


0.4.2.c4-cosbench.zip测v4

 0.4.2.c4-cosbench.zip

官方说明文档

obscmdbench-master.zip测v6

 obscmdbench-master.zip

测试对象存储性能搭cosbench测试v4

云主机环境准备：

1、(主节点操作)进入到/etc/yum.repos.d，将每个repo文件备份mv，换源：

```
curl -o /etc/yum.repos.d/CentOS-Base.repo  
https://mirrors.aliyun.com/repo/Centos-8.repo
```

2、清除缓存：

```
yum clean all
```

3、检查yum

```
yum makecache
```

4、给每台云主机安装依赖

```
yum install nmap-ncat java curl java-1.8.0-openjdk-devel -y
```

云主机没有ansible下载ansible*

安装ansible：

```
sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm -y  
sudo dnf install -y ansible
```

用每一台的总数totalops÷op/s对应的结果，就是时间了，单位是秒

cosbench安装

用到的包：0.4.2.c4-cosbench.zip

下载依赖：

```
ansible all -i hbhost -m shell -a 'yum install -y unzip nmap-ncat java curl  
java-1.8.0-openjdk-devel '  
yum install -y unzip nmap-ncat java curl java-1.8.0-openjdk-devel
```

1、所有节点做互信

```
sh auth.sh root XSW@zaq1 hosts(互信)
```

2、将包传到所有节点

```
ansible all -i hosts -m copy -a 'src=/root/0.4.2.c4-cosbench.zip dest=/root'
```

3、批量解压包

```
ansible all -i hbhost -m shell -a 'unzip 0.4.2.c4-cosbench.zip'
```

4、解压后修改cosbench-start.sh文件

在cosbench-start.sh原文件的/usr/bin/nohupjava后新增三个参数，如下黑色斜体标注内容(有这些内容则不管)

```
-Duser.timezone=Asia/Beijing  
-XX:-OmitStackTraceInFastThrow  
-Dcom.amazonaws.services.s3.disableGetObjectMD5Validation=true  
/usr/bin/nohupjava -Duser.timezone=Asia/Beijing -XX:-OmitStackTraceInFastThrow  
-Dcom.amazonaws.services.s3.disableGetObjectMD5Validation=true -  
Dcosbench.tomcat.config=$TOMCAT_CONFIG-server-  
cpmain/*org.eclipse.equinox.launcher.Main-configuration$OSGI_CONFIG-  
console$OSGI_CONSOLE_PORT1>$BOOT_LOG2>&1&
```

例子

```
/usr/bin/nohup java -Duser.timezone=Asia/Beijing -XX:-  
OmitStackTraceInFastThrow -  
Dcom.amazonaws.services.s3.disableGetObjectMD5Validation=true -  
Dcosbench.tomcat.config=$TOMCAT_CONFIG-server-  
cpmain/*org.eclipse.equinox.launcher.Main-configuration $OSGI_CONFIG-console  
$OSGI_CONSOLE_PORT1 > $BOOT_LOG 2>&1 &
```

同时注意 本文件

```
TOOL_PARAMS="" #一定不填 不然要报错
```

5、将主节点cosbench-start.sh复制到每个节点

```
ansible -i hosts all -m copy -a 'src=/root/0.4.2.c4/cosbench-start.sh  
dest=/root/0.4.2.c4/cosbench-start.sh'
```

6、修改0.4.2.c4/conf/controller.conf文件，增加对应driver的ip地址

这里填写的ip地址为客户端的业务网卡ip（底层用的管理网 云主机用的公网ip或着内网ip）

```
[controller]  
drivers = 40（多少台）  
log_level = INFO  
log_file = log/system.log  
archive_dir = archive  
  
[driver1]  
name = driver1  
url = http://10.230.134.1:18088/driver（节点ip）  
  
[driver2]  
name = driver2  
url = http://10.230.134.2:18088/driver  
以此类推，有多少台写到多少台
```

