**rados bench二三事**

原创 *2017-10-25* *李宏杰／荆文军* [苏研大云人](https://mp.weixin.qq.com/s?__biz=MzIzMzk0MDgxNQ==&mid=2247484012&idx=1&sn=7a40f6ecfb30c9950f2e3dc84f379b1e&chksm=e8fcb04cdf8b395a259b1aa3e9251d557df714f4dff9ca09dc36569c7138048cebf9f4fb1b68&mpshare=1&scene=23&srcid=10253mgRsazQ9pyoWwquz8tS##)

记得关注哦

概述

http://mmbiz.qpic.cn/mmbiz_png/ibd8szTIphHj2o5SYa8yCUy6xm9hJ9uicoGmriboeElcTLgvudV481ynjMhh7eNl8MAUTBmu7Qy2kuenljJJz8YoQ/640?wx_fmt=png&wxfrom=5&wx_lazy=1

rados提供了两种辅助测试的命令，本文关注于rados bench，其用于测试集群后端性能是比较合适的，下面就线上环境中使用rados bench的过程和结果做一个简要记录与分享。

http://mmbiz.qpic.cn/mmbiz_png/ibd8szTIphHj2o5SYa8yCUy6xm9hJ9uicoAJaw4ETl2NmGMRbGGibezzCicOBueRzouE2FzDHbRO4MB5z76ibIXsRLw/640?wx_fmt=png&wxfrom=5&wx_lazy=1

1

**rados bench概述**

**1.1 命令概述**

rados bench的基本命令格式如下。

rados bench -p <pool\_name> <seconds> <write|seq|rand> [-b block\_size] [-t concurrent\_operations] [-k /.../ceph.client.admin.keyring] [-c /.../ceph.conf] [--no-cleanup] [--run-name run\_name]

<pool\_name>：测试所针对的存储池

<seconds>：测试所持续的秒数

<write|seq|rand>：操作模式，write：写，seq：顺序读；rand：随机读

-t：读/写并发数，默认为16

--run-name：默认为benchmark\_last\_metadata。若做多客户端测试，这个值必须自行设置，否则会导致多客户端读失败。设一下的好，默认值长这么丑，不觉得么。

-b：block size，即块大小，默认为 4M（单位也可以是K）。此外，如果这里只填写一个数字，那么，默认单位是字节。若存储池为纠删码策略，则rados bench的最小测试文件大小与EC Stripe Width（默认4K）值相等，小于这个值则文件大小会被直接被调整为EC Stripe Width的值，例如：INFO: op\_size has been rounded to 65536。该参数只适用于写；

--no-cleanup：表示写完成后不删除测试用数据，加在读上不起作用；

-k：指定本rados命令所使用的ceph.client.admin.keyring

-c：指定本rados命令所使用的ceph.conf

**1.2 测试结果解析（基于源码）**

**1.2.1 结果项定义**

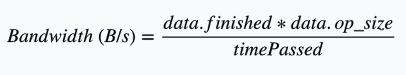
测试过程中各项含义见下表。

| **参数** | **含义** |
| --- | --- |
| started | 已发出的请求 |
| finished | 已完成的请求 |
| avg MB/s | 累计至当前时刻的带宽均值 |
| cur MB/s | 当前时刻实时速度 |
| last lat(s) | 当前时延 |
| avg lat(s) | 累计至当前时刻的时延均值 |
| Cur ops | 并发数-1 |

读和写的测试结果格式是有所区别的。Stddev的相关统计项代表的是标准差，反映该测试项样本的偏差程度，越小越好。

**1.2.2 结果项计算方法**

在最终结果中，**Bandwidth**的计算方法是：



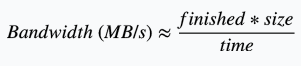
其中参数的定义为：

data.finished：完成的操作数   
data.op\_size：每个操作的大小，即参数-b指定的大小，默认是1<<22，即4M   
timePassed：单个对象的执行时间

**Average IOPS的**计算方法是：

http://mmbiz.qpic.cn/mmbiz_png/ibd8szTIphHjAJZ91tAuEFR1vibKv1ibiaKVAicru2PnW9INH7ib4DsiaKONo8l1PGrVHK5HhvsdwxXn7rV7Bs07C7UicA/640?wx_fmt=png&wxfrom=5&wx_lazy=1

所以，带宽的计算公式为：

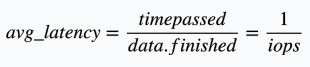


**Average Latency**的计算方法是：

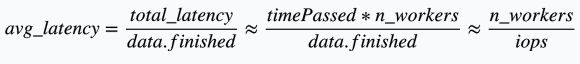
http://mmbiz.qpic.cn/mmbiz_png/ibd8szTIphHjAJZ91tAuEFR1vibKv1ibiaKV05xZI3nrAyicaBrXLeodFf8zPwI5opVJLMTlR5LPIia8l1icns2T5v5OQ/640?wx_fmt=png&wxfrom=5&wx_lazy=1

