

**2015** 级

《物联网数据存储与管理》课程

**实 验 报 告**

**姓 名 陈鑫**

**学 号 U201514873**

**班 号 物联网1501班**

**日 期 2018.04.23**

**目 录**

[一、实验目的 1](#_Toc512239877)

[二、实验背景 1](#_Toc512239878)

[三、实验环境 1](#_Toc512239879)

[四、实验内容 2](#_Toc512239880)

[五、实验过程 2](#_Toc512239881)

[1. 基础实验 2](#_Toc512239882)

[1.1 搭建Minio服务器 2](#_Toc512239883)

[1.2 验证服务器 3](#_Toc512239884)

[1.3 使用minio client客户端 3](#_Toc512239885)

[1.4 COSBench测试 5](#_Toc512239886)

[1.5 COSBench分析 6](#_Toc512239887)

[2. 中等难度实验 9](#_Toc512239888)

[2.1搭建mock-s3服务器 9](#_Toc512239889)

[2.2验证服务器 9](#_Toc512239890)

[2.3使用s3cmd客户端 10](#_Toc512239891)

[2.4 COSBench测试 10](#_Toc512239892)

[2.5 Topic选题研究 11](#_Toc512239893)

[六、实验总结 12](#_Toc512239894)

[参考文献 12](#_Toc512239895)

# 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

物联网数据存储挑战：

1. 庞大：随着物联网的发展，联网设备数量以几何级数增长，所需要处理和存储的数据量也越来越庞大。
2. 复杂：数据内容结构越来越丰富，越来越复杂。

# 三、实验环境

***1.硬件***

实验所使用的硬件如下图3.1所示.



图3.1 硬件描述

***2.软件***

实验分为基础，中等 两个部分，以下表3.2来描述实验系统软件构成和组件关系。

|  |  |  |
| --- | --- | --- |
|  | 基础 | 中等 |
| Test | Sample | Sample |
| Client | Minio Client | S3cmd |
| Server | Minio | Mock-s3 |

表3.2 软件组件部分

# 四、实验内容

依照实验环境中给出的表3.2，针对不同难度的实验，搭建好服务器，开启客户端，使用测试程序进行测试，并且相应地对结果进行性能分析和课题研究。

# 五、实验过程

## 1. 基础实验

### 1.1 搭建Minio服务器

下载、安装好Minio后，直接用命令行打开服务器，如下图5.1所示。

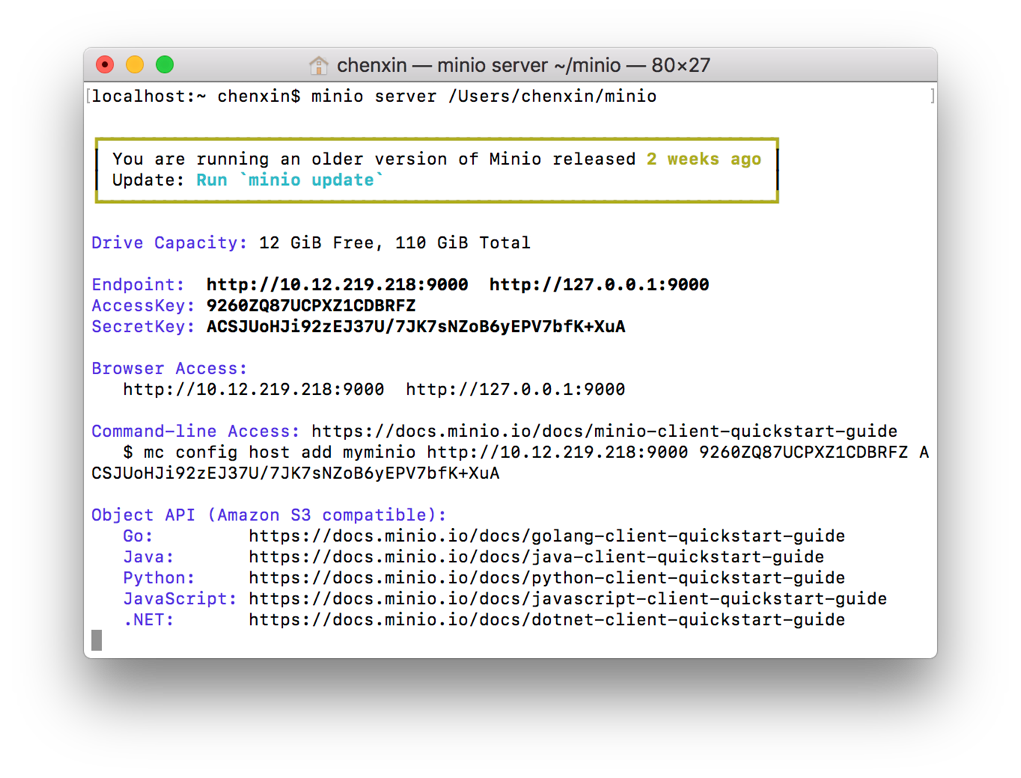


图5.1 搭建minio服务器

分析上述命令行结果：

Drive Capacity:即存储容量，显示有12g空闲，总容量为110g

Endpoint：即端点，指访问的位置，提供了一个本地访问位置和远程访问位置。

AccessKey: 即存储关键字，指的是登陆用户名

SecretKey: 即密码

Browser Access: 浏览器访问位置

Command-line Access: 命令行访问位置

Object API (Amazon S3 compatible): 和amazon s3兼容的对象api接口

### 1.2 验证服务器

为了验证服务器是否搭建成功，我们可以用浏览器来验证，如下图5.2，直接访问127.0.0.1:9000.

