2018 级

《物联网数据存储与管理》课程

实 验 报 告

姓　名　徐 屹

学　号　U201814727

班　号　物联网 1801 班

日　期　2021.06.21

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-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。1999 年成立的全球网络存储工业协会（SNIA）的对象存储设备工作组发布了 ANSI 的 X3T10 标准。总体上来讲，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。

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只能，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。

（1）对象

对象是系统中数据存储的基本单位，每个 Obeject 是数据和数据属性集的综合体，数据属性可以根据应用的需求进行设置，包括数据分布、服务质量等。在传统的存储系统中用文件或块作为基本的存储单位，块设备要记录每个存储数据块在设备上的位置。Object 维护自己的属性，从而简化了存储系统的管理任务，增加了灵活性。Object 的大小可以不同，可以包含整个数据结构，如文件、数据表项等。在存储设备中，所有对象都有一个对象标识，通过对象标识 OSD 命令访问对象。通常由多种类型的对象，存储设备上的根对象标识存储设备和该设备的各种属性，组队象是存储设上共享资源管理策略的对象集合等。

（2）对象存储设备

每个 OSD 都是一个智能设备，具有自己的存储介质、处理器、内存以及网络系统等，负责管理本地的 Object，是对象存储系统的核心。OSD 同块设备的不同不在于存储介质，而在于两者提供的访问接口。OSD 的主要功能包括数据存储和安全访问、目前国际上通常采用刀片式结构实现对象存储设备。OSD 提供三个主要功能：

① 数据存储。OSD 管理对象数据，并将它们放置在标准的磁盘系统上，OSD 不提供接口访问方式，Client 请求数据时用对象 ID、偏移进行数据读写。

② 智能分布。OSD 用其自身的 CPU 和内存优化数据分布，并支持数据的预取。由于 OSD 可以智能地支持对象的预取，从而可以优化磁盘的性能。

③ 每个对象数据的管理。OSD 管理存储在其它对象上的元数据，该元数据与传统的 inode 元数据相似，通常包括对象的数据块和对象的长度。而在传统的 NAS 系统中，这些元数据是由文件服务器提供的，对象存储架构将系统中主要的元数据管理工作由 OSD 来完成，降低了 Client 的开销。

（3）元数据服务器（Metadata Server，MDS）

MDS 控制 Client 与 OSD 对象的交互，为客户端提供元数据，主要是文件的逻辑视图，包括文件与目录的组织关系、每个文件所对应的 OSD 等。主要提供以下几个功能：

① 对象存储访问。MDS 构造、管理描述每个文件分布的视图，允许 Client 直接访问对象。MDS 为 Client 提供访问该文件所含对象的能力，OSD 在接收到每个请求时先验证该能力，然后才可以访问。

② 文件和目录访问管理。MDS 在存储系统上构建一个文件结构，包括限额控制、目录和文件的创建和删除、访问控制等。

③ Client Cache 一致性。为了提高 Client 性能，在对象存储系统设计时通常支持 Client 的 Cache。由于引入 Client 方的 Cache，带来了 Cache 一致性的问提，MDS 支持基于 Client 的文件 Cache，当 Cache 的文件发生改变时，将通知 Client 刷新 Cvashe，从而防止 Cache 不一致引发的问提。

（4）对象存储系统的客户端 Client

为了有效支持 Client 支持访问 OSD 上的对象，需要在计算节点实现对象存储系统的 Client。现有的应用对数据的访问大部分都是通过 POSIX 文件方式进行的，同时为了提高性能，也具有对数据的 Cache 功能和文件的条带功能。同时，文件系统必须维护不同客户端上 Cache 的一致性，保证文件系统的数据一致。文件系统访问流程：

① 客户端应用发出读请求；

② 文件系统向元数据服务器发送请求，获取要读取的数据所在的 OSD；

③ 然后直接向每个 OSD 发送数据读取请求；

④ OSD 得到请求后，判断要读取的 Object，并根据此 Object 的认证方式，对客户端进行认证，如果客户端得到收授权，则将 Object 的数据返回给客户端；

