《数据中心技术》课程

实 验 报 告

姓　名　王道宇

学　号　Y202102005

院　系　研究生院

日　期　2022.01.07

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

随着互联网的不断扩张和云计算技术的进一步推广，海量的数据在个人、企 业、研究机构等源源不断地产生，如何有效、快速、可靠地存取这些日益增长的 海量数据成了关键的问题。

传统的存储解决方案面临越来越多的困难，比如数据量的指数级增长对不断扩容的存储空间提出要求、实时分析海量的数据对存储计算能力提出要求等。因此，需要能处理海量数据、高性能、易扩展、可伸缩、高可用的新型存储方案。对象存储系统（Object-Based Storage System）是综合了 NAS 和 SAN 的优点， 同时具有 SAN 的高速直接访问和 NAS 的数据共享等优势，提供了高可靠性、跨平台性以及安全的数据共享的存储体系结构。

三、实验环境

准备实验的基础环境，选择自己熟悉的操作系统平台和语言环境。这里我选择的实验环境如下：

- Arch Linux amd64
- Python 3.9.7
- Go 1.17.3

四、实验内容

1. 对象存储技术实践
   搭建实验基本环境
   搭建对象存储服务器端
   搭建对象存储客户端

2. 对象存储性能分析
   搭建性能评测工具
   测试并观察记录对象尺寸对延迟的影响
   测试并观察记录样本数对延迟的影响
   测试并观察记录并发数对延迟的影响

3. 尾延迟处理
   尝试应对对冲请求、关联请求

五、实验过程

实验一：系统搭建

实验环境

准备实验的基础环境，选择自己熟悉的操作系统平台和语言环境。这里我选择的实验环境如下：

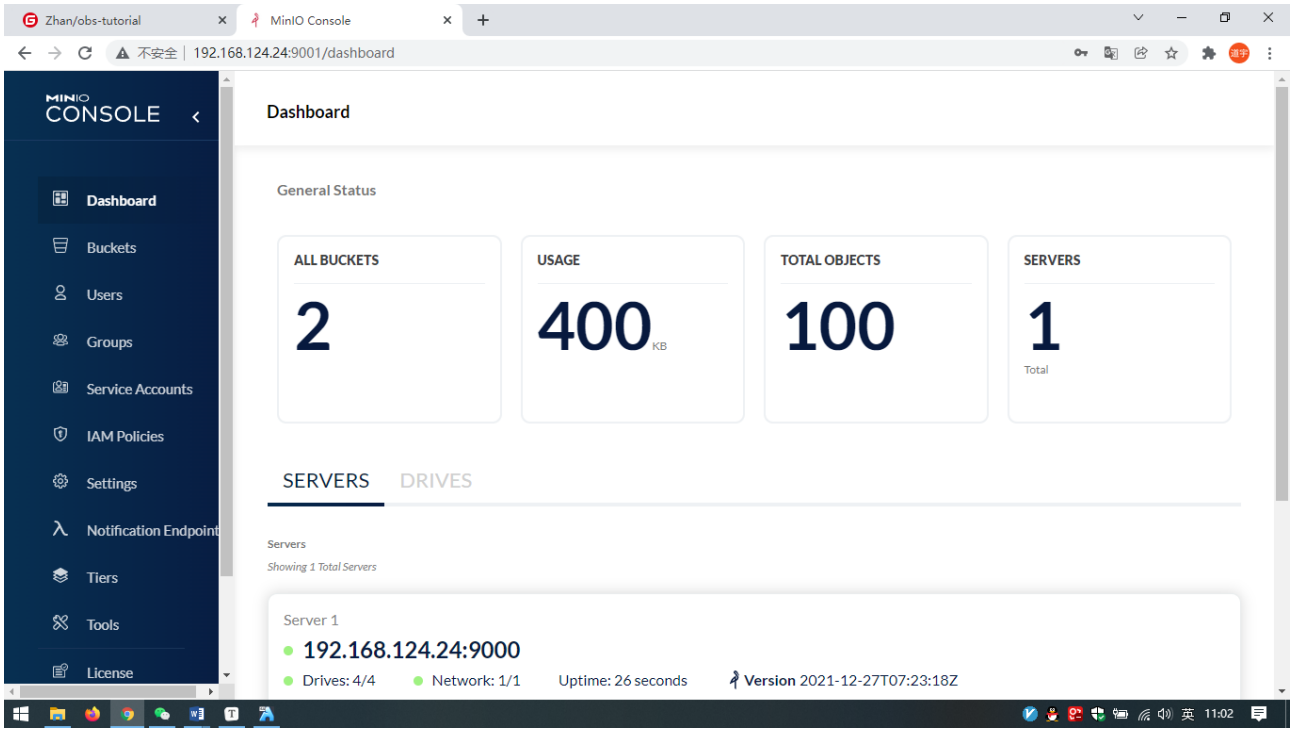
- Arch Linux amd64
- Python 3.9.7
- Go 1.17.3

