dingtong note

Hbase split的三种方式和split的过程

在Hbase中split是一个很重要的功能，Hbase是通过把数据分配到一定数量的region来达到负载均衡的。一个table会被分配到一个或多个region中，这些region会被分配到一个或者多个regionServer中。在自动split策略中，当一个region达到一定的大小就会自动split成两个region。table在region中是按照row key来排序的，并且一个row key所对应的行只会存储在一个region中，这一点保证了Hbase的强一致性 。

在一个region中有一个或多个stroe，每个stroe对应一个column families(列族)。一个store中包含一个memstore 和 0 或 多个store files。每个column family 是分开存放和分开访问的。

Pre-splitting

当一个table刚被创建的时候，Hbase默认的分配一个region给table。也就是说这个时候，所有的读写请求都会访问到同一个regionServer的同一个region中，这个时候就达不到负载均衡的效果了，集群中的其他regionServer就可能会处于比较空闲的状态。解决这个问题可以用pre-splitting,在创建table的时候就配置好，生成多个region。

在table初始化的时候如果不配置的话，Hbase是不知道如何去split region的，因为Hbase不知道应该那个row key可以作为split的开始点。如果我们可以大概预测到row key的分布，我们可以使用pre-spliting来帮助我们提前split region。不过如果我们预测得不准确的话，还是可能导致某个region过热，被集中访问，不过还好我们还有auto-split。最好的办法就是首先预测split的切分点，做pre-splitting,然后后面让auto-split来处理后面的负载均衡。

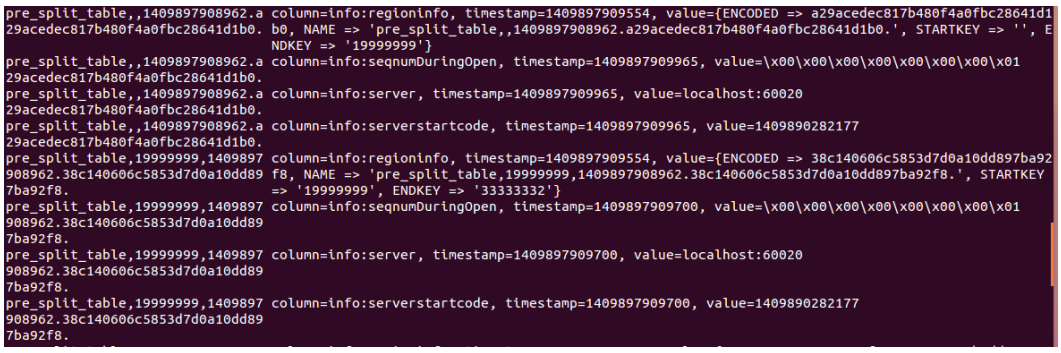
Hbase自带了两种pre-split的算法，分别是 HexStringSplit 和 UniformSplit 。如果我们的row key是十六进制的字符串作为前缀的，就比较适合用 HexStringSplit ，作为pre-split的算法。例如，我们使用HexHash(prefix)作为row key的前缀，其中Hexhash为最终得到十六进制字符串的hash算法。我们也可以用我们自己的split算法。

在hbase shell 下：

```
hbase org.apache.hadoop.hbase.util.RegionSplitter pre_split_table HexStringSplit -c 10 -f f1
```

-c 10 的意思为，最终的region数目为10个；-f f1为创建一个那么为f1的 column family.

执行scan 'hbase:meta' 可以看到meta表中的，



只截取了meta表中的2个region的记录（一共10个region），分别是rowkey范围是 '' ''~19999999 和19999999~33333332的region。

我们也可以自定义切分点，例如在hbase shell下使用如下命令：

```
create 't1', 'f1', {SPLITS => ['10', '20', '30', '40']}
```

自动splitting

当一个reion达到一定的大小，他会自动split称两个region。如果我们的Hbase版本是0.94 ，那么默认的有三种自动split的策略，ConstantSizeRegionSplitPolicy , IncreasingToUpperBoundRegionSplitPolicy 还有 KeyPrefixRegionSplitPolicy .

在0.94版本之前 ConstantSizeRegionSplitPolicy 是默认和唯一的split策略。当某个store (对应一个column family) 的大小大于配置值 'hbase.hregion.max.filesize'的时候 (默认10G) region就会自动分裂。

而0.94版本中，IncreasingToUpperBoundRegionSplitPolicy 是默认的split策略。

这个策略中，最小的分裂大小和table的某个region server的region 个数有关，当store file的大小大于如下公式得出的值的时候就会split，公式如下

```
Min (R^2 * "hbase.hregion.memstore.flush.size", "hbase.hregion.max.filesize")  R为同一个table中在同一个region server中region的个数。
```

