**1**[**Hbase**](http://lib.csdn.net/base/hbase)**日常运维**

**1.1       监控Hbase运行状况**

**1.1.1**[**操作系统**](http://lib.csdn.net/base/operatingsystem)

**1.1.1.1 IO**

a.    群集网络IO，磁盘IO，HDFS IO

IO越大说明文件读写操作越多。当IO突然增加时，有可能：1.compact队列较大，集群正在进行大量压缩操作。

2.正在执行mapreduce作业

可以通过CDH前台查看整个集群综合的数据或进入指定机器的前台查看单台机器的数据：

b.    Iowait

磁盘IO对集群的影响比较大，如果io wait时间过长需检查系统或磁盘是否有异常。通常IO增加时io wait也会增加，现在FMS的机器正常情况io wait在50ms以下

跟主机相关的指标可以在CDH前台左上角先点“主机”选项卡然后选要查看的主机：

**1.1.1.2 CPU**

        如果CPU占用过高有可能是异常情况引起集群资源消耗，可以通过其他指标和日志来查看集群正在做什么。

**1.1.1.3 内存**

**1.1.2**[**Java**](http://lib.csdn.net/base/java)

GC 情况

        regionserver长时间GC会影响集群性能并且有可能会造成假死的情况

**1.1.3       重要的hbase指标**

**1.1.3.1 region情况**

需要检查

1. region的数量（总数和每台regionserver上的region数）

2. region的大小

如果发现异常可以通过手动merge region和手动分配region来调整

从CDH前台和master前台以及regionServer的前台都可以看到region数量，如master前台：

在regionserver前台可以看到storeFile大小：

**1.1.3.2 缓存命中率**

缓存命中率对hbase的读有很大的影响，可以观察这个指标来调整blockcache的大小。

从regionserverweb页面可以看到block cache的情况：

**1.1.3.3 读写请求数**

通过读写请求数可以大概看出每台regionServer的压力，如果压力分布不均匀，应该检查regionServer上的region以及其它指标

master web上可以看到所以regionServer的读写请求数

regionServer上可以看到每个region的读写请求数

**1.1.3.4 压缩队列**

压缩队列存放的是正在压缩的storefile，compact操作对hbase的读写影响较大

通过cdh的hbase图表库可以看到集群总的压缩队列大小：

可以通过CDH的hbase主页查询compact日志：

点击“压缩”进入：

**1.1.3.5 刷新队列**

单个region的memstore写满(128M)或regionServer上所有region的memstore大小总合达到门限时会进行flush操作,flush操作会产生新的storeFile

同样可以通过CDH的hbase前台查看flush日志：

**1.1.3.6 rpc调用队列**

没有及时处理的rpc操作会放入rpc操作队列，从rpc队列可以看出服务器处理请求的情况

**1.1.3.7 文件块保存在本地的百分比**

datanode和regionserver一般都部署在同一台机器上，所以region server管理的region会优先存储在本地，以节省网络开销。如果block locality较低有可能是刚做过balance或刚重启，经过compact之后region的数据都会写到当前机器的datanode，block locality也会慢慢达到接近100：

**1.1.3.8 内存使用情况**

内存使用情况,主要可以看used Heap和memstore的大小，如果usedHeadp一直超过80-85%以上是比较危险的

memstore很小或很大也不正常

从regionServer的前台可以看到：

**1.1.3.9 slowHLogAppendCount**

写HLog过慢（>1s）的操作次数，这个指标可以作为HDFS状态好坏的判断

在regionServer前台查看：

**1.1.4       CDH检查日志**

CDH有强大的系统事件和日志搜索功能，每一个服务(如：[**Hadoop**](http://lib.csdn.net/base/hadoop),hbase)的主页都提供了事件和告警的查询，日常运维除了CDH主页的告警外，需要查看这些事件以发现潜在的问题：

选择“事件搜索”中的标签（“警报”、“严重”）可以进入相关的事件日志，如“严重”：

**1.2       检查数据一致性以及修复方法**

数据一致性是指：

1.     每个region都被正确的分配到一台regionserver上，并且region的位置信息及状态都是正确的。

2.     每个table都是完整的，每一个可能的rowkey 都可以对应到唯一的一个region.

