2020 级

《物联网数据存储与管理》课程

实 验 报 告

姓　名　郭雯

学　号　U202015585

班　号　计算机 2010 班

日　期　2023.05.15

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，分析性能问题，架设应用实践。

二、实验背景

随着社会的发展，数据量越来越大，数据存储也逐渐被重视起来，本次实验对对象存储系统进行简单的实践。

对象存储是一种将数据作为对象进行管理的计算机数据存储体系结构，与其他存储体系结构（例如将数据作为文件层级管理的文件系统）以及将数据作为块和扇区内的块进行管理的块存储相对。每个对象通常包括数据本身，可变数量的元数据和全局唯一标识符。

对象存储系统是一种用于存储非结构化数据的数据存储技术。相比于传统的文件系统或块存储，对象存储系统具有更高的可扩展性、可靠性和安全性，能够轻松地处理海量数据。

而本次实验主要使用 minio 作为服务器端，mc 作为客户端，并使用 s3-bench 进行测试。

minio 是在 GNU Affero 通用公共许可证 v3.0 下发布的高性能对象存储。它与 Amazon S3 云存储服务 API 兼容。使用 minio 为机器学习、分析和应用程序数据工作负载构建高性能基础架构。

S3bench 提供了针对兼容 S3 的端点运行非常基本的吞吐量基准测试的功能。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

计算机环境：

3-1

软件环境：

对象存储客户端：Minio

对象存储服务端：mc

对象存储测试工具：s3-bench

四、实验内容

实验主要需要完成：

（1） 对象存储技术的实践，即通过 minio 和 mc 创建服务器端和客户端，并进行简单的操作实践。

（2） 对象存储性能的分析。即使用 s3-bench，通过改变不同的参数，进行性能的分析测试，并对结果进行记录分析。

4.1 对象存储技术实践

1. 搭建 minio 服务器端，进行简单的操作。
2. 搭建 mc 客户端，通过客户端对服务器进行操作实践。
3. 通过 s3-bench 脚本，学会使用脚本。

4.2 对象存储性能分析

1. 通过改变不同参数，如对象尺寸 objectSize、并发数 numClients、样本数 numSamples，测试吞吐率 Throughput、延迟 Latency 等指标，并记录结果制成表格。
2. 分析不同参数下的结果，总结各参数对存储性能的影响。

五、实验过程

5.1 对象存储技术实践

到官网下载 Minio 的服务器 minio 和客户端 mc。

将其放到 D 盘的 minio 文件夹下。

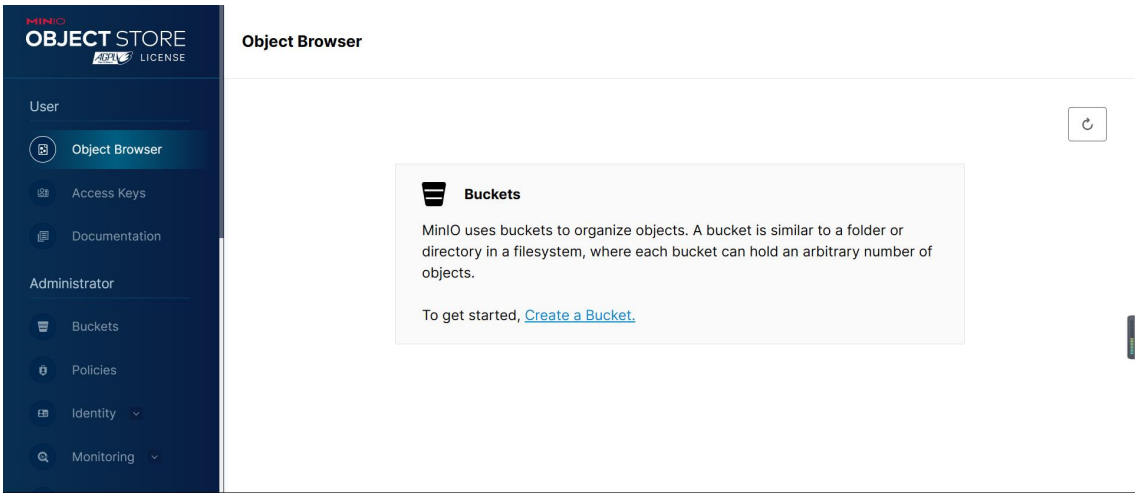
在命令行输入 minio.exe server D:minio 运行 Minio。