其中total\_latency是所有操作的延迟，在计算所有操作的延迟时，是将每一个操作的延迟累加得到的。

单线程情况下，每一个操作的延迟累加得到的即该次测试的总的时间timePassed，即total\_latency=timepassed, 则：

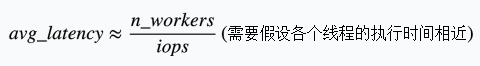


多线程情况下：timePassed = max{各个线程的执行时间}，而total\_latency=各个线程的执行时间之和。在这里我们需要作个假设，即各个线程的执行时间相近，则total\_latency≈timePassed\*n\_wokers(线程数)，故有：



综上，iops， 平均延迟，带宽的关系如下：

http://mmbiz.qpic.cn/mmbiz_png/ibd8szTIphHjAJZ91tAuEFR1vibKv1ibiaKVdzWzHibv3DcDagNyZELr2rjBbNRXQyxDQqIyZBjicOq6ojVG6uWgPFBQ/640?wx_fmt=png&wxfrom=5&wx_lazy=1



**1.2.3 示例输出**

[root@node01 ~]# rados bench -p isa-ecpool 10 write -t 48 --no-cleanup

Maintaining 48 concurrent writes of 4194304 bytes to objects of size 4194304 for up to 10 seconds or 0 objects

Object prefix: benchmark\_data\_node01\_2299803

 sec Cur ops   started  finished  avg MB/s  cur MB/s last lat(s)  avg lat(s)

   0       0         0         0         0         0           -           0

   1      47       338       291   1163.93      1164     0.13229    0.146624

   2      47       681       634   1267.89      1372    0.143106    0.142564

   3      47      1037       990   1319.88      1424    0.130413    0.139999

   4      47      1396      1349   1348.87      1436    0.131054    0.138121

   5      47      1762      1715   1371.87      1464    0.133674     0.13676

   6      47      2130      2083   1388.53      1472    0.130367    0.135656

   7      47      2500      2453   1401.58      1480     0.12737     0.13481

   8      47      2861      2814   1406.86      1444    0.126316    0.134541

   9      47      3235      3188   1416.74      1496    0.126937    0.133777

  10      47      3602      3555   1421.85      1468    0.136775    0.133404

Total time run:         10.074614

Total writes made:      3603

Write size:             4194304

Object size:            4194304

Bandwidth (MB/sec):     1430.53

Stddev Bandwidth:       97.1642

Max bandwidth (MB/sec): 1496

Min bandwidth (MB/sec): 1164

Average IOPS:           357

Stddev IOPS:            24

Max IOPS:               374

Min IOPS:               291

Average Latency(s):     0.132988

Stddev Latency(s):      0.00914917

Max latency(s):         0.205097

Min latency(s):         0.0741599

[root@node01 ~]# rados bench -p isa-ecpool 30 seq -t 48

 sec Cur ops   started  finished  avg MB/s  cur MB/s last lat(s)  avg lat(s)

   0       0         0         0         0         0           -           0

   1      47       495       448   1791.46      1792   0.0938372   0.0977953

   2      47       987       940   1879.61      1968   0.0912792    0.097713

   3      47      1493      1446   1927.65      2024   0.0913603    0.096425

   4      47      1986      1939   1938.68      1972   0.0866233   0.0966401

   5      47      2480      2433   1945.93      1976   0.0940523    0.096554

   6      47      2977      2930   1952.78      1988   0.0899998   0.0964149

   7      47      3488      3441   1965.66      2044   0.0986758   0.0959167

Total time run:       7.307804

Total reads made:     3603

Read size:            4194304

Object size:          4194304

Bandwidth (MB/sec):   1972.14

Average IOPS          493

Stddev IOPS:          20

Max IOPS:             511

Min IOPS:             448

Average Latency(s):   0.0959921

Max latency(s):       0.213512

Min latency(s):       0.0450312

**1.3 部署独立测试客户端**

有时候为了压测，需要在没有安装ceph软件包的测试节点上运行rados bench，这时就需要进行简单配置。   
a) 安装ceph-common：yum install ceph-common -y   
b) 把集群节点中的 ceph.client.admin.keyring文件拷到测试节点上对应路径下（默认是/etc/ceph），并赋予当前用户访问权限，或是指定-c和-k参数指向本机任意位置存放的keyring和ceph.conf，以使得独立测试节点能够访问集群；   
c) 有时需要临时在OSD上执行rados bench，照上一步执行即可。注意，这种临时操作，操作完后请善后（删掉keyring）；

**1.4 清除测试数据**

rados -p <pool\_name> cleanup [--run-name run\_name] [--prefix prefix]