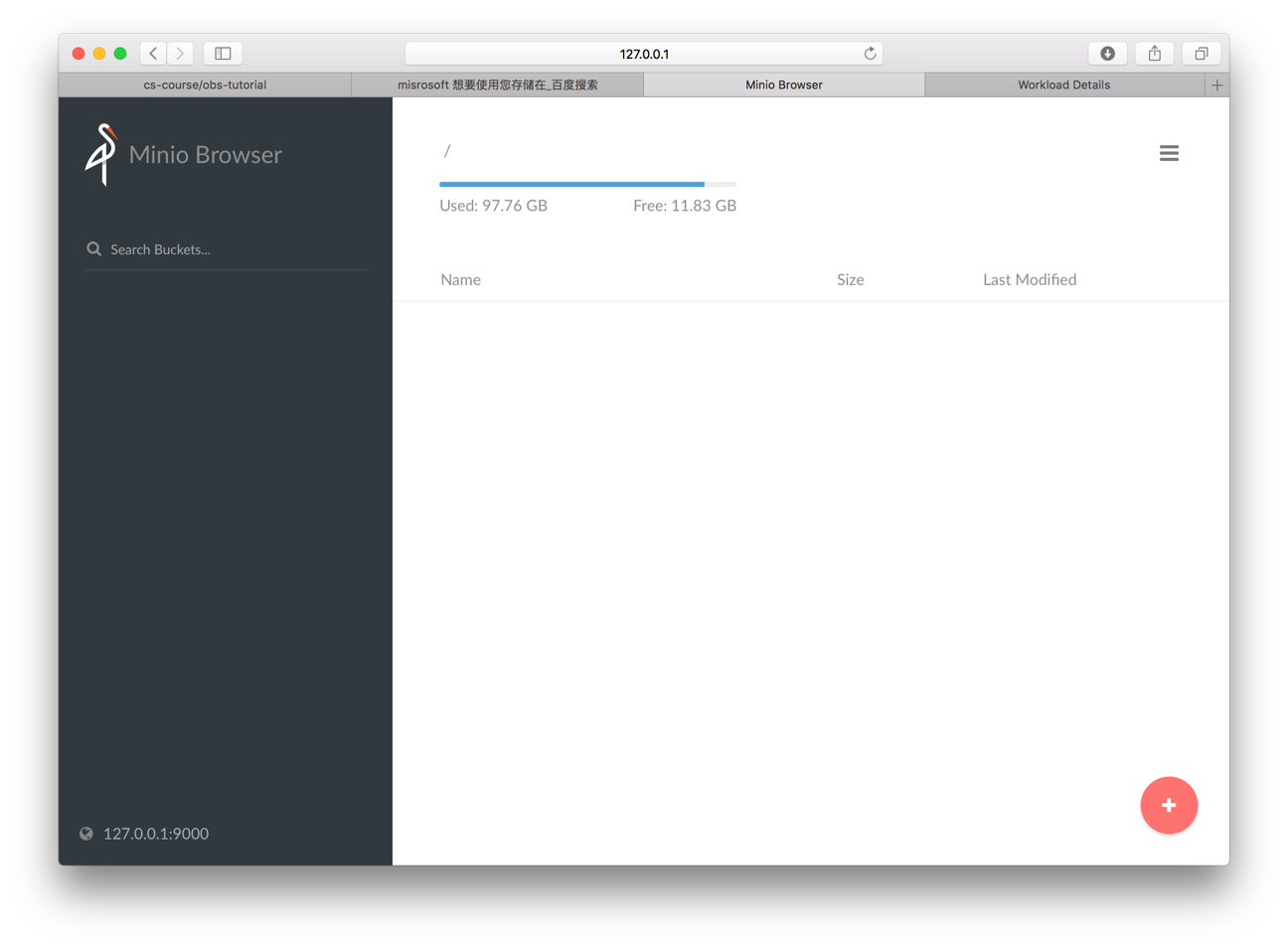


图5.2 验证minio服务器

### 1.3 使用minio client客户端

命令行直接打开客户端，如下图5.3所示.

