Урок 5. Потоковая обработка данных. Flume/Flink/SparkStreaming

Выполнил Колеганов Н.Д.

Задание

Для части по SQOOP
Провести импорт таблицы из вашего сервера БД в Hadoop с использованием SQOOP в любых двух вариантах из перечисленных ниже.
a. в Hive-таблицу (--hive-import)
b. в HDFS в формате avro (--as-avrodatafile)
c. в HDFS в формате sequencefile (--as-sequencefile)
Если у вас нет своего сервера то можно использовать тот Postgres, который я показал на лекции. Пароль expoter_pass
Для части по потоковой обработке (Flume)
1. Создать Flume-агент с именем, соответствующим имени своего пользователя (например Flume4_20)
2. Создать любой Flume поток используя Flume сервис соответствующего номера.
• Тип источника источник – exec
• Тип канала – memory
• Тип слива – hdfs
3. Убедиться что данные поступают в слив.
4. Создать поверх данных в hdfs таблицу через которую можно просмотреть полученные данные.
5. [Продвинутый вариант] Сделать то-же самое используя несколько сливов в разные места, например в HDFS и в Hive одновременно
6. [Продвинутый вариант] Повторить стандартный пример с выборкой сообщений из Twitter.

Создал БД pg_db_test1
Создал таблицу character в формате parquet без компрессии

sqoop import --connect jdbc:postgresql://node3.novalocal/pg_db --username exporter -P --table character --hive-import --hive-database pg_db_test1 —as-parquetfile

```
... 8 more
[student4_2@manager ~]$ hdfs dfs -du -h -s /user/hive/warehouse/pg_db_test1.db/character
69.6 K  208.8 K /user/hive/warehouse/pg_db_test1.db/character
```

Итоговый размер около 69.6 кб

Попробовал различные запросы на таблице character:

SELECT SUM(speechcount) FROM pg_db_test1.character WHERE abbrev='First Musician'

SELECT count(*) FROM pg_db_test1.character WHERE description IS NOT NULL

SELECT charname, SUM(speechcount) FROM pg_db_test1.character WHERE description IS NOT

	charname	_c1
1	Aaron	57
2	Abhorson	13
3	Abraham	5
4	Achilles	74
5	Adam	10
6	Adrian	9

NULL GROUP BY charname

Импорт таблицы sales_large в формате parquet

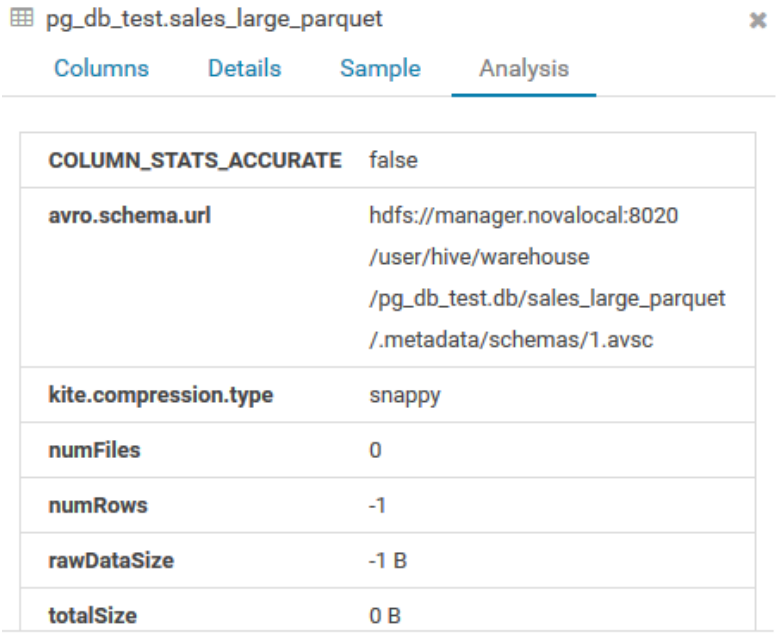
sqoop import --connect jdbc:postgresql://node3.novalocal/pg_db --username exporter -P --table sales_large --hive-import --hive-database pg_db_test1 --hive-table sales_large_parquet –as-parquetfile -m 1

```
[student4_2@manager ~]$ sqoop import --connect jdbc:postgresql://node3.novalocal/pg_db --username exporter -P --table sales_la
rge --hive-import --hive-database pg_db_test1 --hive-table sales_large_parquet –as-parquetfile -m 1
```

```
[student4_2@manager ~]$ hdfs dfs -du -h -s /user/hive/warehouse/pg_db_test1.db/sales_large_parquet
```

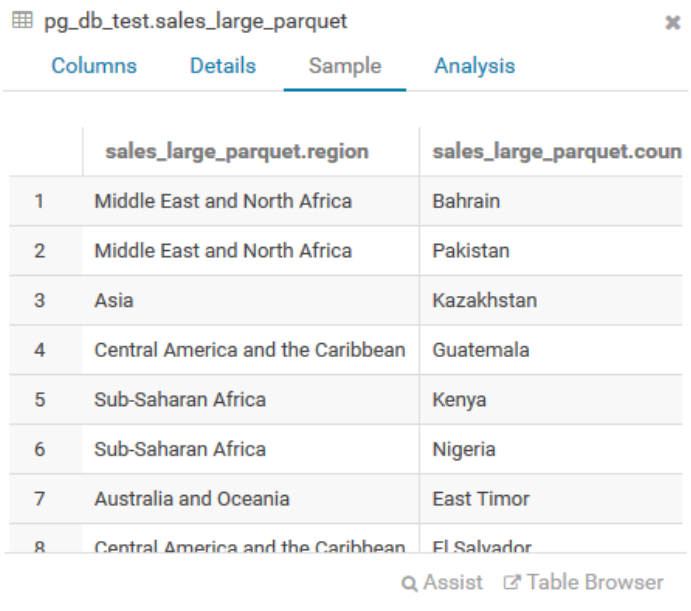
```
458.7 M 1.3 G /user/hive/warehouse/pg_db_test1.db/sales_large_parquet
```

Размер таблицы 458.7 Мб.



