

2019 级

《物联网数据存储与管理》课程

实 验 报 告

姓 名 陆云龙

学 号 U201914987

班 号 计算机 1903 班

日 期 2022.04.19

目 录

一、实验目的	1
二、实验背景	1
三、实验环境	1
四、实验内容	2
4.1 对象存储技术实践	2
4.2 对象存储性能分析	2
五、实验过程	3
六、实验总结	5
参考文献	7

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

存储局域网（SAN）和网络附加存储（NAS）是目前两种主流网络存储架构，而对象存储（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。

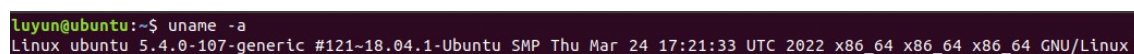
本次实验对象存储服务端使用了 Openstack Swift，由 Rackspace 开发的用来为云计算提供可扩展存储的项目。Swift 是 OpenStack 的对象存储组件，无需采用 RAID（磁盘冗余阵列），也没有中心单元或主控结点。Swift 通过在软件层面引入一致性哈希技术和数据冗余性，牺牲一定程度的数据一致性来达到高可用性（High Availability，简称 HA）和可伸缩性，支持多租户模式、容器和对象读写操作，适合解决互联网的应用场景下非结构化数据存储问题。

对象存储客户端使用了基于 python 的 python-swiftclient 终端客户端。

对象存储测评工具为开源工具 swift-bench。通过编写脚本模拟各种情况下的请求，从而得到性能结果。

三、实验环境

在实验中，我选择了 Linux 虚拟机作为实验进行的环境，虚拟机操作系统为 Ubuntu18.04，如图 1 所示：



```
luyun@ubuntu:~$ uname -a
Linux ubuntu 5.4.0-107-generic #121-18.04.1-Ubuntu SMP Thu Mar 24 17:21:33 UTC 2022 x86_64 x86_64 x86_64 GNU/Linux
```

图 1 虚拟机环境

虚拟机下安装 docker，便于使用 Openstack Swift 镜像，docker 版本信息如图 2 所示：

```
Client:
Version:      20.10.7
API version:  1.41
Go version:   go1.13.8
Git commit:   20.10.7-0ubuntu5~18.04.3
Built:        Mon Nov  1 01:04:14 2021
OS/Arch:      linux/amd64
Context:      default
Experimental: true

Server:
Engine:
Version:      20.10.7
API version:  1.41 (minimum version 1.12)
Go version:   go1.13.8
Git commit:   20.10.7-0ubuntu5~18.04.3
Built:        Fri Oct 22 00:57:37 2021
OS/Arch:      linux/amd64
Experimental: false
containerd:
Version:      1.5.5-0ubuntu3~18.04.2
GitCommit:
runc:
Version:      1.0.1-0ubuntu2~18.04.1
GitCommit:
docker-init:
Version:      0.19.0
GitCommit:
```

图 2 docker 环境

Openstack Swift 服务端使用的是老师提供的开箱即用容器版。相应的客户端选择使用基于 python 的 python-swiftclient 终端客户端。python 版本如图 3 所示：

```
luyun@ubuntu:~$ python3 --version
Python 3.6.9
```

图 3 python 环境

四、实验内容

4.1 对象存储技术实践

1. 搭建虚拟机环境，配置 python 环境，下载相应依赖软件；
2. 部署 Openstack Swift 服务端，安装 swift，使用 swift 命令初步使用对象存储服务，进行文件的上传，下载，删除容器以及增加容器等操作；

4.2 对象存储性能分析

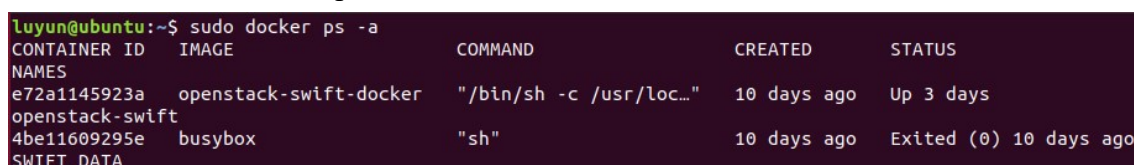
1. 安装 Swift-Bench，配置相应设置，使之能够访问 Openstack，进行简易的测试。
2. 编写代码，使用 Swift-Bench 进行多组测试，汇总测试结果并做出分析。