使用mc version命令，可以查看mc的版本。

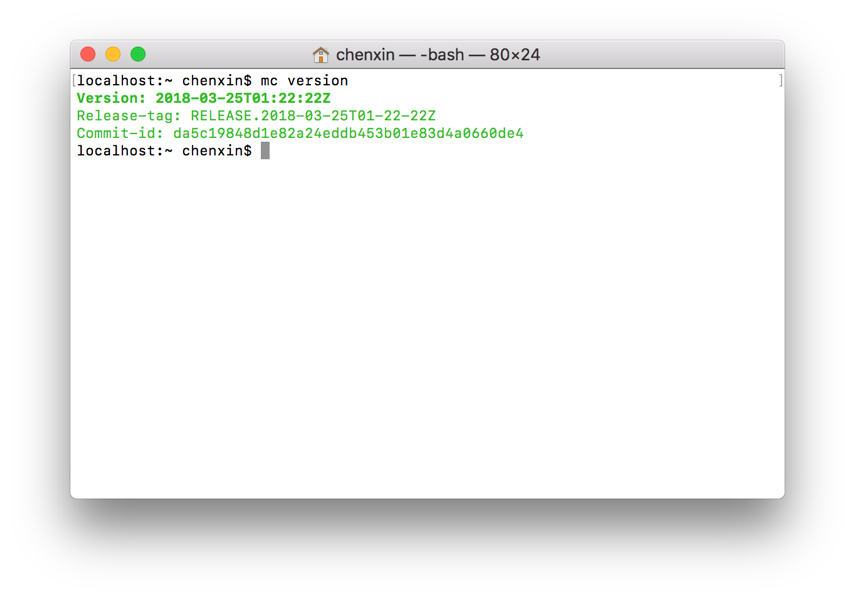


图5.3 mc版本

使用mc ls可以查看搭建的服务器上的文件信息，如下图5.4所示。

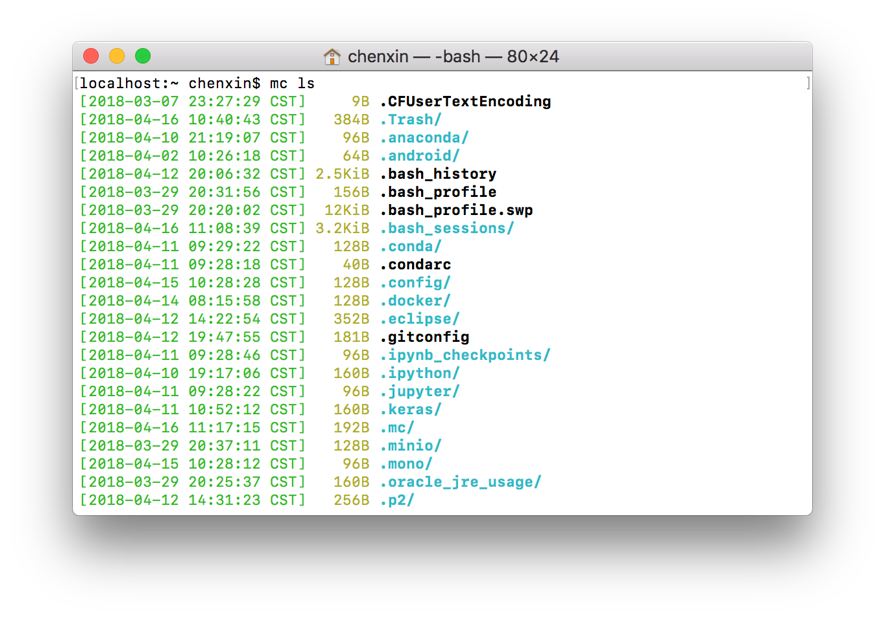


图5.4 mc ls命令

使用mc find命令可以查找服务器上的特定文件。

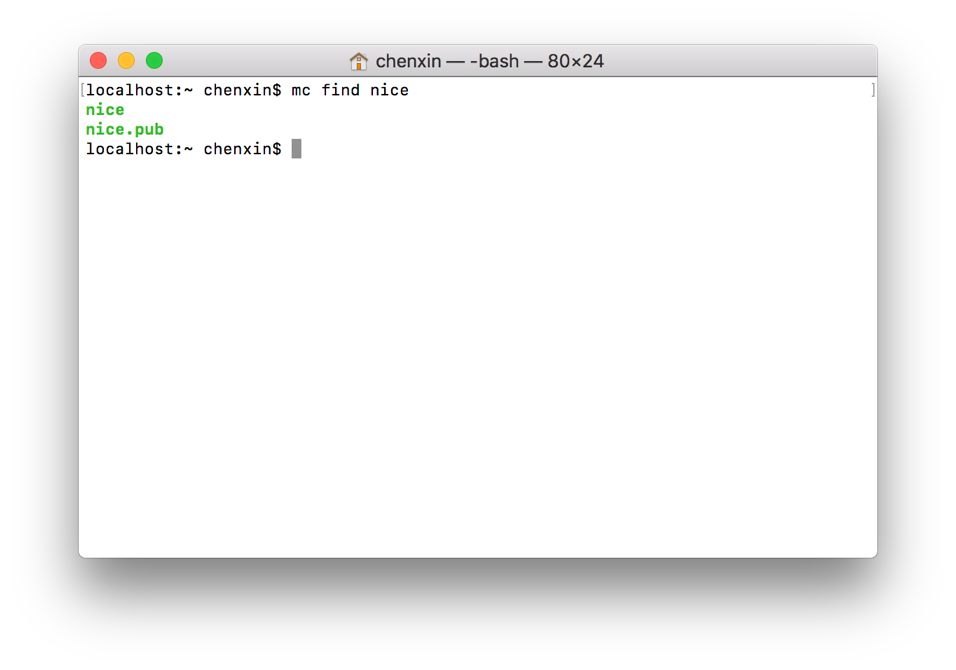


图5.5 mc find命令

### 1.4 COSBench测试

首先命令行cd到目标目录，接着命令行启动测试程序，如下图5.6所示。

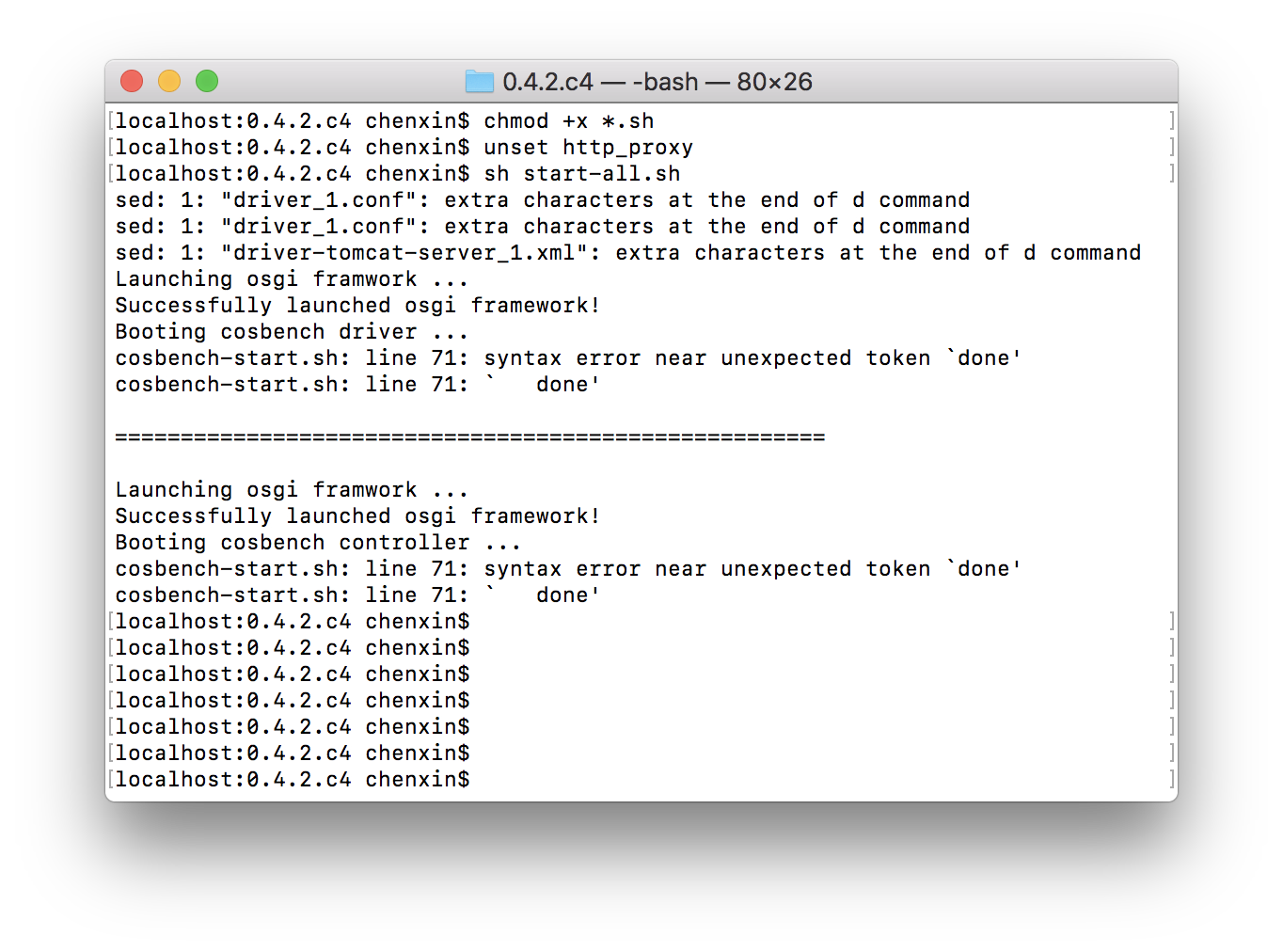


