Hive是将符合SQL语法的字符串解析生成可以在Hadoop上执行的MapReduce的工具。使用Hive尽量按照分布式计算的一些特点来设计sql，和传统关系型数据库有区别，所以需要去掉原有关系型数据库下开发的一些固有思维。

基本原则：

1：尽量尽早地过滤数据，减少每个阶段的数据量,对于分区表要加分区，同时只选择需要使用到的字段

select ... from A

join B

on A.key = B.key

where A.userid>10

and B.userid<10

and A.dt='20120417'

and B.dt='20120417';

应该改写为：

select .... from (select .... from A

where dt='201200417'

and userid>10

) a

join ( select .... from B

where dt='201200417'

and userid < 10

) b

on a.key = b.key;

2、对历史库的计算经验 (这项是说根据不同的使用目的优化使用方法)

历史库计算和使用，分区

3：尽量原子化操作，尽量避免一个SQL包含复杂逻辑

可以使用中间表来完成复杂的逻辑

4 jion操作 小表要注意放在join的左边（目前TCL里面很多都小表放在join的右边）。

否则会引起磁盘和内存的大量消耗

5：如果union all的部分个数大于2，或者每个union部分数据量大，应该拆成多个insert into 语句，实际测试过程中，执行时间能提升50%

insert overwite table tablename partition (dt= ....)

select ..... from (

select ... from A

union all

select ... from B

union all

select ... from C

) R

where ...;

可以改写为：

insert into table tablename partition (dt= ....)

select .... from A

WHERE ...;

insert into table tablename partition (dt= ....)

select .... from B

WHERE ...;

insert into table tablename partition (dt= ....)

select .... from C

WHERE ...;

5：写SQL要先了解数据本身的特点，如果有join ,group操作的话，要注意是否会有数据倾斜

如果出现数据倾斜，应当做如下处理：

set hive.exec.reducers.max=200;

set mapred.reduce.tasks= 200;---增大Reduce个数

set hive.groupby.mapaggr.checkinterval=100000 ;--这个是group的键对应的记录条数超过这个值则会进行分拆,值根据具体数据量设置

set hive.groupby.skewindata=true; --如果是group by过程出现倾斜 应该设置为true

set hive.skewjoin.key=100000; --这个是join的键对应的记录条数超过这个值则会进行分拆,值根据具体数据量设置

set hive.optimize.skewjoin=true;--如果是join 过程出现倾斜 应该设置为true

(1) 启动一次job尽可能的多做事情，一个job能完成的事情,不要两个job来做

通常来说前面的任务启动可以稍带一起做的事情就一起做了,以便后续的多个任务重用,与此紧密相连的是模型设计,好的模型特别重要.

(2) 合理设置reduce个数

reduce个数过少没有真正发挥hadoop并行计算的威力，但reduce个数过多，会造成大量小文件问题，数据量、资源情况只有自己最清楚，找到个折衷点,

(3) 使用hive.exec.parallel参数控制在同一个sql中的不同的job是否可以同时运行，提高作业的并发

2、让服务器尽量少做事情，走最优的路径，以资源消耗最少为目标

比如:

(1) 注意join的使用

若其中有一个表很小使用map join，否则使用普通的reduce join，注意hive会将join前面的表数据装载内存,所以较小的一个表在较大的表之前,减少内存资源的消耗

(2)注意小文件的问题

在hive里有两种比较常见的处理办法

第一是使用Combinefileinputformat，将多个小文件打包作为一个整体的inputsplit，减少map任务数

set mapred.max.split.size=256000000;

set mapred.min.split.size.per.node=256000000

set Mapred.min.split.size.per.rack=256000000

set hive.input.format=org.apache.hadoop.hive.ql.io.CombineHiveInputFormat

第二是设置hive参数，将额外启动一个MR Job打包小文件

hive.merge.mapredfiles = false 是否合并 Reduce 输出文件，默认为 False

hive.merge.size.per.task = 256\*1000\*1000 合并文件的大小

(3)注意数据倾斜

在hive里比较常用的处理办法

第一通过hive.groupby.skewindata=true控制生成两个MR Job,第一个MR Job Map的输出结果随机分配到reduce做次预汇总,减少某些key值条数过多某些key条数过小造成的数据倾斜问题

