**** **HBase 官方配置**

Copyright © 2012 Apache Software Foundation。保留所有权利。 Apache Hadoop, Hadoop, MapReduce, HDFS,Zookeeper, HBase 及 HBase项目 logo 是Apache Software Foundation的商标。

作者：陈少钦 于2018/01/08 23:53修改完毕

**摘要**

HBase是一个分布式，版本化，面向列的数据库，构建在 Apache Hadoop和 Apache ZooKeeper之上。该文档截取自《HBase 官方文档中文版》的配置部份，并经过实际验证作适当修改。实际测试中各主键版本及环境采用如下：

|  |  |
| --- | --- |
| **环境或主键** | **版本** |
| Linux | CentOS 6.7 64位 |
| Jdk | Jdk 1.8.0\_151 |
| Hadoop | hadoop-2.6.5 |
| Zookeeper | zookeeper-3.4.5 |
| Hbase | hbase-1.2.6 |

# 配置

本章是慢速开始配置指导。

HBase有如下需要，请仔细阅读本章节以确保所有的需要都被满足。如果需求没有能满足，就有可能遇到莫名其妙的错误甚至丢失数据。

HBase使用和Hadoop一样配置系统。要配置部署，编辑conf/hbase-env.sh文件中的环境变量——该配置文件主要启动脚本用于获取已启动的集群——然后增加配置到XML文件，如同覆盖HBase缺省配置，告诉HBase用什么文件系统， 全部ZooKeeper位置。

在分布模式下运行时，在编辑HBase配置文件之后，确认将conf目录复制到集群中的每个节点。HBase不会自动同步。使用**rsync**.

小心编辑XML。确认关闭所有元素。采用 **xmllint** 或类似工具确认文档编辑后是良好格式化的。

* 1. **基础条件**

This section lists required services and some required system configuration.

以下条件已在前面的文档配置完成。

* + 1. **java**

和Hadoop一样，HBase需要Oracle版本的Java.除了那个有问题的u18版本其他的都可以用，最好用最新的。

* + 1. **操作系统**
       1. **ssh**

必须安装**ssh** ， **sshd** 也必须运行，这样Hadoop的脚本才可以远程操控其他的Hadoop和HBase进程。ssh之间必须都打通，不用密码都可以登录，详细方法可以Google一下 ("ssh passwordless login").

* + - 1. **DNS**

HBase使用本地 hostname 才获得IP地址. 正反向的DNS都是可以的.

如果你的机器有多个接口，HBase会使用hostname指向的主接口.

如果还不够，你可以设置 hbase.regionserver.dns.interface 来指定主接口。当然你的整个集群的配置文件都必须一致，每个主机都使用相同的网络接口

还有一种方法是设置 hbase.regionserver.dns.nameserver来指定nameserver，不使用系统带的.

* + - 1. **Loopback IP**

HBase expects the loopback IP address to be 127.0.0.1. Ubuntu and some other distributions, for example, will default to 127.0.1.1 and this will cause problems for you.

/etc/hosts should look something like this:

127.0.0.1 localhost

127.0.0.1 ubuntu.ubuntu-domain ubuntu

* + - 1. **NTP**

集群的时钟要保证基本的一致。稍有不一致是可以容忍的，但是很大的不一致会造成奇怪的行为。 运行 [NTP](http://en.wikipedia.org/wiki/Network_Time_Protocol) 或者其他什么东西来同步你的时间.

如果你查询的时候或者是遇到奇怪的故障，可以检查一下系统时间是否正确!

* + - 1. **ulimit** **和 nproc**

HBase是数据库，会在同一时间使用很多的文件句柄。大多数linux系统使用的默认值1024是不能满足的，会导致FAQ: Why do I see "java.io.IOException...(Too many open files)" in my logs?异常。还可能会发生这样的异常

2010-04-06 03:04:37,542 INFO org.apache.hadoop.hdfs.DFSClient: Exception increateBlockOutputStream java.io.EOFException

2010-04-06 03:04:37,542 INFO org.apache.hadoop.hdfs.DFSClient: Abandoning block blk\_-6935524980745310745\_1391901

所以你需要修改你的最大文件句柄限制。可以设置到10k。大致的数学运算如下：每列族至少有1个存储文件(StoreFile) 可能达到5-6个如果区域有压力。将每列族的存储文件平均数目和每区域服务器的平均区域数目相乘。例如：假设一个模式有3个列族，每个列族有3个存储文件，每个区域服务器有100个区域，JVM 将打开3 \* 3 \* 100 = 900 个文件描述符(不包含打开的jar文件，配置文件等)

你还需要修改 hbase 用户的 nproc，在压力下，如果过低会造成 OutOfMemoryError异常。

需要澄清的，这两个设置是针对操作系统的，不是HBase本身的。有一个常见的错误是HBase运行的用户，和设置最大值的用户不是一个用户。在HBase启动的时候，第一行日志会现在ulimit信息，确保其正确。

* + - * 1. **在Ubuntu上设置ulimit**

如果你使用的是Ubuntu,你可以这样设置:

在文件 /etc/security/limits.conf 添加一行，如:

hadoop - nofile 32768

可以把 hadoop 替换成你运行HBase和Hadoop的用户。如果你用两个用户，你就需要配两个。还有配nproc hard 和 soft limits. 如:

hadoop soft/hard nproc 32000

在 /etc/pam.d/common-session 加上这一行:

session required pam\_limits.so

否则在 /etc/security/limits.conf上的配置不会生效.

还有注销再登录，这些配置才能生效!

* + - 1. **Windows**

HBase没有怎么在Windows下测试过。所以不推荐在Windows下运行.

如果你实在是想运行，需要安装[Cygwin](http://cygwin.com/) 并虚拟一个unix环境.详情请看 [Windows 安装指导](http://hbase.apache.org/cygwin.html) . 或者 [搜索邮件列表](http://search-hadoop.com/?q=hbase+windows&fc_project=HBase&fc_type=mail+_hash_+dev)找找最近的关于windows的注意点

* + 1. **hadoop**

选择 Hadoop 版本对HBase部署很关键。下表显示不同HBase支持的Hadoop版本信息。基于HBase版本，应该选择合适的Hadoop版本。我们没有绑定 Hadoop 发行版选择。可以从Apache使用 Hadoop 发行版，或了解一下Hadoop发行商产品： <http://wiki.apache.org/hadoop/Distributions%20and%20Commercial%20Support>

**Table 2.1. Hadoop version support matrix**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **HBase-0.92.x** | **HBase-0.94.x** | **HBase-0.96** |
| Hadoop-0.20.205 | S | X | X |
| Hadoop-0.22.x | S | X | X |
| Hadoop-1.0.x | S | S | S |
| Hadoop-1.1.x | NT | S | S |
| Hadoop-0.23.x | X | S | NT |
| Hadoop-2.x | X | S | S |

S = supported and tested,支持

X = not supported,不支持

NT = not tested enough.可以运行但测试不充分

由于 HBase 依赖 Hadoop，它配套发布了一个Hadoop jar 文件在它的 lib 下。该套装jar仅用于独立模式。在分布式模式下，Hadoop版本必须和HBase下的版本一致。用你运行的分布式Hadoop版本jar文件替换HBase lib目录下的Hadoop jar文件，以避免版本不匹配问题。确认替换了集群中所有HBase下的jar文件。Hadoop版本不匹配问题有不同表现，但看起来都像挂掉了。

* + - 1. **Hadoop 安全性**

HBase运行在Hadoop 0.20.x上，就可以使用其中的安全特性 -- 只要你用这两个版本0.20S 和CDH3B3，然后把hadoop.jar替换掉就可以了.