⑤ 文件系统收到 OSD 返回的数据以后，读操作完成。

三、实验环境

硬件环境：

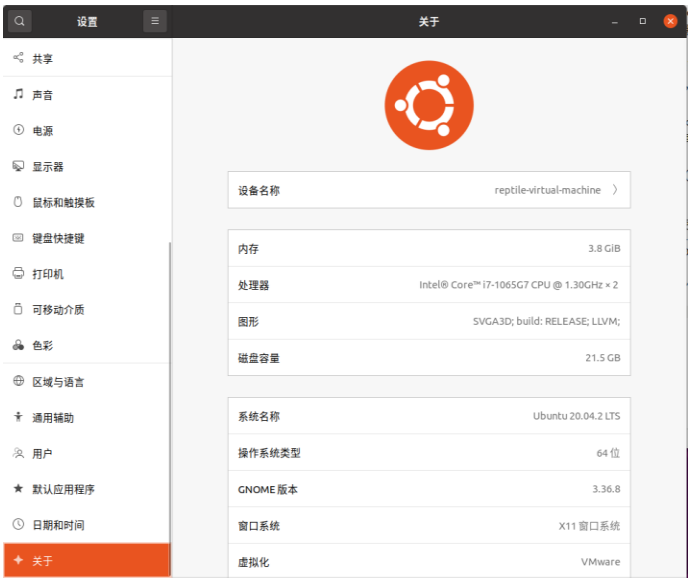

图 1 实验环境

操作系统：Ubuntu 20.04.2 LTS
CPU：Intel® Core™ i7-1065G7 CPU @ 1.30GHz × 2
内存：4G

软件环境：
对象存储客户端：采用 mock_s3 客户端进行测试；
对象存储服务器端：采用 osm 作为服务器端；
对象存储测试工具：采用 s3bench 进行测试。

四、实验内容

本次实验中采用 mock_s3 作为服务端，osm 作为客户端，并用 s3bench 来进行评测。

4.1 对象存储技术实践

熟悉和掌握操作服务端+客户端+评测
（1）对象存储服务端 mock_s3
（2）对象存储客户端 osm
（3）对象存储评测工具 s3bench

4.2 对象存储性能分析

修改 run-s3bench.sh 的内容，accessKey、accessSecret、endpoint 默认的就可以，主要是设计桶的名称 bucket、并行运行的客户端的数目 numClients、测试对象的数目 numSamples、测试对象的前缀名 objectNamePrefix、测试对象的大小 objectSize 等等。观察和分析配置参数和性能指标之间的潜在关系。

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
python3 setup.py install
python3 mock_s3/main.py --hostname 0.0.0.0 --port 9000 --root ./root

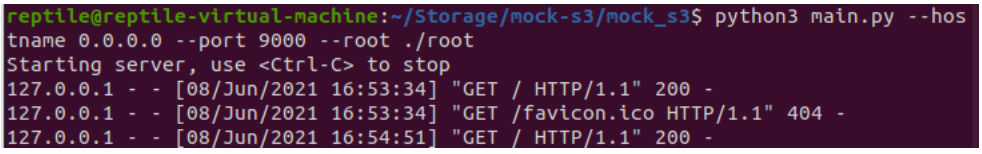

图 2 启动服务端

打开 web 访问 127.0.0.1:9000，发现终端会返回记录，状态码为 200，成功访问。

也可以网页端看到 bucket 的 html 显示：

图 3 查看 bucket

（2）对象存储服务端 osm 的安装、配置和启动

① go get -u github.com/appscode/osm（会自动下载到主目录下的 go 下的 bin 下面）

② cd 到该目录下，添加可执行权限（下载的是二进制文件，需要赋予执行权限）

chmod +x osm

③ sudo mv osm /usr/local/bin/，终端输入 osm 可以发现安装成功了

```
reptile@reptile-virtual-machine:~/Storage/mock_s3+osm+S3Bench/evaluate$ osm
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
```

