Hive12 其他文件格式和压缩方法

> Hive的一个功能：Hive不会强制要求将数据转换为特定的格式才能使用。Hive利用Hadoop的InputFormat API来从不同的数据源中读取数据，例如：文本格式、sequeuece文件格式，甚至用户自定义格式。

确定安装编解码器

基于用户所使用的Hadoop版本，会提供不同的编解码器。Hive可以通过set命令查看Hive配置文件中或Hadoop配置文件中配置的值。

通过查看属性 io.compression.codec， 可以看到编解码器之间是按照逗号分隔的；

```
# hive -e "set io.compression.codecs"
io.compression.codecs=org.apache.hadoop.io.compress.DefaultCodec,
com.hadoop.compression.lzo.LzoCodec,
com.hadoop.compression.lzo.LzopCodec,
org.apache.hadoop.io.compress.GzipCodec,
org.apache.hadoop.io.compress.BZip2Codec
```

选择一种压缩编/解码器

使用压缩的优势可以最小化所需要的磁盘存储空间，以及减少磁盘和网络I/O操作，不过文件压缩过程和解压过程会增加CPU的开销，因此，对于压缩密集型的job最好使用压缩，特别是额外有CPU资源或磁盘存储空间比较稀缺的情况。

压缩种类	压缩率	CPU开销	压缩速度	是否可分割
BZip2	最高	最大	一般	可
GZip2	较高	较大	一般	不可
LZO	一般	一般	快	可
Snappy	一般	一般	快	不可

文件是怎样分割成行的：

> 文本文件使用`\n`作为默认的行分隔符，当用户没有使用默认的文本文件格式的时候，用户需要告诉`Hive`使用的`InputFormat` 和 `OutputFormat`是什么。实际上，用户需要指定对于输入和输出格式实现的`Java`类的名称，`InputFormat` 中定义了如何读取划分，以及如何将划分分隔成记录，而`OutputFormat`中定义了如何将这些划分协会到文件。

如何分隔成字段的

> `Hive`使用`^A`作文本中默认的字段分隔符。`Hive`使用`SerDe`作为对输入反序列化进行分隔以及序列化的模板，用户只需要指定一个`java`类即可

开启中间压缩

对中间数据进行压缩可以减少job中`map`和`reduce teask`间的数据传输量。对于中间数据压缩，选择一个低CPU开销的编/解码器要比选择一个压缩率高的编/解码器要重要得多。

`hadoop`压缩的默认编解码器是`DefaultCodec`，可以通过设置参数`mapred.map.output.compression.codec`来进行相应调整，这是一个`hadoop`配置项，可以在`hadoop mapred-site.xml`文件更改或`hive-site.xml`文件进行更改。`SnappyCodec`是一个比较好的编解码器，拥有低cpu开销和好的压缩率。

```
<property>
       <name>hive.exec.compress.intermediate</name>
       <value>true</value>
       <description> This controls wheteher intermediate files produced by Hive
between multiple map-reduce jobs are compressed.The compression codec and other
options are determined from hadoop config varuables mapred.output.compress*
       </description>
</property>

 <property>
        <name>mapred.map.output.compression.codec</name>
        <value> org.apache.hadoop.io.compress.SnappyCodec    </value>
</property>
</configuration>
```

最终输出结果压缩

当Hive将输出写入表中，输出内容同样可以进行压缩，属性`hive.exec.compress.output`控制着这个功能，用户保持默认配置文件中得默认值false，这样默认得输出就是非压缩的纯文本文件了，用户可以通过查询语句或执行脚本中设置这个值为true。

如果`hvie.exec.compression.output`的值设置为true，那么这个时候需要为其指定一个编解码器。对于输出文件，使用GZip进行压缩是一个不错的注意，其通常可以大幅度降低文件的大小。

```
<property>
       <name>hive.exec.compress.output</name>
       <value>true</value>
       <description> This controls wheteher intermediate files produced by Hive
between multiple map-reduce jobs are compressed.The compression codec and other
options are determined from hadoop config varuables mapred.output.compress*
       </description>
</property>

 <property>
        <name>mapred.output.compression.codec</name>
        <value> org.apache.hadoop.io.compress.GzipCodec     </value>
</property>
```

sequence file 存储格式

Hadoop 所支持的`sequence file`存储格式可以将一个文件划分为多个块，然后采用一种可分割的方式进行压缩。

如果想在Hive中使用 `sequence file` 存储格式，那么需要在 `CREATE TABLE` 语句中通过 `STORED AS SEQUENCEFILE` 语句进行指定：

```
CREATE TABLE a_sequence_file_table STORED AS SEQUENCEFILE;
```

