

**2017** 级

《物联网数据存储与管理》课程

**实 验 报 告**

**姓 名 李振明**

**学 号 U201714772**

**班 号 物联网1701班**

**日 期 2020.06.04**

**目 录**

[一、实验目的 1](#_Toc509412095)

[二、实验背景 1](#_Toc509412096)

[三、实验环境 1](#_Toc509412097)

[四、实验内容 1](#_Toc509412098)

[4.1 对象存储技术实践 1](#_Toc509412099)

[4.2 对象存储性能分析 1](#_Toc509412100)

[五、实验过程 2](#_Toc509412101)

[六、实验总结 8](#_Toc509412102)

# 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

物联网应用随着移动互联网的兴起而蓬勃发展，快速增长的物联网数据带来了存储挑战：以几何级数不断增长的数据量，对存储和处理带来挑战、数据内容多样性带来的数据异构，使得数据结构越来越复杂。因此，需要一个高可用、可扩展的海量数据存储系统来满足物联网存储需求。

本次实验为对象存储入门实验，其中主要的部分有：基础环境搭建；对象存储服务器端准备；对象存储客户端准备；对象存储测评工具的使用

实验难度的选择：

服务端：mock-s3

客户端：boto3 API python程序与aws-shell

系统测试：s3bench

# 三、实验环境

CPU：Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70GHz

内存：12.0 GB

操作系统：ubuntu 16.04 LTS

对象存储服务端：mock-s3

对象存储客户端：aws-shell和boto3

对象存储评测工具：S3 bench

python：Python 3.7.6

Go：go1.14.2 linux/amd64

# 四、实验内容

根据实验给出的教程，选择对象存储服务端、对象存储客户端、对象存储评测工具。

## 4.1 对象存储技术实践

1. 搭建对象存储服务端，选择使用mock-s3。

2. 搭建对象存储客户端，选择使用aws-shell并且使用boto3API进行编程使用。

3. 测试对象存储基本功能是否正常。

## 4.2 对象存储性能分析

1. 搭建对象评测工具环境，选择S3 Bench，作为Go的一个模块进行安装。

2. 调整对象存储评测参数，包括客户端数量、对象数量、对象大小，观察数据存储性能。

3. 编写shell脚本实现批量测试，将终端输出结果重定向到文本文件，以便进一步分析。

4. 整理和分析不同参数下存储性能数据。

# 五、实验过程

## 5.1 搭建对象存储服务端

mock-s3服务端是基于Python的，搭建好Python环境后，在mock-s3的程序包内运行main.py即可启动服务。



图 5-1 运行mock-s3服务端

## 5.2 搭建对象存储客户端

1.安装aws，使用pip3安装，为提高速度，使用清华的源进行安装，命令为

pip3 install -I <https://pypi.tuna.tsinghua.edu.cn/simple> aws-shell

安装成功后输入aws-shell运行，如下图所示



图5-2 检查aws安装情况

2.使用boto3API编程实现，下图为将文件添加到bucket中

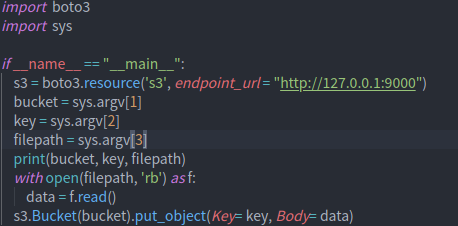


图5-3 boto3使用

## 5.3 测试对象存储功能

1. 运行mock-s3服务端



图5-4 mock-s3服务端

2. 利用aws上传文件



图5-5 aws上传文件

3.查看bucket中上传的文件



图5-6 上传文件成功

4.使用boto3编程实现上传文件



图5-7 boto3上传文件

5. 使用boto3编程查看上传的文件

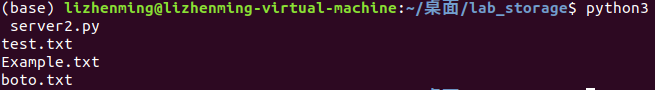


图5-8 boto3查看桶

3. 运行mock-s3服务端，在网页访问127.0.0.1:9000.