* + - 1. **dfs.datanode.max.xcievers**

一个 Hadoop HDFS Datanode 有一个同时处理文件的上限. 这个参数叫 xcievers (Hadoop的作者把这个单词拼错了). 在你加载之前，先确认下你有没有配置这个文件conf/hdfs-site.xml里面的xceivers参数，至少要有4096:

<property>

<name>dfs.datanode.max.xcievers</name>

<value>4096</value>

</property>

对于HDFS修改配置要记得重启.

如果没有这一项配置，你可能会遇到奇怪的失败。你会在Datanode的日志中看到xcievers exceeded，但是运行起来会报 missing blocks错误。例如: 10/12/08 20:10:31 INFO hdfs.DFSClient: Could not obtain block blk\_XXXXXXXXXXXXXXXXXXXXXX\_YYYYYYYY from any node: java.io.IOException: No live nodes contain current block. Will get new block locations from namenode and retry... [[5](http://abloz.com/hbase/book.html" \l "ftn.d613e474)]

* 1. **HBase运行模式:单机和分布式**

HBase有两个运行模式: [“单机模式”](http://abloz.com/hbase/book.html#standalone) 和 [“分布式模式”](http://abloz.com/hbase/book.html#distributed). 默认是单机模式，如果要分布式模式你需要编辑 conf 文件夹中的配置文件.

不管是什么模式，你都需要编辑 conf/hbase-env.sh来告知HBase **java**的安装路径.在这个文件里你还可以设置HBase的运行环境，诸如 heapsize和其他 JVM有关的选项, 还有Log文件地址，等等. 设置 JAVA\_HOME指向 **java**安装的路径.

* + 1. **单机模式**

这是默认的模式. 在单机模式中，HBase使用本地文件系统，而不是HDFS ，所有的服务和zooKeeper都运作在一个JVM中。zookeep监听一个端口，这样客户端就可以连接HBase了。

* + 1. **分布式模式**

分布式模式分两种。*伪分布式模式*是把进程运行在一台机器上，但不是一个JVM.而*完全分布式模式*就是把整个服务被分布在各个节点上了.

分布式模式需要使用 *Hadoop Distributed File System* (HDFS).可以参见 [HDFS需求和指导](http://hadoop.apache.org/common/docs/current/api/overview-summary.html#overview_description)来获得关于安装HDFS的指导。在操作HBase之前，你要确认HDFS可以正常运作。

在我们安装之后，你需要确认你的*伪分布式模式*或者 *完全分布式模式*的配置是否正确。这两个模式可以使用同一个验证脚本Section 1.2.3, “运行和确认你的安装”。

* + - 1. **伪分布式模式**

略

* + - 1. **完全分布式模式**

要想运行完全分布式模式，你要进行如下配置，先在 hbase-site.xml, 加一个属性 hbase.cluster.distributed 设置为 true 然后把 hbase.rootdir 设置为HDFS的NameNode的位置。 例如，你的namenode运行在bigdata服务上 你期望的目录是 /hbase,使用如下的配置

<configuration>

...

<property>

<name>hbase.rootdir</name>

<value>hdfs://bigdata/hbase</value>

<description>The directory shared by RegionServers.

</description>

</property>

<property>

<name>hbase.cluster.distributed</name>

<value>true</value>

<description>The mode the cluster will be in. Possible values are

false: standalone and pseudo-distributed setups with managed Zookeeper

true: fully-distributed with unmanaged Zookeeper Quorum (see hbase-env.sh)

</description>

</property>

...

</configuration>

* + - * 1. **regionservers**

完全分布式模式的还需要修改conf/regionservers.一行写一个host (就像Hadoop里面的 slaves 一样). 列在这里的server会随着集群的启动而启动，集群的停止而停止.

slave1

slave2

* + - * 1. **hadoop配置文件**

添加hadoop配置文件：core-site.xml和hdfs-site.xml到conf目录下。

* + - * 1. **HDFS客户端配置**

如果你希望Hadoop集群上做*HDFS 客户端配置* ，例如你的HDFS客户端的配置和服务端的不一样。按照如下的方法配置，HBase就能看到你的配置信息:

* 在hbase-env.sh里将HBASE\_CLASSPATH环境变量加上HADOOP\_CONF\_DIR 。
* 在${HBASE\_HOME}/conf下面加一个 hdfs-site.xml (或者 hadoop-site.xml) ，最好是软连接
* 如果你的HDFS客户端的配置不多的话，你可以把这些加到 hbase-site.xml上面.

例如HDFS的配置 dfs.replication.你希望复制5份，而不是默认的3份。如果你不照上面的做的话，HBase只会复制3份。

* + 1. **运行和确认你的安装**

首先确认你的HDFS是运行着的。你可以运行HADOOP\_HOME中的 bin/start-hdfs.sh 来启动HDFS.你可以通过**put**命令来测试放一个文件，然后有**get**命令来读这个文件。通常情况下HBase是不会运行mapreduce的。所以比不需要检查这些。

*如果*你自己管理ZooKeeper集群，你需要确认它是运行着的。如果是HBase托管，ZoopKeeper会随HBase启动。

用如下命令启动HBase:

bin/start-hbase.sh

这个脚本在HBASE\_HOME目录里面。

你现在已经启动HBase了。HBase把log记在 logs 子目录里面. 当HBase启动出问题的时候，可以看看Log.

HBase也有一个界面，上面会列出重要的属性。默认是在Master的60010端口上H (HBase RegionServers 会默认绑定 60020端口，在端口60030上有一个展示信息的界面 ).如果Master运行在 master，端口是默认的话，你可以用浏览器在 http://master:60010看到主界面. .

一旦HBase启动，可以建表，插入数据，scan你的表，还有disable这个表，最后把它删掉。

可以在HBase Shell停止HBase

$ ./bin/stop-hbase.sh

stopping hbase...............

停止操作需要一些时间，你的集群越大，停的时间可能会越长。如果你正在运行一个分布式的操作，要确认在HBase彻底停止之前，Hadoop不能停.

