**COSBench测Ceph对象存储：那些网上找不到的细节**

原创 *2017-09-28* *李宏杰／童德奎* [苏研大云人](https://mp.weixin.qq.com/s?__biz=MzIzMzk0MDgxNQ==&mid=2247483927&idx=1&sn=815a233cf405f50a05f4c997a3a8fd39&chksm=e8fcb037df8b39215260957688e7cd545dbf8fb9e7a0eef1a6d411e22de975db5e41287b7df4&mpshare=1&scene=23&srcid=##)

关注我们哦

http://mmbiz.qpic.cn/mmbiz_gif/ibd8szTIphHiaZy46X2yGYgwjpGo1kicCYMGxqUejV4QkksJyZMJpGJhmfICE48aDzgIt5OBQibslMcgqslXia14ibsw/0?wx_fmt=gif&wxfrom=5&wx_lazy=1

COSBench可以用来进行CEPH对象存储的性能测试。由于COSBench相关资料比较缺乏，这里就我们遇到的一些问题进行了小结，供参考。

http://mmbiz.qpic.cn/mmbiz_gif/ibd8szTIphHiaZy46X2yGYgwjpGo1kicCYMGxqUejV4QkksJyZMJpGJhmfICE48aDzgIt5OBQibslMcgqslXia14ibsw/0?wx_fmt=gif&wxfrom=5&wx_lazy=1

**1.环境准备**

部署节点应选择专用测试节点或者具备两组万兆卡，能够实现分离业务网与存储网的纯OSD节点。**不要在有重要服务的节点上执行COSBench。**

依赖主要有java, nmap-ncat, curl。如缺少nmap-ncat，会报出找不到nc；如缺少curl，启动driver后可能无法停止，且COSBench可能无法运作；如缺少Java，COSBench起不来，一般会报出nc拒绝连接。

**1.1 修改cosbench-start.sh**

修改cosbench-start.sh，加上参数，其作用是阻止下载后验证。如无特殊需求，建议加上，否则可能会导致读terminate。

# 加上-Dcom.amazonaws.services.s3.disableGetObjectMD5Validation=true  
  
/usr/bin/nohup java -Dcom.amazonaws.services.s3.disableGetObjectMD5Validation=true -Dcosbench.tomcat.config=$TOMCAT\_CONFIG -server -cp main/\* org.eclipse.equinox.launcher.Main -configuration $OSGI\_CONFIG -console $OSGI\_CONSOLE\_PORT 1> $BOOT\_LOG 2>&1 &

**1.2 修改COSBench的端口**

由于现场环境安全策略限制，有时需要修改COSBench的端口，可以通过修改conf/controller-tomcat-server.xml完成。

# 找到该行，直接修改端口号即可  
<Connector port="19088" protocol="HTTP/1.1" />

同样的，修改客户端的端口也可以通过在客户端所属机器上修改conf/driver-tomcat-server.xml完成。随后开启/关闭相关防火墙端口。**注意：不应该开的防火墙端口在测试完成后务必关闭。**

**1.3 多客户端设定**

修改服务端的controller.conf文件，在下面加上driver条目。注意，中括号里的是指定的，编号按顺序增加。name随意，url如果是本机，可以用本地IP加上不同的端口；其它机器务必指明IP地址，并且在该客户端起来后，curl验证一下是否能够联通。正常工作的情况下，Driver与Controller必须能互相curl通。其它机器不需要修改配置文件。一个增加客户端的例子如下。

[controller]

drivers = 2  
log\_level = DEBUG

log\_file = log/system.log

archive\_dir = archive

[driver1]

name = driver1

url = http://127.0.0.1:18088/driver  
[driver2]

name = drv2

url = http://1.2.3.4:18088/driver

**1.4 添加路由**

COSBench的图形界面对于监控任务执行状态而言是很方便的，截图也好使。远程时，通常需要在本机上访问，那就需要添加相关路由，或者在终端上做端口映射了。

**1.5 启停COSBench**

如果需要同时在本机上启动服务端与客户端，直接执行start-all.sh脚本即可。如果只需要启动服务端或客户端，执行start-driver.sh或start-controller.sh。停止可以直接执行stop-all.sh。用浏览器访问完整地址http://127.0.0.1:19088/controller/index.html，即可看到Web界面。