服务端

首先从 minio 官网选择 Linux 平台下最新的 minio 二进制文件到项目的根目录下，然后自定义用户名，密码，存储目录，console 端口号。在终端运行如下命令：

```
MINIO_ROOT_USER=$(USERNAME) MINIO_ROOT_PASSWORD=$(PASSWORD) \
./minio server $(DATADIR) --console-address ":$(PORT)"
```

在浏览器中输入 http://<IP>:<PORT>可以访问 minio console，可以通过 web 页面对 minio 进行简单的管理。



客户端

官方的 mc 客户端我用着不习惯，这里选择了 s3cmd，一个简单的命令行客户端。安装并配置 s3cmd，按照提示输入相应的信息。

```
pip install s3cmd && s3cmd --configure
```

s3cmd 的配置信息会保存在`$HOME/.s3cfg`中，这里我们主要修改了下面几个配置：

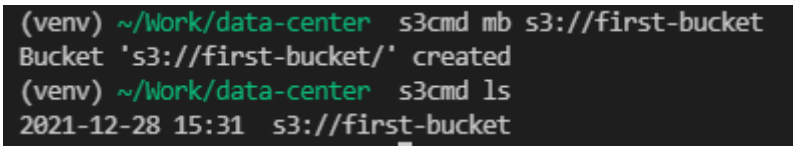
```
access_key = <USERNAME>
secret_key = <PASSWORD>
host_base = <IP>:<PORT>
host_bucket =
use_https = False
```

使用 s3cmd -h 可以查看该命令的一些用法。比如在根目录下创建一个叫 first-bucket 的桶，可以使用：

```
s3cmd mb s3://first-bucket
```

查看已有桶可以使用：

```
s3cmd ls
```



实验二：性能观测

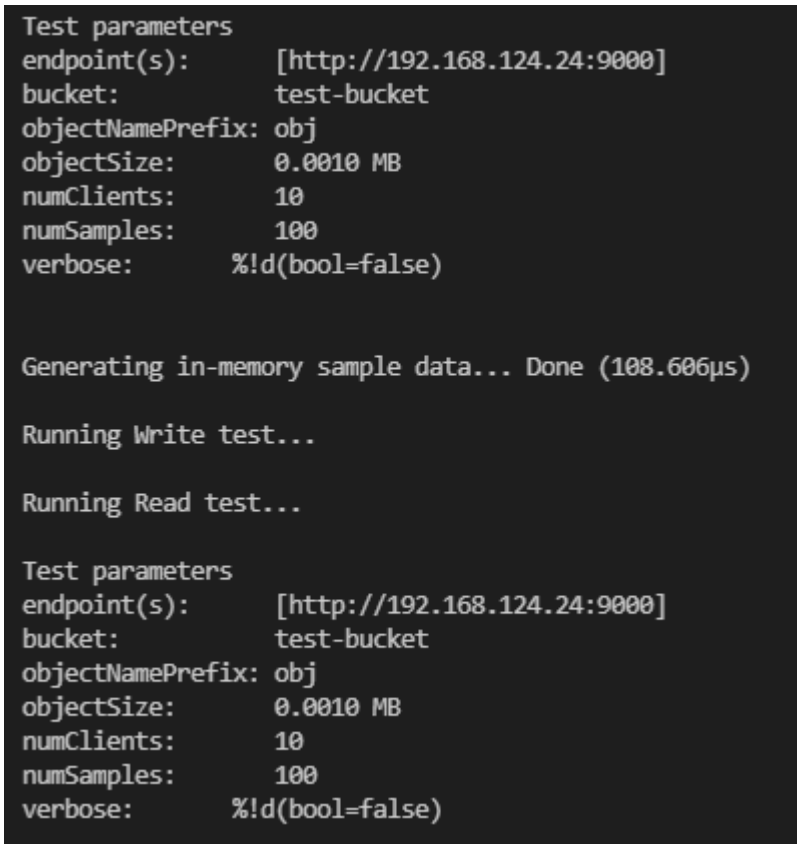
使用 s3bench 进行性能观测。首先准备好 go 语言环境，注意把 go/bin 和 gopath/bin 加入环境变量。
s3bench 安装

```
go get -u github.com/igneous-systems/s3bench
```