* 1. **配置文件**

HBase的配置系统和Hadoop一样。在conf/hbase-env.sh配置系统的部署信息和环境变量。 -- 这个配置会被启动shell使用 -- 然后在XML文件里配置信息，覆盖默认的配置。告知HBase使用什么目录地址，ZooKeeper的位置等等信息。

当你使用分布式模式的时间，当你编辑完一个文件之后，记得要把这个文件复制到整个集群的conf 目录下。HBase不会帮你做这些，你得用 **rsync**.

* + 1. **hbase-site.xml 和 hbase-default.xml**

正如Hadoop放置HDFS的配置文件hdfs-site.xml，HBase的配置文件是 conf/hbase-site.xml. 你可以在 Section 1.3.1.1, “HBase 默认配置”找到配置的属性列表。你也可以看有代码里面的hbase-default.xml文件，他在src/main/resources目录下。

不是所有的配置都在 hbase-default.xml出现.只要改了代码，配置就有可能改变，所以唯一了解这些被改过的配置的办法是读源代码本身。

要注意的是，要重启集群才能是配置生效。

* + - 1. **HBase 默认配置**

**HBase 默认配置**

该文档是用hbase默认配置文件生成的，文件源是 hbase-default.xml

hbase.rootdir

这个目录是region server的共享目录，用来持久化HBase。URL需要是'完全正确'的，还要包含文件系统的scheme。例如，要表示hdfs中的'/hbase'目录，namenode 运行在bigdata服务上。则需要设置为hdfs://bigdata/hbase。默认情况下HBase是写到/tmp的。不改这个配置，数据会在重启的时候丢失。

默认: file:///tmp/hbase-${user.name}/hbase

hbase.master.port

HBase的Master的端口.

默认: 60000

hbase.cluster.distributed

HBase的运行模式。false是单机模式，true是分布式模式。若为false,HBase和Zookeeper会运行在同一个JVM里面。

默认: false

hbase.tmp.dir

本地文件系统的临时文件夹。可以修改到一个更为持久的目录上。(/tmp会在重启时清楚)

默认:${java.io.tmpdir}/hbase-${user.name}

hbase.local.dir

作为本地存储，位于本地文件系统的路径。

默认: ${hbase.tmp.dir}/local/

hbase.master.info.port

HBase Master web 界面端口. 设置为-1 意味着你不想让他运行。

默认: 60010

hbase.master.info.bindAddress

HBase Master web 界面绑定的端口

默认: 0.0.0.0

hbase.client.write.buffer

HTable客户端的写缓冲的默认大小。这个值越大，需要消耗的内存越大。因为缓冲在客户端和服务端都有实例，所以需要消耗客户端和服务端两个地方的内存。得到的好处是，可以减少RPC的次数。可以这样估算服务器端被占用的内存： hbase.client.write.buffer \* hbase.regionserver.handler.count

默认: 2097152

hbase.regionserver.port

HBase RegionServer绑定的端口

默认: 60020

hbase.regionserver.info.port

HBase RegionServer web 界面绑定的端口 设置为 -1 意味这你不想与运行 RegionServer 界面.

默认: 60030

hbase.regionserver.info.port.auto

Master或RegionServer是否要动态搜一个可以用的端口来绑定界面。当hbase.regionserver.info.port已经被占用的时候，可以搜一个空闲的端口绑定。这个功能在测试的时候很有用。默认关闭。

默认: false

hbase.regionserver.info.bindAddress

HBase RegionServer web 界面的IP地址

默认: 0.0.0.0

hbase.regionserver.class

RegionServer 使用的接口。客户端打开代理来连接region server的时候会使用到。

默认: org.apache.hadoop.hbase.ipc.HRegionInterface

hbase.client.pause

通常的客户端暂停时间。最多的用法是客户端在重试前的等待时间。比如失败的get操作和region查询操作等都很可能用到。

默认: 1000

hbase.client.retries.number

最大重试次数。所有需重试操作的最大值。例如从root region服务器获取root region，Get单元值，行Update操作等等。这是最大重试错误的值。  默认: 10.

默认: 10

hbase.bulkload.retries.number

最大重试次数。 原子批加载尝试的迭代最大次数。 0 永不放弃。默认: 0.

默认: 0

hbase.client.scanner.caching

当调用Scanner的next方法，而值又不在缓存里的时候，从服务端一次获取的行数。越大的值意味着Scanner会快一些，但是会占用更多的内存。当缓冲被占满的时候，next方法调用会越来越慢。慢到一定程度，可能会导致超时。例如超过了hbase.regionserver.lease.period。

默认: 100

hbase.client.keyvalue.maxsize

一个KeyValue实例的最大size.这个是用来设置存储文件中的单个entry的大小上界。因为一个KeyValue是不能分割的，所以可以避免因为数据过大导致region不可分割。明智的做法是把它设为可以被最大region size整除的数。如果设置为0或者更小，就会禁用这个检查。默认10MB。

默认: 10485760

hbase.regionserver.lease.period

客户端租用HRegion server 期限，即超时阀值。单位是毫秒。默认情况下，客户端必须在这个时间内发一条信息，否则视为死掉。

默认: 60000

hbase.regionserver.handler.count

RegionServers受理的RPC Server实例数量。对于Master来说，这个属性是Master受理的handler数量

默认: 10

hbase.regionserver.msginterval

RegionServer 发消息给 Master 时间间隔，单位是毫秒

默认: 3000

hbase.regionserver.optionallogflushinterval

将Hlog同步到HDFS的间隔。如果Hlog没有积累到一定的数量，到了时间，也会触发同步。默认是1秒，单位毫秒。

默认: 1000

hbase.regionserver.regionSplitLimit

region的数量到了这个值后就不会在分裂了。这不是一个region数量的硬性限制。但是起到了一定指导性的作用，到了这个值就该停止分裂了。默认是MAX\_INT.就是说不阻止分裂。

默认: 2147483647

hbase.regionserver.logroll.period

提交commit log的间隔，不管有没有写足够的值。

默认: 3600000

hbase.regionserver.hlog.reader.impl

HLog file reader 的实现.

默认: org.apache.hadoop.hbase.regionserver.wal.SequenceFileLogReader

hbase.regionserver.hlog.writer.impl

HLog file writer 的实现.

默认: org.apache.hadoop.hbase.regionserver.wal.SequenceFileLogWriter

hbase.regionserver.nbreservationblocks

储备的内存block的数量(译者注:就像石油储备一样)。当发生out of memory 异常的时候，我们可以用这些内存在RegionServer停止之前做清理操作。

默认: 4

hbase.zookeeper.dns.interface

当使用DNS的时候，Zookeeper用来上报的IP地址的网络接口名字。

默认: default

hbase.zookeeper.dns.nameserver

当使用DNS的时候，Zookeepr使用的DNS的域名或者IP 地址，Zookeeper用它来确定和master用来进行通讯的域名.