图 4 osm

可以用 osm 命令查看 bucket：

```
reptile@reptile-virtual-machine:~/Storage/mock_s3+osm+S3Bench$ cd osm
reptile@reptile-virtual-machine:~/Storage/mock_s3+osm+S3Bench/osm$ osm lc
loadgen
Found 1 container in
```

图 5 osm 命令

还可以用 osm 命令查看 bucket 下的对象：

```
reptile@reptile-virtual-machine:~/Storage/mock_s3+osm+S3Bench/osm$ osm ls loadgen
loadgen190
loadgen98
loadgen120
loadgen235
loadgen198
loadgen54
loadgen60
loadgen116
loadgen149
loadgen151
loadgen187
loadgen64
loadgen155
loadgen153
loadgen245
loadgen94
loadgen125
loadgen21
loadgen82
loadgen195
loadgen218
loadgen148
loadgen227
loadgen200
loadgen35
loadgen89
loadgen7
loadgen169
loadgen16
loadgen28
loadgen63
loadgen219
loadgen215
loadgen213
loadgen42
loadgen44
loadgen134
loadgen96
loadgen206
loadgen59
loadgen99
loadgen2
loadgen39
loadgen127
loadgen222
loadgen150
loadgen223
loadgen9
loadgen77
loadgen4
Found 50 items in container loadgen
```

图 6 osm 命令

（3）对象存储评测工具 s3bench 的安装、配置和启动

① go get github.com/igneous-systems/s3bench（会自动下载到主目录下的 go 下的 bin 下面）

② cd 到该目录下，添加可执行权限（下载的是二进制文件，需要赋予执行权限）：chmod +x s3bench

③ 配置：

bash s3bench \ -accessKey=hust -accessSecret=hust_obs \ -endpoint=http://127.0.0.1:9000 \ -bucket=loadgen -objectNamePrefix=loadgen \ -numClients=8 -numSamples=256 -objectSize=1024

④ 启动（服务端启动情况下，进行评测）：

bash run-s3bench.sh

```
reptile@reptile-virtual-machine:~/Storage/minio+mc+S3Bench/evaluate$ bash run-s3bench.sh
Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           loadgen
objectNamePrefix: loadgen
objectSize:       0.0312 MB
numClients:       8
numSamples:       256
verbose:       %!d(bool=false)


Generating in-memory sample data... Done (12.591223ms)

Running Write test...

Running Read test...

Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           loadgen
objectNamePrefix: loadgen
objectSize:       0.0312 MB
numClients:       8
numSamples:       256
verbose:       %!d(bool=false)


Results Summary for Write Operation(s)
Total Transferred: 8.000 MB
Total Throughput:  0.77 MB/s
Total Duration:    10.358 s
Number of Errors:  0
---------------------------------
Write times Max:       1.377 s
Write times 99th %ile: 1.265 s
Write times 90th %ile: 0.507 s
Write times 75th %ile: 0.373 s
Write times 50th %ile: 0.281 s
Write times 25th %ile: 0.202 s
Write times Min:       0.089 s


Results Summary for Read Operation(s)
Total Transferred: 8.000 MB
Total Throughput:  0.71 MB/s
Total Duration:    11.218 s
Number of Errors:  0
---------------------------------
Read times Max:       1.392 s
Read times 99th %ile: 1.286 s
Read times 90th %ile: 0.504 s
Read times 75th %ile: 0.391 s
Read times 50th %ile: 0.313 s
Read times 25th %ile: 0.241 s
Read times Min:       0.147 s


Cleaning up 256 objects...
Deleting a batch of 256 objects in range {0, 255}... Succeeded
Successfully deleted 256/256 objects in 151.500222ms
```

