# 回顾

1 什么是分区表？

就是在HDFS上创建独立的目录，按照业务需求把数据存放在不同的目录中，方便查询，和提高检索效率，避免全表扫描

2 什么是分桶表？

是基于数据划分的，主要区别与分区表

3 管理表与外部表

4 数据导入方式

5 数据导出

# 今天预计讲解内容

1 排序 （重点）

2 压缩和存储

3 函数 如果实现一个自定义函数

4 hive的企业级调优

## 具体内容

## 3.1 hive中的排序

### 3.1.1 全局排序

Order by ：全局排序 在一个MapReduce中进行排序

以上这种操作，在实际工作中要慎用，因为使用的如果不是数据库，而是数据仓库软件例如现在的hive ,操作的是数据仓库执行的数据量是非常庞大的，如果使用了全局排序则可能资源的严重消耗

ASC： 升序 也是默认的

DESC: 降序

范例： 查询员工信息，按照工资进行排序

|  |
| --- |
| select \* from emp order by sal ; |

### 3.1.2 每个MapReduce的内部排序

关键字：sort by :每个mapreduce的内部排序,对全局结果集来说并不是排序

实现步骤：

1 设置reduce个数：

|  |
| --- |
| hive (test3)> set mapreduce.job.reduces=3; |

2 根据部门编号降序查看员工信息

|  |
| --- |
| select \* from emp sort by deptno desc; |

对于以上的语句，是看不到分区排序的效果的，所以可以使用导出语句把结果导出到本地来查看

3 把查询结果导出到文件中

|  |
| --- |
| insert overwrite local directory '/data/sort\_emp'  select \* from emp sort by deptno desc; |

### 3.1.3 分区排序

关键字： distribute by : 类似MR中的partition ,进行分区的时候使用的，一般结合sort by 关键字使用

注意： distribute by 语句必须写在 sort by 之前

对于distribute by 的测试的时候必须给定多个reduce 不然是没有办法分区的

范例： 先按照部门编号分区，在按照员工编号排序

1 必须要设置reduce的数量不为1 （为多个）

2 语句的实现

|  |
| --- |
| insert overwrite local directory '/data/sort/emp2'  select \* from emp distribute by deptno sort by empno desc; |

### 3.1.4 cluster by

当distribute by与sort by 字段相同的时候，可以使用cluster by 关键字

能分区 也可以排序 缺点： 只能对一个字段使用

|  |
| --- |
| insert overwrite local directory '/data/sort/emp4'  select \* from emp cluster by deptno; |

不能指定 asc 或者desc 默认降序排序

## 3.2 Hive的压缩与存储

回顾Hadoop的压缩

对于Hadoop支持的压缩编码格式：

1 default 不可被切分

2 gzip 不可被切分

3 bzip2 可以被切分

4 lzo 可以被切分

5 snappy 不可切分

对于以上的压缩的方式的性能比较

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 压缩算法 | 文件大小 | 压缩之后文件大小 | 压缩速度 | 解压速度 |
| gzip | 10G B | 2GB | 17MB/s | 50MB/s |
| Bzip2 | 10GB | 1.3GB | 3MB/s | 10MB/sl |
| lzo | 10GB | 3GB | 50MB/s | 80MB/s |
| snappy | 10GB | 3GB | 250MB/s | 500MB/s |

### 3.2.1 在hive中设置压缩

在hive可以在两个阶段来设置压缩：

1 在map阶段，可以减少job中map到reduce的数据间的传输量

2 在reduce阶段，hive在导出数据到本地或者hdfs上的时候也可以对内容进行压缩

范例：压缩的实现步骤：

1. Map的压缩输出开启

开启hive中间传输压缩的功能

|  |
| --- |
| set hive.exec.compress.intermediate=true; |

1. 开启mapreduce中map的输出压缩功能

|  |
| --- |
| set mapreduce.map.output.compress=true; |

1. 设置map输出数据压缩的方式

|  |
| --- |
| set mapreduce.map.output.compress.codec=org.apache.hadoop.io.compress.DefaultCodec |

1. 执行查询语句

|  |
| --- |
| select count(\*) from emp; |

5 ）开启reduce的阶段压缩

1. 开启hive的最终输出压缩功能

|  |
| --- |
| set hive.exec.compress.output=true; |

1. 开启mapreduce最终的输出压缩

|  |
| --- |
| set mapreduce.output.fileoutputformat.compress=true; |

8 ） 设置最终输出压缩方式

|  |
| --- |
| set mapreduce.output.fileoutputformat.compress.codec=org.apache.hadoop.io.compress.DefaultCodec |

