



2017 级

《物联网数据存储与管理》课程
实 验 报 告

姓 名 金修旭

学 号 U201714739

班 号 物联网 1701 班

日 期 2020.05.26

目 录

1	实验目的	1
2	实验背景	1
2.1	ACCOUNT SERVER.....	1
2.2	CONTAINER	1
2.3	OBJECT SERVER	2
3	实验环境	3
3.1	系统环境.....	3
3.2	MINIO-SERVER.....	3
3.3	MINIO-PYTHON API.....	5
3.4	OSM.....	7
3.5	S3BENCH	8
3.6	OPENSTACK SWIFT CLI	10
4	实验内容	11
4.1	对象存储技术实践.....	11
4.2	对象存储性能分析.....	11
5	实验过程	11
5.1	环境配置.....	11
6	实验总结	13
	参考文献.....	14

1 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。

2 实验背景

OpenStack Object Storage 是一个高度可用，分布式，最终一致的对象 blob 存储。可以使用 Object Storage API 创建，修改和获取对象和元数据，该 API 是作为一组 Representational State Transfer (REST) Web 服务实现的。

2.1 Account Server

表示层次结构的顶级

服务提供商会创建帐户，并拥有该帐户中的所有资源。该帐户定义容器的命名空间。

容器在两个不同的帐户中可能具有相同的名称。

在 OpenStack 环境中，帐户与项目或租户同义。

2.2 Container

定义对象的命名空间。两个不同容器中具有相同名称的对象表示两个不同的对象。

可以在帐户中创建任意数量的容器。

除了包含对象之外，还可以使用容器通过使用访问控制列表 (ACL) 来控制对对象的访问。不能使用单个对象存储 ACL。

在容器级别配置和控制许多其他功能，例如对象版本控制。

可以在一个请求中批量删除多达 10,000 个容器。

可以在具有云提供商的预定义名称和定义的容器上设置存储策略。

2.3 Object Server

存储数据内容，例如文档，图像等。还可以使用对象存储自定义元数据。

使用 Object Storage API，

存储无限数量的对象。每个对象可以大到 5 GB，这是默认值。可以配置最大对象大小。

- 1) 使用大对象创建上载和存储任何大小的对象。
- 2) 使用跨源资源共享来管理对象安全性。
- 3) 使用内容编码元数据压缩文件。
- 4) 使用内容处置元数据覆盖对象的浏览器行为。
- 5) 安排删除对象。
- 6) 在单个请求中批量删除最多 10,000 个对象。
- 7) 自动提取存档文件。
- 8) 生成一个 URL，该 URL 提供对对象的时间限制 GET 访问。
- 9) 使用表单 POST 中间件从浏览器直接将对象上传到 Object Storage 系统。
- 10) 创建指向其他对象的符号链接。

3 实验环境

3.1 系统环境

```
      000/
      `+0000:
      `+000000:
      -+000000+:
      `/:-:++0000+:
      `/++++/+++++++:
      `/+++++++:
      `/+++++000000000000/`
      /000SSSSO++0SSSSSO+`
      0SSSSSO-````/0SSSSSO+`
      SSSSSSO.      :SSSSSSSO.
      SSSSSS/      0SSSSO+++
      SSSSSS/      +SSSS000/-
      SSO+/:--      -:/+0SSSSO+-
      `            `.-/+oso:
                  `-/+/
                  `/
jyxk@magt
-----
OS: Arch Linux x86_64
Host: 20J6A00GHH ThinkPad T470p
Kernel: 5.6.14-arch1-1
Uptime: 35 mins
Packages: 1234 (pacman)
Shell: zsh 5.8
Resolution: 1920x1080
DE: Plasma
WM: KWin

Theme: Breeze [Plasma], Breeze [GTK2/3]
Icons: breeze [Plasma], breeze [GTK2/3]
Terminal: yakuake
CPU: Intel i7-7700HQ (8) @ 3.800GHz
GPU: Intel HD Graphics 630
GPU: NVIDIA GeForce 940MX
Memory: 4448MiB / 15759MiB
```