例如:

hbase.hregion.memstore.flush.size 默认值 128MB。

hbase.hregion.max.filesize默认值为10GB 。

- 如果初始时R=1,那么Min(128MB,10GB)=128MB,也就是说在第一个flush的时候就会触发分裂操作。
- 当R=2的时候Min(2*2*128MB,10GB)=512MB ,当某个store file大小达到512MB的时候，就会触发分裂。
- 如此类推，当R=9的时候，store file 达到10GB的时候就会分裂，也就是说当R>=9的时候，store file 达到10GB的时候就会分裂。

split 点都位于region中row key的中间点。

KeyPrefixRegionSplitPolicy可以保证相同的前缀的row保存在同一个region中。

指定rowkey前缀位数划分region，通过读取 KeyPrefixRegionSplitPolicy.prefix_length 属性，该属性为数字类型，表示前缀长度，在进行split时，按此长度对splitPoint进行截取。此种策略比较适合固定前缀的rowkey。当table中没有设置该属性，指定此策略效果等同与使用IncreasingToUpperBoundRegionSplitPolicy。

我们可以通过配置 hbase.regionserver.region.split.policy 来指定split策略，我们也可以写我们自己的split策略。

强制split

Hbase 允许客户端强制执行split,在hbase shell中执行以下命令：

```
split 'forced_table', 'b' //其中forced_table 为要split的table , 'b' 为split 点
```

region splits 执行过程：

region server处理写请求的时候，会先写入memstore，当memstore 达到一定大小的时候，会写入磁盘成为一个store file。这个过程叫做memstore flush。当store files 堆积到一定大小的时候，region server 会 执行'compact'操作，把他们合成一个大的文件。 当每次执行完flush 或者compact操作，都会判断是否需要split。当发生split的时候，会生成两个region A 和 region B但是parent region数据file并不会发生复制等操作，而是region A 和region B 会有这些file的引用。这些引用文件会在下次发生compact操作的时候清理掉，并且当region中有引用文件的时候是不会再进行split操作的。这个地方需要注意一下，如果当region中存在引用文件的时候，而且写操作很频繁和集中，可能会出现region变得很大，但是却不split。因为写操作比较频繁和集中，但是没有均匀到每个引用文件上去，所以region一直存在引用文件，不能进行分裂，这篇文章讲到了这个情况，总结得挺好的。 http://koven2049.iteye.com/blog/1199519

虽然split region操作是region server单独确定的，但是split过程必须和很多其他部件合作。region server 在split开始前和结束前通知master,并且需要更新.META.表，这样，客户端就能知道有新的region。在hdfs中重新排列目录结构和数据文件。split是一个复杂的操作。在split region的时候会记录当前执行的状态，当出错的时候，会根据状态进行回滚。下图表示split中，执行的过程。 (红色线表示region server 或者master的操作，绿色线表示client的操作。)

1.region server 决定split region，第一步，region server在zookeeper中创建在

/hbase/region-in-transition/region-name 目录下，创建一个znode，状态为SPLITTING.

2.因为master有对 region-in-transition 的znode做监听，所以，mater的得知parent region需要split

3.region server 在hdfs的parent region的目录下创建一个名为".splits"的子目录

4.region server 关闭parent region。强制flush缓存，并且在本地数据结构中标记region为下线状态。如果这个时候客户端刚好请求到parent region，会抛出NotServingRegionException。这时客户端会进行补偿性重试。

5.region server在.split 目录下分别为两个daughter region创建目录和必要的数据结构。然后创建两个引用文件指向parent regions的文件。

6.region server 在HDFS中，创建真正的region目录，并且把引用文件移到对应的目录下。

7.region server 发送一个put的请求到.META.表中，并且在.META.表中设置parent region为下线状态，并且在parent region对应的row中两个daughter region的信息。但是这个时候在.META.表中daughter region 还不是独立的row。这个时候如果client scan .META.表，会发现parent region正在split，但是client还看不到daughter region的信息。当这个put 成功之后，parent region split会被正在的执行。如果在RPC 成功之前 region server 就失败了，master和下次打开parent region的region server 会清除关于这次split的脏状态。但是当RPC返回结果给到parent region ，即.META.成功更新之后，，region split的流程还会继续进行下去。相当于是个补偿机制，下次在打开这个parent region的时候会进行相应的清理操作。

8.region server 打开两个daughter region接受写操作。

9.region server 在.META.表中增加daughters A 和 B region的相关信息，在这以后，client就能发现这两个新的regions并且能发送请求到这两个新的region了。client本地具体有.META.表的缓存，当他们访问到parent region的时候，发现parent region下线了，就会重新访问.META.表获取最新的信息，并且更新本地缓存。

10.region server 更新 znode 的状态为SPLIT。master就能知道状态更新了，master的平衡机制会判断是否需要把daughter regions 分配到其他region server 中。

11.在split之后，meta和HDFS依然会有引用指向parent region. 当compact 操作发生在daughter regions中，会重写数据file，这个时候引用就会被逐渐的去掉。垃圾回收任务会定时检测daughter regions是否还有引用指向parent files，如果没有引用指向parent files的话，parent region 就会被删除。

参考连接：

http://hortonworks.com/blog/apache-hbase-region-splitting-and-merging/ Hbase split

http://hbase.apache.org/book/regions.arch.html Hbase 官方文档(region)

http://blog.javachen.com/2014/01/16/hbase-region-split-policy/ split策略

http://blackproof.iteye.com/blog/2037159 split源码解析

posted @ 2014-12-31 16:59 albeter 阅读(4652) 评论(1) 编辑 收藏