运行测试

```
s3bench -accessKey=$(USERNAME) -accessSecret=$(PASSWORD) \
  -endpoint=http://192.168.124.24:9000 -bucket=$(BUCKET) \
  -objectNamePrefix=obj -numClients=10 -numSamples=100 -objectSize=1024
```

测试结果如下：



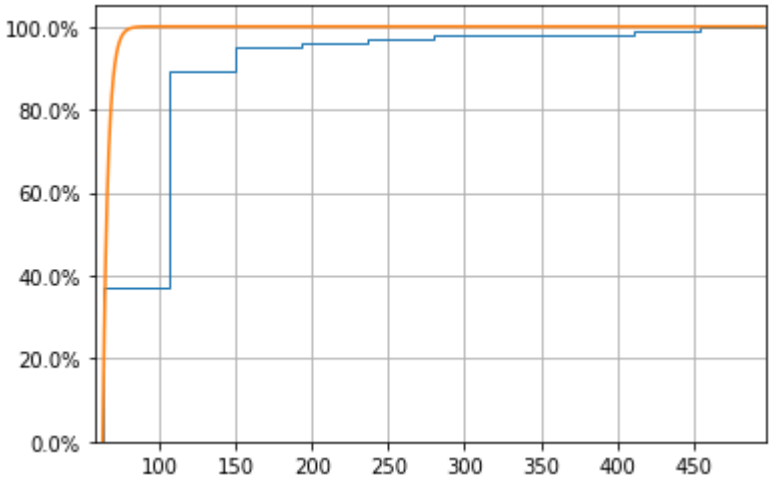
```
Results Summary for Write Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  0.02 MB/s
Total Duration:    4.989 s
Number of Errors:  0
------------------------------------
Write times Max:       1.294 s
Write times 99th %ile: 1.294 s
Write times 90th %ile: 1.254 s
Write times 75th %ile: 1.014 s
Write times 50th %ile: 0.196 s
Write times 25th %ile: 0.129 s
Write times Min:       0.077 s


Results Summary for Read Operation(s)
Total Transferred: 0.098 MB
Total Throughput:  0.22 MB/s
Total Duration:    0.443 s
Number of Errors:  0
------------------------------------
Read times Max:       0.112 s
Read times 99th %ile: 0.112 s
Read times 90th %ile: 0.083 s
Read times 75th %ile: 0.067 s
Read times 50th %ile: 0.043 s
Read times 25th %ile: 0.016 s
Read times Min:       0.004 s


Cleaning up 100 objects...
Deleting a batch of 100 objects in range {0, 99}... Succeeded
Successfully deleted 100/100 objects in 463.160369ms
```

实验三：尾延迟挑战

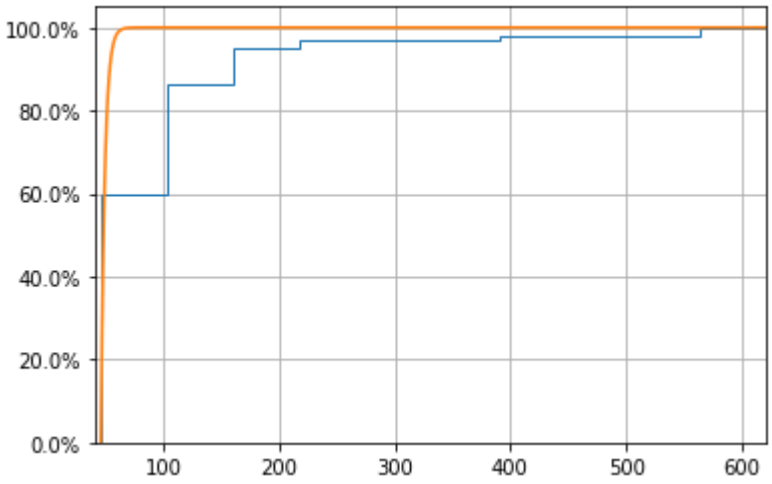
使用 boto3 进行观察测试，首先安装 boto3：

```
pip install boto3
```