默认: default

hbase.regionserver.dns.interface

当使用DNS的时候，RegionServer用来上报的IP地址的网络接口名字。

默认: default

hbase.regionserver.dns.nameserver

当使用DNS的时候，RegionServer使用的DNS的域名或者IP 地址，RegionServer用它来确定和master用来进行通讯的域名.

默认: default

hbase.master.dns.interface

当使用DNS的时候，Master用来上报的IP地址的网络接口名字。

默认: default

hbase.master.dns.nameserver

当使用DNS的时候，RegionServer使用的DNS的域名或者IP 地址，Master用它来确定用来进行通讯的域名.

默认: default

hbase.balancer.period

Master执行region balancer的间隔。

默认: 300000

hbase.regions.slop

当任一区域服务器有average + (average \* slop)个分区，将会执行重新均衡。默认 20% slop .

默认:0.2

hbase.master.logcleaner.ttl

Hlog存在于.oldlogdir 文件夹的最长时间, 超过了就会被 Master 的线程清理掉.

默认: 600000

hbase.master.logcleaner.plugins

LogsCleaner服务会执行的一组LogCleanerDelegat。值用逗号间隔的文本表示。这些WAL/HLog cleaners会按顺序调用。可以把先调用的放在前面。你可以实现自己的LogCleanerDelegat，加到Classpath下，然后在这里写下类的全称。一般都是加在默认值的前面。

默认: org.apache.hadoop.hbase.master.TimeToLiveLogCleaner

hbase.regionserver.global.memstore.upperLimit

单个region server的全部memtores的最大值。超过这个值，一个新的update操作会被挂起，强制执行flush操作。

默认: 0.4

hbase.regionserver.global.memstore.lowerLimit

当强制执行flush操作的时候，当低于这个值的时候，flush会停止。默认是堆大小的 35% . 如果这个值和 hbase.regionserver.global.memstore.upperLimit 相同就意味着当update操作因为内存限制被挂起时，会尽量少的执行flush(译者注:一旦执行flush，值就会比下限要低，不再执行)

默认: 0.35

hbase.server.thread.wakefrequency

service工作的sleep间隔，单位毫秒。 可以作为service线程的sleep间隔，比如log roller.

默认: 10000

hbase.server.versionfile.writeattempts

退出前尝试写版本文件的次数。每次尝试由 hbase.server.thread.wakefrequency 毫秒数间隔。

默认: 3

hbase.hregion.memstore.flush.size

当memstore的大小超过这个值的时候，会flush到磁盘。这个值被一个线程每隔hbase.server.thread.wakefrequency检查一下。

默认:134217728

hbase.hregion.preclose.flush.size

当一个region中的memstore的大小大于这个值的时候，我们又触发了close.会先运行“pre-flush”操作，清理这个需要关闭的memstore，然后将这个region下线。当一个region下线了，我们无法再进行任何写操作。如果一个memstore很大的时候，flush操作会消耗很多时间。"pre-flush"操作意味着在region下线之前，会先把memstore清空。这样在最终执行close操作的时候，flush操作会很快。

默认: 5242880

hbase.hregion.memstore.block.multiplier

如果memstore有hbase.hregion.memstore.block.multiplier倍数的hbase.hregion.flush.size的大小，就会阻塞update操作。这是为了预防在update高峰期会导致的失控。如果不设上界，flush的时候会花很长的时间来合并或者分割，最坏的情况就是引发out of memory异常。(译者注:内存操作的速度和磁盘不匹配，需要等一等。原文似乎有误)

默认: 2

hbase.hregion.memstore.mslab.enabled

体验特性：启用memStore分配本地缓冲区。这个特性是为了防止在大量写负载的时候堆的碎片过多。这可以减少GC操作的频率。(GC有可能会Stop the world)(译者注：实现的原理相当于预分配内存，而不是每一个值都要从堆里分配)

默认: true

hbase.hregion.max.filesize

最大HStoreFile大小。若某个列族的HStoreFile增长达到这个值，这个Hegion会被切割成两个。 默认: 10G.

默认:10737418240

hbase.hstore.compactionThreshold

当一个HStore含有多于这个值的HStoreFiles(每一个memstore flush产生一个HStoreFile)的时候，会执行一个合并操作，把这HStoreFiles写成一个。这个值越大，需要合并的时间就越长。

默认: 3

hbase.hstore.blockingStoreFiles

当一个HStore含有多于这个值的HStoreFiles(每一个memstore flush产生一个HStoreFile)的时候，会执行一个合并操作，update会阻塞直到合并完成，直到超过了hbase.hstore.blockingWaitTime的值

默认: 7

hbase.hstore.blockingWaitTime

hbase.hstore.blockingStoreFiles所限制的StoreFile数量会导致update阻塞，这个时间是来限制阻塞时间的。当超过了这个时间，HRegion会停止阻塞update操作，不过合并还有没有完成。默认为90s.

默认: 90000

hbase.hstore.compaction.max

每个“小”合并的HStoreFiles最大数量。

默认: 10

hbase.hregion.majorcompaction

一个Region中的所有HStoreFile的major compactions的时间间隔。默认是1天。 设置为0就是禁用这个功能。

默认: 86400000

hbase.storescanner.parallel.seek.enable

允许 StoreFileScanner 并行搜索 StoreScanner, 一个在特定条件下降低延迟的特性。

默认: false

hbase.storescanner.parallel.seek.threads

并行搜索特性打开后，默认线程池大小。

默认: 10

hbase.mapreduce.hfileoutputformat.blocksize

MapReduce中HFileOutputFormat可以写 storefiles/hfiles. 这个值是hfile的blocksize的最小值。通常在HBase写Hfile的时候，bloocksize是由table schema(HColumnDescriptor)决定的，但是在mapreduce写的时候，我们无法获取schema中blocksize。这个值越小，你的索引就越大，你随机访问需要获取的数据就越小。如果你的cell都很小，而且你需要更快的随机访问，可以把这个值调低。

默认: 65536

hfile.block.cache.size

分配给HFile/StoreFile的block cache占最大堆(-Xmx setting)的比例。默认0.25意思是分配25%，设置为0就是禁用，但不推荐。

默认:0.25

hbase.hash.type

哈希函数使用的哈希算法。可以选择两个值:: murmur (MurmurHash) 和 jenkins (JenkinsHash). 这个哈希是给 bloom filters用的.

默认: murmur

hfile.block.index.cacheonwrite

这允许在写入索引时将非根多级索引块写入到块缓存。

默认: false

hfile.index.block.max.size

当多层次块索引中的叶级、中间层或根级索引块的大小增长到这个大小时，该块将被写入，并开始一个新的块。

默认: 131072

hfile.format.version

新文件使用的HFile格式版本。设置为1来测试向后兼容性。此选项的默认值应符合FixedFileTrailer.MAX\_VERSION.