运行结果如下：

```
D:\minio>minio.exe server D:minio
WARNING: Detected default credentials 'minioadmin:minioadmin', we recommend that yo
T_USER' and 'MINIO_ROOT_PASSWORD' environment variables
MinIO Object Storage Server
Copyright: 2015-2023 MinIO, Inc.
License: GNU AGPLv3 <https://www.gnu.org/licenses/agpl-3.0.html>
Version: RELEASE.2023-04-13T03-08-07Z (go1.20.3 windows/amd64)

Status:         1 Online, 0 Offline.
API: http://192.168.43.81:9000  http://127.0.0.1:9000
RootUser: minioadmin
RootPass: minioadmin
Console: http://192.168.43.81:51556 http://127.0.0.1:51556
RootUser: minioadmin
RootPass: minioadmin

Command-line: https://min.io/docs/minio/linux/reference/minio-mc.html#quickstart
   $ mc.exe alias set myminio http://192.168.43.81:9000 minioadmin minioadmin

Documentation: https://min.io/docs/minio/linux/index.html
Warning: The standard parity is set to 0. This can lead to data loss.

+------------------------------------------------------------+
| You are running an older version of MinIO released 3 weeks ago |
| Update: Run `mc admin update`                              |
+------------------------------------------------------------+
```

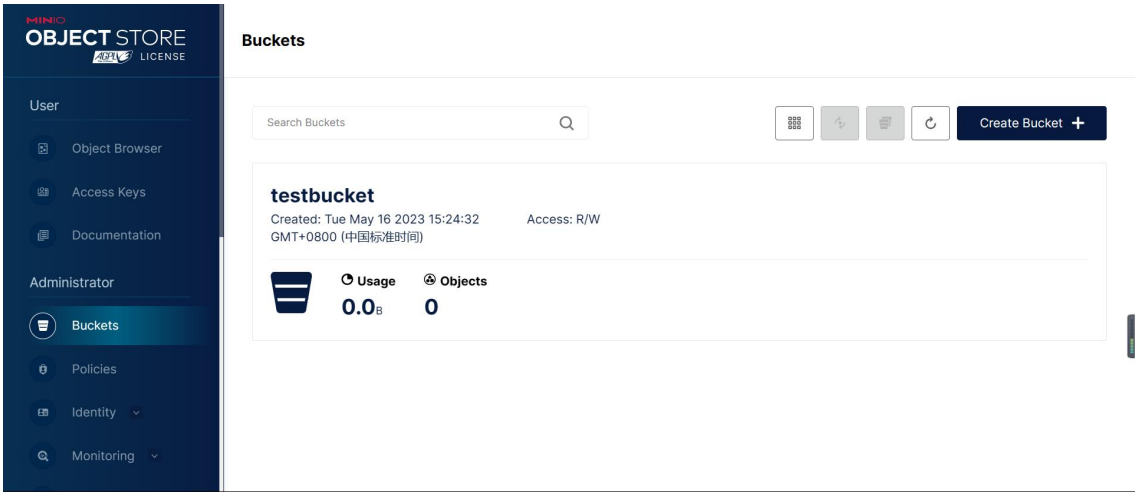
根据结果可得知，服务端调用的 URL 为本机网络 IP 的 9000 端口，服务器控制台为本机网络 IP 的 51556 端口，登录的初始账号名和密码为默认值 minioadmin。

用浏览器访问 http://192.168.43.81:51556，输入上述用户名和密码后，进入如图所示界面。



5-2

点击 Create a Bucket 来创建一个 bucket，如图所示。



5-3

随后运行 minio 客户端 mc 进行服务器的访问。

根据之前运行服务器命令行给出的提示，重新打开一个新的命令行输入 mc.exe alias set myminio http://192.168.43.81:9000 minioadmin minioadmin，通过 mc 访问服务器。结果如下：

```
D:\minio>mc.exe alias set myminio http://192.168.43.81:9000 minioadmin minioadmin
Added `myminio` successfully.
```

5-4

通过 mc.exe mb myminio/newbucket 创建新的 bucket，结果如下：

```
D:\minio>mc.exe mb myminio/newbucket
Bucket created successfully `myminio/newbucket`.
```