图 7 评测结果

可以发现服务端进行处理，接收 Create | INSERT | PUT / POST 的请求：

```
reptile@reptile-virtual-machine:~/Storage/mock-s3$ python3 mock_s3/main.py --hostname 0.0.0.0 --port 9000 --root ./root
Starting server, use <Ctrl-C> to stop
127.0.0.1 - - [08/Jun/2021 18:56:20] "PUT /loadgen/loadgen1 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:20] "PUT /loadgen/loadgen3 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:20] "PUT /loadgen/loadgen0 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:20] "PUT /loadgen/loadgen7 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:20] "PUT /loadgen/loadgen4 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:20] "PUT /loadgen/loadgen2 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:20] "PUT /loadgen/loadgen6 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:21] "PUT /loadgen/loadgen5 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:21] "PUT /loadgen/loadgen8 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:21] "PUT /loadgen/loadgen11 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:21] "PUT /loadgen/loadgen9 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:21] "PUT /loadgen/loadgen12 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:21] "PUT /loadgen/loadgen13 HTTP/1.1" 200 -
```

图 8 服务端进行处理

```
127.0.0.1 - - [08/Jun/2021 18:56:41] "GET /loadgen/loadgen239 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:41] "GET /loadgen/loadgen240 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:41] "GET /loadgen/loadgen241 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:41] "GET /loadgen/loadgen244 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:41] "GET /loadgen/loadgen242 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:41] "GET /loadgen/loadgen243 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:41] "GET /loadgen/loadgen247 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:41] "GET /loadgen/loadgen248 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:41] "GET /loadgen/loadgen245 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:41] "GET /loadgen/loadgen246 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:41] "GET /loadgen/loadgen250 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:41] "GET /loadgen/loadgen249 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:42] "GET /loadgen/loadgen251 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:42] "GET /loadgen/loadgen252 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:42] "GET /loadgen/loadgen255 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:42] "GET /loadgen/loadgen253 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:42] "GET /loadgen/loadgen254 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:42] "GET /loadgen/loadgen237 HTTP/1.1" 200 -
127.0.0.1 - - [08/Jun/2021 18:56:42] "POST /loadgen?delete= HTTP/1.1" 200 -
```

图 9 服务端进行处理

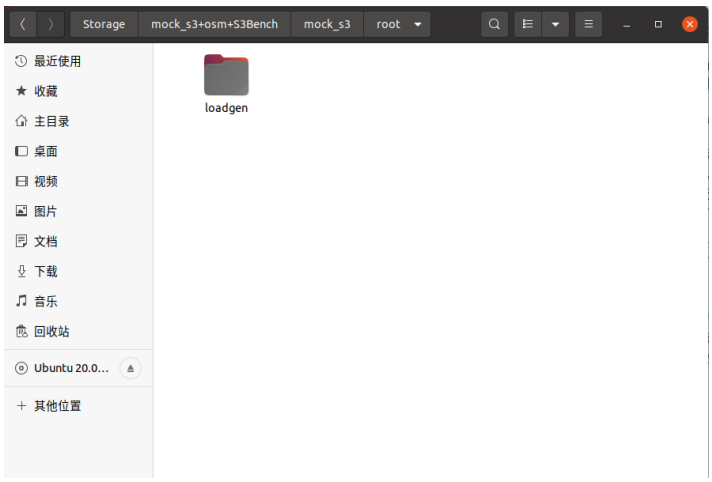

图 10 评测

同时发现 mock_s3 下的 root 文件夹下出现第三步配置的测试 bucket，打开发现对象有 256 个。

下面的评测修改参数时可以使用第三步配置的命令，也可以直接打开文件 run-s3bench.sh 进行修改。

5.2 评测

（1）启动服务端，再打开评测工具：bash run-s3bench.sh

（2）修改一下参数进行分析：

-numClients=8 \

-numSamples=256 \

-objectSize=$((1024*32))

