添加jar包：

add jar /usr/hdp/2.4.0.0-169/hive/aux\_lib/fastjson-1.2.7.jar;

add jar /usr/hdp/2.4.0.0-169/hive/aux\_lib/demo-hive-0.0.1-SNAPSHOT.jar;

创建函数：

create temporary function row\_json as 'com.tx.demo\_hive.UDF\_JSON';

create temporary function col\_json as 'com.tx.demo\_hive.UDAF\_COL\_JSON';

建表结构

create table cluss(class\_name string,student\_id int,student\_name string)

row format delimited fields terminated by '\001' stored as textfile;

建临时表存放行json查询：

create temporary table json\_1 as

select class\_name as class\_name

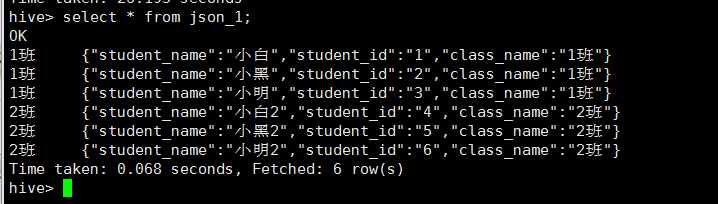
,row\_json(concat('class\_name:',class\_name)

,concat('student\_id:',student\_id)

,concat('student\_name:',student\_name)) as rowJson

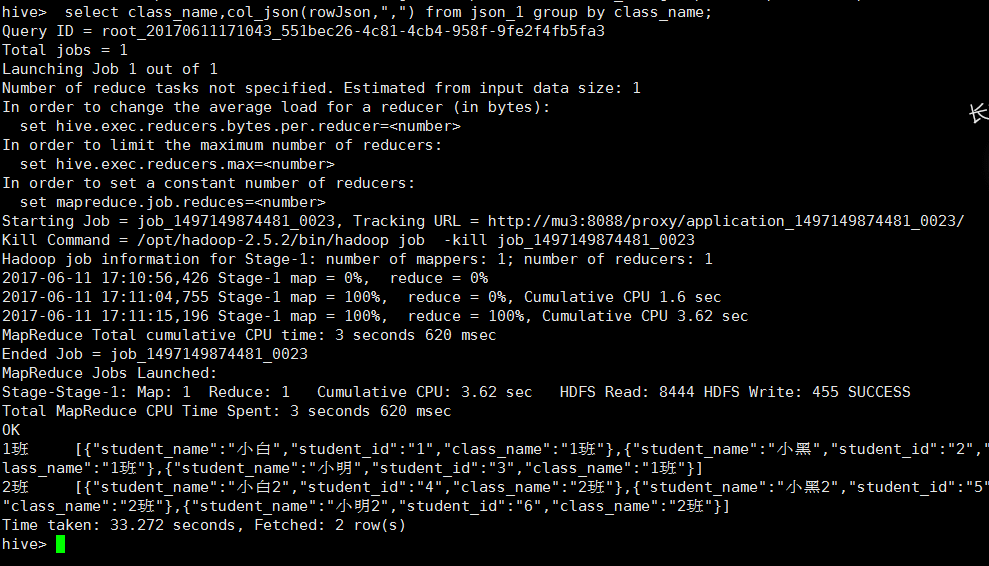
from cluss;

Select \* from json\_1;



按class\_name聚合：

select class\_name,col\_json(rowJson,",") from json\_1 group by class\_name;



Bigdatabench大数据测试工具

YCSB

第一步：./prepare 准备测试环境

hadoop dfs -mkdir /data-PageRank

--color=[true/false]

set mapreduce.job.queuename=queueName

**本地模式:**

有时hive的输入数据量是非常小的。在这种情况下，为查询出发执行任务的时间消耗可能会比实际job的执行时间要多的多。对于大多数这种情况，hive可以通过本地模式在单台机器上处理所有的任务。对于小数据集，执行时间会明显被缩短。