图5.6 启动测试程序

接着打开浏览器，访问<http://127.0.0.1:19088/controller>，见下图5.7。

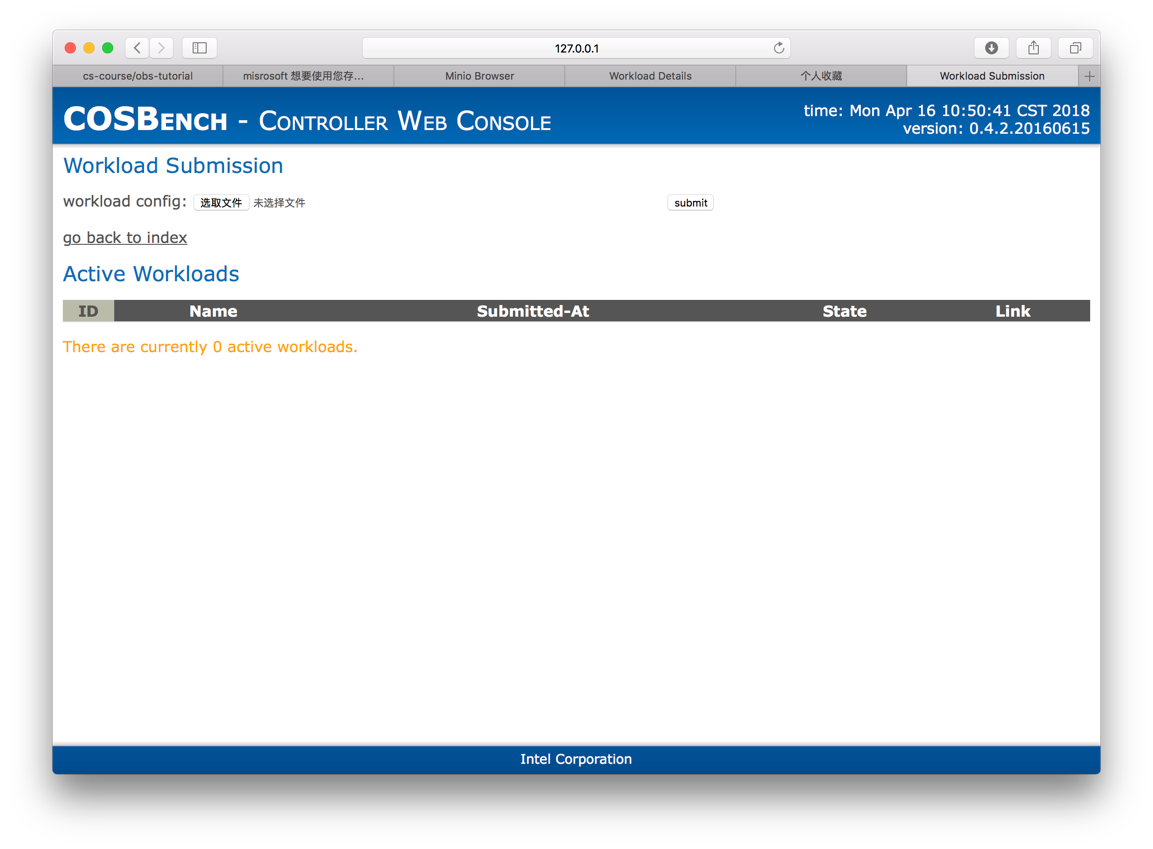


图5.7COSBench

导入workload，进行测试。

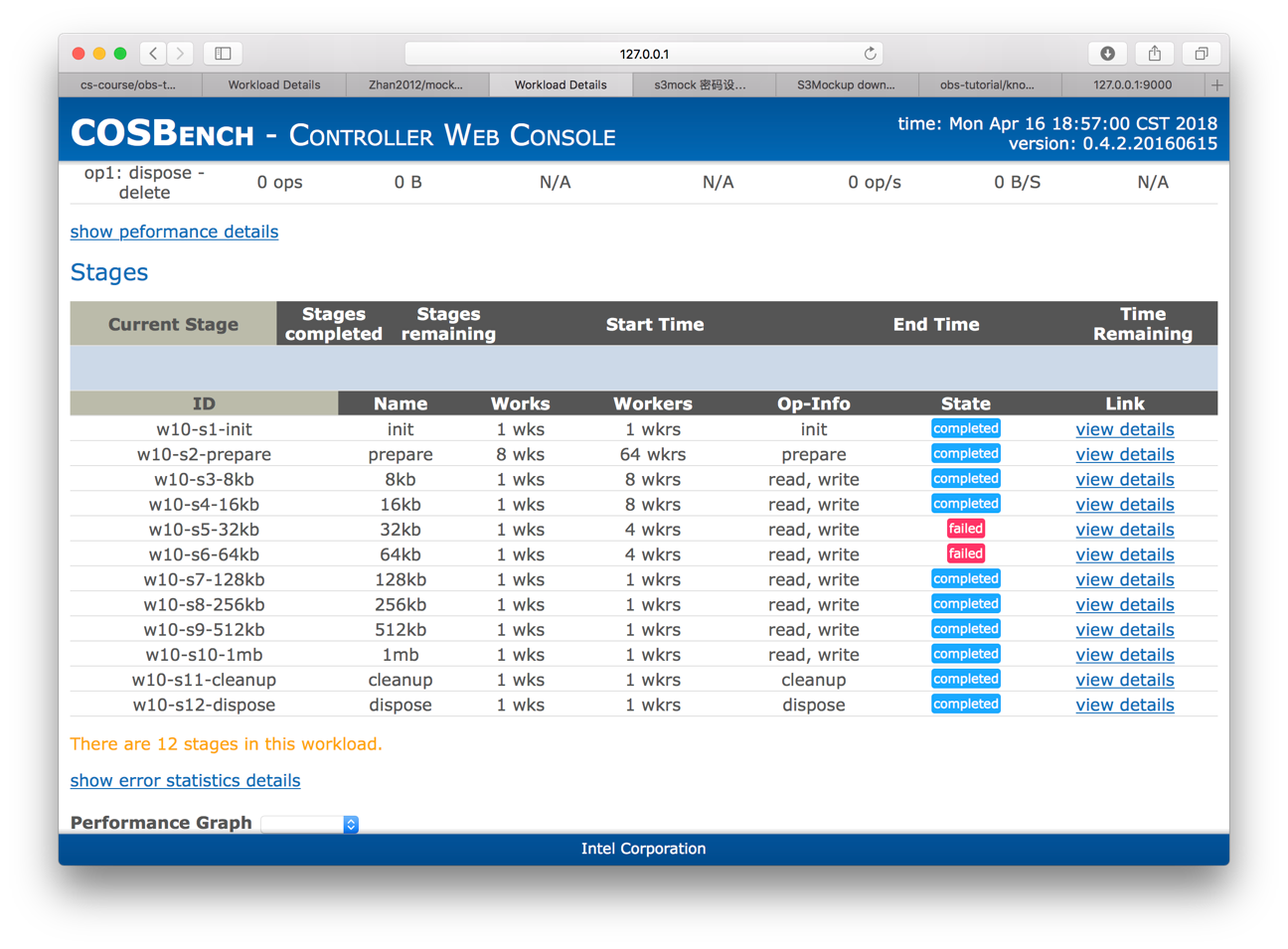


图5.8 COSBench测试结果

由上述实验结果可以发现，测试失败。下面我们进行分析。

### 1.5 COSBench分析

依照在班上交流的经验，上述COSBench几乎在每个windows电脑上都是成功的，唯独在macos系统的电脑中会失败，我猜想，这可能是跟系统的某种设置或者某种执行策略有关。