**1.2.1       检查**

**hbase hbck**

注：有时集群正在启动或region正在做split操作，会造成数据不一致

**hbase hbck -details**

加上–details会列出更详细的检查信息，包括所以正在进行的split任务

**hbase hbck Table1 Table2**

如果只想检查指定的表，可以在命令后面加上表名，这样可以节省操作时间

**CDH**

通过CDH提供的检查报告也可以看到hbck的结果，日常只需要看CDH hbck的报告即可：

选择“最近的Hbck结果”：

**1.2.2       修复**

**1.2.2.1 局部的修复**

如果出现数据不一致，修复时要最大限度的降低可能出现的风险，使用以下命令对region进行修复风险较低：

**1.2.2.1.1       hbase hbck -fixAssignments**

修复region没有分配(unassigned)，错误分配（incorrectly assigned）以及多次分配（multiply assigned）的问题

**1.2.2.1.2       hbase hbck -fixMeta**

删除META表里有记录但HDFS里没有数据记录的region

添加HDFS里有数据但是META表里没有记录的region到META表

**1.2.2.1.3       hbase hbck -repairHoles**

等价于：hbase hbck -fixAssignments -fixMeta -fixHdfsHoles

-fixHdfsHoles的作用：

如果rowkey出现空洞，即相邻的两个region的rowkey不连续，则使用这个参数会在HDFS里面创建一个新的region。创建新的region之后要使用-fixMeta和-fixAssignments参数来使用挂载这个region，所以一般和前两个参数一起使用

**1.2.2.2 Region重叠修复**

进行以下操作非常危险，因为这些操作会修改文件系统，需要谨慎操作！

进行以下操作前先使用**hbck –details**查看详细问题，如果需要进行修复先**停掉应用**，如果执行以下命令时同时有数据操作可能会造成不可期的异常。

**1.2.2.2.1       hbase hbck -fixHdfsOrphans**

将文件系统中的没有metadata文件(.regioninfo)的region目录加入到hbase中，即创建.regioninfo目录并将region分配到regionser

**1.2.2.2.2       hbase hbck -fixHdfsOverlaps**

通过两种方式可以将rowkey有重叠的region合并：

1.merge:将重叠的region合并成一个大的region

2.sideline:将region重叠的部分去掉，并将重叠的数据先写入到临时文件，然后再导入进来。

如果重叠的数据很大，直接合并成一个大的region会产生大量的split和compact操作，可以通过以下参数控制region过大：

-maxMerge <n> 合并重叠region的最大数量

-sidelineBigOverlaps 假如有大于maxMerge个数的 region重叠, 则采用sideline方式处理与其它region的重叠.

-maxOverlapsToSideline <n> 如果用sideline方式处理重叠region,最多sideline n个region .

**1.2.2.2.3       hbase hbck -repair**

      以下命令的缩写：

hbase hbck -fixAssignments -fixMeta -fixHdfsHoles -fixHdfsOrphans -fixHdfsOverlaps -fixVersionFile –sidelineBigOverlaps

     可以指定表名：

hbase hbck -repair Table1 Table2

**1.2.2.2.4       hbase hbck -fixMetaOnly –fixAssignments**

如果只有META表的region不一致，则可以使用这个命令修复

**1.2.2.2.5       hbase hbck –fixVersionFile**

  Hbase的数据文件启动时需要一个version file，如果这个文件丢失，可以用这个命令来新建一个，但是要保证hbck的版本和Hbase集群的版本是一样的

**1.2.2.2.6       hbase org.apache.hadoop.hbase.util.hbck.OfflineMetaRepair**

如果ROOT表和META表都出问题了Hbase无法启动，可以用这个命令来创建新的ROOT和META表。

这个命令的前提是Hbase已经关闭，执行时它会从hbase的home目录加载hbase的相关信息（.regioninfo），如果表的信息是完整的就会创建新的root和meta目录及数据

**1.2.2.2.7       hbase hbck –fixSplitParents**

当region做split操作的时候，父region会被自动清除掉。但是有时候子region在父region被清除之前又做了split。造成有些延迟离线的父region存在于META表和HDFS中，但是没有部署，HBASE又不能清除他们。这种情况下可以使用此命令重置这些在META表中的region为在线状态并且没有split。然后就可以使用之前的修复命令把这个region修复

**1.3       手动merge region**

进行操作前先将balancer关闭，操作完成后再打开balancer

经过一段时间的运行之后有可能会产生一些很小的region，需要定期检查这些region并将它们和相邻的region合并以减少系统的总region数，减少管理开销

      合并方法：

1.找到需要合并的region的encoded name

2.进入hbase shell

3.执行merge\_region‘region1’,’region2’