配置如下参数，可以开启Hive的本地模式：  
hive> set hive.exec.mode.local.auto=true;(默认为false)

当一个job满足如下条件才能真正使用本地模式：  
1.job的输入数据大小必须小于参数：hive.exec.mode.local.auto.inputbytes.max(默认128MB)  
2.job的map数必须小于参数：hive.exec.mode.local.auto.tasks.max(默认4)  
3.job的reduce数必须为0或者1

但是你会发现job确实是以本地模式运行了（看job名字就能看出来,中间有local字样）,但是还是会报错，各种找不到jar包。

这里还要运行一个语句：set fs.defaultFS=file:///

常用语法:

查看分区: show partitions people;

修改表备注:

ALTER TABLE table\_name SET TBLPROPERTIES('comment' = new\_comment);

修改分隔符: alter table table\_name set serdeproperties(‘filed.delim’=’’);

重命名分区: ALTER TABLE table\_name PARTITION partition\_spec RENAME TO PARTITION partition\_spec;

删除既存的指定列  
ALTER TABLE name DROP [COLUMN] column\_name

修改列信息: ALTER TABLE employee CHANGE name ename String;  
ALTER TABLE employee REPLACE COLUMNS (eid INT empid Int,ename STRING name String);

hive> alter table ljn005 SET SERDEPROPERTIES('serialization.null.format' = 'a');  
alter table ljn005 SET SERDEPROPERTIES('field.delim'='\t');  
alter table tablename change column c\_Old c\_New int comment 'XXXXXX'  
after severity;//可以把该列放到指定列的后而，或者使用’first’放到第一位。

修改表格式：

ALTER TABLE foo SET FILEFORMAT   
 INPUTFORMAT "com.hadoop.mapred.DeprecatedLzoTextInputFormat"   
 OUTPUTFORMAT "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutpu

重命名分区：

ALTER TABLE table\_name PARTITION partition\_spec RENAME TO PARTITION partition\_spec;

hive中四个hook  
使用hive.semantic.analyzer.hook 可以用来修改权限

Hive乱码：

1 进入mysql ，执行

show create database hive

查看hive 数据库当前编码，如果是utf8 则执行下面sql

alter database hive default character set latin1

将 hive 数据库默认编码改成 latin1

2 执行下面sql ，修改表

alter database hive default character set latin1 ;

use hive;

alter table COLUMNS\_V2 modify column COMMENT varchar(256) character set utf8;

alter table TABLE\_PARAMS modify column PARAM\_VALUE varchar(4000) character set utf8;

alter table PARTITION\_PARAMS modify column PARAM\_VALUE varchar(4000) character set utf8;

alter table PARTITION\_KEYS modify column PKEY\_COMMENT varchar(4000) character set utf8;

alter table INDEX\_PARAMS modify column PARAM\_VALUE varchar(4000) character set utf8;

**运行时参数:**

set system:user.name=hive;  
set hive.execution.engine=tez; 来启动Tez  
mapreduce.reduce.memory.mb=5120 //设置reduce container的内存大小  
mapreduce.reduce.java.opts=-Xms2000m -Xmx4600m; //设置reduce任务的JVM参数

**Map任务个数：**

**调优:**

1. Hadoop的主要性能瓶颈是IO负载，降IO负载是优化的重头戏。   
   2、对中间结果的压缩   
   3、合理设置分区，静态分区和动态分区

**优化方法** （从任务数量和数据倾斜考虑，任务数量合适，数据分布均匀才好）  
合并小文件 （起太多小任务浪费时间）  
避免数据倾斜,解决数据倾斜 (任务数据分配不均，不能合理利用资源)  
减少job数据（合并job，大job的拆分…）

**优化手段:**

1,合理控制Map和Reduce数

1》减少map数量可以通过合并小文件，这点是针对文件源

2》增加map数可通过控制上一个作业的reduce数量来控制

**输出合并：**

set hive.merge.mapfiles = true;（默认）#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

set hive.merge.smallfiles.avgsize=16000000 #当输出文件的平均大小小于该值时，启动一个独立的map-reduce任务进行文件merge