–run-name：默认为benchmark\_last\_metadata。若做多客户端测试，这个值必须自行设置；

对于不需要压数据的测试，建议记住自己使用的--run-name，以便使用这种方式进行温柔的清理，毕竟有时不方便删池；

强烈建议善始善终，谁做的测试，谁负责恢复原状；

2

**测试流示例**

下面以一个较为完整的测试流作为使用的例子

**2.1 环境预准备**

***2.1.1 清缓存***

整个测试任一阶段前都应该清除一次全集群的缓存， 以使得测试数据更为准确。清缓存的命令如下。

sudo sync && echo 3 | sudo tee /proc/sys/vm/drop\_caches

***2.1.2 系统性能数据监控准备（略）***

***2.1.3 集群状态***

测试前搜集一次，测试后搜集一次，至少需要关注对象数目、已用空间等信息。可以使用ceph df detail等命令查看。

**2.2 测试实例**

***2.2.1 预测试：最优参数***

如果是对比测试，可以跳过这一步，因为对比测试重在对比，而无关是否最优。

预测试的目的在于寻找最优参数，如并发数，以探寻使用何种参数能够产生最大压力，以及测试多客户端是否能够正常工作。设带宽为10Gb，默认并发是16，则可以依次测试16、32、48、64……的并发数，执行10秒即可，观察哪种并发下效率最高：rados bench -p isa-ecpool 10 write -t <并发数>。这种情况下不需要加什么特别的参数，跑就是了。

例如，16M写测试，首先进行10秒短测试，得到如下结果，自左向右分别为并发数，带宽，IOPS，时延。

36 337 21 1558

32 337 21 1393

24 343 21 1054

16 342 21 711

12 343 21 543

8 345 21 369

6 350 21 274

4 230 14 278

据测试结果，6并发应该是最优的了。随后加测5分钟长测试，得到如下结果，确认6并发是最优的。故而在正式的压力测试中，使用6并发即可。

12  369 23 519

6   358 22 268

***2.2.2 预测试：性能估算***

清除缓存后，执行一个时间稍长的用例（譬如5分钟），观察结果，以进一步掌握集群性能概况。

rados bench -p isa-ecpool 300 write -t <潜在最优并发数> --no-cleanup --run-name aaa

rados bench -p isa-ecpool 300 seq -t <潜在最优并发数> --no-cleanup --run-name aaa

rados -p isa-ecpool cleanup --run-name aaa

***2.2.3 写测试***

在选好的一台或多台客户端上同时执行rados bench.

a）运行压力写测试

使用nohup执行即可。考虑到我们经常采用多客户端的形式进行测试，故而注意尽量规范输出文件的命名，注意添加–run-name。

在提交测试任务后，确认已成功执行。

b）搜集/提取/处理测试结果

***2.2.4 读测试***

注意：请先完成写测试流再执行读测试。

有几个要点需要注意的。

* 先写再读，如果读失败，考虑是没有数据的缘故；
* --run-name读写需要对应；
* 随机读一定会执行到规定好的时间，而顺序读不会。顺序读读完就会停止。考虑到读通常比写快，因此通常不会执行到规定时间，但这并不影响结果的准确性。
* 不要忘记清缓存。
* 时间允许的情况下顺序读和随机读都可以做，如果时间紧张，优先做顺序读。这是因为，随机读由于可能会对同一个对象进行多次读，该对象在第一次读时被从磁盘中提到缓存，之后都会在缓存命中，所以随机读的结果一定会是有所虚高的。

3

**其它问题**

rados bench无法按预期执行时间终止

现象：rados bench在极端情况下，执行到规定时间后不会立即停止，还会继续往前执行。往前执行的过程中，不会新增加请求（start不增加），但是完成请求数（finish）是有可能增加的。

复现：开一个超级大的并发数的任务，就能观察到该现象。如：rados bench -p <pool-name> 10 write -b 64K -t 16384

原因：经分析源码，到达stopTime仍然执行，是因为要等到所有发出去的请求返回才算整个写的过程结束。从rados bench的计算逻辑来看，其计算结果准确性没有问题，它的逻辑也是科学的，能够排除一些例如并发数无限（过大）的极端情况。但rados bench这个等待所有任务返回后才停止这个逻辑很坑，如果有请求“永远”无法完成，则rados bench会一直执行，不再输出结果。

对象统计数多1（日志在同一块盘则是2）

现象：rados bench执行后，avgcount所统计到的数目均是对象数+1，而其它任何写对象方法都没出现这一情况，op统计均是准确的。此外，通过数据大小统计，发现多了24bit，这应是这个额外的写操作所致。

问题：多出的额外写操作是什么？

原因：rados bench会在写操作结束后，写入一个以run\_name为标志的一个元数据文件，并落盘。其大小为24bit。

// persist meta-data for further cleanup or read

sync\_write(run\_name\_meta, b\_write, sizeof(int)\*3);