3.2 Minio-Server

```
MINIO_ACCESS_KEY
MINIO_ACCESS_KEY
Attempting encryption of all config, IAM users and policies on MinIO backend
Endpoint: http://192.168.1.106:9000 http://192.168.0.2:9000 http://127.0.0.1:9000
AccessKey: hust
SecretKey: hust_obs

Browser Access:
http://192.168.1.106:9000 http://192.168.0.2:9000 http://127.0.0.1:9000

Command-line Access: https://docs.min.io/docs/minio-client-quickstart-guide
$ mc config host add myminio http://192.168.1.106:9000 hust hust_obs

Object API (Amazon S3 compatible):
Go: https://docs.min.io/docs/golang-client-quickstart-guide
Java: https://docs.min.io/docs/java-client-quickstart-guide
Python: https://docs.min.io/docs/python-client-quickstart-guide
JavaScript: https://docs.min.io/docs/javascript-client-quickstart-guide
.NET: https://docs.min.io/docs/dotnet-client-quickstart-guide
```

图 3.1 Minio-Server 启动验证

此时在浏览器打开 127.0.0.1:9000(endpoint), 在打开的 minio browser, 输入自己的 AccessKey 和 SerectKey 登陆。

华中科技大学课程实验报告

此时可以选择右下角红色标记随意添加 bucket 和上传文件，方便地实现类似网盘的功能。

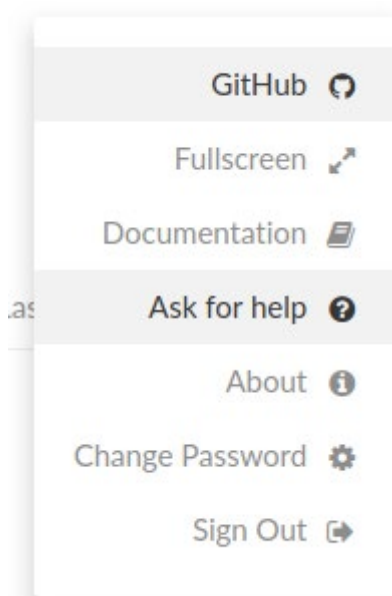


图 3.2 账户管理相关选项

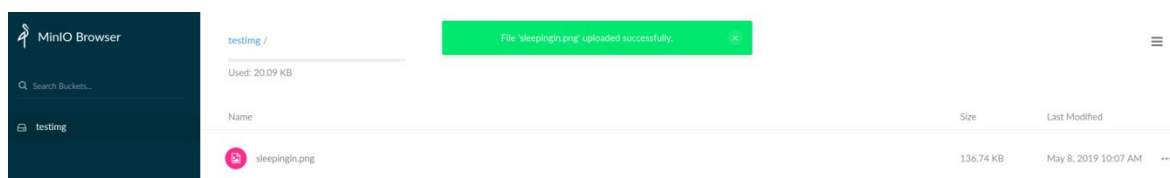


图 3.3 Bucket/file 测试图

注意，bucket 有相应的命名限制，只能采用小写字母，而不允许下划线和相应大写字母的出现。

华中科技大学课程实验报告

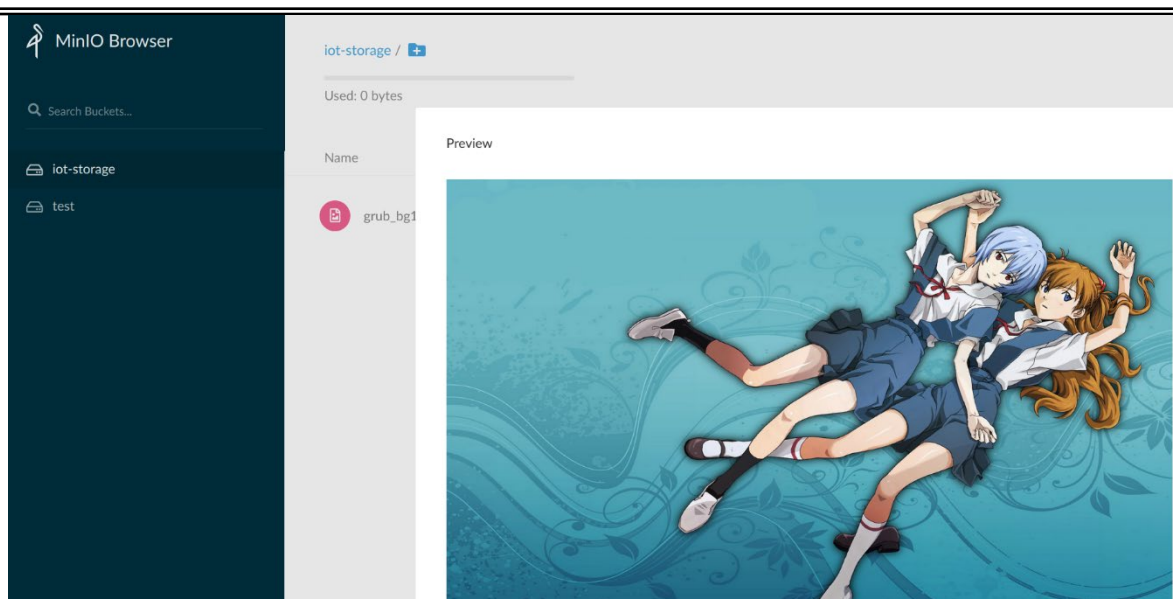


图 3.4 本地文件验证

3.3 Minio-python API

使用 python-tutorial 进行 API 编程测试，使用官方测试仓库，代码如下