第二通过hive.map.aggr = true(默认为true)在Map端做combiner,假如map各条数据基本上不一样, 聚合没什么意义，做combiner反而画蛇添足,hive里也考虑的比较周到通过参数hive.groupby.mapaggr.checkinterval = 100000 (默认)hive.map.aggr.hash.min.reduction=0.5(默认),预先取100000条数据聚合,如果聚合后的条数/100000>0.5，则不再聚合

(4)善用multi insert,union all

multi insert适合基于同一个源表按照不同逻辑不同粒度处理插入不同表的场景，做到只需要扫描源表一次，job个数不变，减少源表扫描次数

union all用好，可减少表的扫描次数，减少job的个数,通常预先按不同逻辑不同条件生成的查询union all后，再统一group by计算,不同表的union all相当于multiple inputs,同一个表的union all,相当map一次输出多条

(5) 参数设置的调优

集群参数种类繁多,举个例子比如

可针对特定job设置特定参数,比如jvm重用,reduce copy线程数量设置(适合map较快，输出量较大)

如果任务数多且小，比如在一分钟之内完成，减少task数量以减少任务初始化的消耗。可以通过配置JVM重用选项减少task的消耗

-----------------------------------------------------------

一、控制Hive中Map和reduce的数量

Hive中的sql查询会生成执行计划，执行计划以MapReduce的方式执行，那么结合数据和集群的大小，map和reduce的数量就会影响到sql执行的效率。

除了要控制Hive生成的Job的数量，也要控制map和reduce的数量。

1、 map的数量，通常情况下和split的大小有关系，之前写的一篇blog“map和reduce的数量是如何定义的”有描述。

hive中默认的hive.input.format是org.apache.hadoop.hive.ql.io.CombineHiveInputFormat，对于combineHiveInputFormat,它的输入的map数量

由三个配置决定，

mapred.min.split.size.per.node， 一个节点上split的至少的大小

mapred.min.split.size.per.rack 一个交换机下split至少的大小

mapred.max.split.size 一个split最大的大小

它的主要思路是把输入目录下的大文件分成多个map的输入, 并合并小文件, 做为一个map的输入. 具体的原理是下述三步:

a、根据输入目录下的每个文件,如果其长度超过mapred.max.split.size,以block为单位分成多个split(一个split是一个map的输入),每个split的长度都大于mapred.max.split.size, 因为以block为单位, 因此也会大于blockSize, 此文件剩下的长度如果大于mapred.min.split.size.per.node, 则生成一个split, 否则先暂时保留.

b、现在剩下的都是一些长度效短的碎片,把每个rack下碎片合并, 只要长度超过mapred.max.split.size就合并成一个split, 最后如果剩下的碎片比mapred.min.split.size.per.rack大, 就合并成一个split, 否则暂时保留.

c、把不同rack下的碎片合并, 只要长度超过mapred.max.split.size就合并成一个split, 剩下的碎片无论长度, 合并成一个split.

举例: mapred.max.split.size=1000

mapred.min.split.size.per.node=300

mapred.min.split.size.per.rack=100

输入目录下五个文件,rack1下三个文件,长度为2050,1499,10, rack2下两个文件,长度为1010,80. 另外blockSize为500.

经过第一步, 生成五个split: 1000,1000,1000,499,1000. 剩下的碎片为rack1下:50,10; rack2下10:80

由于两个rack下的碎片和都不超过100, 所以经过第二步, split和碎片都没有变化.

第三步,合并四个碎片成一个split, 长度为150.

如果要减少map数量, 可以调大mapred.max.split.size, 否则调小即可.

其特点是: 一个块至多作为一个map的输入，一个文件可能有多个块，一个文件可能因为块多分给做为不同map的输入， 一个map可能处理多个块，可能处理多个文件。

2、 reduce数量

可以在hive运行sql的时，打印出来，如下：

Number of reduce tasks not specified. Estimated from input data size: 1

In order to change the average load for a reducer (in bytes):

set hive.exec.reducers.bytes.per.reducer=<number>

In order to limit the maximum number of reducers:

set hive.exec.reducers.max=<number>

In order to set a constant number of reducers:

set mapred.reduce.tasks=<number

reduce数量由以下三个参数决定，

mapred.reduce.tasks(强制指定reduce的任务数量)

hive.exec.reducers.bytes.per.reducer（每个reduce任务处理的数据量，默认为1000^3=1G）

hive.exec.reducers.max（每个任务最大的reduce数，默认为999）

计算reducer数的公式很简单N=min( hive.exec.reducers.max ，总输入数据量/ hive.exec.reducers.bytes.per.reducer )

只有一个reduce的场景：

a、没有group by 的汇总

b、order by

c、笛卡尔积

二、join和Group的优化

对于普通的join操作，会在map端根据key的hash值，shuffle到某一个reduce上去，在reduce端做join连接操作，内存中缓存join左边的表，遍历右边的表，一次做join操作。所以在做join操作时候，将数据量多的表放在join的右边。

当数据量比较大，并且key分布不均匀，大量的key都shuffle到一个reduce上了，就出现了数据的倾斜。

对于Group操作，首先在map端聚合，最后在reduce端坐聚合，hive默认是这样的，以下是相关的参数

· hive.[map](http://www.verydemo.com/demo_c152_i9269.html).aggr = true是否在 Map 端进行聚合，默认为 True

· hive.groupby.mapaggr.checkinterval = 100000在 Map 端进行聚合操作的条目数目

对于join和Group操作都可能会出现数据倾斜。

以下有几种解决这个问题的常见思路

1、参数hive.groupby.skewindata = true,解决数据倾斜的万能钥匙，查询计划会有两个 MR [Job](http://www.verydemo.com/demo_c152_i9269.html)。第一个 MR Job 中，Map 的输出结果集合会随机分布到 Reduce 中，每个 Reduce 做部分聚合操作，并输出结果，这样处理的结果是相同的 Group By Key 有可能被分发到不同的 Reduce 中，从而达到负载均衡的目的；第二个 MR Job 再根据预处理的数据结果按照 Group By Key 分布到 Reduce 中（这个过程可以保证相同的 Group By Key 被分布到同一个 Reduce 中），最后完成最终的聚合操作。

2、where的条件写在join里面，使得减少join的数量（经过map端过滤，只输出复合条件的）

3、mapjoin方式，无reduce操作，在map端做join操作（map端cache小表的全部数据），这种方式下无法执行Full/RIGHT OUTER join操作

4、对于count(distinct)操作，在map端以group by的字段和count的字段联合作为key，如果有大量相同的key，那么会存在数据倾斜的问题

5、数据的倾斜还包括，大量的join连接key为空的情况，空的key都hash到一个reduce上去了，解决这个问题，最好把空的key和非空的key做区分

空的key不做join操作。

当然有的hive操作，不存在数据倾斜的问题，比如数据聚合类的操作，像sum、count，因为已经在map端做了聚合操作了，到reduce端的数据相对少一些，所以不存在这个问题。

四、小文件的合并

大量的小文件导致文件数目过多，给HDFS带来压力，对hive处理的效率影响比较大，可以合并map和reduce产生的文件

· hive.merge.mapfiles = true是否和并 Map 输出文件，默认为 True

· hive.merge.mapredfiles = [false](http://www.verydemo.com/demo_c152_i9269.html)是否合并 Reduce 输出文件，默认为 False

· hive.merge.size.per.task = 256\*1000\*1000合并文件的大小

五、in/exists（not）

通过left semi join 实现 in操作，一个限制就是join右边的表只能出现在join条件中

六、分区裁剪

通过在条件中指定分区，来限制数据扫描的范围，可以极大提高查询的效率

七、排序

order by 排序，只存在一个reduce，这样效率比较低。

可以用sort by操作,通常结合distribute by使用做reduce分区键

**hive 调优**  
**1.EXPLAIN**   
 EXPLAIN EXTENDED   
 使用EXPLAIN

可以hive对sql的查询计划信息解析  
  EXPLAIN SELECT COUNT(1) FROM T1;  
 hive将查询转为mapreduce执行，所以优化hive可以转化为优化mr.  
 根据EXPLAIN展示的信息可以做出优化  
**2.限制调整，**  
使用order by;sort by ;distribute by ;cluster by ;limit返回部分结果  
**3.jvm重用**  
jvm执行map和reduce任务，启动jvm过程开销很大，  
开启jvm重用可以使得jvm实例在同一个job中重用n次，n在配置文件中设置。  
**4.其他**  
join优化；小表在左边  
本地模式；并行执行，动态分区，严格模式等  
hive的调优本质上是对hive-default.xml中参数的优化。  
  