默认: 2

io.storefile.bloom.block.size

复合Bloom滤波器的单个块（“chunk”）的字节大小。这个大小是近似的，因为Bloom块只能在数据块边界中插入，并且每个数据块的键数是不同的。

默认: 131072

hfile.block.bloom.cacheonwrite

允许对复合布鲁姆过滤器的内联块写缓存.

默认: false

hbase.rs.cacheblocksonwrite

块结束时,是否 HFile块应添加块缓存。

默认: false

hbase.rpc.server.engine

用于服务器的RPC调用编组的org.apache.hadoop.hbase.ipc.RpcServerEngine 实现.

默认: org.apache.hadoop.hbase.ipc.ProtobufRpcServerEngine

hbase.ipc.client.tcpnodelay

设置RPC套接字连接不延迟。参考 http://docs.oracle.com/javase/1.5.0/docs/api/java/net/Socket.html#getTcpNoDelay()

默认: true

hbase.master.keytab.file

HMaster server验证登录使用的kerberos keytab 文件路径。(译者注：HBase使用Kerberos实现安全)

默认:

hbase.master.kerberos.principal

例如. "hbase/\_HOST@EXAMPLE.COM". HMaster运行需要使用 kerberos principal name. principal name 可以在: user/hostname@DOMAIN 中获取. 如果 "\_HOST" 被用做hostname portion，需要使用实际运行的hostname来替代它。

默认:

hbase.regionserver.keytab.file

HRegionServer验证登录使用的kerberos keytab 文件路径。

默认:

hbase.regionserver.kerberos.principal

例如. "hbase/\_HOST@EXAMPLE.COM". HRegionServer运行需要使用 kerberos principal name. principal name 可以在: user/hostname@DOMAIN 中获取. 如果 "\_HOST" 被用做hostname portion，需要使用实际运行的hostname来替代它。在这个文件中必须要有一个entry来描述 hbase.regionserver.keytab.file

默认:

hadoop.policy.file

RPC服务器使用策略配置文件对客户端请求做出授权决策。仅当HBase启用了安全设置可用.

默认: hbase-policy.xml

hbase.superuser

拥有完整的特权用户或组的列表(逗号分隔), 不限于本地存储的 ACLs, 或整个集群. 仅当HBase启用了安全设置可用.

默认:

hbase.auth.key.update.interval

The update interval for master key for authentication tokens in servers in milliseconds. Only used when HBase security is enabled.

默认: 86400000

hbase.auth.token.max.lifetime

The maximum lifetime in milliseconds after which an authentication token expires. Only used when HBase security is enabled.

默认: 604800000

zookeeper.session.timeout

ZooKeeper 会话超时.HBase把这个值传递改zk集群，向他推荐一个会话的最大超时时间。详见http://hadoop.apache.org/zookeeper/docs/current/zookeeperProgrammers.html#ch\_zkSessions "客户端发送请求超时，服务器响应它可以给客户端的超时"。 单位是毫秒

默认: 180000

zookeeper.znode.parent

ZooKeeper中的HBase的根ZNode。所有的HBase的ZooKeeper会用这个目录配置相对路径。默认情况下，所有的HBase的ZooKeeper文件路径是用相对路径，所以他们会都去这个目录下面。

默认: /hbase

zookeeper.znode.rootserver

ZNode 保存的 根region的路径. 这个值是由Master来写，client和regionserver 来读的。如果设为一个相对地址，父目录就是 ${zookeeper.znode.parent}.默认情形下，意味着根region的路径存储在/hbase/root-region-server.

默认: root-region-server

zookeeper.znode.acl.parent

Root ZNode for access control lists.

默认: acl

hbase.coprocessor.region.classes

一个逗号分隔的协处理器，通过在所有表的默认加载列表。对于任何重写协处理器方法，这些类将按顺序调用。实现你自己的协处理器后，就把它放在HBase的路径，在此添加完全限定名称。协处理器也可以装上设置HTableDescriptor需求。

默认:

hbase.coprocessor.master.classes

一个逗号分隔的org.apache.hadoop.hbase.coprocessor.MasterObserver 微处理器列表，主HMaster进程默认加载。对于任何实现的协处理器方法，列出的类将按顺序调用。实现你自己的MasterObserver后，就把它放在HBase的路径，在此处添加完全限定名称（fully qualified class name）。

默认:

hbase.zookeeper.quorum

Zookeeper集群的地址列表，用逗号分割。例如："host1.mydomain.com,host2.mydomain.com,host3.mydomain.com".默认是localhost,是给伪分布式用的。要修改才能在完全分布式的情况下使用。如果在hbase-env.sh设置了HBASE\_MANAGES\_ZK，这些ZooKeeper节点就会和HBase一起启动。

默认: localhost

hbase.zookeeper.peerport

ZooKeeper节点使用的端口。详细参见：http://hadoop.apache.org/zookeeper/docs/r3.1.1/zookeeperStarted.html#sc\_RunningReplicatedZooKeeper

默认: 2888

hbase.zookeeper.leaderport

ZooKeeper用来选择Leader的端口，详细参见：http://hadoop.apache.org/zookeeper/docs/r3.1.1/zookeeperStarted.html#sc\_RunningReplicatedZooKeeper

默认: 3888

hbase.zookeeper.useMulti

指示HBase使用ZooKeeper的多更新功能。这让某些管理员操作完成更迅速，防止一些问题与罕见的复制失败的情况下（见例hbase-2611版本说明）。重要的是：只有设置为true，如果集群中的所有服务器上的管理员3.4版本不会被降级。ZooKeeper管理员3.4之前的版本不支持多更新不优雅地失败如果多更新时把它放在HBase的路径，在这里添加完全限定名称。 (参考 ZOOKEEPER-1495).

默认: false

hbase.zookeeper.property.initLimit

ZooKeeper的zoo.conf中的配置。 初始化synchronization阶段的ticks数量限制

默认: 10

hbase.zookeeper.property.syncLimit

ZooKeeper的zoo.conf中的配置。 发送一个请求到获得承认之间的ticks的数量限制

默认: 5

hbase.zookeeper.property.dataDir

ZooKeeper的zoo.conf中的配置。 快照的存储位置

默认: ${hbase.tmp.dir}/zookeeper

hbase.zookeeper.property.clientPort

ZooKeeper的zoo.conf中的配置。 客户端连接的端口

默认: 2181

hbase.zookeeper.property.maxClientCnxns

ZooKeeper的zoo.conf中的配置。 ZooKeeper集群中的单个节点接受的单个Client(以IP区分)的请求的并发数。这个值可以调高一点，防止在单机和伪分布式模式中出问题。

默认: 300

hbase.rest.port

HBase REST server的端口

默认: 8080

hbase.rest.readonly

定义REST server的运行模式。可以设置成如下的值： false: 所有的HTTP请求都是被允许的 - GET/PUT/POST/DELETE. true:只有GET请求是被允许的

默认: false

hbase.defaults.for.version.skip

设置为true，跳过 'hbase.defaults.for.version' 检查。 设置为 true 相对maven 生成很有用，例如ide环境. 你需要将该布尔值设为 true，避免看到RuntimException 抱怨: "hbase-default.xml file seems to be for and old version of HBase (\${hbase.version}), this version is X.X.X-SNAPSHOT"

默认: false

hbase.coprocessor.abortonerror

Set to true to cause the hosting server (master or regionserver) to abort if a coprocessor throws a Throwable object that is not IOException or a subclass of IOException. Setting it to true might be useful in development environments where one wants to terminate the server as soon as possible to simplify coprocessor failure analysis.