代码 3.1 Minio-python API

```
from minio import Minio
from minio.error import ResponseError
from minio.error import (ResponseError, BucketAlreadyOwnedByYou,
                          BucketAlreadyExists)

minioClient = Minio('play.min.io:9000',
                    access_key='Q3AM3UQ867SPQQA43P2F',
                    secret_key='zuf+tfteSlswRu7BJ86wekitnifILbZam1KYY3TG',
                    secure=True)

# Make a bucket with the make_bucket API call.
try:
```

华中科技大学课程实验报告

```
minioClient.make_bucket("maylogs", location="us-east-1")
except BucketAlreadyOwnedByYou as err:
    pass
except BucketAlreadyExists as err:
    pass
except ResponseError as err:
    raise
else:
    # Put an object 'pumaserver_debug.log' with contents from 'pumaserver_debug.log'.
    try:
        minioClient.fput_object('maylogs', 'pumaserver_debug.log',
        '/tmp/pumaserver_debug.log')
    except ResponseError as err:
        print(err)
```

运行结果如下

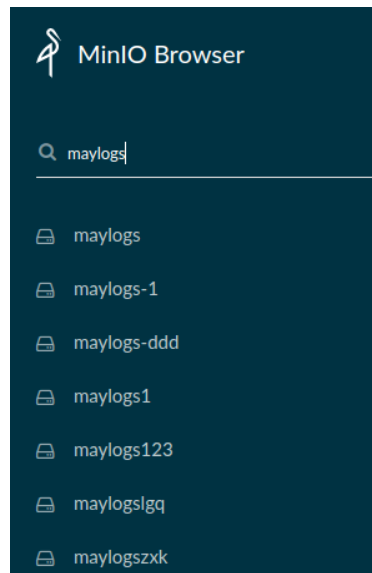


图 3.5 Minio-python 测试结果

3.4 osm

```
# jyxxk @ Magi in ~/Code/IoTStorage [20:30:22]
$ source config-osm.sh
/home/jyxxk/Code/IoTStorage
contexts:
- config:
  access_key_id: hust
  auth_type: accesskey
  disable_ssl: "false"
  endpoint: http://127.0.0.1:9000
  secret_key: hust_obs
  name: osm-minio
  provider: s3
current-context: osm-minio
```