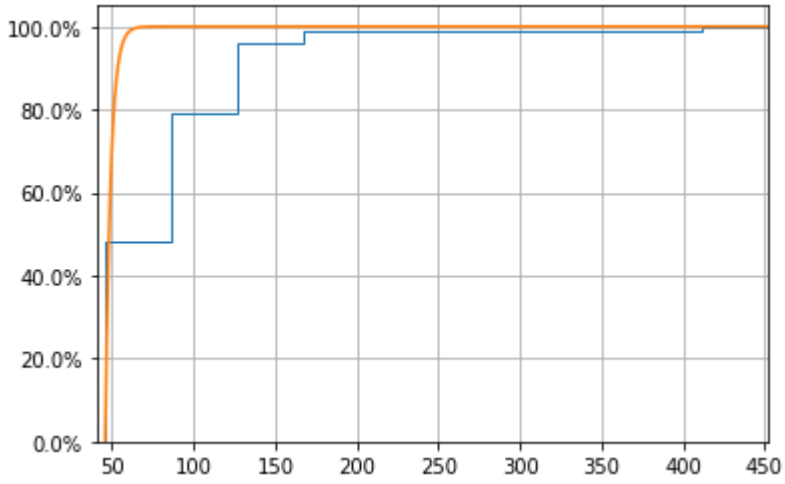
使用方法可以看官网文档，可以看老师仓库里是怎么用的。整体流程如下：先生成一定大小的文件，然后以不同的到达率上传该文件，收集这些请求的延迟，通过排队论进行拟合数据。测试结果如下：



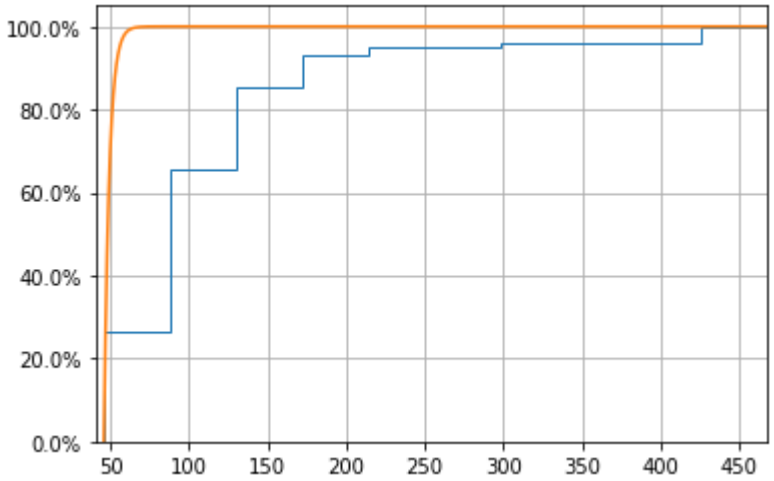
到达率：2 requests / 100 ms，平均延迟 123ms，最大延迟 497ms



到达率：4 requests / 100 ms，平均延迟 111ms，最大延迟 620ms



到达率：8 requests / 100 ms，平均延迟 100ms，最大延迟 451ms



到达率：无限制，平均延迟 128ms，最大延迟 467ms

六、实验总结

这个实验使我对对象存储有了基本的了解和认识。环境配置部分让我重新认识了 Linux，熟悉了一些常见工具的使用。后面的性能观测和尾延迟挑战尽管做的不是很好，但让我对对象存储有了更深的理解，明白了一些在存储领域常见的基本概念。虽然我不是做存储方向的研究，但多了解一些东西也是极好的。