默认: false

hbase.online.schema.update.enable

Set true to enable online schema changes. This is an experimental feature. There are known issues modifying table schemas at the same time a region split is happening so your table needs to be quiescent or else you have to be running with splits disabled.

默认: false

hbase.table.lock.enable

Set to true to enable locking the table in zookeeper for schema change operations. Table locking from master prevents concurrent schema modifications to corrupt table state.

默认: true

dfs.support.append

Does HDFS allow appends to files? This is an hdfs config. set in here so the hdfs client will do append support. You must ensure that this config. is true serverside too when running hbase (You will have to restart your cluster after setting it).

默认: true

hbase.thrift.minWorkerThreads

The "core size" of the thread pool. New threads are created on every connection until this many threads are created.

默认: 16

hbase.thrift.maxWorkerThreads

The maximum size of the thread pool. When the pending request queue overflows, new threads are created until their number reaches this number. After that, the server starts dropping connections.

默认: 1000

hbase.thrift.maxQueuedRequests

The maximum number of pending Thrift connections waiting in the queue. If there are no idle threads in the pool, the server queues requests. Only when the queue overflows, new threads are added, up to hbase.thrift.maxQueuedRequests threads.

默认: 1000

hbase.offheapcache.percentage

The amount of off heap space to be allocated towards the experimental off heap cache. If you desire the cache to be disabled, simply set this value to 0.

默认: 0

hbase.data.umask.enable

Enable, if true, that file permissions should be assigned to the files written by the regionserver

默认: false

hbase.data.umask

hbase.data.umask.enable为真时，用于写数据文件文件的权限

默认: 000

hbase.metrics.showTableName

Whether to include the prefix "tbl.tablename" in per-column family metrics. If true, for each metric M, per-cf metrics will be reported for tbl.T.cf.CF.M, if false, per-cf metrics will be aggregated by column-family across tables, and reported for cf.CF.M. In both cases, the aggregated metric M across tables and cfs will be reported.

默认: true

hbase.metrics.exposeOperationTimes

是否报告有关在区域服务器上执行操作的时间的度量。 Get, Put, Delete, Increment, 及 Append 都可以有他们的时间，每CF和每个区域都可以通过Hadoop metrics暴露出来。

默认: true

hbase.master.hfilecleaner.plugins

一个以逗号分隔的HFileCleanerDelegate HFileCleaner调用的服务。这些HFile清除服务被按顺序调用,所以把清除大部分文件的清除服务放在最前面。实现自己的HFileCleanerDelegate,需把它放在HBase的类路径,并在此添加完全限定类名。总是添加上述日志清除服务，因为会被hbase-site.xml的配置覆盖。

默认: org.apache.hadoop.hbase.master.cleaner.TimeToLiveHFileCleaner

hbase.regionserver.catalog.timeout

Catalog Janitor从regionserver到 META的超时值.

默认: 600000

hbase.master.catalog.timeout

Catalog Janitor从master到 META的超时值.

默认: 600000

hbase.config.read.zookeeper.config

设置为true，允许HBaseConfiguration读取 zoo.cfg 文件，获取ZooKeeper配置。切换为true是不推荐的，因为从zoo.cfg文件读取ZK配置功能已废弃。

默认: false

hbase.snapshot.enabled

设置为true允许快照被取/恢复/克隆。

默认: true

hbase.rest.threads.max

REST服务器线程池的最大线程数。池中的线程被重用以处理REST请求。该值控制同时处理的最大请求数。这可能有助于控制REST服务器内存，避免OOM问题。如果线程池已满，传入的请求将被排队等待一些空闲线程。默认值是100.

默认: 100

hbase.rest.threads.min

REST服务器线程池的最小线程数。线程池始终具有该数量的线程，以便REST服务器准备好为传入的请求服务。默认值是 2.

默认: 2

* + 1. **hbase-env.sh**

在这个文件里面设置HBase环境变量。比如可以配置JVM启动的堆大小或者GC的参数。你还可在这里配置HBase的参数，如Log位置，niceness(译者注:优先级)，ssh参数还有pid文件的位置等等。打开文件conf/hbase-env.sh细读其中的内容。每个选项都是有详尽的注释的。你可以在此添加自己的环境变量。

这个文件的改动系统HBase重启才能生效。

* + 1. **log4j.properties**

编辑这个文件可以改变HBase的日志的级别，轮滚策略等等。

这个文件的改动系统HBase重启才能生效。 日志级别的更改会影响到HBase UI

* + 1. **连接HBase集群的客户端配置和依赖**

因为HBase的Master有可能转移，所有客户端需要访问ZooKeeper来获得现在的位置。ZooKeeper会保存这些值。因此客户端必须知道Zookeeper集群的地址，否则做不了任何事情。通常这个地址存在 hbase-site.xml 里面，客户端可以从CLASSPATH取出这个文件.

如果你是使用一个IDE来运行HBase客户端，你需要将conf/放入你的 classpath,这样 hbase-site.xml就可以找到了，(或者把hbase-site.xml放到 src/test/resources，这样测试的时候可以使用).

HBase客户端最小化的依赖是 hbase, hadoop, log4j, commons-logging, commons-lang, 和 ZooKeeper ，这些jars 需要能在 CLASSPATH 中找到。

下面是一个基本的客户端 hbase-site.xml 例子：

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>hbase.zookeeper.quorum</name>

<value>master,slave1,slave2</value>

<description>The directory shared by region servers.