9 ) 设置最终数据输出的压缩类型 为block | record

|  |
| --- |
| set mapreduce.output.fileoutputformat.compress.type=BLOCK |

1. 完成之后，就可以测试输出结果是否为压缩文件

|  |
| --- |
| insert overwrite local directory '/data/test'  select \* from emp distribute by deptno sort by empno; |

000000\_0.deflate 000001\_0.deflate 000002\_0.deflate

通过以上的代码测试，发现输出的结果已经被压缩了

### 3.2.2 Hadoop 编译源码支持snappy

在hadoop目录首先需要学会如何查看hadoop支持的压缩编码方式

[hadoop@hadoop01 hadoop-2.7.2]$ bin/hadoop checknative

通过以上的命令，可以查看hadoop的本地库支持的压缩方式

1. 编译的步骤

环境必须linxu能连接外网，ping [www.baidu.com](http://www.baidu.com) ,注意必须使用root来进行编译

1. 需要准备的jar包

|  |
| --- |
| （1） hadoop-2.7.2-src.tar.gz  （2） jdk-8u144-linux-x64.tar.gz  （3） snappy-1.1.3.tar.gz  （4） apache-maven-3.0.5-bin.tar.gz  （5） protobuf-2.5.0.tar.gz |

3 需要安装maven （可以进行编译，测试，打包的一个工具）

解压maven 到app目录下

之后配置MAVEN\_HOME

|  |
| --- |
| export MAVEN\_HOME=/opt/app/apache-maven-3.6.1  export PATH=$MAVEN\_HOME/bin:$PATH |

mvn -version 验证maven是否安装成功

### 3.2.3 配置yum源为国内服务器

1. 备份yum源的配置目录

|  |
| --- |
| mv yum.repos.d/ yum.repos.d.back |

2 新建一个空的yum源的配置目录

|  |
| --- |
| mkdir /etc/yum.repos.d |

3 下载阿里云的配置到目录中

|  |
| --- |
| wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-6.repo |

4 清空缓存 clean all 重建缓存库 makecache

完成以上的步骤之后，就可以进行yum的使用

5 开始编译

|  |
| --- |
| yum install svn  yum install autoconf automake libtool cmake  yum install ncurses-devel  yum install openssl-devel  yum install gcc\*  1） 编译安装snappy  tar -zxvf snappy-1.1.3.tar.gz -C /opt/module/  cd snappy-1.1.3/  ./confifigure  make  make install  查看 snappy 库文件  ls -lh /usr/local/lib |grep snappy  2) 编译安装 protobuf  tar -zxvf protobuf-2.5.0.tar.gz -C /opt/module/  cd protobuf-2.5.0/  ./confifigure  mak  make install  查看 protobuf 版本以测试是否安装成功  protoc --version  3) 编译 hadoop native  tar -zxvf hadoop-2.7.2-src.tar.gz  cd hadoop-2.7.2-src/  mvn clean package -DskipTests -Pdist,native -Dtar -Dsnappy.lib=/usr/local/lib -Dbundle.snappy  执行成功后， /opt/software/hadoop-2.7.2-src/hadoop-dist/target/hadoop-2.7.2.tar.gz 即为新生成的支持  snappy 压缩的二进制安装包 |

### 3.2.4 存储格式

|  |
| --- |
| file\_format:  : SEQUENCEFILE  | TEXTFILE -- (Default, depending on hive.default.fileformat configuration)  | RCFILE -- (Note: Available in Hive 0.6.0 and later)  | ORC -- (Note: Available in Hive 0.11.0 and later)  | PARQUET -- (Note: Available in Hive 0.13.0 and later)  | AVRO -- (Note: Available in Hive 0.14.0 and later)  | JSONFILE -- (Note: Available in Hive 4.0.0 and later)  | INPUTFORMAT input\_format\_classname OUTPUTFORMAT output\_format\_classname |

TextFile ：默认的hive中存储格式（并且是行式存储）

RCFile : 属于列式存储

ORC: 属于列式存储

首先来说行式存储与列式存的区别？

|  |  |  |
| --- | --- | --- |
| 属性A | 属性B | 属性C |
| A1 | B1 | C1 |
| A2 | B2 | C2 |

对于以上的存储数据来说行式存储的格式如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A1 | B1 | C1 | A2 | B2 | C2 |

对于以上的数据列式存储的格式如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A1 | A2 | B1 | B2 | C1 | C2 |