**数据倾斜：**  
shuffle阶段优化过程中，会遇到数据倾斜。  
如何将map输出的数据均匀分配到reduce中是解决数据倾斜的根本。  
  
**原因:**1.key分布不均匀  
   2.业务数据本身的特性  
   3.建表时考虑不周全  
   4.sql本身就数据倾斜  
**表现：**任务长时间在99%不动。单一reduce的记录数与平均记录数差异过大。  
**解决方案：**参数调节,进行负载均衡  
hive.map.aggr=true  
map端聚合，相当于combiner  
hive.groupby.skewindata=true  
保证相同的group by key分布到同一个reduce  
sql语句调节  
join，map join  
**业务场景：**  
日志中信息丢失，空值产生数据倾斜  
**1.**为空的字段不参与关联  
**2.**给空值赋值,io和作业数都少了，空值key赋值字符串加上随机数  
**3.**日志信息中不同字段类型关联产生数据倾斜  
例如：当按照用户id进行两表join操作时，默认hash操作会按int类型的id进行分配，  
这样string类型的id的记录都会分配到一个reducer中，  
解决办法将数字类型转换为字符串类型  
cast(a.user\_id as string)

**hive面试常见问题**  
1.Hive内外部表的区别  
2.Hive如何做到权限管理  
3.Hive的数据倾斜和调优  
4.Hive文件压缩格式有哪些？压缩效率如何  
5.说说对桶表的理解  
6.Hive解析hql转化为MR的执行过程  
7.Hive的分组排序(row\_number())，组内TopN  
8.hive有哪些保存元数据的方式，有什么特点  
9.hive能见多个库吗  
10.hive的底层怎么与数据库交互的。  
11.hive实现统计的查询语句以及与mr的区别  
12.对于hive你写过哪些UDF函数，作用是干什么的  
13.hive与hbase的区别  
14.hive中的压缩格式textfile, sequencefile, rcfile, orc的区别  
15.hive的工作原理  
16.编写hive自定义函数，实现oracle数据库中的addmonths函数功能，然后封装到hive函数库中。  
   Addmonths(data a,int b)求传入日期a经过b个月后的日期。  
17.另外看到一道很不错的面试题

看了很多博客，感觉没有一个说的很清楚，所以我来整理一下。

先看一下这个图

输入分片（Input Split）：在进行map计算之前，mapreduce会根据输入文件计算输入分片（input split），每个输入分片（input split）针对一个map任务，输入分片（input split）存储的并非数据本身，而是一个分片长度和一个记录数据的位置的数组。

Hadoop 2.x默认的block大小是128MB，Hadoop 1.x默认的block大小是64MB，可以在hdfs-site.xml中设置dfs.block.size，注意单位是byte。

分片大小范围可以在mapred-site.xml中设置，mapred.min.split.size mapred.max.split.size，minSplitSize大小默认为1B，maxSplitSize大小默认为Long.MAX\_VALUE = 9223372036854775807

那么分片到底是多大呢？

minSize=max{minSplitSize,mapred.min.split.size}

maxSize=mapred.max.split.size

splitSize=max{minSize,min{maxSize,blockSize}}

我们再来看一下源码

所以在我们没有设置分片的范围的时候，分片大小是由block块大小决定的，和它的大小一样。比如把一个258MB的文件上传到HDFS上，假设block块大小是128MB，那么它就会被分成三个block块，与之对应产生三个split，所以最终会产生三个map task。我又发现了另一个问题，第三个block块里存的文件大小只有2MB，而它的block块大小是128MB，那它实际占用Linux file system的多大空间？

答案是实际的文件大小，而非一个块的大小。

有大神已经验证这个答案了：http://blog.csdn.net/samhacker/article/details/23089157

1、往hdfs里面添加新文件前，hadoop在linux上面所占的空间为 464 MB：

2、往hdfs里面添加大小为2673375 byte(大概2.5 MB)的文件：

2673375 derby.jar

3、此时，hadoop在linux上面所占的空间为 467 MB——增加了一个实际文件大小(2.5 MB)的空间，而非一个block size(128 MB)：

4、使用hadoop dfs -stat查看文件信息：

这里就很清楚地反映出： 文件的实际大小(file size)是2673375 byte， 但它的block size是128 MB。

5、通过NameNode的web console来查看文件信息:

结果是一样的： 文件的实际大小(file size)是2673375 byte， 但它的block size是128 MB。

6、不过使用‘hadoop fsck’查看文件信息，看出了一些不一样的内容—— ‘1（avg.block size 2673375 B）’:

值得注意的是，结果中有一个 ‘1（avg.block size 2673375 B）’的字样。这里的 'block size' 并不是指平常说的文件块大小(Block Size)—— 后者是一个元数据的概念，相反它反映的是文件的实际大小(file size)。以下是Hadoop Community的专家给我的回复：

“The fsck is showing you an "average blocksize", not the block size metadata attribute of the file like stat shows. In this specific case, the average is just the length of your file, which is lesser than one whole block.”