脚本样例

```
<?xml version="1.0" encoding="UTF-8" ?>
<workload name="xxx_write_4M" description="write">
  <workflow>
    <workstage name="xxx_write" closedelay="0">
      <work name="xxx_write_01" workers="100" totalOps="50000" driver="driver1">
        <storage type="s3" config="accesskey=lp3DCuser01;secretkey=lp3DCuser01;endpoint=http://10.195.156.140" />
        <operation type="write" ratio="100" config="cprefix=lp3dcbucket0;containers=c(1);oprefix=sgytest3dct2104t030290a;objects=s(1,50000);sizes=c(4)MB" />
      </work>
      <work name="xxx_write_01" workers="100" totalOps="50000" driver="driver2">
        <storage type="s3" config="accesskey=lp3DCuser01;secretkey=lp3DCuser01;endpoint=http://10.195.156.140" />
        <operation type="write" ratio="100" config="cprefix=lp3dcbucket0;containers=c(1);oprefix=sgytest3dct2104t030290a;objects=s(1,50000);sizes=c(4)MB" />
      </work>
    </workstage>
  </workflow>
</workload>
```

`<workload name="xxx_write_4M" description="write">` 、`<workstage name="xxx_write">` `<work name="xxx_write_01"`
测试的場景名称。建议根据实际测试内容进行描述和填写，方便查找]

`workers="100"`
对应driver的并发数。根据实际测试需求进行填写。并发数的多少和客户自身的配置相关，需根据客户端的网络带宽设置合理的并发数。
例如测试4M文件读写，客户端带宽10G，设置100并发测试，COSBench得到的带宽结果是1.2GB/s，换算成带宽为9.6Gbps，接近客户端的最大带宽，即达到客户端带宽瓶颈，此时并发值不能再增加。
同时需关注avg-time平均时延，保证平均时延符合要求。

`totalOps="50000"`
总的请求数量。此值需和objects=s(1,50000)保持一致。请求数量的多少决定测试时间的长短。测试时间根据实际的IOPS值来进行估算。比如在100并发下，IOPS为500，那么测试10分钟需要的数据量为500*10*60=300000，在设置30万totalOps之前，可以先将totalOps设置为5000，验证在对应并发下的IOPS，然后再按照时长决定totalOps的值

`driver="driver1"`
测试使用的driver，如果需要多个driver，将`<work _ /work>`一段内容复制多个，`driver="driver2"`、`driver="driver3"`按此方式增加

`accesskey, secretkey`
用户对应的ak、sk，通过移动云对象存储菜单中可查找到

`endpoint=http://10.195.156.140"`
访问对象存储的地址。根据实际测试需求进行填写，可填写的类别有公网域名、内网域名、SLB VIP地址

`operation type="write"`
脚本的操作类型。分为write、read、delete，根据实际测试需求填写

`config="cprefix=lp3dcbucket0;containers=c(1)`
桶名。由cprefix+containers组成，上述的桶名为lp3dcbucket01

`oprefix=sgytest3dct2104t030290a;objects=s(1,50000)`
对象名。由oprefix+objects组成。上述的对象名为sgytest3dct2104t030290a1, sgytest3dct2104t030290a2。。sgytest3dct2104t030290a50000
注意：每一个driver写入的对象前缀都要保证不一样

`sizes=c(4)MB`
文件大小。根据实际测试需求填写，上述为4MB文件，写入128KB文件可修改为sizes=c(128)KB

7、批量将driver给起起来

```
ansible -i cosbenchhost all -m shell -a 'cd /root/0.4.2.c4/ && nohup sh start-driver.sh &'
```

8、运行“sh start-all.sh”命令，检查19088、18088端口的状态

在本地使用<http://ip:19088/controller/index.html>访问工具的控制台页面（主节点执行就行）

9、检查cpu模式(云主机不需要)

```
for i in {1..100};do ssh 10.216.76.$i "cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor |grep performance |wc -l";echo $i;done
```

10、查看19088端口

```
netstat -nap |grep USTEN|grep 19088
netstat -ltunp |grep LISTEN|grep 19088
```

11、检查是否有18088/19088端口

(如果页面显示是红点有可能是没有端口，此时启动`start-driver.sh`看看有没有恢复为绿色)

```
netstat -ltunp
```

12、看防火墙是否关闭

应为inactive

```
systemctl status firewalld
```

13、启动完sh start-controller.sh登上网址

<http://36.137.115.114:19088/controller/index.html>

（如果发现有节点为红点，底层启动 `sh start-driver.sh`，刷新后状态变为绿色，若批量推driver启动脚本卡住或个别云主机为红点，单独进该台机器启动`sh driver.sh`，如果报错：将包删掉重新推包）

