HBase随机读写性能测试

本文转载自淘宝网BlueDavy同学的博客，文章基于淘宝对HBase的大量应用，给出了一个HBase的随机读写性能测试结果，对测试环境、配置及性能参数分析都有较详细的描述，推荐给各位NoSQL Fans。

根据最近生产环境使用的经验，更多的项目的采用，以及采用了更加自动的测试平台，对HBase做了更多的场景的测试，在这篇blog中来分享下纯粹的随机写和随机读的性能数据，同时也分享下我们调整过后的参数。

测试环境说明：

- 1、Region Server： 5台，12块1T SATA盘(7200 RPM)，No Raid，物理内存24G，CPU型号为E5620；启动参数为：-Xms16g -Xmx16g -Xmn2g -XX:SurvivorRatio=2 -XX:+UseCMSInitiatingOccupancyOnly -XX:CMSInitiatingOccupancyFraction=85
- 2、Data Node：35台，和Region Server同样的硬件配置，启动参数上-Xms2g -Xmx2g，未设置-Xmn；

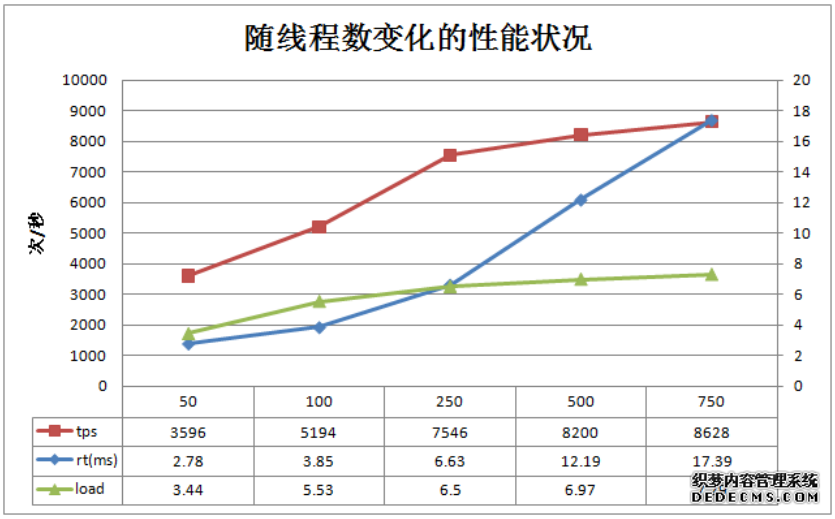
服务端参数：

- hbase.replication false
- hbase.balancer.period 1200000
- hfile.block.cache.size 0.4，随机读20%命中场景使用0.01
- hbase.regionserver.global.memstore.upperLimit 0.35
- hbase.hregion.memstore.block.multiplier 8
- hbase.server.thread.wakefrequency 100
- hbase.regionserver.handler.count 300
- hbase.master.distributed.log.splitting false
- hbase.regionserver.hlog.splitlog.writer.threads 3
- hbase.hregion.max.filesize 1073741824
- hbase.hstore.blockingStoreFiles 20
- hbase.hregion.memstore.flush.size 134217728

客户端参数：

- hbase.client.retries.number 11
- hbase.client.pause 20
- hbase.ipc.client.tcpnodelay true
- ipc.ping.interval 3000

最终随机写的测试性能结果如下（点开可看大图）：



从写的测试来看，可以看到，当客户端线程数在250左右时，此时的响应时间在6ms左右，tps在7.5k左右，差不多是比较好的一个状态。

在随机写的测试中，以及我们的一些项目的测试中，看到的一些现象和问题：

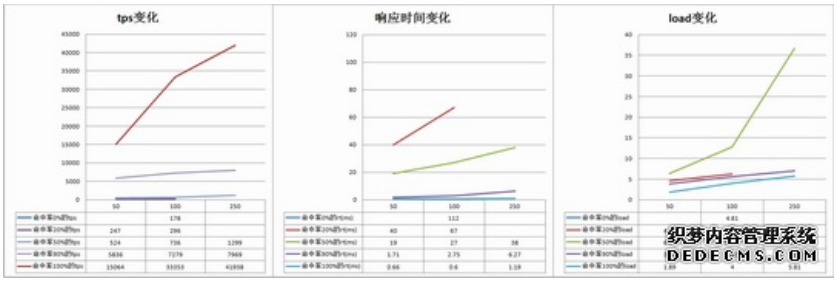
- 1、随着单台机器的region数变多了，tps下降的比较明显，team的同事做了一个改进，保障了随着region数的增多，tps基本不会有太多的下降，具体请见同事的这篇blog；
- 2、当hbase.regionserver.handler.count为100（默认为10，更正常了）时，压力大的情况下差不多100个线程都会

BLOCKED，增加到300后差不多足够了，此时tps也到达瓶颈了；

- 3、当datanode数量比较少时，会导致写tps比较低，原因是此时compact会消耗掉太多的网络IO；
- 4、当写采用gz压缩时，会造成堆外内存泄露，具体请参见同事的这篇blog；
- 5、在压力增大、region数增多的情况下，split和flush会对写的平稳性造成比较大的影响，而通常内存是够用的，因此可以调整split file size和memstore flush size，这个要根据场景来决定是否可调整。

对写的速度影响比较大的因素主要是：请求次数的分布均衡、是否出现Blocking Update或Delaying flush、HLog数量、DataNode数量、Split File Size。

随机读的测试性能结果如下（点开可看大图）：



从读的测试来看，可以看到，读的tps随cache命中率降低会下降的比较厉害，命中率为90%时、客户端线程数为250时，此时的响应时间和tps是比较不错的状况。

在随机读的测试中，以及我们的一些项目的测试中，看到的一些现象和问题：

- 1、随机读的tps随着命中率下降，下降的有点太快，具体原因还在查找和分析中；
- 2、当命中率很低时，读bloomfilter的索引信息需要耗费掉比较多的时间，主要原因是bloomfilter的索引信息并没有在cache优先级中占优，这是一个可以改进的点。

对读的速度影响比较大的因素主要是：请求次数的分布均衡、StoreFile数量、BloomFilter是否打开、Cache大小以及命中率。

ps: 强烈推荐同事的blog，其中记录了很多我们对HBase的改进，以及我们在运维HBase项目时碰到的各种奇怪、诡异的问题。

来源：blog.bluedavy.com