1. Hbase减少节点：./bin/hbase-daemon.sh stop regionserver

若关闭节点时负载均衡还在运行，则在负载均衡和master恢复刚才下线的region服务器之间会产生竞争，若要避免，可设置 hbase>balance\_switch false

重新启用hbase>balance\_switch true

Region服务器会先将其管理的所有region关闭，再把自己的进程停止，region服务器在zookeeper中对应的临时节点会过期，master会注意到regionserver停止，并将其按崩溃处理，并将region重新分配到其他机器上

或采用让服务器逐渐减少负载并停止服务的方法，graceful\_stop.sh脚本

$ ./bin/graceful\_stop.sh hostname

1. 滚动重启

$ for I in `cat conf/regionservers | sort` ; do ./bin/graceful\_stop.sh \

--restart –reload –debug $i ; done &> /tmp/log.txt &

可以tail该文件查看进度

1. 新增服务器

修改配置文件，regionservers中添加机器ip，通过master节点运行start-hbase.sh即可。

1. 导入导出数据

Hbase发布了一些工具，有两个支持导入和导出的mapreduce作业

$ hadoop jar $HBASE\_HOME/hbase-0.91.0-SNAPSHOT.jar export

例如：$ hadoop jar $HBASE\_HOME/HBase-0.91.0-SNAPSHOT.jar export \

testable /user/larsgeorge/backup-testtable 就会启动mapreduce作业，并显示进度

$ hadoop jar $HBASE\_HOME/HBase-0.91.0-SNAPSHOT.jar import \

testable /user/larsgeorge/backup-testtable

优化：

1. 垃圾回收优化

并发模式失败（可能内存不足，或者碎片过多）

第一种失败的模式，是简单的并发模式失败。最好的一个例子：假设有一个8GB堆，已经使用了7GB。当CMS的收集开始第一阶段，它欢快的隆隆的做 着并发标记。与此同时，有更多的数据被分配到老年代。如果老年代增长的速度太快，在CMS完成第一阶段标记工作之前就填满了全部老年代。这时候因为没有自 由空间，CMS就无法工作！CMS必须放弃并行工作，并回落到停止世界（stop-the-world）单线程复制收集算法。此算法开始搬迁堆，检查所有 活动对象，并释放了所有死角。长时间的停顿后，该程序可能会继续。

 但我们可以很容易的通过调整JVM参数避免并发模式失败：我们只需要鼓励CMS提前开始工作！设置 -XX：CMSInitiatingOccupancyFraction = N，其中N是堆在开始收集过程中的百分比。HBase仔细的计算了内存使用，以保持其只使用60％的堆空间，所以我们通常将此值设置为大约70。（译者 注：同时也可以考虑Old区的30%要比Young区大，这样即使Young区在CMS之前全部搬迁到Old区也不会把Old区填满）

一个实验：测量碎片

 我们将运行一个实验证实这一假设。第一步是收集一些有关堆的碎片信息。在探查OpenJDK源代码后，我发现鲜为人知的参数的 -XX：PrintFLSStatistics = 1（译者注：JDK1.6也支持该选项），结合其他详细GC日志记录选项时，会导致CMS之前和之后每打印有关其自由空间的统计信息。特别是，我们关心的 指标是：

* free space - 老年代的空闲内存的总数
* NUM块 - 非连续的内存空闲块总数
* 最大块大小 - 空闲块的最大大小

我启用了这个选项，启动了一个集群，然后运行了Yahoo Cloud Serving Benchmark（YCSB）的三个独立的压力测试：

只写：每行10列，每列100个字节，1亿个row key。

只读（有缓存替换）：随机读取1亿不同的行键的数据，使数据不能完全存储在适LRU缓存。

只读（无缓存替换）：随机读取1万不同的行键的数据，使数据完全符合LRU缓存。

每个压力测试将运行至少一个小时，这样我们可以收集GC行为数据。这个实验的目标是首先要验证我们的假说，暂停是由碎片引起的，第二，以确定造成这些问题的主要原因是读取路径（包括LRU缓存）还是写路径（包

1. 本地memstore分配：0.9版本引入机制缓解region服务器内存碎片问题，这个问题主要是memstore的扰动造成的（不断创建和释放内存空间），本地Memstore分配缓冲区（Memstore-Local Allocation Buffers,MSLAB）

Memstore刷新到磁盘，会在老生代的堆上产生孔洞，由于碎片过多导致没有连续的空间分配，JRE会退回到使应用程序停止垃圾回收器，会导致其重写整个堆空间并压缩剩余的可用对象，减少压缩回收的关键是减少碎片，MSLAB就是为此设计的，关键在于只允许从堆中留下固定大小的孔洞，之后调用相同大小的新对象将会重新使用这些孔洞，就不需要应用程序停止压缩回收了。

1. 压缩
2. 优化拆分和合并
3. 负载均衡

===================================================

**scanner规范：**全表扫描一般不会用，数据量大的时候会死人的。。  
TIMERANGE，   
FILTER，   
LIMIT，   
STARTROW（start row），   
STOPROW（stop row），   
ROWPREFIXFILTER（row prefix filter,行前缀）   
TIMESTAMP，   
MAXLENGTH，   
or COLUMNS，   
CACHE，   
or RAW，   
VERSIONS

scan ‘qy’,{COLUMNS=>[‘name’,’foo’],LIMIT=>1}

scan ‘qy’,{TIMERANGE=>[1448045892646,1448045892647]}

scan ‘qy’,{FILTER=>”PrefixFilter(‘001’)”} rowkey前缀过滤

scan ‘qy’,{FILTER=>”PrefixFilter(‘t’) AND QualifierFilter(>=,’binary:b’)”}

scan ‘qy’,{FILTER=>”TimestampsFilter(1448069941270,1548069941230)” }

shell脚本中date+%s获取当前时间戳

hbase web localhost:60010