**2. COSBench Workload**

**2.1 Workload文件字段说明**

**workload字段**  
<workload>中的name值会显示在Web页面上，尽量取有意义的名字。

**storage字段**  
存储类型，通常我们使用的都是S3。

<storage type="s3" config="accesskey=yourkey;secretkey=yourkey;endpoint=http://127.0.0.1/" />  
  
# type填s3；

# accesskey与secretkey填s3用户对应信息；

# endpoint填RGW地址或VIP；

# 如有需要，还可以再加上timeout与retry参数，但一般用不着；

**workflow、workstage字段**  
一个workflow可以包含多个workstage（建议一个就好），一个workstage可以包含多个work（除了多客户端测试，建议一个就好），一个work可以包含多个operation（除了混合操作测试，建议一个就好）。

workstage与work中的name同样要起有意义的名字，会显示在Web页面上。

建议一个workload里面只有一个单一任务：多任务调控更方便，不会有一个workload内多work造成的统计失误；

建议仅在workload里面放PUT/GET/DEL任务，而桶的创建与删除可以手动进行；

**work字段**  
workers为总并发数，totalOps为总操作数（作为任务终止条件，也可以使用时间来控制），driver为使用的客户端。如果有多个<operation>，则并发数和操作数会按照指定比例近似分布。

<work name="Test" workers="100" totalOps="2000" driver="driver1">

   <operation... />  
</work>

**operation字段**  
type只有3个值，即write, read, delete，对应三种操作。

ratio是该操作的比例，一个<work>里面的比例和必须为100。

操作使用的桶由前缀与编号指定。cprefix为桶前缀，containers为桶后缀。

操作中使用的对象名由前缀与编号指定。oprefix为对象名前缀，objects为对象名后缀。操作中使用的对象大小由sizes参数确定，其中单位可以为KB，MB，GB。

<operation type="write" ratio="100" config="cprefix=bucket;containers=c(1);oprefix=obj;objects=s(1,2000);sizes=c(512)KB" />

变量的指定，常用的有三个，c(1)为常数1，u(1,1000)为范围内随机且可以重复，s(1,1000)为范围内顺序选取。

**2.2 workload示例**

**100% PUT**

<?xml version="1.0" encoding="UTF-8" ?>  
<workload name="write\_100\_4096KB\_1000000" description="write\_100\_4096KB\_1000000">  
    <storage type="s3" config="accesskey=yourkey;secretkey=yourkey;endpoint=http://127.0.0.1/;timeout=100000;retry=3" />

   <workflow>

       <workstage name="write">

           <work name="write\_100\_4096KB\_1000000" workers="100" totalOps="1000000" driver="driver1">

               <operation type="write" ratio="100" config="cprefix=cosbench;containers=c(12);oprefix=obj;objects=s(1,1000000);sizes=c(4096)KB" />

           </work>

       </workstage>

   </workflow>  
</workload>

**混合操作【2:7:1】**

需要注意的是，混合操作需要预先写入读取测试所需要的文件。

<?xml version="1.0" encoding="UTF-8" ?>  
<workload name="Mix Ops Test 2-7-1 500" description="Mix Ops Test 2-7-1 500">

   <storage type="s3" config="accesskey=yourkey;secretkey=yourkey;endpoint=http://127.0.0.1/" />

   <workflow>

       <workstage name="Mix Ops 2-7-1 500">

           <work name="Mix Ops 2-7-1 500" workers="500" totalOps="1000000" driver="driver1">

               <operation type="write" ratio="20" config="cprefix=mix;containers=c(1);oprefix=obj;objects=s(1,200000);sizes=c(512)KB" />

               <operation type="read" ratio="70" config="cprefix=mix;containers=c(2);oprefix=obj;objects=s(1,700000);sizes=c(512)KB" />

               <operation type="delete" ratio="10" config="cprefix=mix;containers=c(3);oprefix=obj;objects=s(1,100000);sizes=c(512)KB" />

           </work>

       </workstage>

   </workflow>  
</workload>

**多客户端写**  
在单客户端的基础上，增加客户端，注意修改客户端名，以及操作涉及的文件的序号。

<?xml version="1.0" encoding="UTF-8" ?>  
<workload name="write\_100\_4096KB\_1000000" description="write\_100\_4096KB\_1000000">  
    <storage type="s3" config="accesskey=yourkey;secretkey=yourkey;endpoint=http://127.0.0.1/;timeout=100000;retry=3" />

   <workflow>

       <workstage name="write">

           <work name="write\_100\_4096KB\_1000000" workers="100" totalOps="1000000" driver="driver1">