五、实验过程

(一)Openstack Swift 部署

由于选用容器版 Openstack Swift，需要先配置 docker 环境。由于 Ubuntu18.04 自带 Python3.6，故不再需要进行其他配置，直接使用 `apt-get install docker.io` 即可进行安装。

下载课程资料提供的 Openstack Swift 容器，在相应目录下使用 `docker build -t openstack-swift-docker .` 命令生成 docker 镜像。随后使用 `docker run -v /srv --name SWIFT_DATA busybox` 为其准备数据卷。执行 `docker run -d --name openstack-swift -p 12345:8080 --volumes-from SWIFT_DATA -t openstack-swift-docker` 命令会开始运行此容器。使用 `docker ps -a` 可以查看所有镜像，结果如图 4 所示：

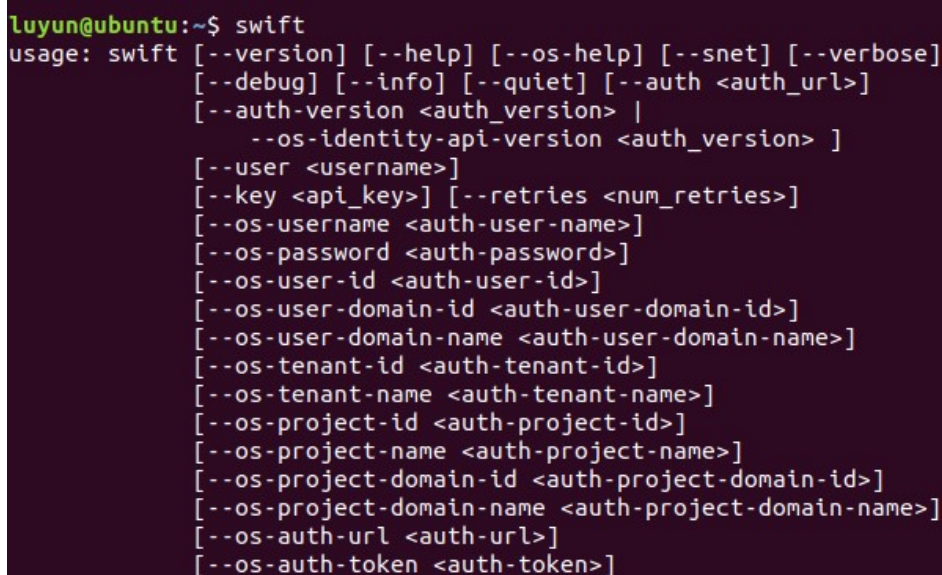


CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
e72a1145923a	openstack-swift-docker	"/bin/sh -c /usr/loc..."	10 days ago	Up 3 days
openstack-swift				
4be11609295e	busybox	"sh"	10 days ago	Exited (0) 10 days ago
SWIFT_DATA				

图 4 Openstack Swift 部署

(二)Swift 安装

在安装容器镜像后，应当已经安装好了 Swift，使用 Swift 命令可以查看安装情况，出现如图 5 所示：



```
luyun@ubuntu:~$ swift
usage: swift [--version] [--help] [--os-help] [--snet] [--verbose]
           [--debug] [--info] [--quiet] [--auth <auth_url>]
           [--auth-version <auth_version> |
           --os-identity-api-version <auth_version> ]
           [--user <username>]
           [--key <api_key>] [--retries <num_retries>]
           [--os-username <auth-user-name>]
           [--os-password <auth-password>]
           [--os-user-id <auth-user-id>]
           [--os-user-domain-id <auth-user-domain-id>]
           [--os-user-domain-name <auth-user-domain-name>]
           [--os-tenant-id <auth-tenant-id>]
           [--os-tenant-name <auth-tenant-name>]
           [--os-project-id <auth-project-id>]
           [--os-project-name <auth-project-name>]
           [--os-project-domain-id <auth-project-domain-id>]
           [--os-project-domain-name <auth-project-domain-name>]
           [--os-auth-url <auth-url>]
           [--os-auth-token <auth-token>]
```