`Sequence file` 提供了三种压缩方式： `NONE`、 `RECORD` 和 `BLOCK`， 默认是 `RECORD` 级别，但是一般 `BLOCK`（块级别）压缩性能最好而且是可以分割的。也需要自己在 `Hadoop` 的 `mapred-site.xml` 或者 `hive-site.xml` 文件中定义：

```
<property>
       <name>mapred.output.compression.type</name>
       <value>BLOCK</value>
       <description> If the job outputs are to compressed as SequenceFiles, how
should they be compressed? Should be one of NONE, RECORD BLOCK
       </description>
</property>
```

使用压缩实践

首先创建一个数据表

```
hive> CREATE TABLE a
    > ( a   int,
    >   b   int)
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY '\t'
    > STORED AS TEXTFILE;
```

在文件外生成导入数据 data.txt

```
# vim data.txt
4     5
3     2
```

将数据导入表中

```
注意修改路径
hive> load data local inpath 'data.txt's Path' into table a;
```

查看表数据

```
hive> SELECT * FROM a;

4   5
3   2

hive> DESCRIBE a;

a                       int
b                       int
```

开启中间压缩功能，以及导入其他数据表

```
hive> set hive.exec.compress.intermediate=true;
hive> CREATE TABLE intermediate_comp_on
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
    > AS SELECT * FROM a;

复制粘贴处：CREATE TABLE intermediate_comp_on ROW FORMAT DELIMITED FIELDS
TERMINATED BY '\t' AS SELECT * FROM a;

Moving data to: file:/user/hive/warehouse/ontermediate_comp_on
Table default.intermediate_comp_on status:[num_partition:0, num_files:1,
num_rows:2, total_size:8, raw_data_size:6]
```

中间数据压缩并没有影响到最终的输出，最终的数据结果依然是非压缩的：

```
hive> dfs -ls /user/hive/warehouse/intermediate_comp_on;
/user/hive/warehouse/intermidate_comp_on/000000_0

hive> dfs -cat  /user/hive/warehouse/intermediate_comp_on/000000_0;
4   5
3   2
```

开启输出结果压缩

```
hive> set hive.exec.compress.output=true;

hive> CREATE TABLE final_comp_on
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\T'
    > AS SELECT * FROM a;
Moving data to: file: /tmp/hive-edward/hive_2020_10_08_13_02_45_884.../-ext-
10001
Moving data to: file:/user/hive/warehouse/final_comp_on
Table default.final_comp_on stats:
[num_partition:0, num_files:1, num_rows:2, total_size:16, raw_data_size:6]

hive>dfs -ls /user/hive/warehouse/final_comp_on;
/user/hive/warehouse/final_comp_on/000000_0.deflate
```

输出信息中的表统计信息显示 total_size(总大小)是16B，但是raw_data_size（裸数据）是6B，多出的存储空间被deflate算法消耗了。不推荐使用cat命令来查看这个压缩文件，因为用户只能看到二进制输出。

```
hive> SELECT * FROM final_comp_on;
4   5
3   2
```

- 使用GZip编解码器

```
hive> set hive.exec.compress.output=true;
hive> set mapred.output.compression.codec  =
org.apache.hadoop.io.compress.GzipCodec;
hive> CREATE TABLE final_comp_gz
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\T'
    > AS SELECT * FROM a;
Moving data to: file:/user/hive/warehouse/final_comp_on_gz
```

```
Table default.final_comp_on stats:
[num_partition:0, num_files:1, num_rows:2, total_size:28, raw_data_size:6]

hive> dfs -ls /user/hive/warehouse/final_comp_on_gz;
/user/hive/warehouse/final_comp_on_gz/000000_0.gz

zcat解压缩观看
hive> ! /bin/zcat /user/hive/warehouse/final_comp_on_gz/000000_0.gz;
4   5
3   2

hive> SELECT * FROM final_comp_on_gz;
4   5
3   2
```

