离散单元的方法，可提供基于分布式系统之上的对象形式的数据存储服务。对象存储和我们经常接触到的块和文件系统等存储形态不同，它提供 RESTful API 数据读写接口及丰富的 SDK 接口，并且常以网络服务的形式提供数据的访问。简单理解，对象存储类似酒店的代客泊车。顾客（前端应用）把车钥匙交给服务生，换来一张收据（对象的标识符）。顾客不用关心车（数据）具体停在哪个车位，这样省事儿、省时间。

早些年，传统数据中心主要是烟囱式架构，系统扩展性较差加上过去硬盘容量偏小、价格偏高，企业主要用存储保存关键数据，对象存储一直不温不火。在新数据时代，传统数据中心向云计算、大数据、人工智能转变，越来越多的应用和企业在使用新的存储形态，“对象存储”这个词变得炙手可热。

在我们的实验中，使用到了 MinIO 作为服务端。MinIO 是一款高性能、分布式的对象存储系统. 它是一款软件产品, 可以 100%的运行在标准硬件。即 X86 等低成本机器也能够很好的运行 MinIO。MinIO 与传统的存储和其他的对象存储不同的是：它一开始就针对性能要求更高的私有云标准进行软件架构设计。因为 MinIO 一开始就只为对象存储而设计。所以他采用了更易用的方式进行设计，它能实现对象存储所需要的全部功能，在性能上也更加强劲，它不会为了更多的业务功能而妥协，失去 MinIO 的易用性、高效性。 这样的结果所带来的好处是：它能够更简单的实现局有弹性伸缩能力的原生对象存储服务。

三、实验环境

本实验的环境如下：
实验所用的工具均支持 windows 操作系统，故没有使用虚拟机，如图 3-1.

图 3-2 实验环境

软件环境：

对象存储客户端：采用 Minio 进行测试
对象存储服务器端：采用 mc
对象存储测试工具：采用 s3-bench 进行测试

四、实验内容

本次实验是对象存储实验入门实践，前置知识包括 Git 与命令行相关操作。然后是选定对象存储服务端与客户端，选择 Minio 和 MC，运行 Minio 后用测试工具 s3-bench 进行测试。

4.1 对象存储技术实践

1. 服务端准备及性能测试

1）下载 Minio。在 Minio 官网 http://www.minio.org.cn/ 下载服务端和客户端。

2）运行 Minio。打开终端，运行 minio，会看到相关入口。

```
D:\study\大四\大数据存储>.\minio.exe server \data

+------------------------------------------------------------------+
| You are running an older version of MinIO released 2 weeks ago |
| Update: Run `mc admin update`                                    |
+------------------------------------------------------------------+

API: http://10.16.110.49:9000  http://192.168.131.1:9000  http://192.168.228.1:9000  http://127.0.0.1:9000
RootUser: minioadmin
RootPass: minioadmin

Console: http://10.16.110.49:58314 http://192.168.131.1:58314 http://192.168.228.1:58314 http://127.0.0.1:58314
RootUser: minioadmin
RootPass: minioadmin

Command-line: https://docs.min.io/docs/minio-client-quickstart-guide
   $ mc.exe alias set myminio http://10.16.110.49:9000 minioadmin minioadmin

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

图 4-1 运行 Minio

3）在浏览器访问服务器。在浏览器中输入 http://127.0.0.1:9000 可以访问服务器，用户名和密码在上图中已默认给出，登录界面如图 4-2。确认登录后，可以看到界面如图 4-3。

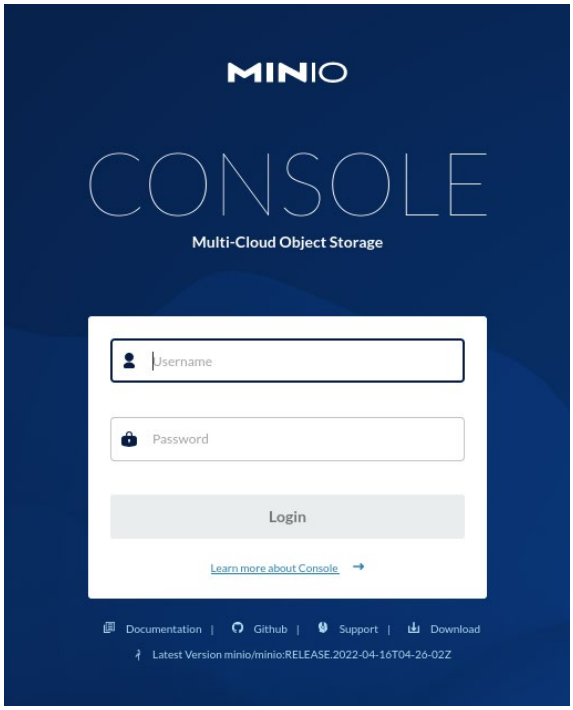

图 4-2 登录服务器

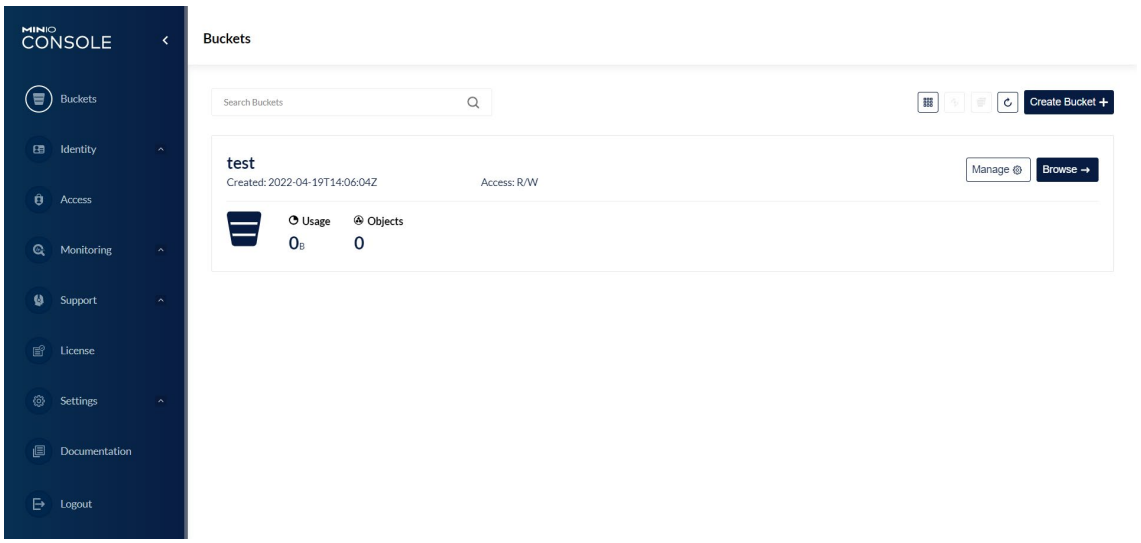

图 4-3 在浏览器中访问服务器

4）在浏览器中可以添加存储对象。点击页面的+号按钮，可以选择新建一个仓库或者上传一个新的存储文件，这里创建一个 test 仓库。

5）运行 MC 客户端进行对服务器的访问。重新打开一个终端，在命令行输入运行服务器之后给的提示，使得可以通过 MC 访问服务器，如图 4-5。

```
D:\study\大四\大数据存储>mc.exe alias set myminio http://10.16.110.49:9000 minioadmin minioadmin
Added `myminio` successfully.
```

```
D:\study\大四\大数据存储>mc.exe ls myminio
[2022-04-19 21:55:08 CST]     0B test/
```

图 4-4 通过 MC 访问服务器

6）下载 s3bench 源码和相关项目并编 https://github.com/igneous-systems/s3bench，预编译的文件 https://share.weiyun.com/BICMfA4G：

7）使用测试工具 s3-bench 进行测试，先修改脚本，再运行，如图 4-5。

```
@rem -accessKey         Access Key
@rem -accessSecret      Secret Key
@rem -bucket=loadgen    Bucket for holding all test objects.
@rem -endpoint=http://127.0.0.1:9000 Endpoint URL of object storage service being tested.
@rem -numClients=8      Simulate 8 clients running concurrently.
@rem -numSamples=256    Test with 256 objects.
@rem -objectNamePrefix=loadgen Name prefix of test objects.
@rem -objectSize=1024          Size of test objects.
@rem -verbose           Print latency for every request.

s3bench.exe ^
    -accessKey=minioadmin ^
    -accessSecret=minioadmin ^
    -bucket=test ^
    -endpoint=http://127.0.0.1:9000 ^
    -numClients=8 ^
    -numSamples=256 ^
    -objectNamePrefix=loadgen ^
    -objectSize=1024
pause
```

```
Generating in-memory sample data... Done (996.2µs)

Running Write test...

