# 获取Cosbench

从svn上checkout：

<http://192.168.88.85/svn/STORAGE_PRODUCT/cosbench/cosbench-bin-0.5.zip>

这是编译好的可执行包，无须再编译。

# 准备环境

## Dstor集群

如下所示，使用3台server搭建Dstor集群。搭建集群的步骤见：Dstor集群部署及维护文档.docx

Server0: 192.168.100.190

Server1: 192.168.100.191

Server2: 192.168.100.192

部署完成后，确保在每个server上，Proxy都是正常服务的。

## 创建测试账户

注：若已经有测试账户，可忽略这一步。

在一台存储server（例如Server0）上：

* cd /usr/local/stor-openresty/nginx/proxy/test/bash
* vim create\_user.sh, 并更改如下两行：

id="xxxxxxxxxx"

key="yyyyyyyyyyyyyyyyyyyyy"

id和key任意选择，不过不要改变风格（大小写，长度等）。在本例中：

id="JTLA6O1TF69Z0YB4I7O1"

key="cwoNbM7TZLxYeMcmQxfiwL7n4Pv0JhPRlNG6m1dq"

* 执行这个脚本，创建测试账户：

sh create\_user.sh -u yuanguotest

## 测试Server

另外选择3台server（最好是Dstor集群之外的server）。

TestServer0：192.168.100.200

TestServer1：192.168.100.201

TestServer2：192.168.100.202

要求：

* 和Dstor在同一网段；
* Java已经安装；
* 依赖nc，需要用yum安装：

yum install -y nc

# 安装cosbench

在**所有测试Server**上：

* mkdir /opt/cosbench
* 从svn下载cosbench-bin-0.5.zip（见第1节：获取cosbench），并放到/opt/cosbench/下；
* cd /opt/cosbench
* unzip cosbench-bin-0.5.zip
* cd cosbench-bin-0.5
* chmod +x \*.sh

# 配置cosbench

TestServer0：作controller;

TestServer1：运行3个driver;

TestServer2：运行3个driver;

## TestServer0

TestServer0作controller，需要把6个driver的信息配置在此：

cd /opt/cosbench/cosbench-bin-0.5

vim conf/controller.conf

[controller]

**drivers = 6**

log\_level = INFO

log\_file = log/system.log

archive\_dir = archive

[driver1]

name = driver1

url = http://192.168.100.201:18088/driver

[driver2]

name = driver2

url = http://192.168.100.201:18188/driver

[driver3]

name = driver3

url = http://192.168.100.201:18288/driver

[driver4]

name = driver4

url = http://192.168.100.202:18088/driver

[driver5]

name = driver5

url = http://192.168.100.202:18188/driver

[driver6]

name = driver6

url = http://192.168.100.202:18288/driver

## TestServer1和TestServer2

不须配置

# 运行cosbench

## 5.1 TestServer1

运行3个driver：

cd /opt/cosbench/cosbench-bin-0.5

./start-driver.sh 3

从输出可以看出，在这个sever上，运行了3个driver。它们分别监听18088,18188,18288三个端口。这也是4.1节配置在controller.conf中的。

若发现端口冲突，导致driver启动失败，请关掉占用这些端口的进程。

## 5.2 TestServer2

和TestServer1相同；

## 5.3 TestServer0

在TestServer0上启动controller：

cd /opt/cosbench/cosbench-bin-0.5

./start-controller.sh

从输出可见，controller使用端口19088。若发现端口冲突，导致controller启动失败，请关掉占用这个端口的进程。

现在，可以测试一下controller是否能够发现所有driver：

# ./cli.sh info

Drivers:

driver1 http://192.168.100.201:18088/driver

driver2 http://192.168.100.201:18188/driver

driver3 http://192.168.100.201:18288/driver

driver4 http://192.168.100.202:18088/driver

driver5 http://192.168.100.202:18188/driver

driver6 http://192.168.100.202:18288/driver

Total: 6 drivers

Active Workloads:

Total: 0 active workloads

# 创建测试bucket

在TestServer0上：

* cd /opt/cosbench/cosbench-bin-0.5
* vim conf/my-init-buck.xml

<?xml version="1.0" encoding="UTF-8" ?>

<workload name="s3-sample" description="sample benchmark for s3">

<workflow>

<workstage name="init">

<storage type="s3" config="accesskey=JTLA6O1TF69Z0YB4I7O1;secretkey=cwoNbM7TZLxYeMcmQxfiwL7n4Pv0JhPRlNG6m1dq;proxyhost=;proxyport=6081;endpoint=http://192.168.100.190:6081" />