**输入合并：**

set mapred.max.split.size = 256000000; #每个Map最大(块)分割大小（hadoop）

set mapred.min.split.size.per.node = l00000000; #一个节点上split的最小值

set mapred.min.split.size.per.rach决定了多个交换机上的文件是否需要合并

**set hive.input.format =org.apache.hadoop.hive.ql.io.CombineHiveInputFormat;**#执行Map前进行小文件合并。合并文件数大小由 mapred.max.split.size 限制的大小决定。

set mapred.map.tasks=100;设置map个数，要看大小是否合理

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

Set hive.exec.reducer.max=<number> reduce最大数量

set mapreduce.job.reduces=<number> reduce固定数量

set mapred.reduce.tasks=10 有group by才会生效....

mapreduce.job.reduces=2 和上面同时出现，这个生效

**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reduce个数的参数，任务中一直都只有一个reduce任务（会产生数据倾斜）

**原因：**   
1、数据量小于hive.exec.reducers.bytes.per.reducer参数值（有时，通常情况下设置reduce个数会起作用）   
2、没有group by的汇总   
3、用了order by

**数据倾斜：**

数据分布不均，造成热点

症状：map快，reduce非常慢。

某些map快，某些满。

某些reduce快，某些reduce奇慢。

数据倾斜场景：数据在节点上分布不均(无法避免)

Join时on关键词中个别值数量很大

Count(distinct)，数据量大的时候容易数据倾斜

使用select count(\*) from (select distinct id from table) t;

在使用SUM，COUNT，MAX，MIN等UDAF函数时，不怕数据倾斜问题，Hadoop在Map端的汇总合并优化过，使数据倾斜不成问题。

select unix\_timestamp(),row\_number() over() from tbls limit 10;

**解决方案:**

**参数调节:**

hive.map.aggr=true map端做聚合，相当于combine

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）

hive.groupby.skewindata=true 有数据倾斜的时候进行负载均衡，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。

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

set hive.exec.parallel=True 是否并行执行job

set hive.exec.parallel.thread.number=8

**Sql语句调节**

1. 关于驱动表的选取：选用join key分布最均匀的表作为驱动表

2，做好列裁剪和filter操作，以达到两表做join的时候，数据量相对变小的效果。

3，大表与小表Join   
使用map join让小的维度表（1000条以下的记录条数）先进内存。在map端完成reduce。

4，大表与大表Join：   
把空值的key变成一个字符串加上随机数，把倾斜的数据分到不同的reduce上，由于null值关联不上，处理后并不影响最终结果。

5，count distinct大量相同特殊值：使用roup by代替

不要select count(distinct order\_no) from order\_snap;

要select count(t.order\_no) from (select order\_no from order\_snap group by order\_no) t;

Count(distinct)只shuffle一个reduce  
count distinct时，将值为空的情况单独处理，如果是计算count distinct，可以不用处理，直接过滤，在最后结果中加1。如果还有其他计算，需要进行group by，可以先将值为空的记录单独处理，再和其他计算结果进行union。

6，group by维度过小：   
采用sum() 与group by的方式来替换count(distinct)完成计算。

**Count(distinct)**

select count(distinct user\_id) from dm\_user where ds=20150701;