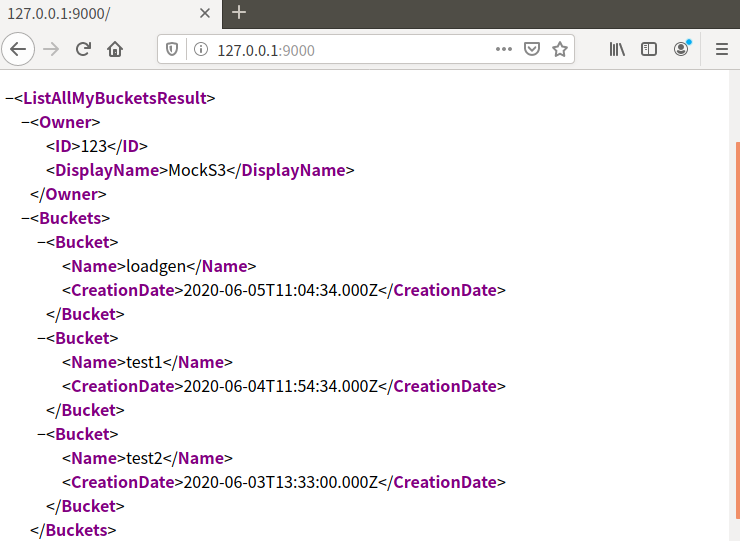


图 5-8 mock-s3服务端

## 5.4 对象存储性能分析

1. S3 Bench作为GO模块加载，下载aws-sdk-go和go-jmespath依赖包进行安装，再安装s3bench，安装完成在GOPATH的bin目录下会出现s3bench可执行文件。

2. S3 Bench没有批量测试和输出数据为文件的功能，编写shell脚本进行批量测试，将bash的输出重定向到文件，测试脚本如下：

①对象存储大小从1KB增加到1MB，并发客户端数量为1，对象数量为100，观察对象大小对存储性能的影响。

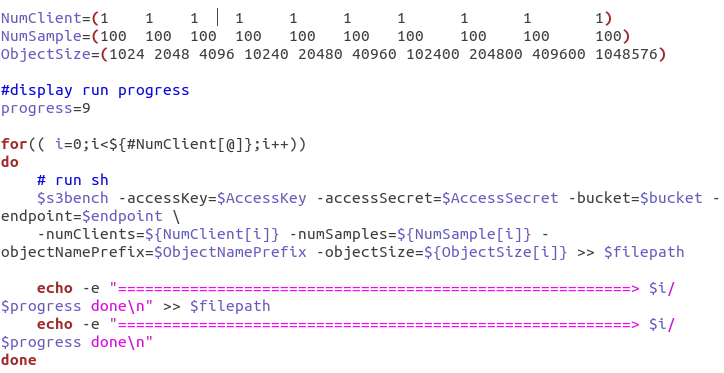


图5-9 s3bench1.sh

②并发客户端从1增加到100，对象数量为100，对象大小为1KB，观察并发客户端数量对存储性能的影响。

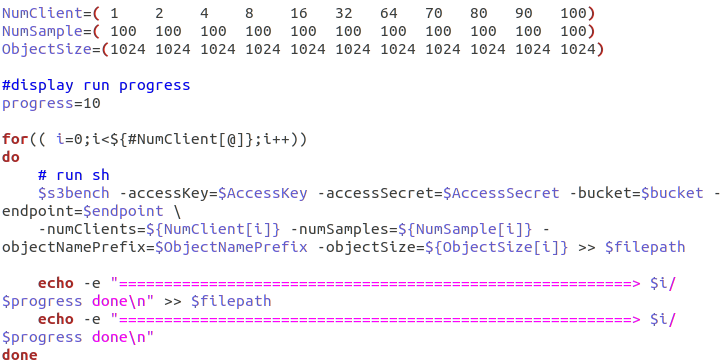


图5-10 s3bench2.sh

③并发客户端从1增加到100，对象数量为100，对象大小为100KB，与测试2内容做对比。

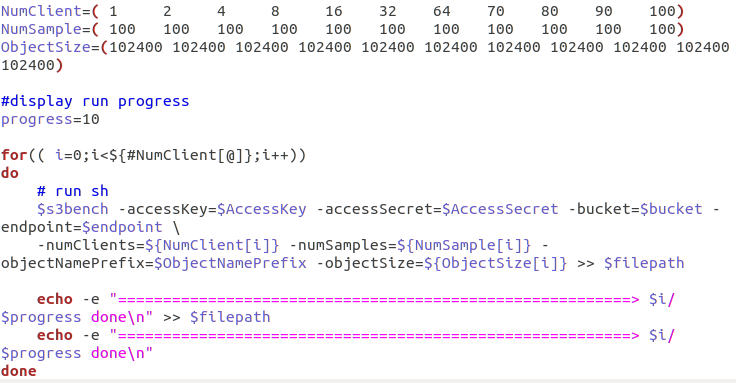