图 5 Swift 测试

随后，使用 `pip3 install python-swiftclient` 命令进行安装 Swift 客户端用于使用与测试。由于 Ubuntu18.04 自带 python2.7 以及 python3.6，通过不同版本安装的工具使用的 python 环境不同，故需指明使用 pip3 安装。

输入正确的地址，用户名以及验证，可以通过 Swift 查看 Openstack 情况，如图 6 所示：

```

luyun@ubuntu:~$ swift -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing stat
Account: AUTH_test
Containers: 1
Objects: 2
Bytes: 0
Containers in policy "policy-0": 1
Objects in policy "policy-0": 2
Bytes in policy "policy-0": 0
X-Timestamp: 1649580817.88162
X-Trans-Id: tx538808164ecf4c5ba6ae1-00625fc752
Content-Type: text/plain; charset=utf-8
Accept-Ranges: bytes

```

图 6 Openstack Swift 状态

(三)对象存储服务使用

在部署好 Openstack Swift 服务端和客户端之后，可以通过其提供基本命令来完成相应的操作。

list 命令可以展示当前存储的容器，在 list 后加上相应容器名称，可以进一步查看该容器中存储的文件，如图 7 所示：

```

luyun@ubuntu:~$ swift -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing list
usr_file
luyun@ubuntu:~$ swift -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing list usr_file
test1.txt
test2.txt

```

图 7 list 命令测试结果

upload 命令可以将本地文件上传到存储系统中，该指令需要指定上传的容器以及文件，若该容器不存在则会新建一个相应的容器，如图 8 所示：

```

luyun@ubuntu:~$ swift -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing upload test HelloWorld.py
HelloWorld.py
luyun@ubuntu:~$ swift -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing list
test
usr_file
luyun@ubuntu:~$ swift -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing list test
HelloWorld.py

```

图 8 upload 命令测试结果

delete 命令则可以将存储系统上的文件删除，该命令既可以删除指定容器下的特定文件，也可以直接删除某一个容器，这个操作会同时删除容器内的文件，如图所示：

```

luyun@ubuntu:~$ swift -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing delete usr_file test1.txt
test1.txt
luyun@ubuntu:~$ swift -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing list usr_file
test2.txt
luyun@ubuntu:~$ swift -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing delete usr_file
test2.txt
usr_file
luyun@ubuntu:~$ swift -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing list
test

```

图 9 delete 命令测试结果

(四)Swift-bench 测试

github 开源项目 Swift-bench 支持对 Openstack Swift 存储系统进行测试，可通过 pip2 install swift-bench 进行安装。由于该项目使用 python2.x 版本，故在安装时需要使用 pip2，随后在运行时，会使用本机 python2.7 环境。如果使用 pip3 安装，则在使用时会因为 python 环境不支持出现无法启动的错误。

执行 swift-bench -h 命令，出现如所示情况，说明安装完成。

```
luyun@ubuntu:~$ swift-bench -h
Usage: swift-bench [OPTIONS] [CONF_FILE]

Conf file with SAIO defaults:

[bench]
auth = http://localhost:8080/auth/v1.0
user = test:tester
key = testing
concurrency = 10
object_size = 1
num_objects = 1000
num_gets = 10000
delete = yes
auth_version = 1.0
policy_name = gold
```