使用disticnt函数，所有的数据只会shuffle到一个reducer上，导致reducer数据倾斜严重

优化后为

set mapred.reduce.tasks=50;

select count(\*) from

(select user\_id from dm\_user where ds=20150701 group by user\_id)t;

order by全局排序，只有一个reduce

sort by 在一个reduce中排序，distribute by 按字段分为不同的reduce

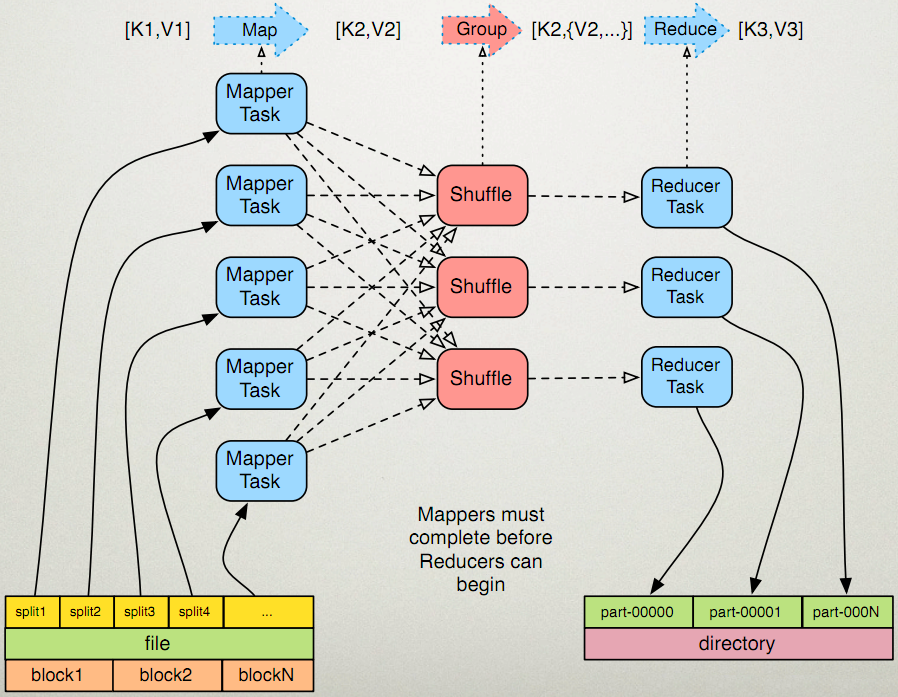
distribute by 先分为不同的reduce，之后在reduce内部排序

distribute by  (类似于分桶)   
根据distribute by指定的字段对数据进行划分到不同的输出reduce 文件中

sort by   (类似于桶内排序)   
sort by不是全局排序，其在数据进入reducer前完成排序。   
因此，如果用sort by进行排序，并且设置mapred.reduce.tasks>1， 则sort by只保证每个reducer的输出有序，不保证全局有序。

cluster by   
cluster by 除了具有 distribute by 的功能外还兼具 sort by 的功能。    
但是排序只能是倒序排序，不能指定排序规则为asc 或者desc。

因此，常常认为cluster by = distribute by + sort by



**Map任务个数并不一定是分片数：**

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。

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

**测试**：Sequencefile,snappy压缩

file=259M,block数量为3

Dfs.block.size=128M

查询语句：Select max() from table xxx

1，直接查询 map数:2，不是3

2，set mapred.map.tasks=3,4都不好使，还是2个map任务

**压缩配置：**

  map/reduce 输出压缩（一般采用序列化文件存储Stored AS SEQUENCEFILE;，text格式的话只会有一个map...）

默认是RECORD类型，它会按单个的record压缩，如果指定为BLOCK类型，它将一组record压缩，压缩效果自然是BLOCK好。

set hive.exec.compress.output=true;

set mapred.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec;

set mapred.output.compression.type=BLOCK;

**任务中间压缩：**

set hive.exec.compress.intermediate=true;

set hive.intermediate.compression.codec=org.apache.hadoop.io.compress.SnappyCodec;（常用）

set hive.intermediate.compression.type=BLOCK;

**总结：**

数据仓库的特点：一次写入、多次读取，因此，整体来看，ORCFile相比其他格式具有较明显的优势。

TextFile 默认格式，加载速度最快，可以采用Gzip、bzip2等进行压缩，压缩后的文件无法split，即并行处理

SequenceFile 压缩率最低，查询速度一般，三种压缩格式NONE，RECORD，BLOCK

RCfile 压缩率最高，查询速度最快，数据加载最慢。

从压缩及查询的空间和时间性能上来说，DefaultCodeC + RCFile的压缩方式均为最优，但使用该方式，会使得Pig 和Impala 无法使用（Impala的不兼容不确定是否是暂时的）。