               <operation type="write" ratio="100" config="cprefix=cosbench;containers=c(12);oprefix=obj;objects=s(1,1000000);sizes=c(4096)KB" />

           </work>

           <work name="write\_100\_4096KB\_1000000" workers="100" totalOps="1000000" driver="driver2">

               <operation type="write" ratio="100" config="cprefix=cosbench;containers=c(12);oprefix=obj;objects=s(1000001,2000000);sizes=c(4096)KB" />

           </work>

       </workstage>

   </workflow>  
</workload>

**多VIP/RGW实例写**  
下例实现了在一个Workload中向多个VIP或者多个RGW实例写的测试方案

<?xml version="1.0" encoding="UTF-8" ?>  
<workload name="put\_2000Workers\_4MB" description="sample benchmark for s3">

   <workflow>

   <workstage name="put\_2000Workers\_4MB">

       <work name="test" workers="300" totalOps="60000" driver="127.0.0.40">

           <storage type="s3" config="timeout=300000;accesskey=yourkey;secretkey=yourkey;endpoint=http://127.0.0.236:8081/;retry=3" />

           <operation type="write" ratio="100" config="cprefix=cosbench-put-4m-;oprefix=2000wks\_64k;containers=c(1);objects=s(1,60000);sizes=c(4)MB" />

       </work>

       <work name="test" workers="300" totalOps="60000" driver="127.0.0.80">

           <storage type="s3" config="timeout=300000;accesskey=yourkey;secretkey=yourkey;endpoint=http://127.0.0.236:8081/;retry=3" />

           <operation type="write" ratio="100" config="cprefix=cosbench-put-4m-;oprefix=2000wks\_64k;containers=c(1);objects=s(60001,120000);sizes=c(4)MB" />

       </work>

   </workstage>

   </workflow>  
</workload>

**3. 设计、执行测试与监控**

**所有测试环节前，强烈建议清除一次缓存。**

**3.1 写/读删独立测试**

**创建桶**

不建议使用COSBench创建桶，有可能带来不必要的麻烦。手工创建即可。

**清缓存**  
执行以下命令：sudo sync && echo 3 | sudo tee /proc/sys/vm/drop\_caches

**系统性能信息搜集【略】**

**集群性能信息搜集【略】**

**拟Workload**

对于Workload，可以手写，也可以写脚本自动生成。

**提交Workloads**

使用命令完成即可，COSBench提供了cli.sh来完成这个任务，如

sh ../cli.sh submit $xmlfile

**检查桶内对象数量**

操作前需要确认桶**存在**并且是空的，操作后要检查桶内对象的数量，因为COSBench并不保证全部上传成功（尽管他会告诉你“我上传成功率100%”）。

radosgw-admin bucket stats --bucket=cosbench$i | grep num\_objects

**3.2 混合操作测试流**

设我们要进行写读删操作比例为2:7:1与2:8:0的4种并发混合操作测试，使用三个预先创建好并且保证是空的桶。总共有五步。

**a) 桶2与3预准备数据**  
向桶2和桶3中上传一千数据（仅为举例方便），并确保上传数量正确。

**b) 执行2:7:1混合操作**  
按照如下表格拟定执行对象范围并依次执行。这里的数据范围只是一个例子，**注意写和删不要有交叉的地方，而且间隔最好大一些，因为COSBench在执行的时候不会严格按照比例执行。**

| **执行顺序** | **并发** | **PUT** | **GET** | **DEL** |
| --- | --- | --- | --- | --- |
| 1 | 100 | (1,200) | (1,700) | (1,100) |
| 2 | 500 | (251,450) | (101,800) | (201,300) |
| 3 | 1000 | (501,700) | (201,900) | (501,600) |
| 4 | 2000 | (751,950) | (201,900) | (701,800) |

**c) 清除桶1数据**  
考虑到2:8:0的操作设计为写桶1，读桶2，不涉及桶3，故而虽然上面对桶3进行了删除操作，但也没必要去恢复它，只需要清除桶1数据即可。

**d) 执行2:8:0混合操作**

| **执行顺序** | **并发** | **PUT** | **GET** |
| --- | --- | --- | --- |
| 1 | 100 | (1,200) | (1,800) |
| 2 | 500 | (251,450) | (101,900) |
| 3 | 1000 | (501,700) | (101,900) |
| 4 | 2000 | (751,950) | (1,800) |