若网页没转出来，可能是没加路由（物理机的话）：

```
route add -net 172.16.160.0/24 gw 10.230.138.254
```

```
route add -net 172.20.152.0/24 gw 10.230.138.254
```

前面的IP是到4A的，后面的IP是用到的物理机节点网关

[illegible]

如果起不来

1、进入0.4.2.c4路径执行停止

stop-all.sh (或分别执行stop-controller.sh和stop-driver.sh)

2、在执行启动

start-all.sh（或cosbench-start.sh和start-driver.sh）

上面的方法都不行 直接换控制节点即可

测试脚本：并发-4M写-41台公v4.xml（endpoint写SLB的vip:80）

```
<?xml version="1.0" encoding="UTF-8" ?>
<workload name="chengdu_write_1536k" description="write">
  <workflow>
    <workstage name="chengdu_write" closedelay="0">
      <work name="chengdu_write_01" workers="40" totalOps="1000000" driver="driver1">
        <storage type="s3" config="accesskey=K4VR5EETB2U1DU6T6AX3;secretkey=JFmKIPK1IDNRYdIRYTSDX46Mw13forXAVJ3XIEFT;endpoint=http://10.231.139.201"/>
        <operation type="write" ratio="100" config="cprefix=test_000;containers=c(1);oprefix=qq1001;objects=s(1,1000000);sizes=c(4096)KB" />
      </work>
    </workstage>
  </workflow>
</workload>
```

参数	描述
workers	并发数，根据被测集群规模、客户端配置进行设置，并发大小会决定时延、IOPS、带宽结果
totalOps	操作的文件数量，需要与objects参数保持一致
driver	客户端名称，配置时需要按顺序进行填写
endpoint	请求地址，根据测试需求填写内/外网域名
cprefix=bucket;containers=c(1)	桶名，根据cprefix、containers进行拼接，左侧的桶名为bucket1
oprefix=testabc;objects(1,300000)	对象名，根据oprefix、objects进行拼接，左侧示例的对象名为testabc1，testabc2.....Testabc300000
sizes=c(16)KB	写入的文件大小，根据需求填入指定的值

底层验证pool上传速度：

rados bench -p ningbo4-zone1-dataset6-replicated-pool-3 30 write -b 4M -t 18

```
Total time run:      30.0549
Total writes made:   8965
Write size:          4194304
Object size:         4194304
Bandwidth (MB/sec):  1193.15
Stddev Bandwidth:    80.6073
Max bandwidth (MB/sec): 1248
Min bandwidth (MB/sec): 820
Average IOPS:        298
Stddev IOPS:         20.1518
Max IOPS:            312
Min IOPS:            205
Average Latency(s):  0.0603301
Stddev Latency(s):   0.0210988
Max latency(s):      0.329191
Min latency(s):      0.0382221
Cleaning up (deleting benchmark objects)
Removed 8965 objects
Clean up completed and total clean up time: 1.20626
```

正常的pool

象存储性能搭obscmdbench测v6

跟cosbenchV4用一样的IP，换用户换桶（虚机内创建的桶）

1、传到所有节点

```
ansible all -i /root/v6host -m copy -a 'src=/root/obscmdbench-master.zip
dest=/root'
```

2、批量解压包

```
ansible all -i /root/v6host -m shell -a 'unzip obscmdbench-master.zip'
```

3、解压后进入：obscmdbench-master

4、测试环境

若config.dat配置文件中，UseDomainName为false(关闭domain的访问)，将OSCs参数填写为对应的SLB ipv6地址即可（对象存储内网v6地址）

若UseDomainName为True，即填写对象存储的域名进行访问，也可以解析到后端ipv6的地址（找网络的同事确认下流量路径，是否走的ipv6）

```
#####test environment#####

# [OSCs]OSC的IP地址。若配置项 [UseDomainName]为True，此项忽略。
# 示例：OSCs = 172.20.41.3,172.20.41.4,172.20.41.2
# 长连接，系统轮询OSC建立连接。配置使用短连接，每次请求随机选择 OSC请求。
# OSCs = 127.0.0.1
# OSCs = 36.134.88.83
#OSCs = [2409:8c5b:ffff::aa9:e7c9]
#OSCs = [2409:8c85:aa10::ad1:9ac9]
OSCs = [2409:8C62:FF00::0AE7:99C9]
#OSCs = 10.209.156.185

#####test case plan#####
```