<work type="init" workers="1" config="cprefix=testbuck;containers=r(1,200)" />

</workstage>

</workflow>

</workload>

注意：这里的assesskey和secretkey就是第2.2节（创建测试用户）中，你选择的id和key。

* ./cli.sh submit conf/my-init-buck.xml

成功之后，cosbench创建了200个bucket，供后续测试用：

testbuck1

testbuck2

……

testbuck200

# 测试写性能

在TestServer0上：

* cd /opt/cosbench/cosbench-bin-0.5
* vim conf/my-write-test.xml

<?xml version="1.0" encoding="UTF-8" ?>

<workload name="s3-sample" description="sample benchmark for s3">

<workflow>

<workstage name="main">

<work name="write-server0" workers="8" runtime="30">

<storage type="s3" config="accesskey=JTLA6O1TF69Z0YB4I7O1;secretkey=cwoNbM7TZLxYeMcmQxfiwL7n4Pv0JhPRlNG6m1dq;proxyhost=;proxyport=6081;endpoint=http://192.168.100.190:6081" />

<operation type="write" ratio="100" config="cprefix=testbuck;containers=u(1,200);objects=u(1,1000);sizes=c(64)KB" />

</work>

<work name="write-server1" workers="8" runtime="30">

<storage type="s3" config="accesskey=JTLA6O1TF69Z0YB4I7O1;secretkey=cwoNbM7TZLxYeMcmQxfiwL7n4Pv0JhPRlNG6m1dq;proxyhost=;proxyport=6081;endpoint=http://192.168.100.191:6081" />

<operation type="write" ratio="100" config="cprefix=testbuck;containers=u(1,200);objects=u(1,1000);sizes=c(64)KB" />

</work>

<work name="write-server2" workers="8" runtime="30">

<storage type="s3" config="accesskey=JTLA6O1TF69Z0YB4I7O1;secretkey=cwoNbM7TZLxYeMcmQxfiwL7n4Pv0JhPRlNG6m1dq;proxyhost=;proxyport=6081;endpoint=http://192.168.100.192:6081" />

<operation type="write" ratio="100" config="cprefix=testbuck;containers=u(1,200);objects=u(1,1000);sizes=c(64)KB" />

</work>

</workstage>

</workflow>

</workload>

说明：

1. workers="8" ：是用来调压力的。为了达到测试效果，请逐渐调大。
2. runtime="30"：运行时间，30秒。
3. cprefix=testbuck：测试bucket前缀，见第6节（创建测试bucket）。
4. containers=u(1,200)：测试bucket, testbuck1到testbuck200。见第6节（创建测试bucket）。
5. objects=u(1,1000)：每个bucket写入的对象：object1到object1000；
6. sizes=c(64)KB：每个对象的大小，64KB。

根据测试需要，都可以调。

运行这个测试：

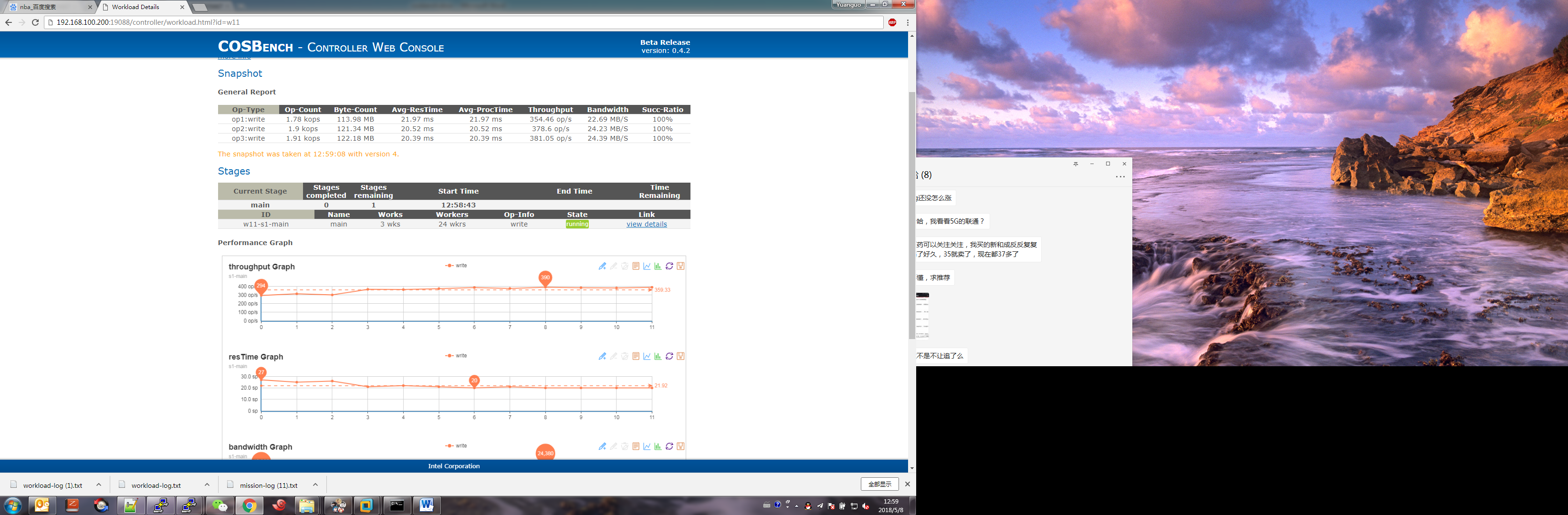
* ./cli.sh submit conf/my-write-test.xml