Running Read test...

Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           test
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:  0.02 MB/s
Total Duration:    10.749 s
Number of Errors:  0
------------------------------------
Write times Max:       0.975 s
Write times 99th %ile: 0.945 s
Write times 90th %ile: 0.571 s
Write times 75th %ile: 0.358 s
Write times 50th %ile: 0.272 s
Write times 25th %ile: 0.228 s
Write times Min:       0.142 s


Results Summary for Read Operation(s)
Total Transferred: 0.250 MB
Total Throughput:  1.23 MB/s
Total Duration:    0.203 s
Number of Errors:  0
------------------------------------
Read times Max:       0.115 s
Read times 99th %ile: 0.113 s
Read times 90th %ile: 0.003 s
Read times 75th %ile: 0.002 s
Read times 50th %ile: 0.001 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 454.3315ms
```

图 4-5 启动驱动程序与控制器

4.2 对象存储性能分析

首先，测试标准的 s3-bench 结果如表 4-1。可以看到我们改变 object size 等参数时运行时间的变化。

Object size	Num clients	Num samples	写速度	写时间	读速度	读时间
0.001MB	8	256	0.02 MB/s	10.729 s	1.23 MB/s	0.203 s

1MB	8	256	9.05 MB/s s	28.282 s	1816.16 MB/s	0.141 s
10MB	8	256	41.47 MB/s	61.732 s	2237.63 MB/s	1.144 s
20MB	8	256	42.94 MB/s	119.229 s	2269.26 MB/s	2.256 s
10MB	16	256	33.10 MB/s	77.335 s	967.95 MB/s	1.301 s
10MB	24	256	29.39 MB/s	87.110 s	1482.03 MB/s	1.727 s
10MB	8	512	32.28 MB/s	158.594 s	1287.94 MB/s	3.975 s
10MB	8	128	29.01 MB/s	44.116 s	1486.86 MB/s	0.861 s

表 4-1 s3bench 测试结果（minio 作为服务端）

我们可以看到：

1） 写入和读取的成功率一直都是 100%；

2） 读取的 Bandwidth 比写入的 Bandwidth 大，这和我们平时了解的读取速度大于写入速度是一致的；

3） 随着每次读取与写入 size 的增大，Bandwidth 渐渐增大，而 Throughput 渐渐减小，相应的平均的休息时间与工作时间也减小。

4） 随着 object size 增大，读写速度也会逐渐增大

5） 在 object size 相等的情况下，理论上若 client 数量更多则读写速率先升高后降低，即存在一个最优水平，但未观测到

6） 在 object size 相等的情况下，理论上若 sample 数量更多则读写速率先降低后升高，即存在一个最低水平，但未观测到

五、实验总结

本门课程是一门基础性的课程，本试验是面向对象存储的入门实验，在实验中，我了解了对象存储技术的基本概念及其优缺点，明白了面向对象存储系统的应用场景。

实验中，使用了 minio 作为服务端，使用测试工具 s3-bench 进行了测试，考虑到执行时间，进行操作的容量都比较小，在实际应用中肯定会有大的多的数据量，但是对于基本的因素的观察已经足矣。

实验一开始遇到的问题是不知道要做什么，对 minio 等程序不知道该如何使用，后通过与同学们沟通，逐渐明朗，最终顺利完成实验。

感谢学校和老师们的付出！

参考文献

[1]https://gitee.com/shi_zhan/obs-tutorial/tree/master#https://gitee.com/link?target=https%3A%2F%2Fshare.weiyun.com%2FBICMfA4G
[2] https://github.com/igneous-systems/s3bench
[3] https://docs.min.io/minio/baremetal/