图5-11 s3bench3.sh

④并发客户端数量为1，对象数量从5增加到1280，对象大小为1KB，观察对象数量对存储性能的影响。

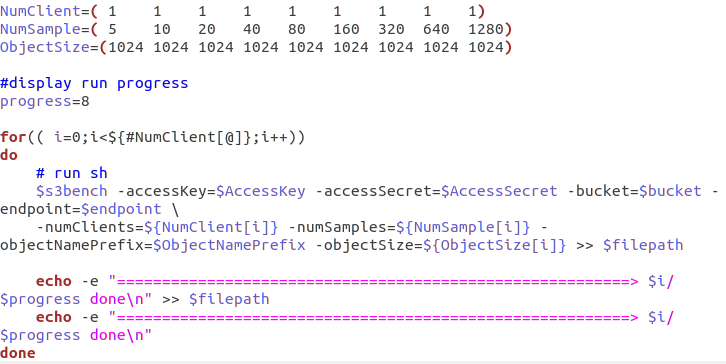


图5-12 s3bench4.sh

3. 执行测试脚本，输出重定向文件，文件内容如图所示。

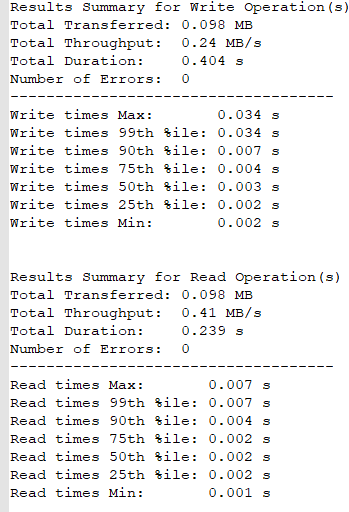


图5-13 重定向文件内容

4. 数据整理



图5-14 mock-s3性能评测数据

5. 分析结果

（1）对象大小对性能的影响

随对象大小的增大，吞吐率在增大。作出百分位延迟散点图。观察图像可知，读取延迟无较大变化，写入延迟随对象大小增大有较为明显的增大。

根据以上分析，随对象大小增大，吞吐率增大，写入延迟在对象较小和较大时较大，读取延迟无较大变化。

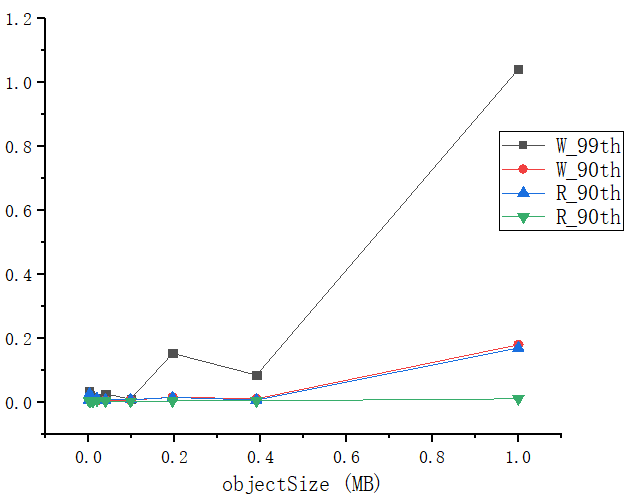


图5-15 改变对象大小

（2）并发客户端数对性能的影响

编号2、3的测试用例中当并发客户端大于32之后，写入发生错误，但是读取并未出错，说明在32个并发客户端后mock-s3的并发数量达到上限，继续增加会发生丢失数据的严重问题。下图为读取延迟。