这时，写测试就运行起来了。我们可以在浏览器上看性能报告：

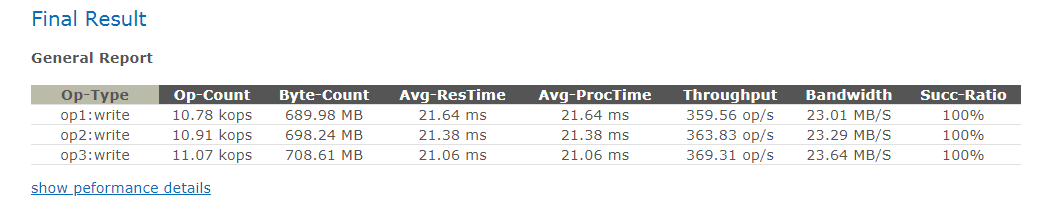
<http://192.168.100.200:19088/controller>



点击 view details：



运行完成后，就得到本轮测试的性能数据：



# 测试读性能

和测试写性能相比，测试读需要一个prepare过程：就是预先写入一些数据。这很好理解，因为，测试读，需要集群里有数据，这样才可以读。

## 预先写入数据

在TestServer0上：

* cd /opt/cosbench/cosbench-bin-0.5
* vim conf/my-prepare.xml

<?xml version="1.0" encoding="UTF-8" ?>

<workload name="s3-sample" description="sample benchmark for s3">

<workflow>

<workstage name="prepare">

<storage type="s3" config="accesskey=JTLA6O1TF69Z0YB4I7O1;secretkey=cwoNbM7TZLxYeMcmQxfiwL7n4Pv0JhPRlNG6m1dq;proxyhost=;proxyport=6081;endpoint=http://192.168.100.190:6081" />

<work type="prepare" workers="8" config="cprefix=testbuck;containers=r(1,200);objects=r(10000,20000);sizes=c(64)KB" />

</workstage>

</workflow>

</workload>

注意：

1. prepare的worker不要太大（压力不要太大。Final Result中的Succ-Ratio是100%）。因为，压力太大，有些写入可能失败，导致读测试时，文件不存在（也不影响测试，只是读操作成功率<100%时，我们不容易区分：是文件不存在导致失败，还是压力过大，系统发生超时）。
2. 在上面例子中，我们写入的对象是:

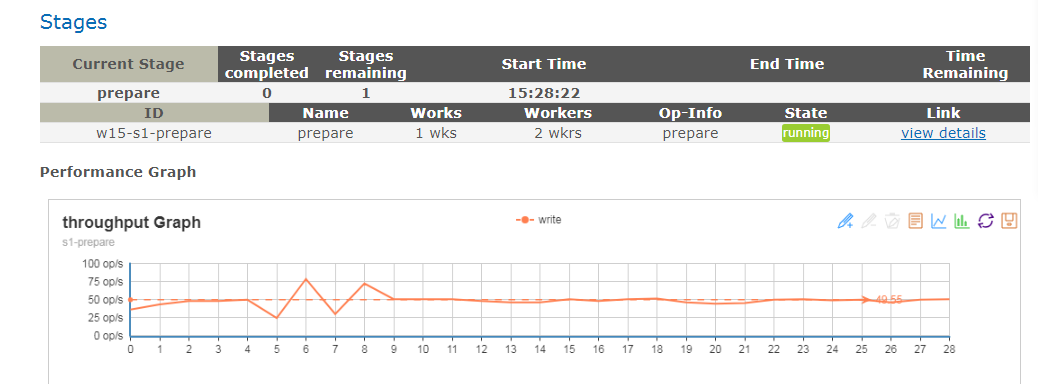
每个bucket（bucket1到bucket200）下，object10000到object20000。