通过咨询老师，我得知可能与并发值有关，下面是其中第五第六部分（即出错部分）的代码。

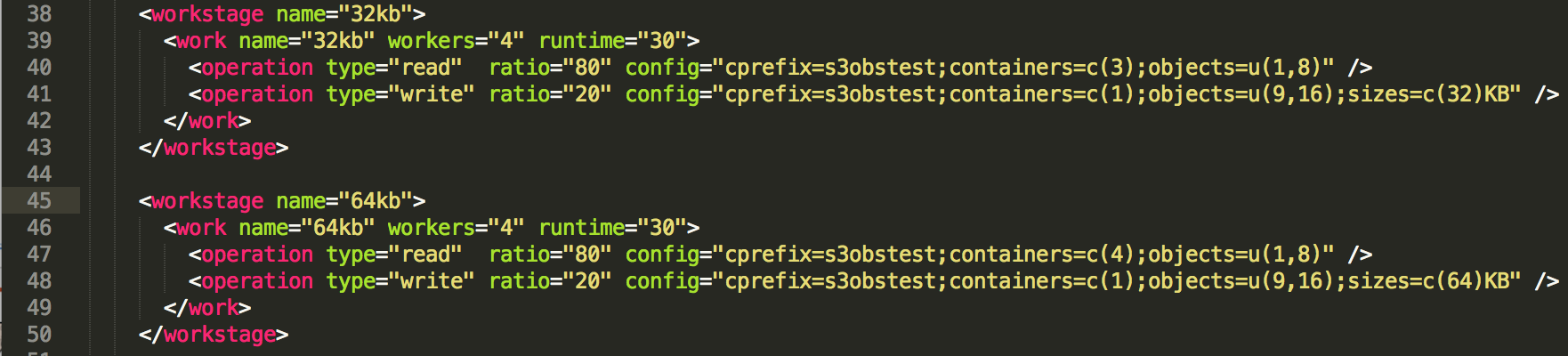


图5.9 COSBench出错部分代码

我们可以清晰的看到并发值，即图5.9中的workers均为4，我们将其修改为8，如下图5.10所示。

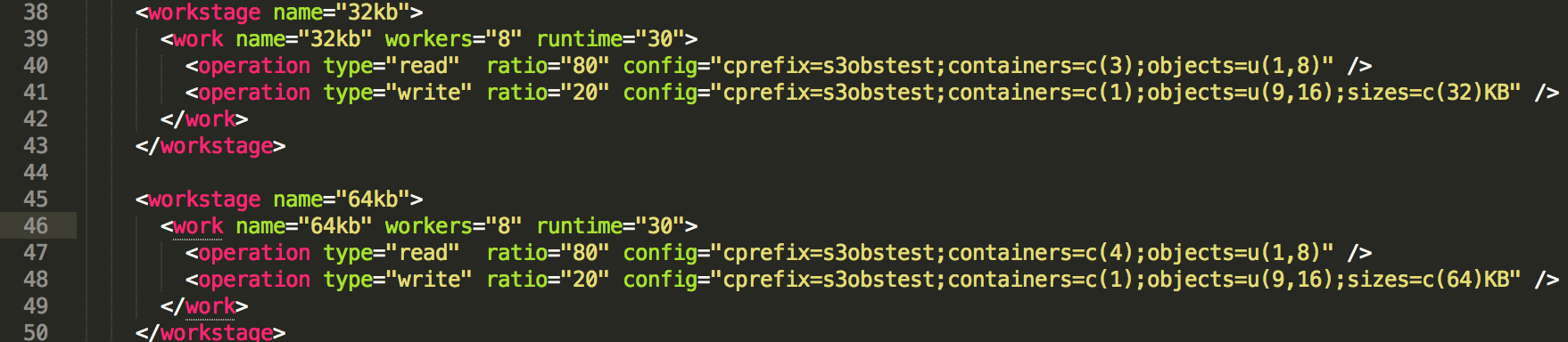


图5.10修改错误处的并发值

修改后我们再跑一次COSBench，这一遍的结果是正确的，如下图5.11所示。

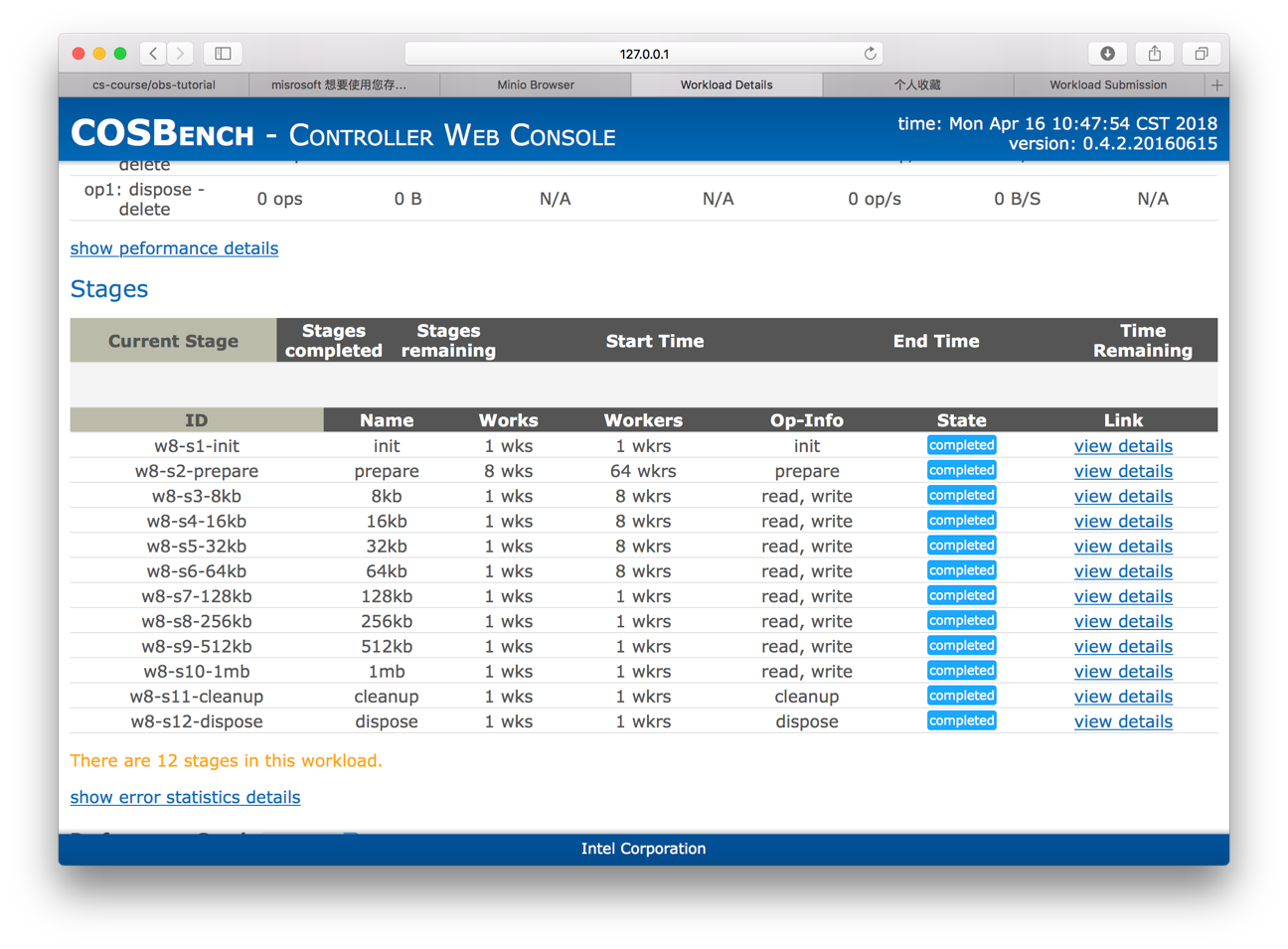


图5.11修改错误并发值

可见错误被解决了，但是问题仍然悬而未决，我并不清楚为什么这样做可以解决错误。

令人疑惑的地方是，单单从并发值（workers）角度研究，并发值有8、4、1三种情况，但唯独并发值为4会导致错误，既不是三者中的最大值，也非最小值，为了深入研究这个问题，我将原本正确的其他几部分并发值均改为4，查看此次的结果。