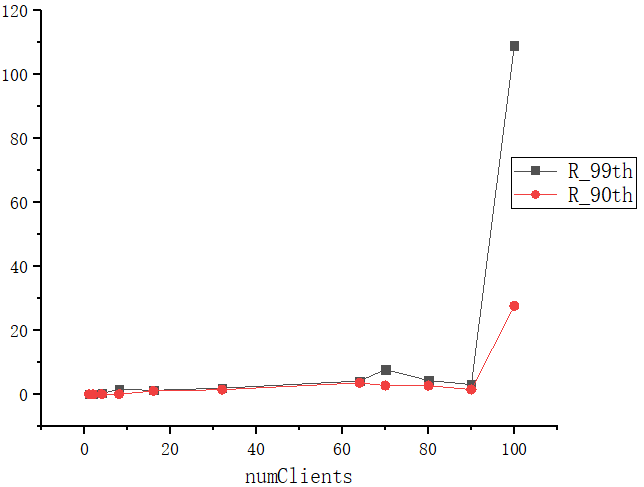


图5-16 改变并发客户端，大小为0.0977

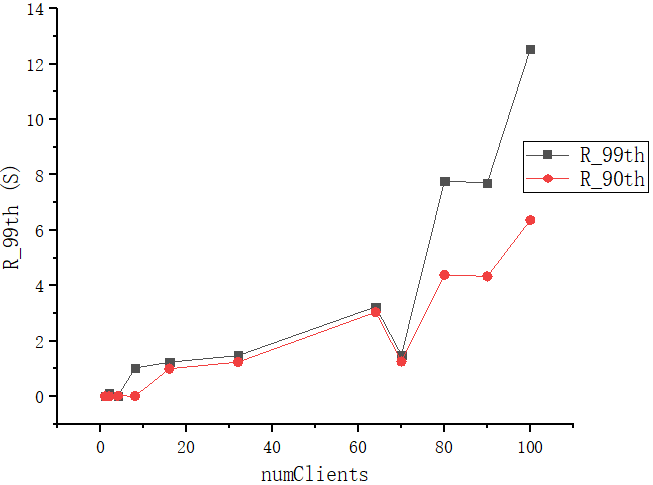


图5-17 改变并发客户端，大小为0.001

（3）对象数量对性能影响

观察数据可知，在并发客户端数量为1，对象数量从5增加到1280，对象大小为1KB的条件下，mock-s3的吞吐率变化不大，延迟先增大后减小，然后趋于不变，因为此时只有一个客户端，所以数据是串行到达，对象数量只是影响处理时间，对吞吐率和延迟的作用不大。

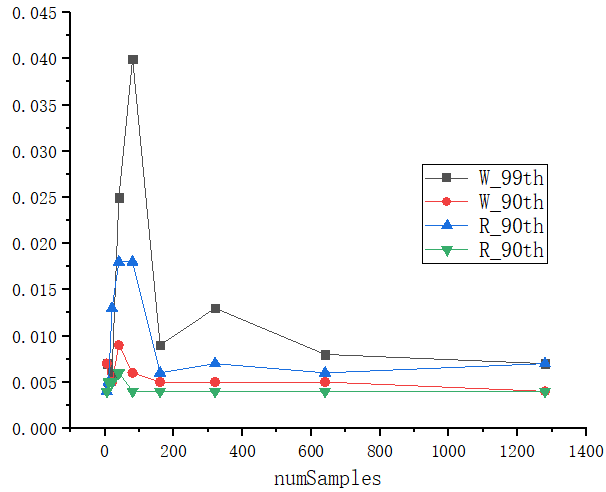


图5-18 改变对象数量

# 六、实验总结

本次实验使我对对象存储有了大致的了解，了解了对象存储服务端、客户端的形式和作用。本次实验虽然不算难，但是由于对此不了解和环境的搭建不太顺利，因此花费了较长的时间。

本次实验在选择客户端时，先使用aws-shell了解了一下，然后就使用代码进行存放对象等操作，并使用aws检查是否正确运行。此次实验虽然不算难，但是学到了较多的东西，对Go语言有了初步的了解。

在配置评测工具，安装S3 bench时，不知是什么原因，运行go get语句之后，界面一直没有反应，因此自己从网上下载所需的依赖和S3 bench的文件到本地，然后使用go install进行安装。由于S3 Bench没有批量测试和输出结果文件功能，所以利用shell脚本实现了批量测试和终端输出的重定向。

实验过程中虽有挫折，但是老师和同学对我的疑问进行了解答，同时也参考了往届学长的实验，收获较大。