</description>

</property>

</configuration>

* + - 1. **Java客户端配置**

**Java是如何读到hbase-site.xml 的内容的**

Java客户端使用的配置信息是被映射在一个HBaseConfiguration 实例中. HBaseConfiguration有一个工厂方法, HBaseConfiguration.create();,运行这个方法的时候，他会去CLASSPATH,下找hbase-site.xml，读他发现的第一个配置文件的内容。 (这个方法还会去找hbase-default.xml ; hbase.X.X.X.jar里面也会有一个an hbase-default.xml). 不使用任何hbase-site.xml文件直接通过Java代码注入配置信息也是可以的。例如，你可以用编程的方式设置ZooKeeper信息，只要这样做:

Configuration config = HBaseConfiguration.create();

config.set("hbase.zookeeper.quorum", "localhost"); // Here we are running zookeeper locally

如果有多ZooKeeper实例，你可以使用逗号列表。(就像在hbase-site.xml 文件中做得一样). 这个 Configuration 实例会被传递到 HTable, 之类的实例里面去.

* 1. **配置示例**
     1. **简单的分布式HBase安装**

这里是一个3节点的HBase的简单示例，这里的配置都是基本的，节点名为 master, slave1... 一直到 slave2 . HBase Master 和 HDFS namenode 运作在同一个节点 master上. RegionServers 运行在节点 slave1-slave2. 一个 3-节点 ZooKeeper 集群运行在master, slave1, 和 slave2，端口保持默认. ZooKeeper 的数据保存在目录 /data/zookeeper. 下面我们展示主要的配置文件-- hbase-site.xml, regionservers, 和 hba

* + - 1. **hbase-site.xml**

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>hbase.zookeeper.quorum</name>

<value>master,slave1,slave2</value>

<description>The directory shared by RegionServers.

</description>

</property>

<property>

<name>hbase.rootdir</name>

<value>hdfs://bigdata/hbase</value>

<description>The directory shared by RegionServers.

</description>

</property>

<property>

<name>hbase.cluster.distributed</name>

<value>true</value>

<description>The mode the cluster will be in. Possible values are

false: standalone and pseudo-distributed setups with managed Zookeeper

true: fully-distributed with unmanaged Zookeeper Quorum (see hbase-env.sh)

</description>

</property>

</configuration>

* + - 1. **regionservers**

这个文件把RegionServer的节点列了下来。在这个例子里面我们让所有的节点都运行RegionServer,除了第一个节点 master，它要运行 HBase Master 和 HDFS namenode

slave1

slave2

* + - 1. **hbase-env.sh**

下面我们用**diff** 命令来展示 hbase-env.sh 文件相比默认变化的部分. 我们把HBase的堆内存设置为4G而不是默认的1G.

$ git diff hbase-env.sh

diff --git a/conf/hbase-env.sh b/conf/hbase-env.sh

index e70ebc6..96f8c27 100644

--- a/conf/hbase-env.sh

+++ b/conf/hbase-env.sh

@@ -31,7 +31,7 @@ export JAVA\_HOME=/opt/jdk1.8.0\_151/

# export HBASE\_CLASSPATH=

# The maximum amount of heap to use, in MB. Default is 1000.

-# export HBASE\_HEAPSIZE=1000

+export HBASE\_HEAPSIZE=4096

# Extra Java runtime options.

# Below are what we set by default. May only work with SUN JVM.

你可以使用 **rsync** 来同步 conf 文件夹到你的整个集群.

* 1. **重要的配置**

下面我们会列举*重要* 的配置. 这个章节讲述必须的配置和那些值得一看的配置。(译者注:淘宝的博客也有本章节的内容，[HBase性能调优](http://rdc.taobao.com/team/jm/archives/975)，很详尽)。

* + 1. **必须的配置**

参考 Section 1.2, “操作系统” 和 Section 1.3, “Hadoop” 节.

* + 1. **推荐配置**
       1. **zookeeper.session.timeout**

这个默认值是3分钟。这意味着一旦一个server宕掉了，Master至少需要3分钟才能察觉到宕机，开始恢复。你可能希望将这个超时调短，这样Master就能更快的察觉到了。在你调这个值之前，你需要确认你的JVM的GC参数，否则一个长时间的GC操作就可能导致超时。（当一个RegionServer在运行一个长时间的GC的时候，你可能想要重启并恢复它）.

要想改变这个配置，可以编辑 hbase-site.xml, 将配置部署到全部集群，然后重启。

我们之所以把这个值调的很高，是因为我们不想一天到晚在论坛里回答新手的问题。“为什么我在执行一个大规模数据导入的时候Region Server死掉啦”，通常这样的问题是因为长时间的GC操作引起的，他们的JVM没有调优。我们是这样想的，如果一个人对HBase不很熟悉，不能期望他知道所有，打击他的自信心。等到他逐渐熟悉了，他就可以自己调这个参数了。

* + - 1. **ZooKeeper 实例个数**

参考 Section 1.5, “ZooKeeper”.

* + - 1. **hbase.regionserver.handler.count**

这个设置决定了处理用户请求的线程数量。默认是10，这个值设的比较小，主要是为了预防用户用一个比较大的写缓冲，然后还有很多客户端并发，这样region servers会垮掉。有经验的做法是，当请求内容很大(上MB，如大puts, 使用缓存的scans)的时候，把这个值放低。请求内容较小的时候(gets, 小puts, ICVs, deletes)，把这个值放大。

当客户端的请求内容很小的时候，把这个值设置的和最大客户端数量一样是很安全的。一个典型的例子就是一个给网站服务的集群，put操作一般不会缓冲,绝大多数的操作是get操作。

把这个值放大的危险之处在于，把所有的Put操作缓冲意味着对内存有很大的压力，甚至会导致OutOfMemory.一个运行在内存不足的机器的RegionServer会频繁的触发GC操作，渐渐就能感受到停顿。(因为所有请求内容所占用的内存不管GC执行几遍也是不能回收的)。一段时间后，集群也会受到影响，因为所有的指向这个region的请求都会变慢。这样就会拖累集群，加剧了这个问题。

你可能会对handler太多或太少有感觉，在单个RegionServer启动log并查看log末尾 (请求队列消耗内存)。

* + - 1. **大内存机器的配置**

HBase有一个合理的保守的配置，这样可以运作在所有的机器上。如果你有台大内存的集群-HBase有8G或者更大的heap,接下来的配置可能会帮助你. TODO.

* + - 1. **压缩**

应该考虑启用ColumnFamily 压缩。有好几个选项，通过降低存储文件大小以降低IO，降低消耗且大多情况下提高性能。

* + - 1. **较大 Regions**

更大的Region可以使你集群上的Region的总数量较少。 一般来言，更少的Region可以使你的集群运行更加流畅。(你可以自己随时手工将大Region切割，这样单个热点Region就会被分布在集群的更多节点上)。

较少的Region较好。一般每个RegionServer在20到小几百之间。 调整Region大小以适合该数字。

0.90.x 版本, 默认情况下单个Region是256MB。Region 大小的上界是 4Gb. 0.92.x 版本, 由于 HFile v2 已经将Region大小支持得大很多， (如, 20Gb).

可能需要实验，基于硬件和应用需要进行配置。

可以调整hbase-site.xml中的 hbase.hregion.max.filesize属性. RegionSize 也可以基于每个表设置：  HTableDescriptor.