结果如下图5.12所示，并发值改为4后，有的失败了，而有的成功了。

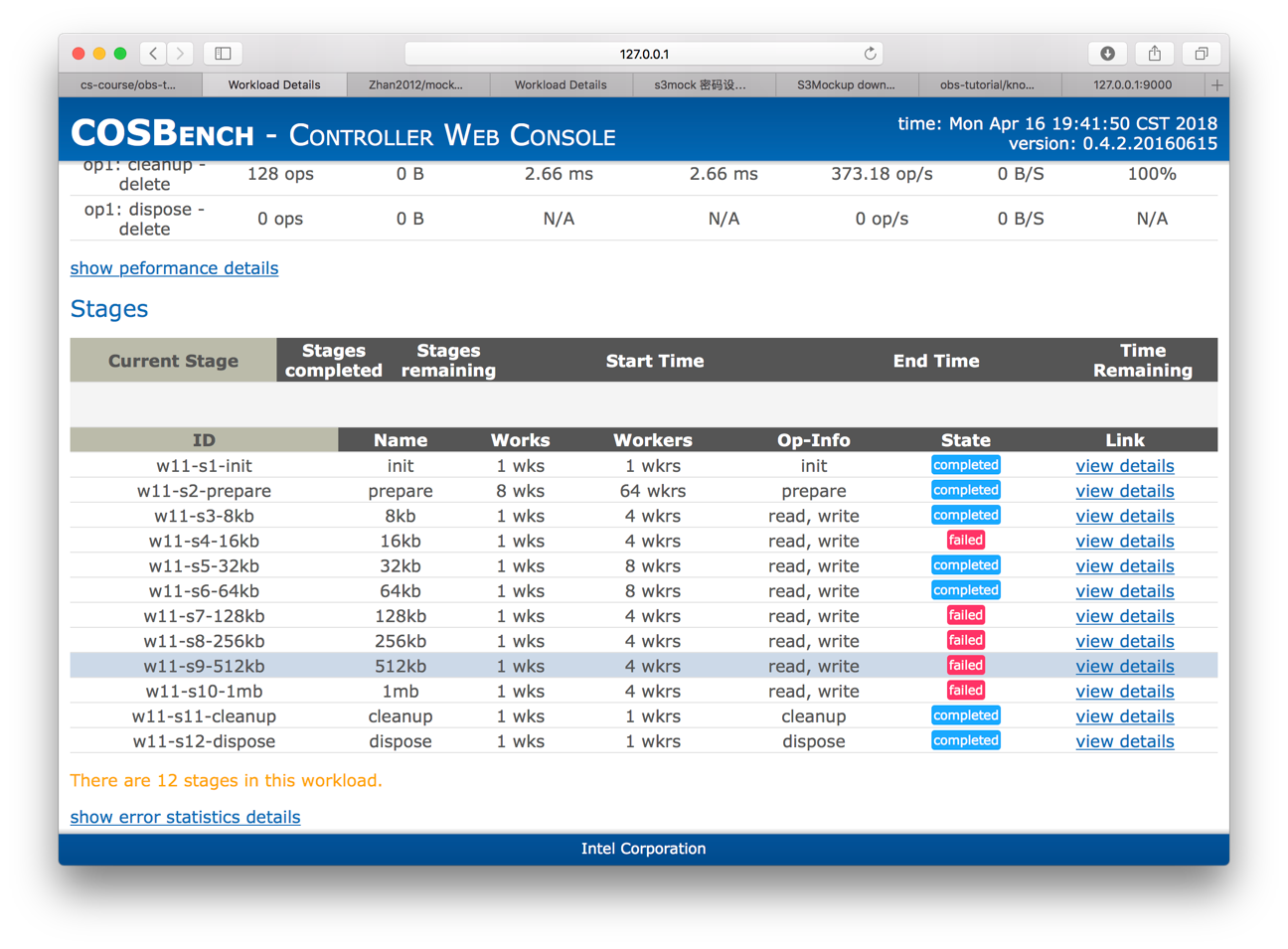


图5.12 修改原本正确的并发值为4

我们可以看到结果并不直观，并发值同为4的情况下，有的操作是成功的，而有的失败了。值得注意的是，成功的标准并不是read操作和write操作百分百成功执行，而是达到一个足够高的值(80%)。如下图所示，read操作成功度为80.2%，但它依旧是成功的。