根据用例的上传下载删除来改配置

```
#####test case plan#####

# 配置测试用例，用例对应操作见下。
Testcase = 204      根据用例改

# 201 = PutObject    上传
# 202 = GetObject    下载
# 204 = DeleteObject 删除
```

并发数根据跑出来的时延大小来改

```
# 每个用户对应的的并发数，默认为1，表示1个用户对应1个并发。1个并发表示1个线程。
# 若配置项 [LongConnection]为True，一个反复使用1个HTTP/HTTPs连接。
ThreadsPerUser = 30
```

桶名改成新创的

```
##### Advanced Configuration #####  
  
# 固定的桶名，默认为空。若配置，所有并发的所有操作均对该桶名进行。  
# 示例: BucketNameFixed = fixedbucket-01  
BucketNameFixed = biaozhun01 #晏总后面创了改  
#BucketNameFixed = test0200001
```

域名改为相应的内网域名

```
# 是否使用域名。如使用域名，系统会从域名获取OSC。  
#UseDomainName = True  
UseDomainName = false  
# 是否使用虚拟主机方式请求，若使用虚拟主机方式，需要保证域名配置正确。  
VirtualHost = false  
# 域名地址  
#DomainName = eos-huhehaote-3-internal.cmecloud.cn  
#DomainName = eos-huhehaote-3.cmecloud.cn  
DomainName = eos-chengdu-4-internal.cmecloud.cn
```

大小改为4M的，这里是以B为单位， $4M=410241024=4194304$ ， $128K=1281024=131072$ ， $1.5M=1.510241024=1572864$

```
##### "201=PutObject" #####  
  
# 每个并发向自身的BucketsPerUser个桶中上传对象。  
#[ObjectSize]上传的对象大小（字节）  
# 示例: ObjectSize = 4096 上传指定大小对象。4096=4K，104857600=100MB，65536=64K  
# 示例: ObjectSize = 0~1024 上传随机大小对象。0 ~ 1024 bytes  
# 示例: ObjectSize = 0,1024,2048 随机上传大小为0,1024或2048大小的对象。  
#ObjectSize = 4096~104857600  
#ObjectSize = 10485760  
ObjectSize = 131072 #4M=4194304  
#ObjectSize = 1572864  
#ObjectSize = 104857600
```

总写入文件量=对象数并发数10

$502000*10=1000000$ （一百万）

```
# 每个并发在每个桶中上传的对象数  
ObjectsPerBucketPerThread = 200  
# 每个对象名上传次数，多次上传覆盖。多用于多版本或对象覆盖测试。  
PutTimesForOneObj = 1
```

5、将改好的config.dat传到所有节点

```
ansible all -i /root/v6host -m copy -a 'src=/root/obscmdbench-master/config.dat dest=/root/obscmdbench-master/config.dat'
```

6、新建users.dat文件，

内容如下，第一列的accountName暂时填写为test（未发现有什么用处），第二、三列分别为accesskey、secretkey


```
[root@CDJD-PSC-P11F3-SP0D5-PM-0S01-BC0NEST-SLICER-02 obscmdbench-master]# vi users.dat
test,TZUTVUDW049JEGJ9ZBAA,H0DBxs9NsQ9TcMBca0p4ux07XwXJj3i00ViXiChX
~
```

7、将改好的users.dat文件传到每个节点

```
ansible all -i /root/v6host -m copy -a 'src=/root/obscmdbench-master/users.dat
dest=/root/obscmdbench-master/users.dat'
```