- 使用sequence file（可以分割）

```
hive> set mapred.output.compression.type=BLOCK;
hive> set hive.exec.compress.output=true;
hive> set
mapred.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec;

hive> CREATE TABLE final_comp_on_gz_seq
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'
    > STORED AS SEQUENCEFILE
    > AS SELECT * FROM a;
Moving data to: file:/user/hive/warehouse/final_comp_on_seq
Table default.final_comp_on stats:
[num_partition:0, num_files:1, num_rows:2, total_size:199, raw_data_size:6]

hive> dfs -ls /user/hive/warehouse/final_comp_on_gz_seq;
/user/hive/warehouse/final_comp_on_gz_seq/000000_0

Sequence filw 是二进制格式的，但是我们可以很容易查询文件头，通过查看文件头可以确认结果是否我们需要的
hive> dfs -cat /user/hive/warehouse/final_comp_on_gz_seq/000000_0;
SEQ[]org.apache.io.BytesWritable[]org.apache.hadoop.io.BytesWritable[]
org.apache.hadoop.io.compress.GzipCodec[]

hive> dfs -text /user/hive/warehouse/final_comp_on_gz_seq/000000_00
4   5
3   2
```

常用配置

```
hive> set
mapred.map.output.compression.codec=org.apache.hadoop.io.compress.SnappyCodec;
hive> set hive.exec.compress.intermediate=true;
hive> set mapred.output.compression.type=BLOCK;
hive> set hive.exec.compress.output=true;
hive> set
mapred.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec;

hive> CREATE TABLE final_comp_on_gz_int_compress_snappy_seq
    > ROW FORMAT DELIMITED FIFLDS TERMINATED BY '\t'
    > STORED AS SEQUENCEFILE AS SELECT * FROM a;
```

存档分区

Hadoop有一种存储格式名为HAR，也就是Hadoop Archive（Hadoop归档文件）的简写。一个HAR文件文件就像HDFS文件系统的一个TAR文件一样是一个单独的文件。其内部可以存放多个文件和文件夹。

如果某个特定的分区下保存的文件有成千上万的话，那么需要HDFS中的NameNode消耗非常大的代价来管理这些文件，通过将分区下的文件归档成一个巨大的，但是同时可以被Hive访问的文件，可以减轻NameNode的压力。

缺点： HAR文件查询效率不高，同时HAR是非压缩的，不会节约空间

- 首先，创建一个分区表，然后将Hive包自带的文件加载到表中

```
hive> CREATE TABLE hive_text (line STRING) PARTITIONED BY (folder STRING);

hive> ! ls $HIVE_HOME
LICENSE
README.txt
RELEASE_NOTES.txt

hive> ALTER TABLE hive_text ADD PARTITION (folder='docs');

hive> LOAD DATA INPATH '${env:HIVE_HOME}/README.txt'
    > INTO TABLE hive_text PATITION (folder='docs');
Loading data to table default.hive_text partition(folder=docs)

hvie> LOAD DATA INPATH '${env:HIVE_HOME}/RELEASE_NOTES.txt'
    > INTO TABLE hive_text PARTITION (folder="docs");
Loading data to table default.hive_text partition(folder=docs)

hvie> SELECT * FROM hive_text WHERE line LIKE '%hive%' LIMIT 2;
http://hive.apache.org/
```

- 这个表下面的底层目录结构

```
hive> dfs -ls /user/hive/warehouse/hive_text/folder=docs;
/user/hive/warehouse/hive_text/folder=docs/README.txt
/user/hive/warehouse/hive_text/folder=docs/RELEASE_NOTES.txt
```

- `ALTER TABLE ... ARCHIVE PARTITION`语句将表转化为一个归档表

```
hive> SET hive.archive.enable=true;
hive> ALTER TABLE hive_text ARCHIVE PARTITION (folder='docs');
....
```

- 查看归档文件

```
hive> dfs -ls /user/hvie/warehouse/hive_text/folder=docs;
/user/hive/warehouse/hive_text/folser=docs/data.har
```

- ALTER TABLE ... UNARCHIVE PARTITION 命令可以将HAR 中的文件提取出来然后重新放回到HDFS中：

```
ALTER TABLE hive_text UNARCHIVE PARTITION (folder='docs');
```