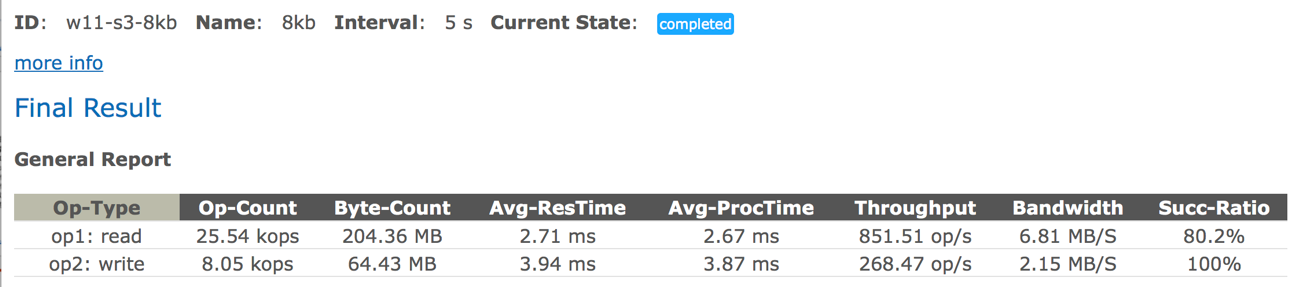


图5.13 成功标准

为了研究并发值和read操作成功率的关系，我控制其他变量均相同，研究8kb情况。

如下表所示。

|  |  |
| --- | --- |
| 并发值 | 性能结果 |
| 8 |  |
| 4 |  |

表5.1并发值和read成功率的关系

可以看到当并发值从8减少到4，成功率不增反减，由原来的94%降低到80%。

通过询问老师，我发现这些实际上跟检验和（checksum）有关。

## 2. 中等难度实验

### 2.1搭建mock-s3服务器

下载安装完mock-s3后，使用命令行搭建服务器，如下图5.14所示。

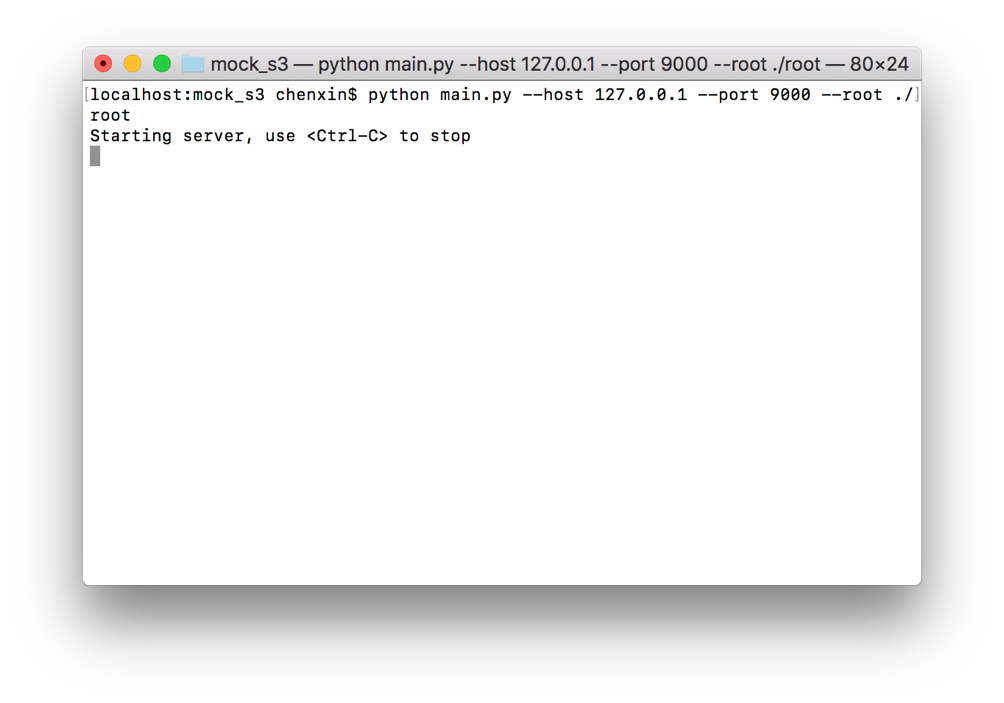


图5.14 mock-s3服务器

### 2.2验证服务器

为了验证客户端，我们打开网页http:// 127.0.0.1:9000,如下图所示。

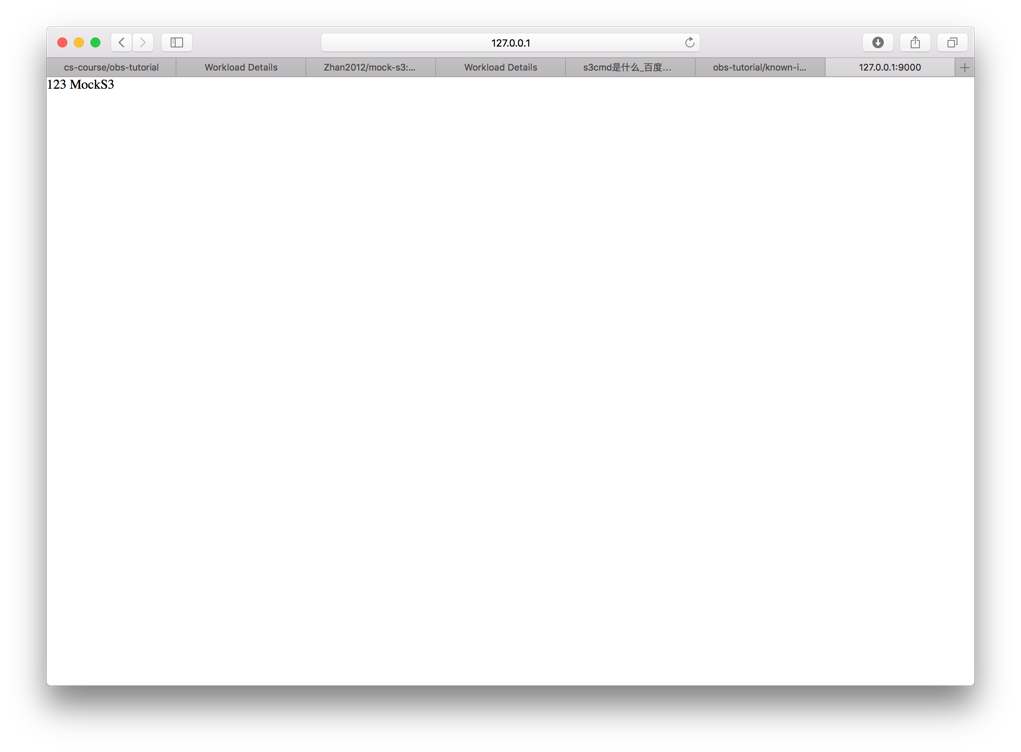


图5.15 mock-s3服务器

### 2.3使用s3cmd客户端

安装完s3cmd客户端，使用命令行s3cmd开头的命令即可访问，如上述mc使用类似，不赘述。如下图5.16所示。

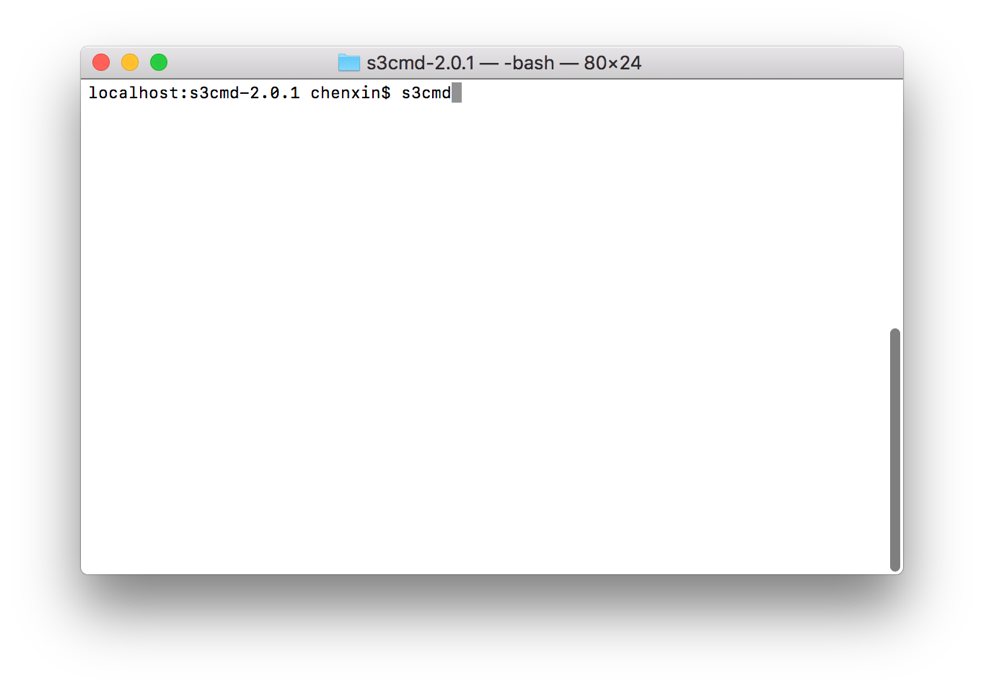


图5.16 s3cmd客户端

### 2.4 COSBench测试

首先为了测试搭建的服务器的性能，我们使用COSBench测试。

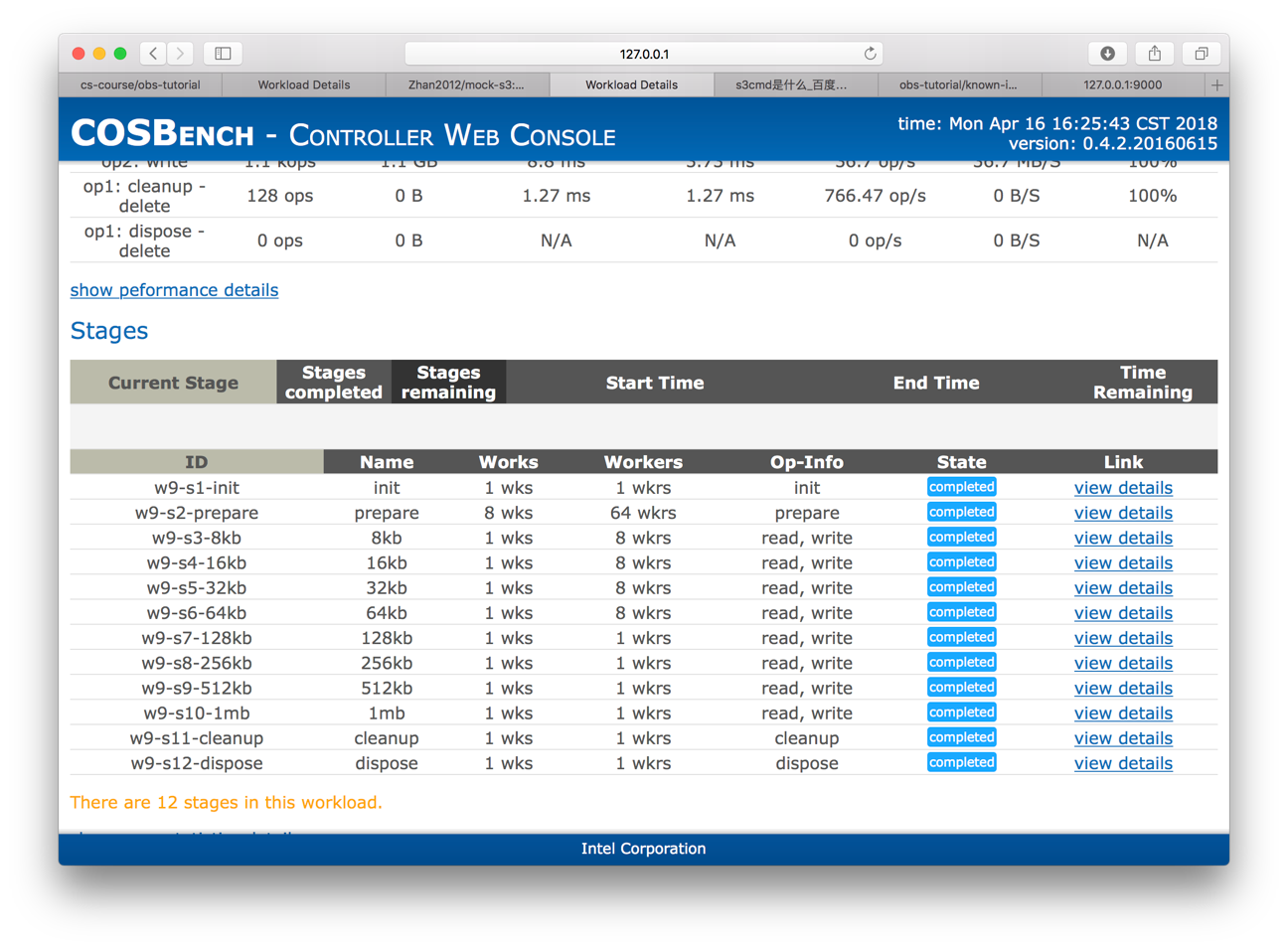


图5.17 中等难度实验cosbench

由上述测试结果可知，测试成功。

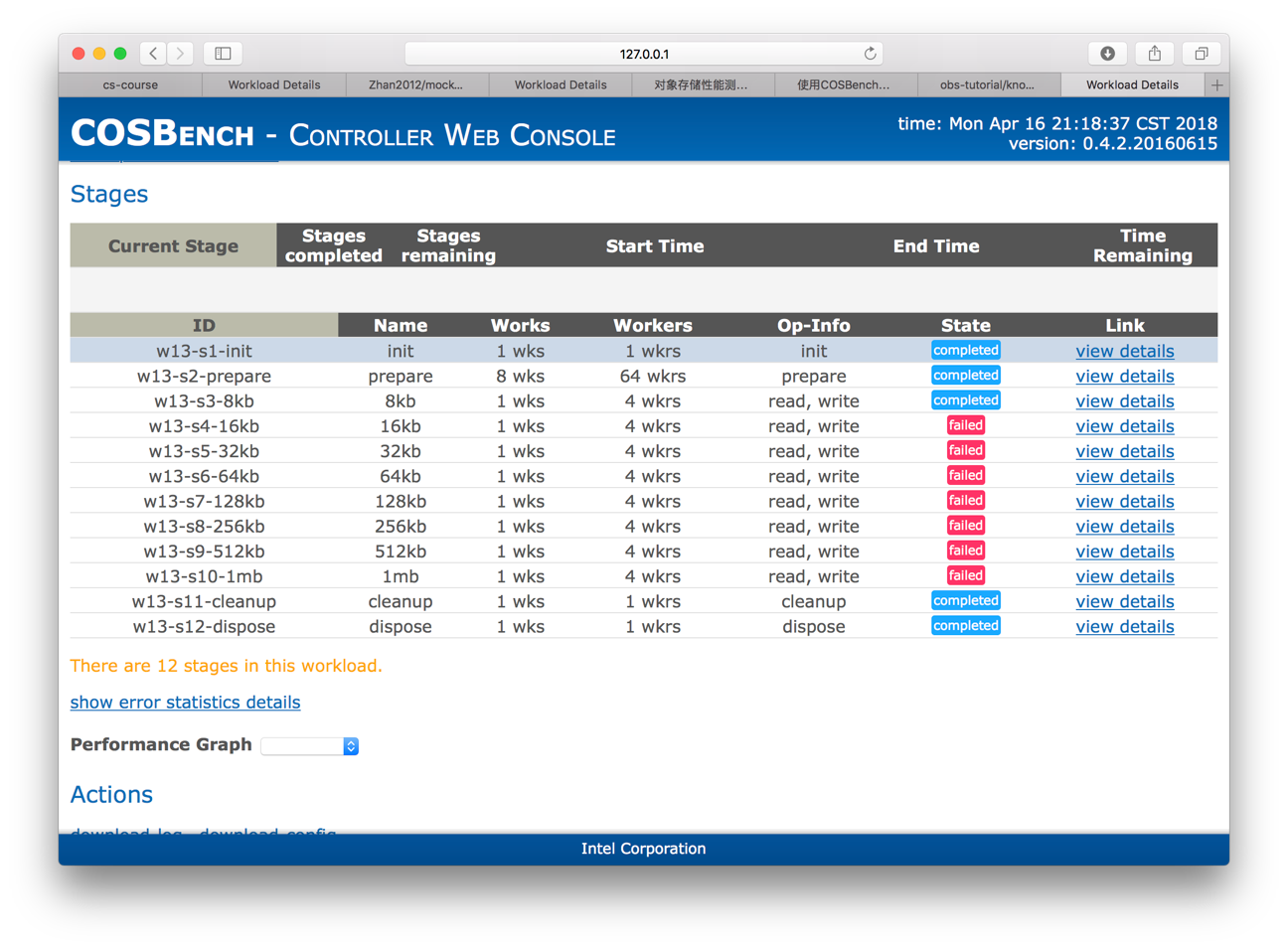
### 2.5 Topic选题研究

我选择的题目是：***对象尺寸如何影响性能?***