**e) 清理工作**  
清空桶1,2,3，有始有终。

**3.3 高可用测试流**

高可用测试使用COSBench的主要目的是产生**足够覆盖所有OSD又比较少**的数据量，测试流是比较固定的，一般只需要使用一个桶。此外要注意监控对象数量变化。 

**a) 准备数据**

向桶内上传序号为（1,2000）的数据。

**b) 触发故障**

测什么高可用就触发什么故障呗。

**c) 读数据**

读取数据（1,2000），验证是否能正常读取。注意步骤c-e的顺序，先读，再写，再删。如果有步骤不正常，需要终止测试流并排查原因（除非预期结果就是不正常）。

**d) 删数据**

删除数据（1,2000），验证是否能够正常删除。

**e) 写数据**

写入数据（2001,4000），验证是否能够正常写入。有些故障下写入是会被卡住的，如日志盘故障。

**f) 修复故障，恢复正常**

**g) 清理工作**

删除桶内数据（1,4000）

**3.4 监控测试执行情况**

COSBench中有每隔5秒刷新一次的图表，在测试时建议不时查看测试情况。若图表中曲线出现异常波动，如突然降级，建议终止当前及后续测试，排查影响因素后再继续测试。

此外，COSBench实时报出的性能数据仅供参考，实时数据和最终数据可以相差很大，这是因为COSBench的性能指标是计算值，5秒的计算值和几个或者几十个小时的计算值相比，自然5秒的值是不够精确的。

**4. COSBench相关分析**

**4.1 性能指标**

**Bandwidth**  
COSBench中的带宽指标有一个小坑，据其手册，带宽中1 MB/s=1000\*1000bytes，而带宽是计算值，是总的数据传输量除以总时间得到。也就是说，在带宽原始数据（传送字节数）准确的情况下，由于进制不同，实际数据将会有所偏高。

由于一般情况下带宽利用率并不会达到100%，因此可以认为COSBench中显示速率超过1.1GB/s即已充分利用带宽，即此时性能瓶颈之一在于网卡。

**Throughput**  
吞吐量也是计算值。据其手册，计算方法为全部成功的请求除以总时间。 

注意：这里的“全部成功的请求”，指的是COSBench认为已成功的请求，而不是实际成功的请求。带宽也有相同的影响因素。

**4.2 COSBench自身问题分析**

**上传或删除不全干净**

对于COSBench的上传、下载和删除，这里有一个基于实际高并发测试的统计。

上传成功率为99.21%

下载上，对于所有完全上传的桶，都能正常进行。对于有对象缺失的桶，都会Terminate。

删除前实有数据22618870，遗留数据26803，删除成功率为99.88%。

故而结论为：在高并发条件下，COSBench可能会上传、删除不完全；若读取上传不完全的数据，会发生Terminate错误。

此外，关于retry和timeout参数，加上确实能改善（但不是解决）上传与下载的成功率，但不建议删除时加上，有可能会导致性能下降。事实上，经查看源码，这两个参数已经有默认设定了。

另一种上传不完全的情况是**并发数选取不合适**，例如300并发上传10万文件，最终会出现仅上传99900个对象的情况（除不尽）。

**超大规模测试性能不理想**  
在一次实际测试中，直接提交一个低并发小文件操作3亿的任务，通过计算，理论上任务能够撑满带宽，且通过1千万数量相同文件大小与并发的实验证明能够撑满带宽。但直接提交3亿文件任务时，所使用的带宽仅仅是上限的1/4。造成这一现象的原因目前还不明确。

因此，若要使用COSBench进行压力测试，建议把超大规模文件测试任务切分成多个小任务执行。譬如，对于3亿文件操作，建议切分成30个1千万文件用例并写入30个桶来操作，用例顺次执行。这样的好处还有三点，一为若单个任务被中断，不会影响后续任务的进行；二为便于观察任务执行进度，并动态调控任务进度（改变测试规模，或是提前终止）；三为可以在监看时发现问题（如性能异常）并及时分析调整。

**虚拟上传与删除**  
COSBench即使对不存在的bucket进行写入和删除COSBench也能全部成功（但不能读）。在这种情况下，性能数据不准确。

**性能数据超过理论上限**  
COSBench的带宽和OP数据均为计算值，计算取值为“COSBench认为已成功的文件总大小/文件总数”除以“本用例总共运行时间”得到。这能够解释COSBench性能数据有时高得离谱的问题，尤其是删除场景下。   
  
