1. Yarn 框架下，通过rm ，am，nm这几个个管理器之间的作用，所有任务包括mapreduce任务是运行在一个资源容器 container中的，目前container只包含内存和cpu，我们只关心内存。

yarn.scheduler.minimum-allocation-mb 是container的最小值，一个任务占用的资源肯定是yarn.scheduler.minimum-allocation-mb的整数倍。

yarn.scheduler.minimum-allocation-mb对集群的并发有直接影响。

相关参数：yarn.scheduler.maximum-allocation-mb container 最小值（非常重要）

yarn.scheduler.maximum-allocation-mb container 最大值，

mapreduce.map.memory.mb map任务分配内存， 一般=container

mapreduce.reduce.memory.mb reduce任务分配内存， 一般=2\*container

mapreduce.map.java.opts map任务java虚拟机内存， 一般=0.8\*container

mapreduce.reduce.java.opts reduce任务java虚拟机内存，一般=0.8\*2\*container

另：

yarn.scheduler.minimum-allocation-mb 这个参数是要初始化就设置好的，所以修改完要重启集群。

mapreduce.task.io.sort.mb 这个是map段缓冲用的内存， 好像没啥作用。（不确定）

另外：

如果用了tez框架， tez对yarn的container有个缓存池，会初始化一些container， tez的缓存的container的大小，只能是yarn的整数倍。

参数：hive.tez.container.size。

jinhb\_text\_nocompress3 表有140个小文件，可以用下面sql来测试并发：

select count(\*), logindate from jinhb\_text\_nocompress3 group by logindate

Container 的大小非常重要， 直接影响集群并发，建议提前根据硬件，确定好并发，然后计算出container。 非常重要。

1. Hive的优化包含两发面，一是执行计划优化， 二是单个mapreduce优化。先说单个mapreduce优化。
2. 块大小：dfs.blocksize

太大了会导致map数很少， 太小了导致每个map处理很小的数据，而且对namenode 有压力。一般128m， 海量数据256m。

非常重要 初始设定好，以后不许修改。

1. 分片大小：

mapred.max.split.size ----最大分片

mapred.min.split.size ------最小分片

分片大小也很重要，太小会启用很多map，不仅消耗资源，而且速度不快。

在可能有很多小文件的时候，如果不启用map段合并，每个小文件会启用一个map，非常低效。

在tez框架下，通过设置分片大小和合并参数控制最终分片大小：

set hive.tez.input.format=org.apache.hadoop.hive.ql.io.CombineHiveInputFormat 开启map段合并（set hive.tez.input.format=org.apache.hadoop.hive.ql.io.HiveInputFormat 默认不合并）

set mapred.max.split.size=256000000；设置最大分片。

在集群有大量小文件时，非常重要。

可以用下面sql测试：

select count(\*), logindate from jinhb\_text\_nocompress3 group by logindate

1. Output合并：

为了解决集群大量小文件的问题，可以启用output合并，当输出大量下文件时，进行合并。

set hive.merge.mapfiles = true;#在Map-only的任务结束时合并小文件，map阶段Hive自动对小文件合并。

set hive.merge.mapredfiles = true;#默认false， true时在MapReduce的任务结束时合并小文件

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

hive.merge.smallfiles.avgsize：在作业输出文件小于该值时，起一个额外的map/reduce作业将小文件合并为大文件，小文件的基本阈值，设置大点可以减少小文件个数，需要mapfiles和

mapredfiles为true，默认值是16MB；

hive.merge.tezfiles =true; ---------------

用下面sql来测试：

set mapred.reduce.tasks=10; ----设置reduce数，每个reduce一个小文件

insert overwrite table jinhb\_result

select count(\*), logindate from jinhb\_small\_zip group by logindate

1. Map压缩;

启用就好：

set hive.exec.compress.intermediate=true; -----启用压缩

set mapred.map.output.compression.codec=com.hadoop.compression.lzo.LzoCodec;---压缩算法