***\* 对于熟悉的某类应用，根据其数据访问特性，怎样适配对象存储最合适?***

首先部分回答前一问，对象尺寸确实会影响到访问性能，但就COSBench的测试结果来看，虽然会影响，但只会影响read性能，对write性能几乎没有影响。

采用控制变量法，将并发值都设为4，探究结果。如下图所示。



从state结果上看，尺寸越大（从16kb开始，到1mb），read性能越差

具体从数值上分析，搜集数据，建立下表5.2.

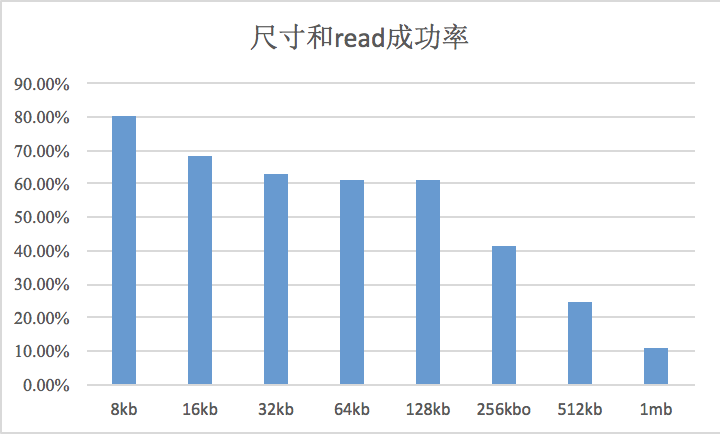


表5.2 尺寸与read成功率

总体上，尺寸越大，read成功率越低，并且当尺寸过大时，每提高一倍尺寸大小，read成功率几乎下降一半。

***对于熟悉的某类应用，根据其数据访问特性，怎样适配对象存储最合适？***

从上述结果来看，对象尺寸越小，存储性能越好，所以，一般的，我们应当选择合适范围内较小的对象尺寸。

# 六、实验总结

***实验结果分析***：基础实验和中等难度实验均成功完成。通过cosbench测试，发现不同的服务器server的性能还是有些差异的，但受限于只是在我自己电脑上测试，并不能完全肯定哪种比较好（不同的使用场景可能会有不同的结论）。

***实验体会***：本次实验过程很有意义，可以在本地开启了一个server，然后通过本地的客户端或者终端，甚至是远程的客户端，来访问这个服务器。除此以外，为了比较性能，还通过cosbench测试了对不同尺寸的对象存储的效果。我了解了许多数据存储方面的软件，了解到不同的软件的应用场景是不同的，有针对普通用户或者出于教育目的的，也有企业级的，我对他们的使用感觉和使用体验完全是不同的。一般来说，针对普通用户的操作更简单一些，结果更直观一些，而针对企业用户的看上去不够直观，但通过查阅资料，我了解到他们在性能上面的表现更加出色。

# 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.