通过观察总结以下行式与列式存储优缺点：

行式存储：

优点：

1 相关的数据保存在一起，这种方式比较方便进行插入和修改

D

2 如果查询只涉及少数几个列，会把整行的数据都读取出来，如果数据量比较少，问题不大，但是如果数据量比较大，就产生缓存的异常

3 由于每一行中列的数据类型并不一致，导致并不能设置一个比较高的压缩比，所以空间的利用率不高

列式存储 ：

优点：

1 查询时只有涉及到的列才会查询出来，并不是把所有的列都查询出来

2 一个列中的数据类型都是一致的，所以可以提供一个较高的压缩比

缺点：

Insert /update的时候比较麻烦

不适合查询小量的数据

### 3.2.5 RCFile文件存储格式

1 RCFile 他是一个列式存储的存储格式 ，并且通过列进行一个数据压缩，列都是相同的数据类型，所以压缩比比较高

2 可以跳过不必要读取的列读取

3 RCFile把数据分成多个block ,每一个block里以rowgroup为单位，存储在HDFS上的一个block里面的所有记录会被划分为多个rowgroup

4 rowgroup 里包括了3个部分 ：

1 sync: 行组头部的一个同步标志，主要用于隔离HDFS两个连续块的行组

2 metadataHeader: 行组的元数据，记录每一个列的字节数，以及列的大小

3 实际存储数据的顺序

### 3.2.6 ORCFile文件存储格式

ORC 的存储格式与RCFile的rowgroup的存储格式类似，ORC是按照stripes组织记录的（默认大小为250MB）

ORC的存储格式分为以下几个部分：

1 多个stripe 每一个stripe为250MB的大小

每一个stripe又包含了一个indexData （数据索引），RowData(行数据)

StripeFooter ,默认大小也是250MB 块大的好处就是HDFS的读取效率高

2 indexData :包含了行的偏移量，以及每一列的最大值与最小值

3 RowData: 保存的是实际的数据

4 StripeFooter 数据读取的列的位置

### 3.2.7 PARQUET文件存储格式

是面向分析型的列式存储格式，并且是以二进制方式存储的，所以parquet 文件是不能直接读取的，其里面包含了文件的数据，以及文件的元数据，所以parquet是支持自解析的一个存储格式类似于一个小型的数据库

### 3.2.8 以上文件格式的对比

测试的方向；对以上的文件进行存储，压缩，查询方面的比较

1. 测试的数据使用log日志数据使用textFile默认的存储格式

存储到HDFS上的大小为35.37MB

查询速度 25.858

2 使用ORC的方式进行存储

存储HDFS上的大小为4.48MB

查询速度 23.901

3 使用parquet方式进行存储

存储在HDFS上大小为14.31MB

查询速度 27.138

## 3.3 Hive的调优

### 3.3.1 Fetch抓取

所谓fetch抓取hive中的某些查询可以不必使用mapreduce来计算，例如select \* from emp这个语句就没有使用mapreduce ，这种情况下，hive可以简单的读取emp目录对应的数据文件

首先需要找到hive-default.xml配置文件中进行属性的相关查找

|  |
| --- |
| <property>  <name>hive.fetch.task.conversion</name>  <value>more</value>  <description>  Expects one of [none, minimal, more].  Some select queries can be converted to single FETCH task minimizing latency.  Currently the query should be single sourced not having any subquery and should not have  any aggregations or distincts (which incurs RS), lateral views and joins.  0. none : disable hive.fetch.task.conversion  1. minimal : SELECT STAR, FILTER on partition columns, LIMIT only  2. more : SELECT, FILTER, LIMIT only (support TABLESAMPLE and virtual columns)  </description> |

Fecth抓取存在三种模式：

1 none: 禁用fetch抓取，当禁用fetch抓取之后，所有的操作都会进行MR的运行

2 minimal: select \* 分区表中的分区列的过滤条件， 以及limit 不会运行MR

3 more: 查询，提交过滤 limit 限制查询 ，包括 抽样查询 虚拟列 都不会运行MR

什么时候使用fetch抓取以及什么时候不用是根据数据量来的

### 3.3.2 本地模式

Hadoop的任务，是可以提供一个可扩展性来处理大数据集的优势，但是有时hive输入的数据量非常的小，在这种情况下，任务的资源分配与调度的消耗时间，可能会比数据的分析时间要多的多，所以在这种情况下，hive可以在本地单台节点上处理任务。对于小的数据可明显提高执行的速度