而DefaultCodeC+ SequenceFile 在压缩比，查询性能上略差于RCFile (压缩比约 6:5), 但可以支持 Impala实时查询。

在hive中使用压缩需要灵活的方式，如果是数据源的话，采用RCFile+bz或RCFile+gz的方式，这样可以很大程度上节省磁盘空间；  
 而在计算的过程中，为了不影响执行的速度，可以浪费一点磁盘空间，建议采用RCFile+snappy的方式，这样可以整体提升hive的执行速度。  
 至于lzo的方式，也可以在计算过程中使用，只不过综合考虑（速度和压缩比）还是考虑snappy适宜。

**高可用HA：**

Hive从0.14开始，使用Zookeeper实现了HiveServer2的HA功能（ZooKeeperService Discovery），Client端可以通过指定一个nameSpace来连接HiveServer2，而不是指定某一个host和port。

hive-site.xml配置

<property>

<name>hive.server2.support.dynamic.service.discovery</name>

<value>true</value>

</property>

<property>

<name>hive.server2.zookeeper.namespace</name>

<value>hiveserver2\_zk</value>

</property>

<property>

<name>hive.zookeeper.quorum</name>

<value>zkNode1:2181,zkNode2:2181,zkNode3:2181</value>

</property>

<property>

<name>hive.zookeeper.client.port</name>

<value>2181</value>

</property>

<property>

<name>hive.server2.thrift.bind.host</name>

<value>0.0.0.0</value>

</property>

<property>

<name>hive.server2.thrift.port</name>

<value>10001</value> //两个HiveServer2实例的端口号要一致

</property>

Hiveserver2启动

nohup hive --service hiveserver2

--hiveconf hive.server2.thrift.port=10001

--hiveconf hive.metastore.uris='thrift://192.168.10.34:9083,thrift://192.168.10.35:9084'

--hiveconf hive.server2.zookeeper.namespace=hiveserver2/ssjf

--hiveconf hive.log.dir=/hive/logs/server2\_10001

--hiveconf hive.exec.scratchdir=/user/ssjf/tmpdata &

其中：

指定端口:10001

指定metastore的uri：thrift://192.168.10.34:9083,thrift://192.168.10.35:9084，

指定server2的轮询分组命名空间:hiveserver2/ssjf

指定日志目录:/hive/logs/server2\_10001

指定临时文件目录:/user/ssjf/tmpdata

JDBC连接

JDBC连接的URL格式为：

jdbc:hive2://<zookeeper quorum>/<dbName>;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2

其中：

<zookeeper quorum> 为Zookeeper的集群链接串，如zkNode1:2181,zkNode2:2181,zkNode3:2181

<dbName> 为Hive数据库，默认为default

serviceDiscoveryMode=zooKeeper 指定模式为zooKeeper

zooKeeperNamespace=hiveserver2 指定ZK中的nameSpace，即参数hive.server2.zookeeper.namespace所定义，我定义为hiveserver2/ssjf

连接hiveserver2

客户端轮询方式连接

beeline -u "jdbc:hive2://jfhadoop005:9501,jfhadoop006:9501,jfhadoop007:9501/ssjf;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2/ssjf;?mapreduce.job.queuename=root.ssfj"

-n ssjf

-pXXXX

-d org.apache.hive.jdbc.HiveDriver

--hiveconf property=value

--hivevar name=value

--color=[true/false]

set hive.plan.serialization.format=javaXML;序列化

 动态分区可通过设定hive.exec.dynamic.partition=true打开DP特性。使用方法：

  INSERT OVERWRITETABLE tbl partition (col1[=value][, col2[=value] …])