5-5

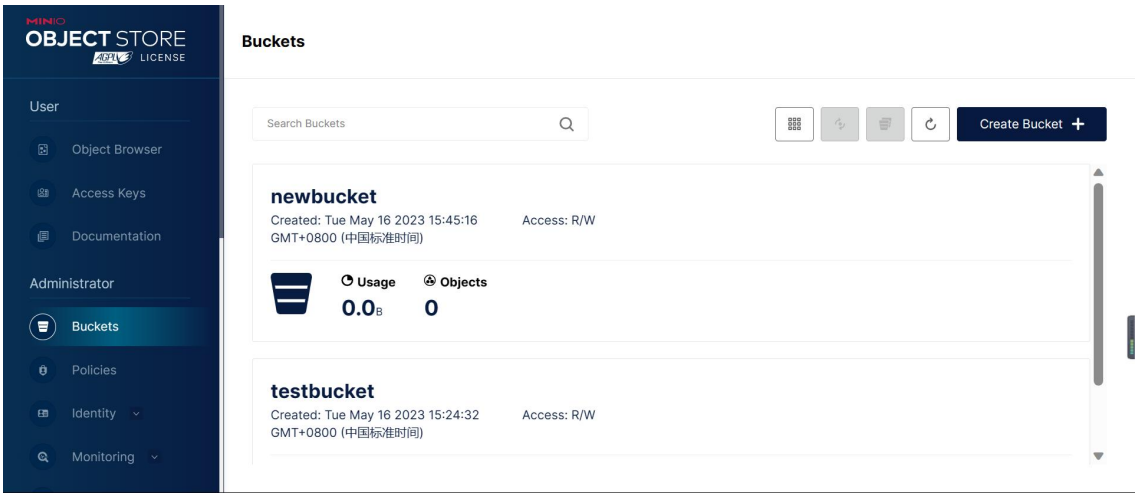
通过 mc.exe ls myminio 来查看服务端的 bucket 个数。通过以下结果可发现目

前有创建成功的两个 bucket：

```
D:\minio>mc.exe ls myminio
[2023-05-16 15:45:16 CST]     0B newbucket/
[2023-05-16 15:24:32 CST]     0B testbucket/
```

5-6

浏览器显示结果如下，表明创建成功：



5-7

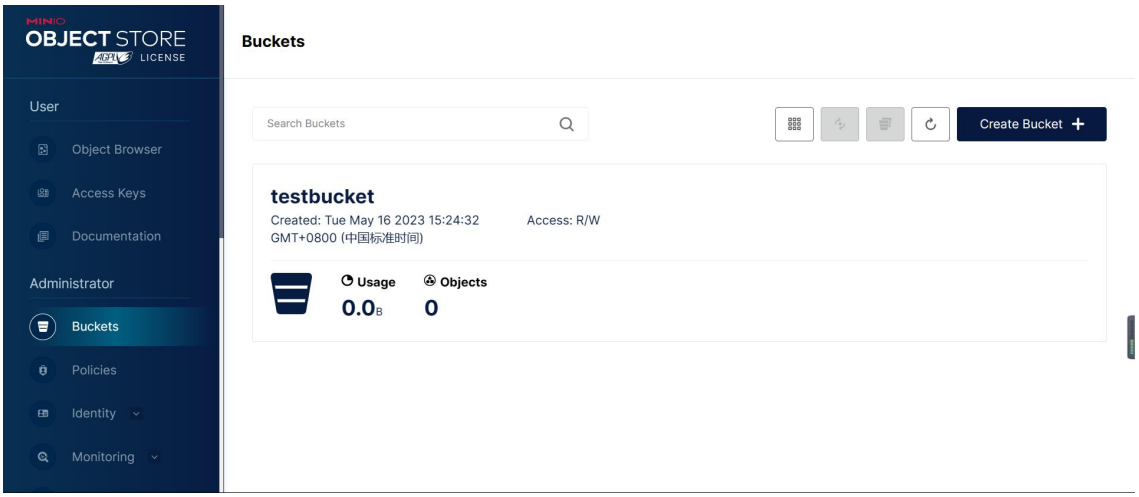
通过 mc rb myminio/newbucket 来删除刚刚创建的 bucket，结果如下：

```
D:\minio>mc rb myminio/newbucket
Removed `myminio/newbucket` successfully.
```

```
D:\minio>mc.exe ls myminio
[2023-05-16 15:24:32 CST]     0B testbucket/
```

5-8

此时查看浏览器，发现只有一开始创建的 bucket，删除 bucket 成功。



5-9

S3 Bench 安装运行

首先下载提供链接中的 s3bench.exe，之后下载运行脚本，将二者置于同一目录下。

编辑 run-s3bench.cmd，修改用户名、密码、bucket。修改结果如下：

```
@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=testbucket ^
    -endpoint=http://127.0.0.1:9000 ^
    -numClients=8 ^
    -numSamples=256 ^
    -objectNamePrefix=loadgen ^
    -objectSize=1024
pause
```