**1.4       手动分配region**

如果发现台regionServer资源占用特别高，可以检查这台regionserver上的region是否存在过多比较大的region,通过hbase shell将部分比较大的region分配给其他不是很忙的regions server：

move ‘regionId’,’serverName’

例：

move '54fca23d09a595bd3496cd0c9d6cae85','vmcnod05,60020,1390211132297'

**1.5       手动major\_compact**

进行操作前先将balancer关闭，操作完成后再打开balancer

选择一个系统比较空闲的时间手工major\_compact，如果hbase更新不是太频繁，可以一个星期对所有表做一次 major\_compact，这个可以在做完一次major\_compact后，观看所有的storefile数量，如果storefile数量增加到 major\_compact后的storefile的近二倍时，可以对所有表做一次major\_compact，时间比较长，操作尽量避免高锋期

注：fms现在生产上开启了自动major\_compact，不需要做手动major compact

**1.6       balance\_switch**

balance\_switchtrue 打开balancer

balance\_switchflase 关闭balancer

配置master是否执行平衡各个regionserver的region数量，当我们需要维护或者重启一个regionserver时，会关闭balancer，这样就使得region在regionserver上的分布不均，这个时候需要手工的开启balance。

**1.7       regionserver重启**

graceful\_stop.sh --restart --reload--debug nodename

进行操作前先将balancer关闭，操作完成后再打开balancer

这个操作是平滑的重启regionserver进程，对服务不会有影响，他会先将需要重启的regionserver上面的所有 region迁移到其它的服务器，然后重启，最后又会将之前的region迁移回来，但我们修改一个配置时，可以用这种方式重启每一台机子，对于hbaseregionserver重启，不要直接kill进程，这样会造成在zookeeper.session.timeout这个时间长的中断，也不要通过bin/hbase-daemon.sh stop regionserver去重启，如果运气不太好，-ROOT-或者.META.表在上面的话，所有的请求会全部失败

**1.8       regionserver关闭下线**

bin/graceful\_stop.sh nodename

进行操作前先将balancer关闭，操作完成后再打开balancer

和上面一样，系统会在关闭之前迁移所有region，然后stop进程。

**1.9       flush表**

所有memstore刷新到hdfs，通常如果发现regionserver的内存使用过大，造成该机的 regionserver很多线程block，可以执行一下flush操作，这个操作会造成hbase的storefile数量剧增，应尽量避免这个操 作，还有一种情况，在hbase进行迁移的时候，如果选择拷贝文件方式，可以先停写入，然后flush所有表，拷贝文件

**1.10  Hbase迁移**

**1.10.1   copytable方式**

bin/hbaseorg.apache.hadoop.hbase.mapreduce.CopyTable--peer.adr=zookeeper1,zookeeper2,zookeeper3:/hbase 'testtable'

  这个操作需要添加hbase目录里的conf/mapred-site.xml，可以复制hadoop的过来。

1.10.2  **Export/Import**

 bin/hbaseorg.apache.hadoop.hbase.mapreduce.Export testtable /user/testtable[versions] [starttime] [stoptime]

bin/hbaseorg.apache.hadoop.hbase.mapreduce.Import testtable  /user/testtable

**1.10.3   直接拷贝hdfs对应的文件**

首先拷贝hdfs文件，如bin/hadoop distcp hdfs://srcnamenode:9000/hbase/testtable/hdfs://distnamenode:9000/hbase/testtable/

然后在目的hbase上执行bin/hbase org.jruby.Main bin/add\_table.rb /hbase/testtable

生成meta信息后，重启hbase

**2        Hadoop日常运维**

**2.1      监控Hadoop运行状况**

1.nameNode、ResourseManager内存(namenode要有足够内存)

2.DataNode和NodeManager运行状态

3.磁盘使用情况

4.服务器负载状态

**2.2      检查HDFS文件健康状况**

命令：hadoop fsck

2.3      **开启垃圾箱(trash)功能**

trash功能它默认是关闭的，开启后，被你删除的数据将会mv到操作用户目录的".Trash"文件夹，可以配置超过多长时间，系统自动删除过期数据。这样一来，当操作失误的时候，可以把数据mv回来