一共200\*10000 = 200万个对象。

8.2节中，读性能测试，会读这些对象。

执行这个prepare：

./cli.sh submit conf/my-prepare.xml



## 测试读性能

在TestServer0上；

* cd /opt/cosbench/cosbench-bin-0.5
* vim conf/my-read-test.xml

<?xml version="1.0" encoding="UTF-8" ?>

<workload name="s3-sample" description="sample benchmark for s3">

<workflow>

<workstage name="main">

<work name="read-server0" workers="8" runtime="30">

<storage type="s3" config="accesskey=JTLA6O1TF69Z0YB4I7O1;secretkey=cwoNbM7TZLxYeMcmQxfiwL7n4Pv0JhPRlNG6m1dq;proxyhost=;proxyport=6081;endpoint=http://192.168.100.190:6081" />

<operation type="read" ratio="100" config="cprefix=testbuck;containers=u(1,200);objects=u(10000,20000)" />

</work>

<work name="read-server1" workers="8" runtime="30">

<storage type="s3" config="accesskey=JTLA6O1TF69Z0YB4I7O1;secretkey=cwoNbM7TZLxYeMcmQxfiwL7n4Pv0JhPRlNG6m1dq;proxyhost=;proxyport=6081;endpoint=http://192.168.100.191:6081" />

<operation type="read" ratio="100" config="cprefix=testbuck;containers=u(1,200);objects=u(10000,20000)" />

</work>

<work name="read-server2" workers="8" runtime="30">

<storage type="s3" config="accesskey=JTLA6O1TF69Z0YB4I7O1;secretkey=cwoNbM7TZLxYeMcmQxfiwL7n4Pv0JhPRlNG6m1dq;proxyhost=;proxyport=6081;endpoint=http://192.168.100.192:6081" />

<operation type="read" ratio="100" config="cprefix=testbuck;containers=u(1,200);objects=u(10000,20000)" />

</work>

</workstage>

</workflow>

</workload>

* ./cli.sh submit conf/my-read-test.xml

和测试写性能一样，可以到<http://192.168.100.200:19088/controller/>观察运行状态，以及最终结果。

# 测试读写混合性能

在TestServer0上：

* cd /opt/cosbench/cosbench-bin-0.5
* vim conf/my-read-test.xml

<?xml version="1.0" encoding="UTF-8" ?>

<workload name="s3-sample" description="sample benchmark for s3">

<workflow>

<workstage name="main">

<work name="readwrite-server0" workers="8" runtime="30">

<storage type="s3" config="accesskey=JTLA6O1TF69Z0YB4I7O1;secretkey=cwoNbM7TZLxYeMcmQxfiwL7n4Pv0JhPRlNG6m1dq;proxyhost=;proxyport=6081;endpoint=http://192.168.100.190:6081" />

<operation type="read" ratio="80" config="cprefix=testbuck;containers=u(1,200);objects=u(10000,20000)" />

<operation type="write" ratio="20" config="cprefix=testbuck;containers=u(1,200);objects=u(1,1000);sizes=c(64)KB" />

</work>

<work name="readwrite-server1" workers="8" runtime="30">

<storage type="s3" config="accesskey=JTLA6O1TF69Z0YB4I7O1;secretkey=cwoNbM7TZLxYeMcmQxfiwL7n4Pv0JhPRlNG6m1dq;proxyhost=;proxyport=6081;endpoint=http://192.168.100.191:6081" />

<operation type="read" ratio="80" config="cprefix=testbuck;containers=u(1,200);objects=u(10000,20000)" />

<operation type="write" ratio="20" config="cprefix=testbuck;containers=u(1,200);objects=u(1,1000);sizes=c(64)KB" />

</work>

<work name="readwrite-server2" workers="8" runtime="30">

<storage type="s3" config="accesskey=JTLA6O1TF69Z0YB4I7O1;secretkey=cwoNbM7TZLxYeMcmQxfiwL7n4Pv0JhPRlNG6m1dq;proxyhost=;proxyport=6081;endpoint=http://192.168.100.192:6081" />

<operation type="read" ratio="80" config="cprefix=testbuck;containers=u(1,200);objects=u(10000,20000)" />

<operation type="write" ratio="20" config="cprefix=testbuck;containers=u(1,200);objects=u(1,1000);sizes=c(64)KB" />

</work>

</workstage>

</workflow>

</workload>

* ./cli.sh submit conf/my-read-test.xml

和测试写性能一样，可以到<http://192.168.100.200:19088/controller/>观察运行状态，以及最终结果。

# 总结

本文较为详细的讲解了cosbench的操作过程。**若需要更详细的信息，请参考安装路径下的COSBenchUserGuide.pdf文档**（/opt/cosbench/cosbench-bin-0.5/COSBenchUserGuide.pdf）。