最后一个问题是： 如果hdfs占用Linux file system的磁盘空间按实际文件大小算，那么这个”块大小“有必要存在吗？

其实块大小还是必要的，一个显而易见的作用就是当文件通过append操作不断增长的过程中，可以通过来block size决定何时split文件。以下是Hadoop Community的专家给我的回复：

“The block size is a meta attribute. If you append tothe file later, it still needs to know when to split further - so it keeps that value as a mere metadata it can use to advise itself on write boundaries.”

补充：我还查到这样一段话

原文地址：http://blog.csdn.net/lylcore/article/details/9136555

一个split的大小是由goalSize, minSize, blockSize这三个值决定的。computeSplitSize的逻辑是，先从goalSize和blockSize两个值中选出最小的那个（比如一般不设置map数，这时blockSize为当前文件的块size，而goalSize是文件大小除以用户设置的map数得到的，如果没设置的话，默认是1）。

hadooop提供了一个设置map个数的参数mapred.map.tasks，我们可以通过这个参数来控制map的个数。但是通过这种方式设置map的个数，并不是每次都有效的。原因是mapred.map.tasks只是一个hadoop的参考数值，最终map的个数，还取决于其他的因素。

为了方便介绍，先来看几个名词：

block\_size : hdfs的文件块大小，默认为64M，可以通过参数dfs.block.size设置

total\_size : 输入文件整体的大小

input\_file\_num : 输入文件的个数

（1）默认map个数

如果不进行任何设置，默认的map个数是和blcok\_size相关的。

default\_num = total\_size / block\_size;

（2）期望大小

可以通过参数mapred.map.tasks来设置程序员期望的map个数，但是这个个数只有在大于default\_num的时候，才会生效。

goal\_num = mapred.map.tasks;

（3）设置处理的文件大小

可以通过mapred.min.split.size 设置每个task处理的文件大小，但是这个大小只有在大于block\_size的时候才会生效。

split\_size = max(mapred.min.split.size, block\_size);

split\_num = total\_size / split\_size;

（4）计算的map个数

compute\_map\_num = min(split\_num, max(default\_num, goal\_num))

除了这些配置以外，mapreduce还要遵循一些原则。 mapreduce的每一个map处理的数据是不能跨越文件的，也就是说min\_map\_num >= input\_file\_num。 所以，最终的map个数应该为：

final\_map\_num = max(compute\_map\_num, input\_file\_num)

经过以上的分析，在设置map个数的时候，可以简单的总结为以下几点：

（1）如果想增加map个数，则设置mapred.map.tasks 为一个较大的值。

（2）如果想减小map个数，则设置mapred.min.split.size 为一个较大的值。

（3）如果输入中有很多小文件，依然想减少map个数，则需要将小文件merger为大文件，然后使用准则2。

# [Hive数据倾斜解决方法总结](https://www.cnblogs.com/kongcong/p/7777092.html)

      数据倾斜是进行大数据计算时最经常遇到的问题之一。当我们在执行HiveQL或者运行MapReduce作业时候，如果遇到一直卡在map100%,reduce99%一般就是遇到了数据倾斜的问题。数据倾斜其实是进行分布式计算的时候，某些节点的计算能力比较强或者需要计算的数据比较少，早早执行完了，某些节点计算的能力较差或者由于此节点需要计算的数据比较多，导致出现其他节点的reduce阶段任务执行完成，但是这种节点的数据处理任务还没有执行完成。

　　在hive中产生数据倾斜的原因和解决方法：

　　1）group by,我使用Hive对数据做一些类型统计的时候遇到过某种类型的数据量特别多，而其他类型数据的数据量特别少。当按照类型进行group by的时候，会将相同的group by字段的reduce任务需要的数据拉取到同一个节点进行聚合，而当其中每一组的数据量过大时，会出现其他组的计算已经完成而这里还没计算完成，其他节点的一直等待这个节点的任务执行完成，所以会看到一直map 100%  reduce 99%的情况。