Как видно на скриншоте по умолчанию для parquet установлена компрессия snappy.
Если пробовать импорт с ключом -z компрессия все равно остается snappy.

Проверим читабельность импортируемых данных:



Импорт таблицы sales_large в формате avro

sqoop import --connect jdbc:postgresql://node3.novalocal/pg_db --username exporter -P --table sales_large --as-avrodatafile --target-dir /user/hive/warehouse/pg_db_test.db/sales_large_avro -m 1

```
[student4_2@manager ~]$ hdfs dfs -du -h -s /user/hive/warehouse/pg_db_test.db/sales_large_avro
```

```
1.5 G  4.5 G  /user/hive/warehouse/pg_db_test.db/sales_large_avro
```

Размер таблицы 1.5 Gb.
В Hue таблицу не видно, поэтому создаем там EXTERNAL TABLE:

```
CREATE EXTERNAL TABLE pg_db_test.sales_large_avro (
region string,
country string,
itemtype string,
saleschannel string,
orderpriority string,
orderdate string,
orderid int,
shipdate string,
unitssold decimal(10,0),
unitprice decimal(10,0),
unitcost decimal(10,0),
totalrevenue decimal(10,0),
totalcost decimal(10,0),
totalprofit decimal(10,0))
STORED AS AVRO;
```

Проверим читабельность импортируемых данных:

pg_db_test.sales_large_avro

Columns | Details | Sample | Analysis

	sales_large_avro.region	sales_large_avro.country
1	Middle East and North Africa	Bahrain
2	Middle East and North Africa	Pakistan
3	Asia	Kazakhstan
4	Central America and the Caribbean	Guatemala
5	Sub-Saharan Africa	Kenya
6	Sub-Saharan Africa	Nigeria
7	Australia and Oceania	East Timor
8	Central America and the Caribbean	El Salvador

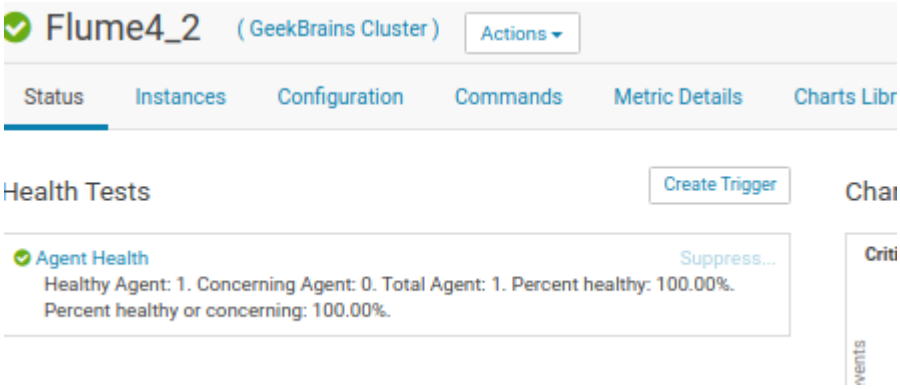
Импорт таблицы sales_large в формате avro с компрессией gzip

sqoop import --connect jdbc:postgresql://node3.novalocal/pg_db --username exporter -P --table sales_large --as-avrodatafile --target-dir /user/hive/warehouse/pg_db_test.db/sales_large_avro -z -m 1

Размер таблицы получается около 450 Мб.

Для части по потоковой обработке (Flume)

Создал Flum



Flume конфиг:

Flume4_2

onfiguration File

Agent Default Group

```
# Naming the components on the current agent
Flume4_2.sources = printer
Flume4_2.channels = MemChannel
Flume4_2.sinks = output

# insert timestamp
# Flume4_2.sources.printer.interceptors = ts_interceptor
# Flume4_2.sources.printer.interceptors.ts_interceptor.type = timestamp

# Describing/Configuring the source
Flume4_2.sources.printer.type = exec
Flume4_2.sources.printer.command = python3 printer.py
Flume4_2.sources.printer.channels = MemChannel
Flume4_2.sources.printer.bind = 89.208.222.201
Flume4_2.sources.printer.port = 7401

# Describing/Configuring the sink
Flume4_2.sinks.output.type = hdfs
# Flume4_2.sinks.output.hdfs.path = /user/student4_2/output/snap_day=%Y%m%d/
Flume4_2.sinks.output.hdfs.path = /user/student4_2/output/
Flume4_2.sinks.output.hdfs.filePrefix = flume_data
Flume4_2.sinks.output.hdfs.fileType = SequenceFile
Flume4_2.sinks.output.hdfs.codeC = gzip

# Describing/Configuring the channel
Flume4_2.channels.MemChannel.type = memory
Flume4_2.channels.MemChannel.capacity = 100
Flume4_2.channels.MemChannel.transactionCapacity = 10

# Bind the source and sink to the channel
```

Проверил статус: