一、 控制hive任务中的map数:

1. 通常情况下，作业会通过input的目录产生一个或者多个map任务。

主要的决定因素有： input的文件总个数，input的文件大小，集群设置的文件块大小(目前为128M, 可在hive中通过set dfs.block.size;命令查看到，该参数不能自定义修改)；

2. 举例：

a) 假设input目录下有1个文件a,大小为780M,那么hadoop会将该文件a分隔成7个块（6个128m的块和1个12m的块），从而产生7个map数

b) 假设input目录下有3个文件a,b,c,大小分别为10m，20m，130m，那么hadoop会分隔成4个块（10m,20m,128m,2m）,从而产生4个map数

即，如果文件大于块大小(128m),那么会拆分，如果小于块大小，则把该文件当成一个块。

3. 是不是map数越多越好？

答案是否定的。如果一个任务有很多小文件（远远小于块大小128m）,则每个小文件也会被当做一个块，用一个map任务来完成，

而一个map任务启动和初始化的时间远远大于逻辑处理的时间，就会造成很大的资源浪费。

而且，同时可执行的map数是受限的。

4. 是不是保证每个map处理接近128m的文件块，就高枕无忧了？

答案也是不一定。比如有一个127m的文件，正常会用一个map去完成，但这个文件只有一个或者两个小字段，却有几千万的记录，

如果map处理的逻辑比较复杂，用一个map任务去做，肯定也比较耗时。

针对上面的问题3和4，我们需要采取两种方式来解决：即减少map数和增加map数；

如何合并小文件，减少map数？

假设一个SQL任务：

Select count(1) from popt_tbaccountcopy_mes where pt = ‘2012-07-04’；

该任务的
inputdir /group/p_sdo_data/p_sdo_data_etl/pt/popt_tbaccountcopy_mes/pt=2012-07-04

共有194个文件，其中很多是远远小于128m的小文件，总大小9G，正常执行会用194个map任务。

Map总共消耗的计算资源： SLOTS_MILLIS_MAPS= 623,020

我通过以下方法来在map执行前合并小文件，减少map数：

set mapred.max.split.size=100000000;

set mapred.min.split.size.per.node=100000000;

set mapred.min.split.size.per.rack=100000000;

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

再执行上面的语句，用了74个map任务，map消耗的计算资源：
SLOTS_MILLIS_MAPS= 333,500

对于这个简单SQL任务，执行时间上可能差不多，但节省了一半的计算资源。

大概解释一下，100000000表示100M, set
hive.input.format=org.apache.hadoop.hive.ql.io.CombineHiveInputFormat;这个参数表示执行前进行小文件合并，

前面三个参数确定合并文件块的大小，大于文件块大小128m的，按照128m来分隔，小于128m,大于100m的，按照100m来分隔，把那些小于100m的（包括小文件和分隔大文件剩下的），

进行合并,最终生成了74个块。

如何适当的增加map数？

当input的文件都很大，任务逻辑复杂，map执行非常慢的时候，可以考虑增加Map数，来使得每个map处理的数据量减少，从而提高任务的执行效率。

假设有这样一个任务：

```
Select data_desc,
    count(1),
    count(distinct id),
    sum(case when ...),
    sum(case when ...),
    sum(...)
from a group by data_desc
```

如果表a只有一个文件，大小为120M，但包含几千万的记录，如果用1个map去完成这个任务，肯定是比较耗时的，这种情况下，我们要考虑将这一个文件合理的拆分成多个，

这样就可以用多个map任务去完成。

```
set mapred.reduce.tasks=10;

create table a_1 as

select * from a

distribute by rand(123);
```

这样会将a表的记录，随机的分散到包含10个文件的a_1表中，再用a_1代替上面sql中的a表，则会用10个map任务去完成。

每个map任务处理大于12M（几百万记录）的数据，效率肯定会好很多。

看上去，貌似这两种有些矛盾，一个是要合并小文件，一个是要把大文件拆成小文件，这点正是重点需要关注的地方，

根据实际情况，控制map数量需要遵循两个原则：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；

二、 控制hive任务的reduce数：

1. Hive自己如何确定reduce数：

reduce个数的设定极大影响任务执行效率，不指定reduce个数的情况下，Hive会猜测确定一个reduce个数，基于以下两个设定：

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(参数2，总输入数据量/参数1)

即，如果reduce的输入（map的输出）总大小不超过1G,那么只会有一个reduce任务；

如：select pt,count(1) from popt_tbaccountcopy_mes where pt = '2012-07-04' group by pt;

/group/p_sdo_data/p_sdo_data_etl/pt/popt_tbaccountcopy_mes/pt=2012-07-04 总大小为9G多，因此这句有10个reduce

2. 调整reduce个数方法一：

调整hive.exec.reducers.bytes.per.reducer参数的值；

set hive.exec.reducers.bytes.per.reducer=500000000;（500M）

select pt,count(1) from popt_tbaccountcopy_mes where pt = '2012-07-04' group by pt; 这次有20个reduce

3. 调整reduce个数方法二；

set mapred.reduce.tasks = 15;

select pt,count(1) from popt_tbaccountcopy_mes where pt = '2012-07-04' group by pt;这次有15个reduce

4. reduce个数并不是越多越好；

同map一样，启动和初始化reduce也会消耗时间和资源；

另外，有多少个reduce,就会有多少个输出文件，如果生成了很多个小文件，那么如果这些小文件作为下一个任务的输入，则也会出现小文件过多的问题；

5. 什么情况下只有一个reduce；

很多时候你会发现任务中不管数据量多大，不管你有没有设置调整reduce个数的参数，任务中一直都只有一个reduce任务；

其实只有一个reduce任务的情况，除了数据量小于hive.exec.reducers.bytes.per.reducer参数值的情况外，还有以下原因：

a) 没有group by的汇总，比如把select pt,count(1) from popt_tbaccountcopy_mes where pt = '2012-07-04' group by pt; 写成 select count(1) from popt_tbaccountcopy_mes where pt = '2012-07-04';

这点非常常见，希望大家尽量改写。

b) 用了Order by

c) 有笛卡尔积

通常这些情况下，除了找办法来变通和避免，我暂时没有什么好的办法，因为这些操作都是全局的，所以hadoop不得不用一个reduce去完成；

同样的，在设置reduce个数的时候也需要考虑这两个原则：使大数据量利用合适的reduce数；使单个reduce任务处理合适的数据量。