吞吐量（Throughput）的计算方式和带宽是一样的。故而，如果出现上述上传不完全的情况，测试结果将会突破天际。

此外，对于运行时间很短的用例，也会有可能出现虽然一切正常但统计数据异常的情况。

**部署节点有要求**   
节点不要有重要服务。如，在NTP节点上运行COSBench会立刻导致时钟漂移，其原因为COSBench占满带宽，影响了时钟的同步。

**4.3 常见问题检查单**

* **COSBench启动失败**

1. 检查依赖是否都具备。如都具备但起不来，考虑netcat的版本有误。
2. 调整COSBench中传给netcat的参数。

* **无法访问COSBench页面**

1. 检查是否开启对应防火墙端口，以及自己设定COSBench使用的端口是多少。
2. 对于线上环境，请检查集群到本机的网络链路是否通畅，如，转发路由是否正确设置，堡垒机这边的防火墙策略是否有设置。
3. 请输入完整的带端口的URL：http://IP:PORT/controller/index.html

* **客户端运作不正常**

1. 正常情况下，客户端与服务端之间应能互相curl通。出现这种问题，请检查网络连通性。
2. 检查控制端controller.conf的内容是否正确。客户端字段包含三个基本内容，是有要求的。
3. 客户端端口与防火墙设置有误

[driver1]                                # 格式固定，只能更改数字编号

name = driver1

url = http://127.0.0.1:18088/driver      # COSBench根据URL去找客户端，这条务必保证地址准确，端口准确。

* **测试进行时集群发生关键服务故障**

请务必把COSBench部署在专用独立测试节点，或是具备业务网与存储网分离条件的纯OSD节点上。

* **读测试总是被Terminate**

1. 请检查各个bucket内对象数量是否与预定数据规模一致。
2. 不修改cosbench-start.conf的话，COSBench会增加一步校验步骤，使用的校验源默认为Amazon，该操作可能导致读terminate。

* **性能数据超过理论上限**

COSBench并没有完成相关操作，使用了错误的数据进行统计。应重测用例，或调整用例参数。

**5. 使用建议**

**5.1 COSBench部署建议**

1. 根据实际情况决定客户端数量，以能压满带宽为准；
2. 运行COSBench的节点需要万兆网或更大带宽。为测试出性能瓶颈，客户端出口应大于或等于集群入口；
3. 运行COSBench的节点为专用集群外测试节点或至少是业务网与存储网分离的纯OSD节点（没有其它任何服务）；

**5.2 测试用例设计建议**

1. 进行预测试，对各规模用例在集群中的表现进行预估，为正式测试用例和压力测试用例的设计与选取作参考；
2. 并发数的选取，应以能够执行完成为准。其标准是，大规模文件数量的上传测试能够上传完全；
3. 文件大小的选取，如无特殊要求，下限为分片后大于磁盘条带宽度，上限为大于rgw分片大小一倍，并应考虑横向对比需要；
4. 文件数的选取，应以用例执行时间为准，以单个用例执行时间在30分钟至90分钟为佳。选取标准参考预测试的结果；
5. 调优用例设计应选取低并发用例，不能触及带宽与磁盘I/O等物理瓶颈。
6. 一组写读删Workload使用一个桶进行操作，分离任务；

**5.3 用例执行建议**

1. 执行前务必确认桶存在；
2. 对于所有测试用例，建议先依次执行全部写用例，清缓存后再依次执行全部读用例，再依次删除；
3. 将大规模用例拆分成多个相同的小用例，确保单个桶内的对象不会太多，上限可定为1千万；
4. 写完成后，务必使用

radosgw-admin bucket stats –bucket=bucketName查看桶内对象数是否达到指定用例规模；

1. 读之前，应手动清除内存缓存；
2. 用例执行过程中，应不时监看用例执行情况，若发现性能表现异常，应及时进行调整并重新提交执行测试用例；
3. 删除完成后，务必查看桶内对象数，以确认删除干净；

**5.4 结果分析建议**

1. 确认结果的合理性：执行时间过短不可用（数据不准），数据异常不可用（计算错误），运行时间不成比例不可用（说明有性能显著升降），上传不完全不可用（导致结果异常）；
2. 带宽需要手动计算，提取COSBench中的原始网速数据（以字节为单位）进行手工换算即可；
3. 在混合操作测试中，操作数不会严格按照比例分布，但不影响性能结果；

参考资料

https://github.com/intel-cloud/cosbench/blob/master/COSBenchUserGuide.pdf