* + - 1. **管理 Splitting**

除了让HBase自动切割你的Region,你也可以手动切割。 [[12](http://abloz.com/hbase/book.html" \l "ftn.d613e1974)] 随着数据量的增大，splite会被持续执行。如果你需要知道你现在有几个region,比如长时间的debug或者做调优，你需要手动切割。通过跟踪日志来了解region级的问题是很难的，因为他在不停的切割和重命名。data offlineing bug和未知量的region会让你没有办法。如果一个 HLog 或者 StoreFile由于一个奇怪的bug，HBase没有执行它。等到一天之后，你才发现这个问题，你可以确保现在的regions和那个时候的一样，这样你就可以restore或者replay这些数据。你还可以调优你的合并算法。如果数据是均匀的，随着数据增长，很容易导致split / compaction疯狂的运行。因为所有的region都是差不多大的。用手的切割，你就可以交错执行定时的合并和切割操作，降低IO负载。

为什么我关闭自动split呢？因为自动的splite是配置文件中的 hbase.hregion.max.filesize决定的. 你把它设置成Long.MAX\_VALUE是不推荐的做法，要是你忘记了手工切割怎么办.推荐的做法是设置成100GB，一旦到达这样的值，至少需要一个小时执行 major compactions。

那什么是最佳的在pre-splite regions的数量呢。这个决定于你的应用程序了。你可以先从低的开始，比如每个server10个pre-splite regions.然后花时间观察数据增长。有太少的region至少比出错好，你可以之后再rolling split.一个更复杂的答案是这个值是取决于你的region中的最大的storefile。随着数据的增大，这个也会跟着增大。 你可以当这个文件足够大的时候，用一个定时的操作使用Store的合并选择算法(compact selection algorithm)来仅合并这一个HStore。如果你不这样做，这个算法会启动一个 major compactions，很多region会受到影响，你的集群会疯狂的运行。需要注意的是，这样的疯狂合并操作是数据增长造成的，而不是手动分割操作决定的。

如果你 pre-split 导致 regions 很小,你可以通过配置HConstants.MAJOR\_COMPACTION\_PERIOD把你的major compaction参数调大

如果你的数据变得太大，可以使用org.apache.hadoop.hbase.util.RegionSplitter 脚本来执行针对全部集群的一个网络IO安全的rolling split操作。

* + - 1. **管理 Compactions**

通常管理技术是手动管理主紧缩(major compactions)， 而不是让HBase 来做。 缺省HConstants.MAJOR\_COMPACTION\_PERIOD 是一天。主紧缩可能强行进行，在你并不太希望发生的时候——特别是在一个繁忙系统。关闭自动主紧缩，设置该值为0.

重点强调，主紧缩对存储文件(StoreFile)清理是绝对必要的。唯一变量是发生的时间。可以通过HBase shell进行管理，或通过 HBaseAdmin.

* + - 1. **预测执行 (Speculative Execution)**

MapReduce任务的预测执行缺省是打开的，HBase集群一般建议在系统级关闭预测执行，除非在某种特殊情况下需要打开，此时可以每任务配置。设置mapred.map.tasks.speculative.execution 和 mapred.reduce.tasks.speculative.execution 为 false.

* + 1. **其他配置**
       1. **负载均衡**

负载均衡器(LoadBalancer)是在主服务器上运行的定期操作，以重新分布集群区域。通过hbase.balancer.period 设置，缺省值300000 (5 分钟).

* + - 1. **禁止块缓存(Blockcache)**

不要关闭块缓存 (通过hbase.block.cache.size 为 0 来设置)。当前如果关闭块缓存会很不好，因为区域服务器会花很多时间不停加载hfile指数。如果工作集如此配置块缓存没有好处，最少应保证hfile指数保存在块缓存内的大小(可以通过查询区域服务器的UI，得到大致的数值。可以看到网页的上方有块指数值统计).

* + - 1. **Nagle算法 或小包问题**

如果操作HBase时看到大量40ms左右的偶然延时，尝试Nagles配置。如，参考用户邮件列表线索, [Inconsistent scan performance with caching set to 1](http://search-hadoop.com/m/pduLg2fydtE/Inconsistent+scan+performance+with+caching+set+&subj=Re+Inconsistent+scan+performance+with+caching+set+to+1) ，该议题在其中启用tcpNoDelay (译者注，本英文原文notcpdelay有误)提高了扫描速度。你也可以查看该文档的尾部图表[：HBASE-7008 Set scanner caching to a better default](https://issues.apache.org/jira/browse/HBASE-7008)  (xie liang)，我们的Lars Hofhansl 尝试了各种不同的数据大小，Nagle打开或关闭的测量结果。

# 配置总结

# 解压缩

[hadoop@master pkg]$ tar -zxvf hbase-1.2.6-bin.tar.gz -C /opt/bigdata/

[hadoop@master pkg]$ cd /opt/bigdata/

[hadoop@master bigdata]$ mv hbase-1.2.6/ hbase

[hadoop@master bigdata]$ cd hbase/conf  
# 修改配置文件

[hadoop@master conf]$ vi hbase-site.xml

<configuration>

<property>

<name>hbase.rootdir</name>

<value>hdfs://bigdata/hbase</value>

<description>The directory shared by RegionServers.

</description>

</property>

<property>

<name>hbase.cluster.distributed</name>

<value>true</value>

<description>The mode the cluster will be in. Possible values are

false: standalone and pseudo-distributed setups with managed Zookeeper

true: fully-distributed with unmanaged Zookeeper Quorum (see hbase-env.sh)

</description>

</property>

<property>

<name>hbase.zookeeper.quorum</name>

<value>master,slave1,slave2</value>

<description>The directory shared by region servers.

</description>

</property>

</configuration>

[hadoop@master conf]$ vi regionservers

slave1

slave2

[hadoop@master conf]$ vi hbase-env.sh

export JAVA\_HOME=/opt/jdk1.8.0\_151/

export HBASE\_CLASSPATH=/opt/bigdata/hbase/

export HBASE\_PID\_DIR=/tmp/logs/

export HBASE\_MANAGES\_ZK=false

# 添加hbase配置文件：core-site.xml和hdfs-site.xml

（Hadoop作HA后需要这一步）

[hadoop@master logs]$ cp /opt/bigdata/hadoop/etc/hadoop/core-site.xml /opt/bigdata/hadoop/etc/hadoop/hdfs-site.xml /opt/bigdata/hbase/conf/

# 分发

[hadoop@master conf]$ scp -r /opt/bigdata/hbase/ hadoop@slave1:/opt/bigdata/

[hadoop@master conf]$ scp -r /opt/bigdata/hbase/ hadoop@slave2:/opt/bigdata/

# 启动

[hadoop@master conf]$ ../bin/start-hbase.sh