（3）对象尺寸对性能的影响

numClients	numSamples	objectSize		numClients	numSamples	objectSize
8	256	1024		8	256	32
	write	read			write	read
total transferred / MB	8	8		total transferred / MB	0.25	0.25
total throughput / (MB/s)	2.43	3.67		total throughput / (MB/s)	0.08	0.12
total duration / s	3.293	2.179		total duration / s	3.002	2.079
number of errors	0	0		number of errors	0	0
write times max / s	1.114	1.081		write times max / s	1.115	1.072
write times min / s	0.025	0.008		write times min / s	0.024	0.014
numClients	numSamples	objectSize		numClients	numSamples	objectSize
8	256	10240		8	256	102400
	write	read			write	read
total transferred / MB	80	80		total transferred / MB	800	800
total throughput / (MB/s)	33.17	71.13		total throughput / (MB/s)	20.51	217.09
total duration / s	2.411	1.125		total duration / s	38.997	3.685
number of errors	0	0		number of errors	0	0
write times max / s	1.1	1.067		write times max / s	1.996	1.151
write times min / s	0.028	0.007		write times min / s	1.088	0.034

图 11 对象尺寸对性能的影响

分析：分别修改 objectSize 为 1024、32、10240、102400，传输总量、吞吐量和总持续时间随之变化，但对比评测结果却可以发现对象的尺寸对读写的速度影响几乎没有。

（4）客户端爆满

```
reptile@reptile-virtual-machine:~/Storage/mock_s3+osm+S3Bench/evaluate$ bash run-s3bench.sh
Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           loadgen
objectNamePrefix: loadgen
objectSize:       0.0312 MB
numClients:       256
numSamples:       256
verbose:       %!d(bool=false)


Generating in-memory sample data... Done (3.47032ms)

Running Write test...

Running Read test...

Test parameters
endpoint(s):      [http://127.0.0.1:9000]
bucket:           loadgen
objectNamePrefix: loadgen
objectSize:       0.0312 MB
numClients:       256
numSamples:       256
verbose:       %!d(bool=false)


Results Summary for Write Operation(s)
Total Transferred: 3.781 MB
Total Throughput:  0.51 MB/s
Total Duration:    7.438 s
Number of Errors:  135
---------------------------------
Write times Max:       7.263 s
Write times 99th %ile: 7.254 s
Write times 90th %ile: 3.404 s
Write times 75th %ile: 3.142 s
Write times 50th %ile: 1.291 s
Write times 25th %ile: 0.400 s
Write times Min:       0.163 s


Results Summary for Read Operation(s)
Total Transferred: 3.781 MB
Total Throughput:  0.20 MB/s
Total Duration:    19.020 s
Number of Errors:  135
---------------------------------
Read times Max:       18.828 s
Read times 99th %ile: 10.061 s
Read times 90th %ile: 5.698 s
Read times 75th %ile: 3.359 s
Read times 50th %ile: 2.865 s
Read times 25th %ile: 1.231 s
Read times Min:       0.159 s


Cleaning up 256 objects...
Deleting a batch of 256 objects in range {0, 255}... Succeeded
Successfully deleted 256/256 objects in 14.550996ms
```

图 12 客户端爆满

客户端爆满		
numClients	numSamples	objectSize
256	256	1024
	write	read
total transferred / MB	3.781	3.781
total throughput / (MB/s)	0.51	0.2
total duration / s	7.438	19.02
number of errors	135	135
write times max / s	7.263	18.828
write times min / s	0.163	0.159

图 13 客户端爆满

分析：客户端爆满时，错误率极高（135/256），总传输量减少了一半多（应该为8MB），吞吐量下降，持续时间长，读写速度也十分不稳定

六、实验总结

本次实验采用 mock_s3 作为服务端，osm 作为客户端，并用 s3bench 来进行评测。一开始入门时用的是 minio + mc +COSBench，后来发现用 s3bench 改测试的参数特别的方便，就慢慢地过渡到 mock_s3 + osm + s3bench 了。

面向对象存储的入门实验也使我了解到了一种新的存储技术，并明白了其的优越性，以及为什么在已有基于块和文件的存储系统以及网络附加存储等技术后，仍需要面向对象存储技术。

总的来说，通过这次实验，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，同时自己亲自动手测试了一些性能指标（对象尺寸对性能的影响、客户端爆满），培养了动手能力，也丰富了理论知识。

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.

（可以根据实际需要更新调整）