限制： 必须是小数据 （128MB以内）

可以通过设置一个属性来开启本地模式：

set hive.exec.mode.local.auto=true; 开启本地模式

以上的属性的意思为： 本地模式自动识别，当输入的数据量大于128M本地模式是关闭的状态，数据提交到集群上运行

当输入的数据小于128MB则自动运行本地模式

对于本地模式也是存在两个属性可以设置的：

1 本地模式的数据量大小（默认为128MB ）可以通过以下的方式来进行设置

set hive.exec.mode.local.auto.inputbytes.max; 设置本地模式输入的最大数据量

2 设置本地默认最大文件输入个数，当输入的文件个数小于设置的值采用本地模式，如果大于设置的值则提交到集群上运行

set hive.exec.mode.local.auto.input.files.max=4 表示本地模式最多只能输入4个文件

### 3.3.3 并行执行

在MapReduce阶段中，map的输入阶段，map的输出阶段，shuffle阶段，reduce输入阶段，以及reduce的输出阶段

根据以上的阶段hive对应的是： 抽样阶段，合并阶段，limit阶段或者其他阶段，默认的情况下hive一次只会执行一个阶段，但是要注意某个特定的任务中可能包含多个阶段，这些任务是不完全相互依赖的是可以并行执行的。并行的执行可以把任务的时间缩短。

设置的参数如下：

|  |
| --- |
| set hive.exec.parallel=true; 打开任务的并行执行 |

|  |
| --- |
| hive.exec.parallel.thread.number=8 同一个sql可以并行执行的数量 |

对于并行的执行，对服务器的资源要求是比较空闲的时候，或者性能比较好的集群上才可以发挥优势

### 3.3.4 严格模式

Hive提供了一个严格模式，可以防止用以执行一些影响不好的查询语句

通过以下属性来设置严格模式：

hive.mapred.mode=nonstrict 默认值为非严格模式 ，如果要开启严格模式需要把属性的内容修改为strict

开启严格模式可以禁止三种类型的查询：

1 ） 对于分区表： 除非where子句中必须存在分区字段来继续过滤限制，否则不允许执行

2 ） 使用order by 进行排序必须使用limit子句进行限制，否则不允许执行

3 ） 限制笛卡尔积的查询

### 3.3.5 JVM重用

JVM的调优参数是属于Hadoop部分的设置，Hadoop默认的配置是使用JVM来执行map和reduce任务的，而对于JVM启动的时候就会造成相当大的资源的开销。尤其是一个MR任务中运行的子任务例如maptask或者reducetask 的数量很多时候，JVM启动的时候资源的开销就会更大。

所谓JVM重用在一个JVM的进程中，对一个任务（job） 可以重新使用N次；N是重用的次数，可以在mapred-site.xml配置文件中进行设置

|  |
| --- |
| <property>  <name>mapreduce.job.jvm.numtasks</name>  <value>1</value>  <description>How many tasks to run per jvm. If set to -1, there is  no limit.  </description>  </property> |

对于JVM重用的值通常设置在10-20之间，具体需要设置多少最好根据业务需求，以及服务器的性能来继续测试得出结果

### 3.3.6 推测执行

在分布式的环境下，因为程序的BUG,或者负载的不均衡，或者资源分布不平均，就会造成一个job中多个任务运行的速度不一致，有且任务的运行速度可能明显慢于其他任务，这些任务就拖慢job的整体执行速度。为了避免这种事情发生，所有Hadoop采用了一种推测执行的机制，根据一定的法则来推测出拖后腿的任务，并且为这样的任务启动一个备份任务，这个任务与原始任务处理同一份数据，并且选用最先成功运行完成的任务的结果作为最终输出结果

开启推测执行的方式：

在mapred-site.xml配置文件中进行配置

|  |
| --- |
| <property>  <name>mapreduce.map.speculative</name>  <value>true</value>  <description>If true, then multiple instances of some map tasks  may be executed in parallel.</description>  </property> |

|  |
| --- |
| <property>  <name>mapreduce.reduce.speculative</name>  <value>true</value>  <description>If true, then multiple instances of some reduce tasks  may be executed in parallel.</description>  </property> |

这两个属性默认就是开启的

Hive也提供了属性来控制推测执行

hive.mapred.reduce.tasks.speculative.execution=true；

如果说用户输入的数据非常大，执行需要很长时间，这个时候开启推测执行，就会造成很大的资源消耗

对于推测执行的时候，有一个前提就是：

1 任务必须执行超过5% 的时候才会启动推测执行

2 如果处理的数据出现了数据倾斜则开启推测执行是没有意义的