　　解决方法：set hive.map.aggr=true

　　　　　　　set hive.groupby.skewindata=true

　　原理：hive.map.aggr=true 这个配置项代表是否在map端进行聚合

　　　　　hive.groupby.skwindata=true 当选项设定为 true，生成的查询计划会有两个 MR Job。第一个 MR Job 中，Map 的输出结果集合会随机分布到 Reduce 中，每个 Reduce 做部分聚合操作，并输出结果，这样处理的结果是相同的 Group By Key 有可能被分发到不同的 Reduce 中，从而达到负载均衡的目的；第二个 MR Job 再根据预处理的数据结果按照 Group By Key 分布到 Reduce 中（这个过程可以保证相同的 Group By Key 被分布到同一个 Reduce 中），最后完成最终的聚合操作。

　　2）map和reduce优化。

　　　　1.当出现小文件过多，需要合并小文件。可以通过set hive.merge.mapfiles=true来解决。

 　　　   2.单个文件大小稍稍大于配置的block块的大写，此时需要适当增加map的个数。解决方法：set mapred.map.tasks个数

　   　　 3.文件大小适中，但map端计算量非常大，如select id,count(\*),sum(case when...),sum(case when...)...需要增加map个数。解决方法：set mapred.map.tasks个数，set mapred.reduce.tasks个数

　　3）当HiveQL中包含count（distinct）时

         如果数据量非常大，执行如select a,count(distinct b) from t group by a;类型的SQL时，会出现数据倾斜的问题。

         解决方法：使用sum...group by代替。如select a,sum(1) from (select a, b from t group by a,b) group by a;

　　4）当遇到一个大表和一个小表进行join操作时。

　　  解决方法：使用mapjoin 将小表加载到内存中。

　  　如：select /\*+ MAPJOIN(a) \*/

    　　　　　　a.c1, b.c1 ,b.c2

　　　　　from a join b

　　　　　where a.c1 = b.c1;

　　5）遇到需要进行join的但是关联字段有数据为空，如表一的id需要和表二的id进行关联

　　   解决方法1：id为空的不参与关联

　　　　比如：select \* from log a

　　　　　　join users b

　　　　　　on a.id is not null and a.id = b.id

　　　　　　　union all

　　　　　　　select \* from log a

　　　　　　where a.id is null;

　　　解决方法2：给空值分配随机的key值

　　　　　　如：select \* from log a

　　　　　　　　left outer join users b

　　　　　　　　on

　　　　　　　　case when a.user\_id is null

　　　　　　　　then concat(‘hive’,rand() )

　　　　　　　　else a.user\_id end = b.user\_id;

# hive中控制map和reduce数量的简单实现方法

　由于mapreduce中没有办法直接控制map数量，所以只能曲线救国，通过设置每个map中处理的数据量进行设置；reduce是可以直接设置的。

控制map和reduce的参数

set mapred.max.split.size=256000000; -- 决定每个map处理的最大的文件大小，单位为B

set mapred.min.split.size.per.node=1; -- 节点中可以处理的最小的文件大小

set mapred.min.split.size.per.rack=1; -- 机架中可以处理的最小的文件大小

方法1

set mapred.reduce.tasks=10; -- 设置reduce的数量

方法2

set hive.exec.reducers.bytes.per.reducer=1073741824 -- 每个reduce处理的数据量,默认1GB

1

2

3

4

5

6

7

补充说明：一个集群可以有多个机架，一个机架有1至多个节点，这里的集群是mapreduce不是yarn，yarn没有详细了解过，另外如果想要实现map中的数据合并需要设置下面的参数，集群默认就是这个格式

set hive.input.format=org.apache.hadoop.hive.ql.io.CombineHiveInputFormat;

1

需要确认的问题：

　a.我们该设置多少个map多少个reduce才合适？

　　map数普遍是通过执行时长来确认的，至少应当保证每个map执行时长在1分钟以上，太短的话意味着大量重复的jvm启用和销毁。具体设置要根据具体任务来处理，有些任务占用cpu大，有些占用io大。