图 3.6 配置 osm

```
# jyxxk @ Magi in ~/Code/IoTStorage [20:31:16]
$ osm lc
iot-storage
test
Found 2 containers in
```

图 3.7 此存储下的 bucket

```
# jyxxk @ Magi in ~/Code/IoTStorage [20:31:29]
$ osm mc osm-test
Successfully created container osm-test

# jyxxk @ Magi in ~/Code/IoTStorage [21:11:43]
$ osm lc
iot-storage
osm-test
test
Found 3 containers in
```

图 3.8 创建 bucket

```
# jyxk @ Magi in ~/Code/IoTStorage [21:18:12] C:
$ osm push -c osm-test grub_bg1.png grub_bg1.png
Successfully pushed item grub_bg1.png
```

图 3.9 上载 IMG 测试

3.5 s3bench

使用脚本进行测试，其中参数可自行调节

```
# jyxk @ Magi in ~/Code/IoTStorage [21:26:31]
```

```
$ ./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 (577.435μs)

Running Write test...

Running Read test...

华中科技大学课程实验报告

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: 0.000 MB

Total Throughput: 0.00 MB/s

Total Duration: 0.092 s

Number of Errors: 256

Results Summary for Read Operation(s)

Total Transferred: 0.000 MB

Total Throughput: 0.00 MB/s

Total Duration: 0.047 s

Number of Errors: 256

Cleaning up 256 objects...

Deleting a batch of 256 objects in range {0, 255}... Succeeded

Successfully deleted 256/256 objects in 46.355513ms

华中科技大学课程实验报告

根据原始数据分析，读写最高速度均出现在客户端数量为 32，样本数为 1024，大小为 512*32 时的情况。或许在我的机器上，针对较小的文件大小和较大的文件数，会发挥较好的表现。

3.6 OpenStack Swift CLI

这里利用老师提供的 open stack swift docker 的 dockerfile 进行安装，按照 readme 说明运行以下命令。

Build docker image

```
docker build -t openstack-swift-docker .
```

Prepare datavolume

```
docker run -v /srv --name SWIFT_DATA busybox
```

Run container

```
docker run -d --name openstack-swift -p 12345:8080 --volumes-from SWIFT_DATA -t openstack-swift-docker
```

Check container

```
docker logs openstack-swift
```

```
docker ps
```

Verify functionality

```
swift -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing stat
swift -A http://127.0.0.1:12345/auth/v1.0 -U test:tester -K testing list
```

然后使用类似上述命令，测试 Swift Docker 启动

```
Account: AUTH_test
Containers: 0
Objects: 0
Bytes: 0
Content-Type: text/plain; charset=utf-8
X-Timestamp: 1557886895.94559
X-Put-Timestamp: 1557886895.94559
X-Trans-Id: txa0e0c68a8404424e812fe-005cdb77af
```

4 实验内容

4.1 对象存储技术实践

- 1) 采用 Docker 进行环境相关配置
- 2) 相关 API 测试以及测试程序编写
- 3) 测试数据下载分析

4.2 对象存储性能分析

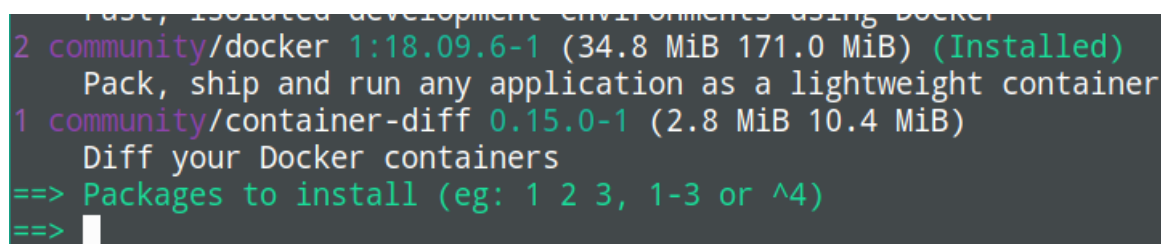
- 1) 对于不同情况下的测试结果进行绘图
- 2) 测试结果分析

5 实验过程

5.1 环境配置

- 1) 安装软件包

由于基于 Arch 系统优良的包管理和 AUR，所需要的包和依赖可以一键安装。