前面已经改了这里只是说明不需要改) obscmdbench工具的核心配置为config.dat文件，该文件定义了需要测试的场景（比如读写删文件），测试的并发用户数、桶名、文件前缀等内容。公共配置说明config.dat文件的最后部分为公共配置区域，具体配置说明如下

```
##### Advanced Configuration #####

# 固定的桶名，默认为空。若配置，所有并发的所有操作均对该桶名进行。
# 示例: BucketNameFixed = fixedbucket-01
BucketNameFixed =  → 桶名，与系统中新建的桶名称保持一致

# 固定的对象名，默认为空。若配置，所有并发的所有操作均对该对象名操作。原操作的次数不影响。
# 示例: ObjectNameFixed = fixedObject-01
ObjectNameFixed =

# 鉴权签名算法,可选AWSV2 | AWSV4 | 空
# 若开启服务端加密功能，工具默认使用AWSV4算法。
# 保持为空：按请求随机，每一次请求时随机选用算法。
AuthAlgorithm = AWSV4 → aws测试时，需要使用V4的签名算法
# 请求所在Region名称，当使用AWSV4算法且环境配置为多Region模式时必须。
Region = cn-northwest-1 → 桶对应的region

# 是否使用域名，如使用域名，系统会从域名获取OSC。
UseDomainName = True → 开启使用域名，否则使用IP地址进行访问（IP地址的配置在config.dat文件的第一行）
# 是否使用虚拟主机方式请求，若使用虚拟主机方式，需要保证域名配置正确。
VirtualHost = false
# 域名地址
DomainName = s3.cn-northwest-1.amazonaws.com.cn → 可理解为endpoint

# 使用HTTP还是HTTPS请求。
IsHTTPS = false

# 是否使用Http2.0
IsHTTP2 = false

# 链接是否多路复用
IsShareConnection = false

# ssl协议版本号配置，当IsHTTPS为True时生效。
# 可选值包括: TLSv1, TLSv1_1, TLSv1_2, SSLv23, SSLv2, SSLv3 (不配置默认为SSLv23)
# 若python 版本 < 2.7.9, 不支持TLSv1_1, TLSv1_2。
# TLSv1 : 选择TLS v1.0协议。
# TLSv1_1 : 需要openssl version 1.0.1+, python >2.7.9
# TLSv1_2 : 当前最安全协议。需要openssl version 1.0.1+,python >2.7.9
# SSLv23: 自动协商最安全协议。
# SSLv2 : 若openssl编译时带了OPENSSL_NO_SSL2 参数不可用。该协议不安全，已不建议使用。
# SSLv3 : 若openssl编译时带了OPENSSL_NO_SSL3 参数不可用。该协议不安全，已不建议使用。

sslVersion =
```

写入测试场景的核心配置说明

Testcase = 201, 即对应了PutObject操作, 因只有一个用户, Users、UserStartIndex两个值分别配置为1、0即可, ThreadsPerUser为并发用户, 在最开始的调试过程中, 建议先配置为1, 调试成功后根据需要进行修改。

```
#####test case plan#####
```

```
# 配置测试用例, 用例对应操作见下。
```

```
Testcase = 202
```

```
# 用户数, 1个用户对应users.dat中的一行用户信息
```

```
Users = 1
```

```
# 从user.dat中加载用户的起始行号,从0开始, 空行跳过不计入。
```

```
UserStartIndex = 0
```

```
# 每个用户对应的的并发数, 默认为1, 表示1个用户对应1个并发。1个并发表示1个线程。
```

```
# 若配置项 [LongConnection]为True, 一个反复使用1个HTTP/HTTPS连接。
```

```
ThreadsPerUser = 1
```

```
# 180 = GetBucketNotification
# 182 = GetBucketMultiPartsUpload
# 185 = GetBucketLocation
# 188 = GetBucketStorageInfo
# 201 = PutObject
# 202 = GetObject
# 203 = HeadObject
# 204 = DeleteObject
# 205 = DeleteMultiObjects
# 206 = CopyObject
```

