HBase 性能优化



# 修改Linux最大文件数

Linux系统最大可打开文件数一般默认的参数值是1024，如果你不进行修改并发量上来的时候会出现“Too Many Open Files”的错误，导致整个HBase不可运行

查看： ulimit -a 结果：open files (-n) 1024

临时修改： ulimit -n 4096

持久修改：

vi /etc/security/limits.conf在文件最后加上：

\* soft nofile 65535

\* hard nofile 65535

\* soft nproc 65535

\* hard nproc 65535

# 修改 JVM 配置

修改hbase-env.sh文件中的配置参数  
HBASE\_HEAPSIZE 4000 #HBase使用的 JVM 堆的大小  
HBASE\_OPTS "‐server ‐XX:+UseConcMarkSweepGC"JVM #GC 选项

参数解释：

**-client，-server**

这两个参数用于设置虚拟机使用何种运行模式，client模式启动比较快，但运行时性能和内存管理效率不如server模式，通常用于客户端应用程序。相反，server模式启动比client慢，但可获得更高的运行性能。

‐XX:+UseConcMarkSweepGC：设置为并发收集

# 修改HBase配置：hbase-site.xml

## zookeeper.session.timeout

默认值：3分钟（180000ms）,可以改成1分钟  
说明：RegionServer与Zookeeper间的连接超时时间。当超时时间到后，ReigonServer会被Zookeeper从RS集群清单中移除，HMaster收到移除通知后，会对这台server负责的regions重新balance，让其他存活的RegionServer接管.

调优：  
这个timeout决定了RegionServer是否能够及时的failover。设置成1分钟或更低，可以减少因等待超时而被延长的failover时间。  
不过需要注意的是，对于一些Online应用，RegionServer从宕机到恢复时间本身就很短的（网络闪断，crash等故障，运维可快速介入），如果调低timeout时间，反而会得不偿失。因为当ReigonServer被正式从RS集群中移除时，HMaster就开始做balance了（让其他RS根据故障机器记录的WAL日志进行恢复）。当故障的RS在人工介入恢复后，这个balance动作是毫无意义的，反而会使负载不均匀，给RS带来更多负担。特别是那些固定分配regions的场景。

## hbase.regionserver.handler.count

默认值：10  
说明：RegionServer的请求处理IO线程数。  
调优：  
这个参数的调优与内存息息相关。  
较少的IO线程，适用于处理单次请求内存消耗较高的Big PUT场景（大容量单次PUT或设置了较大cache的scan，均属于Big PUT）或ReigonServer的内存比较紧张的场景。  
较多的IO线程，适用于单次请求内存消耗低，TPS（吞吐量）要求非常高的场景。

## hbase.hregion.max.filesize

默认值：256M  
说明：在当前ReigonServer上单个Reigon的最大存储空间，单个Region超过该值时，这个Region会被自动split成更小的region。  
调优：  
小region对split和compaction友好，因为拆分region或compact小region里的storefile速度很快，内存占用低。缺点是split和compaction会很频繁。  
特别是数量较多的小region不停地split, compaction，会导致集群响应时间波动很大，region数量太多不仅给管理上带来麻烦，甚至会引发一些Hbase的bug。  
一般512以下的都算小region。  
大region，则不会经常split和compaction，因为做一次compact和split会产生较长时间的停顿，对应用的读写性能冲击非常大。

## hfile.block.cache.size

默认值：0.2  
说明：storefile的读缓存占用内存的大小百分比，0.2表示20%。该值直接影响数据读的性能。  
调优：当然是越大越好，如果写比读少很多，开到0.4-0.5也没问题。如果读写较均衡，0.3左右。如果写比读多，果断默认吧。  
HBase上Regionserver的内存分为两个部分，一部分作为Memstore，主要用来写；另外一部分作为BlockCache，主要用于读。  
写请求会先写入Memstore，Regionserver会给每个region提供一个Memstore，当Memstore满64MB以后，会启动 flush刷新到磁盘。  
读请求先到Memstore中查数据，查不到就到BlockCache中查，再查不到就会到磁盘上读，并把读的结果放入BlockCache。由于BlockCache采用的是LRU策略（Least Recently Used 近期最少使用算法），因此BlockCache达到上限(heapsize \* hfile.block.cache.size \* 0.85)后，会启动淘汰机制，淘汰掉最老的一批数据。  
一个Regionserver上有一个BlockCache和N个Memstore，它们的大小之和不能大于等于内存 \* 0.8，否则HBase不能启动。默认BlockCache为0.2，而Memstore为0.4。对于注重读响应时间的系统，可以将 BlockCache设大些，比如设置BlockCache=0.4，Memstore=0.39，以加大缓存的命中率。

## hbase.hregion.memstore.block.multiplier

默认值：2  
说明：当一个region里的memstore占用内存大小超过hbase.hregion.memstore.flush.size两倍的大小时，block该region的所有请求，进行flush，释放内存。  
虽然我们设置了region所占用的memstores总内存大小，比如64M，但想象一下，在最后63.9M的时候，我Put了一个200M的数据，此时memstore的大小会瞬间暴涨到超过预期的hbase.hregion.memstore.flush.size的几倍。这个参数的作用是当memstore的大小增至超过hbase.hregion.memstore.flush.size 2倍时，block所有请求，遏制风险进一步扩大。  
调优： 这个参数的默认值还是比较靠谱的。如果你预估你的正常应用场景（不包括异常）不会出现突发写或写的量可控，那么保持默认值即可。