   使用hive.exec.dynamic.partition.mode = nonstrict动态分区有一定风险，包括小文件、覆盖数据等。默认分区开关：

    hive.exec.default.dynamic.partition.name

只需要打开hive.enforce.sorting选项即可。这一特性对Sorted merge bucket (map) join非常有用

如果MapJoin中的表都是有序的，这一特性使得Join操作无需扫描整个表，这将大大加速Join操作。可通过

     hive.optimize.bucketmapjoin.sortedmerge=true开启这个功能，获得高的性能提升。

1. **set** hive.mapjoin.cache.numrows=10000000;
2. **set** hive.mapjoin.**size**.**key**=100000;
3. **Insert** overwrite **table** pv\_users
4. **Select** /\*+MAPJOIN(pv)\*/ pv.pageid,u.age
5. **from** page\_view pv
6. join user u **on** (pv.userid=u.userid;

**Hive多分割符问题**

将下面的配置语句，加在配置文件： $HIVE\_INSTALL/conf/[Hive](http://lib.csdn.net/base/hive)-site.xml中，value中hive-contrib-\*.jar的路径为你机器上实际的放置，在$HIVE\_INSTALL/lib目录下寻找。

<property>  
  <name>hive.aux.jars.path</name>  
  <value>file:///home/develop/hive-0.8.0/lib/hive-contrib-0.8.0.jar< /value>  
</property>

create external table test (c1 string, c2 string,c3 string) row format SERDE'org.apache.hadoop.hive.contrib.serde2.MultiDelimitSerDe' with serdeproperties( "field.delim"="^|~" );

**Hive存储格式：**

**Parquet:**

set parquet.compression=SNAPPY 或set parquet.compression=GZIP

压缩比和orc差不多

**Text:**

**Orc:**索引，列式存储，高压缩比

二进制数据，文件元数据在尾部，先读取尾部数据，文件分stripe(group row data),其包含min,max,mean统计信息(row group index/min-max index),为了使Row Group Index有效利用，向表中加载数据时，必须对需要使用索引的字段进行排序，否则，min/max会失去意义。另外，这种索引通常用于数值型字段的查询过滤优化上。

CREATE TABLE lxw1234\_orc2 stored AS ORC

TBLPROPERTIES

('orc.compress'='SNAPPY',

'orc.create.index'='true',

'orc.bloom.filter.fpp'='0.05',

'orc.stripe.size'='10485760',

'orc.row.index.stride'='10000')

AS

SELECT CAST(siteid AS INT) AS id,

pcid

FROM lxw1234\_text

DISTRIBUTE BY id sort BY id;

查询未使用索引：

SELECT COUNT(1) FROM lxw1234\_orc1 WHERE id >= 1382 AND id <= 1399;

使用索引：先根据Row Group Index中的min/max，判断哪些stripes/file包含在内，接着逐行扫描，过滤pcid IN (‘0005E26F0DCCDB56F9041C’,’A’)的记录。

set hive.optimize.index.filter=true;

SELECT COUNT(1) FROM lxw1234\_orc1 WHERE id >= 1382 AND id <= 1399;

可以看到，没有全表扫描，跳过了一部分stripes。这样看来，如果where后面的id范围很大，完全可能会包含所有的文件，再根据pcid过滤时候，又相当于全表扫描了。

对于这种查询场景的优化策略，就是下面的BloomFilter索引。

在建表时候，通过表参数”orc.bloom.filter.columns”=”pcid”来指定为那些字段建立BloomFilter索引，这样，在生成数据的时候，会在每个stripe中，为该字段建立BloomFilter的数据结构，当查询条件中包含对该字段的=号过滤时候，先从BloomFilter中获取是否包含该值，如果不包含，则跳过该stripe.