在config.dat文件中的201=PutObject部分, 按下图中的说明进行配置

```
##### "201=PutObject" #####

# 每个并发向自身的BucketsPerUser个桶中上传对象。
#[ObjectSize]上传的对象大小(字节)
# 示例: ObjectSize = 4096 上传指定大小对象。4096=4K, 10485760=100MB, 65536=64K
# 示例: ObjectSize = 0~1024 上传随机大小对象。 0 ~ 1024 bytes
# 示例: ObjectSize = 0,1024,2048 随机上传大小为0,1024或2048大小的对象。
ObjectSize = 4096
#ObjectSize = 4096~104857600
#ObjectSize = 10485760

# 每个并发在每个桶中上传的对象数
ObjectsPerBucketPerThread = 10
# 每个对象名上传次数, 多次上传覆盖。多用于多版本或对对象覆盖测试。
PutTimesForOneObj = 1

#上传对象同时指定ACL,可选: private | public-read | public-read-write | authenticated-read
# bucket-owner-read | bucket-owner-full-control, 空不携带
#PutWithACL = public-read
PutWithACL =

#对象是否字典序, 若为false, 系统则随机生成对象名。长度15~1024字节。
#若对象名非字典序, 则无法进行乱序下的下载和删除测试。
ObjectLexical = true

#若配置ObjectLexical为true, 对象名格式参考配置项 ObjectNamePartten
ObjectNamePrefix = object.test.1
#对象名partten, 字典序时有效, 一般不修改。若修改需保证对上传对象的其它操作使用相同的配置。
#保证processID, ObjectNamePrefix, Index 3个字符串存在, 字段顺序和之间连接字符(串)可定义。
#Index, 从0-ObjectsPerBucketPerThread-1
#processID, 并发号, 从0开始
ObjectNamePartten = processID-ObjectNamePrefix-Index
#若桶已开启多版本, 则上传对象后把获取到的对象版本信息写在data/objv-{0...}.dat下, 供下载和删除使用。

# 创建对象指定x-default-storage-class, 可选: STANDARD, STANDARD_IA和GLACIER, 也可以STANDARD,STANDARD_IA, GLACIER多选, 系统将从中随机选取类型
ObjectStorageClass = STANDARD
# 设置x-obs-expire头域的值
Expires =

# 以指定对象的内容作为上传对象的实际内容而不从内存中生成(当前图片转码需要)
IsDataFromFile = False

# 指定对象路径, 请给出完整路径。
# 示例: /home/tata.jpg
LocalFilePath =
```

8、改完配置后, 对所有节点run.py增加权限

```
ansible all -i /root/v6host -m shell -a 'chmod 777 /root/obscmdbench-master/run.py'
```

```
[root@promote obscmdbench-master]# ./run.py
-----obscmdbench: v3.1.7, Python: 2.7.5-----
-----Mode: Integrated-----
Config loaded
{prefix: , OSs: s3.cn-northwest-1.amazonaws.com.cn, ObjectStorageClass: STANDARD, VersionStatus: Enabled, AuthAlgorithm: AWSV4, ObjectSize: 4096, Anonym
ous: False, TpsPerThread: , ImageManipulationType: , IsRandomDelete: False, copySrcObjFixed: , CalHashMD5: False, ConnectionHeader: , ConcurrentUpParts:
False, ObjectLexical: True, SrvSideEncryptContext: , IsShareConnection: False, AvoidSinBkOp: True, AllowedMethod: GET, MultiUploadStorageClass: , copyDst
ObjFixed: , Threads: 1, SrvSideEncryptAWSKMSKeyId: , BucketNamePrefix: bucket.test, IsDataFromFile: False, IsMaster: False, ObjectNameFixed: , DomainName
: s3.cn-northwest-1.amazonaws.com.cn, MixLoopCount: 10, ResizeParams: , CdnSTSToken: , StopWindowSeconds: 0, BucketNameFixed: luopeng02, CdnSK: , WindowM
ode: False, SrvSideEncryptAlgorithm: aws:kms, PartSize: 5242880, LatencyPercentileMapSections: 10,50,90,95,99, StatisticsInterval: 3, CropParams: , Recor
dDetails: True, LocalFilePath: None, Range: , RunSeconds: , PartsForEachUploadID: 3, ObjectsPerBucketPerThread: 10, Max-keys: 1000, MixOperations: 100,10
1,104,201,102,202,203,204,103, IsFileInterface: False, ObjNamePatternHash: True, CollectBasicData: False, StorageClass: , LatencyPercentileMap: False, Vi
rtualHost: False, ThreadsPerUser: 1, CdnAK: , UserStartIndex: 0, ConnectTimeout: 30, copySrcSrvSideEncryptType: SSE-C, objectDesFile: , PutTimesForOneObj
: 1, Testcase: 201, RequestsPerThread: 2000, DeleteObjectsPerRequest: 3, RunWindowSeconds: 0, PutTimesForOnePart: 1, ObjectNamePrefix: object.test.1, Reg
ion: cn-northwest-1, LatencyRequestsNumber: False, LatencySections: 500,1000,3000,10000, ImageFormat: , BucketLocation: , TestNetwork: False, ObjectNameP
artten: processID-ObjectNamePrefix-Index, Users: 1, IsRandomGet: False, Expires: , RedirectHostName: example.com, IsHTTP2: False, BadRequestCounted: Fals
e, KeyValueNumber: 10, UseDomainName: True, IsCdn: False, Mode: 1, GetPositionFromMeta: True, PrintProgress: True, LongConnection: True, LatencyRequestsN
umberSections: 20, sslVersion: , RestoreTier: , SrvSideEncryptType: , RestoreDays: , CreateWithACL: public-read-write, PutWithACL: , IsHTTps: False, Buck
etsPerUser: 10}
Testing connection to s3.cn-northwest-1.amazonaws.com.cn SUCCESS → 连接endpoint成功
Start at 09:33:40 10/07/20 EDT, pid:11233. Press Ctrl+C to stop. Screen Refresh Interval: 3 sec
resultWriter started, pid: 11234
[#####100.0%#####] EST:00'00'00.0
[Test Case] PutObject Runtime: 00'00'12.313 → ### 100.0% ### 为动态的执行进度
[RunningThreads] 0
[Requests] 100
[OK] 100 (200,100)
[ClientErrs] 0
[ServerErrs] 0
-OuterFlwCtr 0
-InnerFlwCtr 0
[OtherErrs] 0
[ErrRate] 0.00 % 错误率为0且TPS、时延等有值显示, 说明脚本运行正常
[TPS*] 8.12
[Last5TPS*] [#4,8.0][#3,7.3][#2,8.7][#1,8.3]
[AvgLatency*] 122.14 ms
[latencyPercent*] <=500(100.0%),<=1000(100.0%),<=3000(100.0%),<=10000(100.0%),>10000(0.0%)
[DataSend] 400.00 KB
[DataRecv] --
[SendThroughput*] 32.49 KB/s
[RecvThroughput*] --
[BestReq*]
RequestID Latency(s)
F1C25B72950AAE99 0.110006093979
E5104C626B49B1D 0.110227100002
BA71B2AFD5073CF1 0.110325090038
[WorstReq*]
29E4843BB30046C1 0.204297065735
```