　　我这边的话因为大任务经常搞了上千个map，作为小集群影响还是蛮大的，所以只对监控到的hql产生过大的map和reduce进行调整，经过一些简单测试，map数保证在三四百个其实不影响执行效率。

　b.设置了上面的参数会带来什么影响？

　　设置map的话，影响不是很大，可能会带来更多的集群之间的io，毕竟要做节点之间的文件合并

　　设置reduce的话，如果使用mapred.reduce.tasks，这个影响就大了，至少会造成同一个session每一个mr的job的reduce都是这么多个，而且reduce个数意味着最后的文件数量的输出，如果小文件太多的话可以打开reduce端的小文件合并参数，set hive.merge.mapredfiles=true

1、控制map数量的三个参数的逻辑概念

　　可以简单的理解为集群对一个表分区下面的文件进行分发到各个节点，之后根据mapred.max.split.size确认要启动多少个map数，逻辑如下

　　a.假设有两个文件大小分别为(256M,280M)被分配到节点A，那么会启动两个map，剩余的文件大小为10MB和35MB因为每个大小都不足241MB会先做保留

　　b.根据参数set mapred.min.split.size.per.node看剩余的大小情况并进行合并,如果值为1，表示a中每个剩余文件都会自己起一个map，这里会起两个，如果设置为大于45\*1024\*1024则会合并成一个块，并产生一个map

　　如果mapred.min.split.size.per.node为10\*1024\*1024，那么在这个节点上一共会有4个map，处理的大小为(245MB,245MB,10MB,10MB，10MB，10MB)，余下9MB

　　如果mapred.min.split.size.per.node为45\*1024\*1024，那么会有三个map，处理的大小为(245MB,245MB,45MB)

　　实际中mapred.min.split.size.per.node无法准确地设置成45\*1024\*1024，会有剩余并保留带下一步进行判断处理

　　c. 对b中余出来的文件与其它节点余出来的文件根据mapred.min.split.size.per.rack大小进行判断是否合并，对再次余出来的文件独自产生一个map处理

2、控制map数量的简单实用方式

　　我们执行一个hive语句，发现起了1000个map，这种情况对于当前的数据量来说是完全不必要的，同时还会影响其它用户提交任务

　　这个时候希望map减小到250个左右，很简单

　　将map处理的最大的文件大小增大，256000000\*4=1024000000

参数修改为如下

set mapred.max.split.size=1024000000;

set mapred.min.split.size.per.node=1024000000;

set mapred.min.split.size.per.rack=1024000000;

1

2

3

示例及测试如下

这里写图片描述

　　修改参数之后，如下图，可以看到map数量减少到370，至于为什么单个map处理的数据量增大了，map数却不是按倍数减少到1185/4=296个，原因在文件合并过程中对于如何起map数并不是单纯的看总的文件大小，存在一些更优化的算法，如考虑带宽、IO、当前map分配等

这里写图片描述

3、控制reduce参数

　　修改reduce的个数就简单很多，直接根据可能的情况作个简单的判断确认需要的reduce数量，如果无法判断，根据当前map数减小10倍，保持在1~100个reduce即可(注意，这个不同的集群需求不同哈)

设置参数如下

set mapred.reduce.tasks=10

不建议随意设置reduce参数哈，可能调整参数更好一点

set hive.exec.reducers.bytes.per.reducer=1073741824

4、控制map和rduce数的说明

　　1、因为set参数的设置是session级别的，Toad for Cloud(青蛙)第三方软件中暂时没有发现如何使set的参数设置有效的，所以请使用CRT等工具连接到linux的界面中进行使用，使用hive命令连接hive集群，一次设置只要不结束这次连接，那么参数是始终有效的。

　　2、注意各参数的设置大小，不要冲突，否则会异常，大小顺序如下

mapred.max.split.size <= mapred.min.split.size.per.node <= mapred.min.split.size.per.rack

　　3、这种减少map数的行为是否能带来更短的执行时间，需要具体分析，map数也不是越少越好,减少了map数，单个map处理的数据量就上升，需要更多的时间，同时也因为需要合并节点内、节点间甚至机架之间的数据需要更多的IO和带宽。

　　参数的设置本质是根据文件情况、系统情况、数据计算情况进行的一个平衡考虑，有取有舍，我们需要遵循的规则就是：使大数据量利用合适的map数；使单个map任务处理合适的数据量

　　4、reduce数同3