图 10 swift-bench 安装

六、实验总结

(一) swift-bench 使用

使用 swift-bench 进行测试的时，在测试命令中可以进行自定义的参数配置。可以对对象的大小、并发量、写入次数以及读取次数进行配置，然后设置好的 Openstack Swift 上进行运行。

```
luyun@ubuntu:~$ swift-bench -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing -s 1024 -c 100 -n 100 -g 500
swift-bench 2022-04-20 17:12:45,135 INFO Auth version: 1.0
swift-bench 2022-04-20 17:12:45,760 INFO Auth version: 1.0
swift-bench 2022-04-20 17:12:48,919 INFO 100 PUTS **FINAL** [0 failures], 31.9/s
swift-bench 2022-04-20 17:12:48,920 INFO Auth version: 1.0
swift-bench 2022-04-20 17:12:50,971 INFO 177 GETS [0 failures], 88.3/s
swift-bench 2022-04-20 17:12:54,181 INFO 500 GETS **FINAL** [24 failures], 95.9/s
swift-bench 2022-04-20 17:12:54,181 INFO Auth version: 1.0
swift-bench 2022-04-20 17:12:55,800 INFO 100 DEL **FINAL** [7 failures], 63.1/s
swift-bench 2022-04-20 17:12:55,800 INFO Auth version: 1.0
```

图 11 swift-bench 测试结果

如图 11 所示，对部署好的 Openstack Swift 进行测试，设置文件大小为 1024B(1KB)，并发数为 100，写入次数为 100，读取次数为 500。通过其反馈的结果可以得出，平均写入速度为 31.9/s，平均读取速度为 95.9/s，删除速率为 36.1/s。

(二) 性能测试

通过改变对象大小或者并发数，固定读写次数，进行多次测试，统计数据如图 12 所示。

通过表格数据可以发现，数据对象的大小对数据读写速度有着较大的影响。在数据从 1KB 逐渐增加到 1MB 的过程中，系统的读写速度以及删除速度都随之增大。可见系统处理单个大文件要优于处理多个小文件。

同时，通过固定对象大小，逐渐改变并发量可以发现：在一开始，并发量从 1 增加到 25 的时候，读写速度和删除都随之提升；但是，在并发量从 25 增加到 100

的过程中，读写速度和删除速度一开始尚能维持相近的水平，随后缓慢出现下降。

可见，系统的性能并不可能随着并发数的增加而无限提升。在系统能够支持的范围内，增加并发数，能够提升系统的工作效率；但系统的性能和资源是有上限的，随着并发数越来越多，系统读写的速度不再能够提升，过多的并发会造成拥塞，排队处理时间增加，甚至会影响系统正常工作。

大小/KB	并发数	写入速率	写入速率KB/s	读取速率	读取速率KB/s	删除速率	删除速率KB/s
1	100	38.8	38.8	118	118.0	56.2	56.2
32	100	43.9	1404.8	90.4	2892.8	69.5	2224.0
64	100	38.6	2470.4	83.4	5337.6	61.3	3923.2
128	100	39.7	5081.6	90.9	11635.2	57.6	7372.8
256	100	36.6	9369.6	78.4	20070.4	72.2	18483.2
512	100	23.5	12032.0	48.8	24985.6	58.5	29952.0
1024	100	20.1	20582.4	34.7	35532.8	55.0	56320.0
1024	1	21.3	21811.2	29.2	29900.8	43.4	44441.6
1024	10	27.7	28364.8	36.3	37171.2	53.3	54579.2
1024	25	29.4	30105.6	38.9	39833.6	57.0	58368.0
1024	50	26.1	26726.4	36.0	36864.0	50.7	51916.8
1024	75	24.1	24678.4	36.7	37580.8	48.3	49459.2

图 12 测试数据统计

本次实验主要是搭建一个对象存储服务端，利用相应的客户端以及测试工具，对其进行评测。通过使用和测试加深了我对于对象存储的了解。同时，也让我了解到了影响系统性能的各个因素。在实验初期，由于选用了 Openstack Swift 作为服务端，所以在测试时并没有可用的脚本和工具。在查阅了 swiftclient 的资料后，我动手构造了一个简单的测试，但未能完成多线程并发请求，导致测试结果并不太理想。随后在 github 上发现有用于 Swift 的测试程序，故选择使用此程序作为测试，希望能够以后能够利用借助相关 api 完成自己的代码。

参考文献

- [1] ARNOLD J. OpenStack Swift[M]. O'Reilly Media, 2014
- [2] 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
- [3] Benchmarking tool for OpenStack Swift. Mirror of code maintained at opendev.org.
<https://github.com/openstack/swift-bench>