9、后台执行脚本，将执行结果导入test.txt文件

```
nohup ansible -i /root/v6host all -m shell -a "cd obscmdbench-master && python
run.py" -f 40 >v6xn15msc01.txt>&1 &
```

10、监控文件输出空格翻页：

```
more -s v6xn15msc.txt
```

##查看tps或带宽平均值

```
ansible -i ~/host all -f 100 -m shell -a "cat ~/obscmdbench-
master/result/2021.09.07_17*brief.txt | grep -A3 TPS"
```

```
ansible -i ~/host all -f 100 -m shell -a "cat ~/obscmdbench-
master/result/2021.09.07_17*brief.txt | grep -A3 TPS"|grep Avg
```

```
ansible -i v6host all -f 100 -m shell -a "cat ~/obscmdbench-
master/result/2022.02.19_17.30.36_GetObject_30_brief.txt | grep -A3 Avg"
```

查看脚本进程

```
ansible -i hosts all -m shell -a "ps -ef | grep run.py"
```


11、可以在/obscmdbench-master/result路径下查看生成的结果文件

```
[root@promote result]# ll
total 640
-rw-r--r-- 1 root root 4340 Oct 7 09:08 2020.10.07_09.08.50_ListObjectsInBucket_1_brief.txt
-rw-r--r-- 1 root root 2345 Oct 7 09:08 2020.10.07_09.08.50_ListObjectsInBucket_1_detail.csv
-rw-r--r-- 1 root root 237 Oct 7 09:08 2020.10.07_09.08.50_ListObjectsInBucket_1_realtime.txt
-rw-r--r-- 1 root root 4343 Oct 7 09:11 2020.10.07_09.11.15_ListObjectsInBucket_1_brief.txt
-rw-r--r-- 1 root root 2345 Oct 7 09:11 2020.10.07_09.11.15_ListObjectsInBucket_1_detail.csv
-rw-r--r-- 1 root root 237 Oct 7 09:11 2020.10.07_09.11.15_ListObjectsInBucket_1_realtime.txt
-rw-r--r-- 1 root root 4322 Oct 7 09:18 2020.10.07_09.18.34_ListObjectsInBucket_1_brief.txt
-rw-r--r-- 1 root root 2285 Oct 7 09:18 2020.10.07_09.18.34_ListObjectsInBucket_1_detail.csv
-rw-r--r-- 1 root root 359 Oct 7 09:18 2020.10.07_09.18.34_ListObjectsInBucket_1_realtime.txt
-rw-r--r-- 1 root root 4517 Oct 7 09:26 2020.10.07_09.24.13_GetObject_1_brief.txt
-rw-r--r-- 1 root root 511538 Oct 7 09:26 2020.10.07_09.24.13_GetObject_1_detail.csv
-rw-r--r-- 1 root root 5239 Oct 7 09:26 2020.10.07_09.24.13_GetObject_1_realtime.txt
-rw-r--r-- 1 root root 4519 Oct 7 09:33 2020.10.07_09.33.40_PutObject_1_brief.txt
-rw-r--r-- 1 root root 23825 Oct 7 09:33 2020.10.07_09.33.40_PutObject_1_detail.csv
-rw-r--r-- 1 root root 725 Oct 7 09:33 2020.10.07_09.33.40_PutObject_1_realtime.txt
-rw-r--r-- 1 root root 4517 Oct 7 09:51 2020.10.07_09.50.57_GetObject_1_brief.txt
-rw-r--r-- 1 root root 23825 Oct 7 09:51 2020.10.07_09.50.57_GetObject_1_detail.csv
-rw-r--r-- 1 root root 481 Oct 7 09:51 2020.10.07_09.50.57_GetObject_1_realtime.txt
-rw-r--r-- 1 root root 3065 Oct 7 09:51 archive.csv
```