1. Reduce压缩：

如果采用text这种文件，启用reduce压缩能大大减小文件。

注意选择一个可分割的压缩算法，不然一个map得读取整个文件

set hive.exec.compress.output=true; ------启用压缩

set mapred.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec; ---压缩算法

1. Orc文件;

Orc是一个很高效的列式存储文件，文件更小，速度更快:

set hive.exec.orc.default.stripe.size; -----stripe的大小, 一般=块大小。

建表语法：

CREATE TABLE jinhb\_orc\_compress

STORED AS orc -----文件格式

tblproperties ("orc.compress"="SNAPPY") ----压缩方式

as select \* from jinhb\_text\_nocompress

distribute by id -----可选

sort by id --------排序对某些利用排序列的查询，可以使用orc的索引， 特别高效

set hive.optimize.index.filter=true; ---------启用索引

1. 还有个参数：hive.exec.reducers.bytes.per.reducer=1g

一般情况reduce的数量不手动控制，而是靠这个参数，每个reduce处理1g的数据。

1. Sql优化： sql优化基础查看执行计划：

Explain sql

Explain extended sql ---- 更详细

非常重要，特别重要，是sql调优的基础

1. 一个参数：切换hive的运行框架

set hive.execution.engine=tez;

set hive.execution.engine=mr

1. 动态分区， 动态分区很方便，但是要开启才能用：

set hive.exec.dynamic.partition=true ----开启动态分区

set hive.exec.dynamic.partition.mode=nostrict ---strict模式纯动态分区是不允许的

set hive.exec.max.dynamic.partitions.pernode ------下面都是控制分区数

set hive.exec.max.dynamic.partitions

set hive.exec.max.created.files

测试sql：

CREATE TABLE jinhb\_partition(

id INT,

userid string,

username string,

logindate string,

logintime string,

appid string,

openid string,

yhtchange string,

mobile string,

uuid string,

systag string,

corporationid string,

corpname string,

paid string,

status string,

region string,

industry string

) partitioned by (mon string)

stored as orc

insert overwrite table jinhb\_partition partition(mon)

select a.\* , substr(logindate,1,6) as mon from jinhb\_orc\_compress a

1. 连接优化：

hive.mapjoin.smalltable.filesize：输入表文件的mapjoin阈值，如果输入文件的大小小于该值，则试图将普通join转化为mapjoin，默认25MB；------好像无效了；

hive.auto.convert.join.noconditionaltask.size --------新版本好像是这个参数来控制map连接小文件的大小。

set hive.auto.convert.join.noconditionaltask； ------据说有用，但是好像也没啥效果，不能控制是否开启map连接。 默认成true 就好了。（对tez无效，对mr有效）

hive.auto.convert.join：根据输入文件的大小决定是否将普通join转换为mapjoin的一种优化，默认不开启false； 需要设置成true；

1. 分桶

分桶是一个非常好用的功能，

CREATE TABLE jinhb\_bucket\_small(

id INT,

userid string,

username string,

logindate string,

logintime string,

appid string,

openid string,

yhtchange string,

mobile string,

uuid string,

systag string,

corporationid string,

corpname string,

paid string,

status string,

region string,

industry string

) CLUSTERED BY (id) SORTED BY (id ASC) INTO 4 BUCKETS;

----stored as orc

CREATE TABLE jinhb\_bucket\_big(

id INT,

userid string,

username string,

logindate string,

logintime string,

appid string,

openid string,

yhtchange string,

mobile string,

uuid string,

systag string,

corporationid string,

corpname string,

paid string,

status string,

region string,

industry string

) CLUSTERED BY (id) SORTED BY (id ASC) INTO 4 BUCKETS;

---stored as orc

hive.enforce.bucketing：数据分桶是否被强制执行，默认false，如果开启，则写入table数据时会启动分桶，

insert overwrite table jinhb\_bucket\_small

select \* from jinhb\_small2

insert overwrite table jinhb\_bucket\_big

select \* from jinhb\_orc\_compress

1. . 如果是小表连接大表， 直接mapjoin。

如果是不太大但是又不小的表连接大表， 分桶，然后做连接。

大表小表要根据连接字段分桶， 而且分桶数 是整数倍 关系。

set hive.convert.join.bucket.mapjoin.tez=true; ------------------------开启bucket mapjoin（tez用这个参数）(开启这个参数，好像就不能做merge join了)

大表小表的分桶只要成倍数就行，谁大谁小都能做mapjoin。

1. 再次， sql优化的基础是观察理解执行计划。

多用explain sql

单个mapreduce优化，需要详细理解mapreduce的过程，