CREATE TABLE lxw1234\_orc2 stored AS ORC

TBLPROPERTIES

('orc.compress'='SNAPPY',

'orc.create.index'='true',

"orc.bloom.filter.columns"="pcid",

'orc.bloom.filter.fpp'='0.05',

'orc.stripe.size'='10485760',

'orc.row.index.stride'='10000')

AS

SELECT CAST(siteid AS INT) AS id,

pcid

FROM lxw1234\_text

DISTRIBUTE BY id sort BY id;

然后执行上面的查询：

SET hive.optimize.index.filter=true;

SELECT COUNT(1) FROM lxw1234\_orc1 WHERE id >= 0 AND id <= 1000

AND pcid IN ('0005E26F0DCCDB56F9041C','A');

**使用ocr**

CREATE TABLE ... STORED AS ORC

ALTER TABLE ... [PARTITION partition\_spec] SET FILEFORMAT ORC

SET hive.default.fileformat=Orc

//sqoop import直接创建hive orc表并加载数据 sqoop-import --connect jdbc:oracle:thin:@10.10.11.16:1521/esgyn --username system --password 12345 --table TEST\_GROUPBY --split-by A --create-hcatalog-table --hcatalog-table hive\_orc2 --hcatalog-storage-stanza 'stored as orc tblproperties ("orc.compress"="SNAPPY")'

Hive分桶表：

create table student(sno int, sclass string,sname string,sex string,sage int) clustered by(sno,sclass) into **3** buckets row format delimited fields terminated by ',';

set hive.enforce.bucketing = true;强制分桶。

#设置变量,设置分桶为true, 设置reduce数量是分桶的数量个数set hive.enforce.bucketing = true;set mapreduce.job.reduces=**4**;

#开会往创建的分通表插入数据(插入数据需要是已分桶, 且排序的)

#可以使用distribute by(sno) sort by(sno asc) 或是排序和分桶的字段相同的时候使用Cluster by(字段)

#注意使用cluster by 就等同于分桶+排序(sort)

insert into table stu\_buckselect sno,sdept,sname,sex,sage from student distribute by(sno) sort by(sno asc);

alter table btest3 clustered by(name,age) sorted by(age) into 10 buckets;

Hive中的抽样查询

　　select \* from table\_name tablesample(bucket X out of Y on field);

　　X表示从哪个桶中开始抽取，Y表示相隔多少个桶再次抽取。

　　Y必须为分桶数量的倍数或者因子，比如分桶数为6，Y为6，则表示只从桶中抽取1个bucket的数据；若Y为3，则表示从桶中抽取6/3（2）个bucket的数据

把表或分区划分成bucket有两个理由

1，更快，桶为表加上额外结构，链接相同列划分了桶的表，可以使用map-side join更加高效。

2，取样sampling更高效。没有分区的话需要扫描整个数据集。

这些小文件可以单独排序。如果另外一个表也按照同样的规则分成了一个个小文件。两个表join的时候，就不必要扫描整个表，只需要匹配相同分桶的数据即可。效率当然大大提升。

同样，对数据抽样的时候，也不需要扫描整个文件。只需要对每个分区按照相同规则抽取一部分数据即可。

**LLAP配置**

**yarn队列配置**

要为llap分配一条队列，这条队列有几个要注意的地方.

最大容量和最少容量必须相等，容量一定要足够，即大于hive的hive.llap.daemon.yarn.containter.mb参数加上一个最少的container 容量（driver）。

Priority 设为大于其他队列的值，默认为0，设为1就行

**hive配置**

首先配置hive.llap.daemon.yarn.containter.mb参数。这个参数要大于yarn 的minimum container size小于maximum container size。