对场景的数据统计

每次测试后结果会追加到此文件中

12、每隔3秒（可配置）统计一次，realtime.txt中的结果需要手动计算，汇总的结果可参考archive.csv文件。

```
[root@promote result]# cat 2020.10.07_09.33.40_PutObject_1_realtime.txt
NO      StartTime      OK      Requests      ErrRate(%)      TPS      AvgLatency(s)      SendBytes      RecvBytes
1      10/07_09:33:42.319      25      25      0.0      8.3      0.118      102400      0
2      10/07_09:33:45.319      26      26      0.0      8.7      0.113      106496      0
3      10/07_09:33:48.319      22      22      0.0      7.3      0.138      90112      0
4      10/07_09:33:51.319      24      24      0.0      8.0      0.123      98304      0
5      10/07_09:33:54.319      3      3      0.0      1.0      0.114      12288      0
```

13、读测试场景的核心配置说明

读场景只需要将用例操作配置为202： Testcase = 202 （GetObject） ， 对应的202部分配置如下，不用做修改即可下载201=PutObject中上传的所有对象。

```
##### "202=GetObject" #####
# 按以下顺序查找对象处理：
# 1) 查看是否指定了上传时生成的detail文件。（上传时对象名可非字典序）如objectDesFile=result/abc.csv
objectDesFile =

# 2) 查看是否有任一个并发对应的data/objv-{0...}.dat文件。若桶开启多版本，上传对象会自动生成该文件。
# 文件记录了对象多版本信息，从该文件内读取按指定版本下载。

# 3) 按工具命名规则下载对象，下载201=PutObject中上传的所有对象，上传的对象名要求字典序。

#指定Range下载对象，空表示不指定。格式参考http 1.1协议Range定义,如可选合法值：
# 0-9          #请求对象内容的从第0到第9个字节。
# 9-          #请求对象内容的从第9个及以后的字节。
# -200        #请求对象内容的最后200个字节。
# 0-1,5-      #请求对象内容的多个range段。
# 多个range用分号隔开。如0-1;3-5;6-10(可以重叠)，工具每次请求随机选择。
Range =

# 是否随机获取开关
# 随机获取无需遍历所有201=PutObject中上传对象，该配置属于性能测试需求，对于对象数过大的场景，建议关闭
IsRandomGet = false

# cdn开关，判断是否折扣计费，如果为true，请提供CdnAk与CdnSK以及CdnSTSToken
IsCdn = false

# Cnd的AK、SK，用于计算V2签名
CdnAK =
CdnSK =

# 由IAM提供的cdn Token
CdnSTSToken =
```

以下是可能会遇到的问题

如果执行过程中ErrRate不为0.00%，可在查看 cat /obscmdbench-master/log/obscmdbench.log 日志文件，下图为实例，可以看出提示认证方式不正确，需要使用AWS4-HMAC-SHA256。

如果执行进度为100%，ErrRate为0.00%，则会出现如下提示。

####测试单并发####

用v4环境的最后的五台搭一下环境（把相应的driver添加进去），在另起一台搭环境，两边同时跑，然后看grafana监控

单并发测试5台的环境一直跑写入，1台的跑上传下载删除

#测试

#cosbench

#对象存储