5-10

保存后双击 run-s3bench.cmd 运行，结果如下：

```
Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           testbucket
objectNamePrefix: loadgen
objectSize:       0.0010 MB
numClients:       8
numSamples:       256
verbose:       %!d(bool=false)
tracing:       %!d(bool=false)


Results Summary for Write Operation(s)
Total Transferred: 0.250 MB
Total Throughput:  0.88 MB/s
Total Duration:    0.284 s
Number of Errors:  0
------------------------------------
Write times Max:       0.022 s
Write times 99th %ile: 0.022 s
Write times 90th %ile: 0.012 s
Write times 75th %ile: 0.010 s
Write times 50th %ile: 0.008 s
Write times 25th %ile: 0.007 s
Write times Min:       0.005 s


Results Summary for Read Operation(s)
Total Transferred: 0.250 MB
Total Throughput:  4.44 MB/s
Total Duration:    0.056 s
Number of Errors:  0
------------------------------------
Read times Max:       0.004 s
Read times 99th %ile: 0.003 s
Read times 90th %ile: 0.002 s
Read times 75th %ile: 0.002 s
Read times 50th %ile: 0.002 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 480.4173ms
```

5-11

5.2 对象存储性能分析

使用控制变量，改变不同参数，如对象尺寸 objectSize、并发数 numClients、样本数 numSamples，测试吞吐率 Throughput、读写时间 Duration 等指标。

（1）固定 objectSize 为 1024，numSamples 为 256，分别改变并发数 numClients 为 8、16、24；

（2）固定 objectSize 为 1024，numClients 为 8，分别改变样本数 numSamples 为 128、256、512；

（3）固定 numClients 为 8，numSamples 为 256，分别改变对象尺寸 objectSize 为 1024、2048、4096；

分别进行性能测试，观察不同值下的指标变化。

s3bench 脚本运行得到的各测试结果所构成的表格如下：

numClients	numSamples	objectSize	Write		Read	
			Throughput	Duration	Throughput	Duration
8	256	1024	0.88 MB/s	0.284 s	4.44 MB/s	0.056 s
16	256	1024	1.23 MB/s	0.203 s	4.42 MB/s	0.057 s
24	256	1024	1.16 MB/s	0.215 s	4.31 MB/s	0.058 s
8	128	1024	0.92 MB/s	0.136 s	4.37 MB/s	0.029 s
8	512	1024	1.19 MB/s	0.420 s	4.53 MB/s	0.110 s
8	256	2048	2.05 MB/s	0.244 s	8.80 MB/s	0.057 s
8	256	4096	3.51 MB/s	0.285 s	18.58 MB/s	0.054 s

观察表中数据，经分析可得如下结论：

① 当对象尺寸增大时，吞吐率随之增大，读写时间基本不变。

② 当样本数量增多时，吞吐率基本不变，而读写的时间会随之增大。

③ 当客户端并发数增大时，吞吐率先上升在下降，时间则先下降再上升，但这里读时间的结果由于差异不大并未明显地观察到。

④ 同等条件下，读速率一般远大于写速率。

六、实验总结

本实验我使用了 minio 作为服务器、mc 作为客户端，并用 s3-bench 对不同影响因素下的存储性能对比进行了测试观察，最终得出一定结论。不过由于我选用的测试数据并不多，每组也只有三个数据，只做了部分数据，所以也很可惜无法得到十分准确的结论。

不过本次实验还是让我收获颇多。通过本次实验，我对对象存储技术有了进一步的了解，同时对相关的软件有了一定的认识，比如了解了 minio 和 mc 的基本使用。不过由于本次实验我选用的都是很基本的软件，而且直接在 Windows 上进行实验，所以试验过程中并没有遇到太多困难。不过没能尝试其他更入门一些的软件也是一个小小的遗憾，数据存储应用十分广泛，日后如果有时间还是希望能进一步接触更多的知识。

参考文献

[1] 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.
[2] ARNOLD J. OpenStack Swift[M]. O’Reilly Media, 2014.
[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.
[4] Dean J, Barroso L A. Association for Computing Machinery, 2013. The Tail at Scale[J]. Commun. ACM, 2013, 56(2): 74－80.
[5] Delimitrou C, Kozyrakis C. Association for Computing Machinery, 2018. Amdahl’s Law for Tail Latency[J]. Commun. ACM, 2018, 61(8): 65－72.