配置hive.llap.io.memory.size参数，该参数值可以等于hive.llap.daemon.yarn.containter.mb参数乘以10%。

配置LLAP Daemon Heap Size 参数，该参数值可以等于hive.llap.daemon.yarn.containter.mb参数乘以80%。

hiveServer interactive heap size 2048 M

llap daemon container max headroom 4000 M

llap daemon heap size 7000

memory per daemon 8000 M

1.下列Hive参数对Spark同样起作用。

set hive.exec.dynamic.partition=true; // 是否允许动态生成分区

set hive.exec.dynamic.partition.mode=nonstrict;/ 是否容忍指定分区全部动态生成

set hive.exec.max.dynamic.partitions = 100; // 动态生成的最多分区数

2.运行行为

set spark.sql.autoBroadcastJoinThreshold; // 大表 JOIN 小表，小表做广播的阈值默认值为26214400（25M）

set spark.dynamicAllocation.enabled; // 开启动态资源分配

set spark.dynamicAllocation.maxExecutors; //开启动态资源分配后，最多可分配的Executor数

set spark.dynamicAllocation.minExecutors; //开启动态资源分配后，最少可分配的Executor数

set spark.sql.shuffle.partitions; // 需要shuffle是mapper端写出的partition个数,默认设置为2000

set spark.sql.adaptive.enabled; // 是否开启调整partition功能，如果开启，spark.sql.shuffle.partitions设置的partition可能会被合并到一个reducer里运行

set spark.sql.adaptive.shuffle.targetPostShuffleInputSize; //开启spark.sql.adaptive.enabled后，两个partition的和低于该阈值会合并到一个reducer

set spark.sql.adaptive.minNumPostShufflePartitions; // 开启spark.sql.adaptive.enabled后，最小的分区数

set spark.hadoop.mapreduce.input.fileinputformat.split.maxsize; //当几个stripe的大小大于该值时，会合并到一个task中处理

hive.exec.orc.split.strategy参数控制在读取ORC表时生成split的策略

BI策略以文件为粒度进行split划分；ETL策略会将文件进行切分，多个stripe组成一个split；HYBRID策略为：当文件的平均大小大于hadoop最大split值（默认256 \* 1024 \* 1024）时使用ETL策略，否则使用BI策略。

对于一些较大的ORC表，可能其footer较大，ETL策略可能会导致其从hdfs拉取大量的数据来切分split，甚至会导致driver端OOM，因此这类表的读取建议使用BI策略。

对于一些较小的尤其有数据倾斜的表（这里的数据倾斜指大量stripe存储于少数文件中），建议使用ETL策略。

spark.hadoop.mapreduce.input.fileinputformat.split.maxsize参数可以控制在ORC切分时stripe的合并处理。具体逻辑是，当几个stripe的大小大于spark.hadoop.mapreduce.input.fileinputformat.split.maxsize时，会合并到一个task中处理。可以适当调小该值，如set spark.hadoop.mapreduce.input.fileinputformat.split.maxsize=134217728。以此增大读ORC表的并发。

3.executor能力

set spark.executor.memory; // executor用于缓存数据、代码执行的堆内存以及JVM运行时需要的内存

set spark.yarn.executor.memoryOverhead; //Spark运行还需要一些堆外内存，直接向系统申请，如数据传输时的netty等。

set spark.sql.windowExec.buffer.spill.threshold; //当用户的SQL中包含窗口函数时，并不会把一个窗口中的所有数据全部读进内存，而是维护一个缓存池，当池中的数据条数大于该参数表示的阈值时，spark将数据写到磁盘

set spark.executor.cores; //单个executor上可以同时运行的task数

4.GC优化

打开GC打印：-verbose:gc -XX:+PrintGCDetails -XX:+PrintGCTimeStamps

full GC 频繁：内存不够用，调大spark.executor.memory，调小spark.executor.cores。

minor GC频繁，而full GC比较少：可以适当提高Eden区大小-Xmn

如果OldGen区快要满了，适当提高spark.executor.memory（默认2G）或适当降低spark.memory.fraction（默认为0.3）或适当提高-XX:NewRatio（老年代是年轻代的多少倍，一般默认是2）。

如果spark.executor.memory调的很大且GC仍是程序运行的瓶颈，可以尝试启用G1垃圾回收器（-XX:+UseG1GC）

修改了GC的参数一定要仔细观察GC的频率和时间。

修改方法：set spark.executor.extraJavaOptions="-XX:NewRatio=3 -XX:+UseG1GC ..."