```
Fast, isolated development environments using Docker
2 community/docker 1:18.09.6-1 (34.8 MiB 171.0 MiB) (Installed)
   Pack, ship and run any application as a lightweight container
1 community/container-diff 0.15.0-1 (2.8 MiB 10.4 MiB)
   Diff your Docker containers
==> Packages to install (eg: 1 2 3, 1-3 or ^4)
==> 
```

图 5.1 yay 安装相关包

- 2) 使用 s3bench 脚本测试当前环境。

脚本：

```
s3bench=~/.go/bin/s3bench
```

```
if [ -n "$GOPATH" ]; then
s3bench=$GOPATH/bin/s3bench
```

华中科技大学课程实验报告

fi

```
# -accessKey Access Key
# -accessSecret Secret Key
# -bucket=loadgen Bucket for holding all test objects.
# -endpoint=http://127.0.0.1:9000 Endpoint URL of object storage service
being tested.
# -numClients=8 Simulate 8 clients running concurrently.
# -numSamples=256 Test with 256 objects.
# -objectNamePrefix=loadgen Name prefix of test objects.
# -objectSize=1024 Size of test objects.
for i in {1..15..2}
do
for j in {16..512..16}
do
$s3bench \
-accessKey=hust \
-accessSecret=hust_obs \
-bucket=loadgen \
-endpoint=http://127.0.0.1:9000 \
-numClients=$i \
-numSamples=$j \
-objectNamePrefix=loadgen \
-objectSize=$(( 1024*32 ))
done
done
```

```
# build your own test script with designated '-numClients', '-numSamples'
and '-objectSize'
# 1. Use loop structure to generate test batch (E.g.: to re-evaluate multiple
s3 servers under the same configuration, or to gather data from a range of
parameters);
# 2. Use redirection (the '>' operator) for storing program output to text
```

华中科技大学课程实验报告

files;

3. Observe and analyse the underlying relation between configuration parameters and performance metrics.

结果如下:

```
1 Test parameters
2 endpoint(s):    [http://127.0.0.1:9000]
3 bucket:        loadgen
4 objectNamePrefix: loadgen
5 objectSize:     0.0312 MB
6 numClients:     8
7 numSamples:     256
8 verbose:        %!d(bool=false)
9
10
11 Generating in-memory sample data ... Done (607.186µs)
12
13 Running Write test ...
14
15 Running Read test ...
16
17 Test parameters
18 endpoint(s):    [http://127.0.0.1:9000]
19 bucket:        loadgen
```

具体数据见 test.log。

3.7

6 实验总结

在实验过程中, 环境配置可以说相对来说比较简单, 但是某些玄学 BUG 可能无明显报错信息, 转而转向 Docker, 由于所有的包括服务端和测试端均在 Docker 中

华中科技大学课程实验报告

启动，存在部分 Automated Docker 入口写死，需要自己手动更改 Docker 文件进行相应的编译，在此过程中提高了 Docker 的熟练程度。

使用 minio 与 osm 搭配让我体会了基本的对象存储系统的工作模式，体会到了对象存储系统与其他类别存储系统的不同，感受到了在部署以及使用两方面，对象存储系统的优势。

最后，感谢实验过程中提供帮助的老师 and 同学以及所参阅资料的提供者。

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.
- [4] Swift Document.URL:<https://docs.openstack.org/swift>