

# Урок 5. Поточковая 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.

- в Hive-таблицу (--hive-import)
- в HDFS в формате avro (--as-avrodatafile)
- 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)

- Создать Flume-агент с именем, соответствующим имени своего пользователя (например Flume4\_20)
- Создать любой Flume поток используя Flume сервис соответствующего номера.
  - Тип источника источник – exes
  - Тип канала – memory
  - Тип слива – hdfs
- Убедиться что данные поступают в слив.
- Создать поверх данных в hdfs таблицу через которую можно просмотреть полученные данные.
- [Продвинутый вариант] Сделать то-же самое используя несколько сливов в разные места, например в HDFS и в Hive одновременно
- [Продвинутый вариант] Повторить стандартный пример с выборкой сообщений из Twitter.

**Создал БД pg\_db\_test1**

**Создал таблицу character в формате parquet без компрессии**

```
1.37s pg_db_test1 text ?
1 CREATE DATABASE pg_db_test1;
2
3 SET parquet.compression=none;
4
5 CREATE TABLE pg_db_test1.character (
6     charid string,
7     charname string,
8     abbrev string,
9     description string,
10    speechcount int)
11 STORED AS PARQUET;
```

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

```
[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'
```

```
INFO : MapReduce Jobs Launched:
INFO : Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.87 sec HDFS Read: 40088 HDFS Write: 3 SUCCESS
INFO : Total MapReduce CPU Time Spent: 8 seconds 870 msec
INFO : Completed executing command(queryId=hive_20200311145454_25daf121-983d-4a97-a0f3-92a63288ac24); Time taken: 26.17 seconds
INFO : OK
```

Query History Saved Queries Results (1)

\_c0

```
INFO : MapReduce Jobs Launched:
INFO : Stage-Stage-1: Map: 1 Reduce: 1 Cumulative CPU: 8.07 sec HDFS Read: 35820 HDFS Write: 4 SUCCESS
INFO : Total MapReduce CPU Time Spent: 8 seconds 70 msec
INFO : Completed executing command(queryId=hive_20200311145757_0b85a826-44f8-428e-b7ef-ad5ab05dd6ab); Time taken: 24.219 seconds
INFO : OK
```

Query History Saved Queries Results (1)

\_c0

|  | 1 | 967 |
|--|---|-----|
|--|---|-----|

```
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_large --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.

pg\_db\_test.sales\_large\_parquet

|                       |         |  |          |
|-----------------------|---------|--|----------|
| Columns               | Details | Sample   | Analysis |
|                       |         |  |          |
| COLUMN_STATS_ACCURATE |         | false  |          |
| avro.schema.url       |         | hdfs://manager.novalocal:8020<br>/user/hive/warehouse<br>/pg_db_test.db/sales_large_parquet<br>/.metadata/schemas/1.avsc |          |
| kite.compression.type |         | snappy   |          |
| numFiles              |         | 0  |          |
| numRows               |         | -1   |          |
| rawDataSize           |         | -1 B   |          |
| totalSize             |         | 0 B  |          |

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:

pg\_db\_test.sales\_large\_parquet

|         |                                   |                          |          |
|---------|-----------------------------------|--------------------------|----------|
| Columns | Details                           | Sample                   | Analysis |
|         |                                   |                          |          |
|         | sales_large_parquet.region        | sales_large_parquet.coun |          |
| 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

```
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:

```
1 CREATE EXTERNAL TABLE pg_db_test.sales_large_avro (  
2   region string,  
3   country string,  
4   itemtype string,  
5   saleschannel string,  
6   orderpriority string,  
7   orderdate string,  
8   orderid int,  
9   shipdate string,  
10  unitssold decimal(10,0),  
11  unitprice decimal(10,0),  
12  unitcost decimal(10,0),  
13  totalrevenue decimal(10,0),  
14  totalcost decimal(10,0),  
15  totalprofit decimal(10,0))  
16 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

The screenshot shows the Flume web interface for a cluster named 'GeekBrains Cluster'. The main header displays 'Flume4\_2' with a green checkmark icon and an 'Actions' dropdown menu. Below the header is a navigation bar with tabs: 'Status' (selected), 'Instances', 'Configuration', 'Commands', 'Metric Details', and 'Charts Libr'. The 'Status' tab is active, showing 'Health Tests'. On the right, there is a 'Create Trigger' button and a 'Char' label. The 'Health Tests' section shows a single test: 'Agent Health' with a green checkmark icon. The test details are: 'Healthy Agent: 1. Concerning Agent: 0. Total Agent: 1. Percent healthy: 100.00%. Percent healthy or concerning: 100.00%'. There is a 'Suppress...' link next to the test name. On the far right, there is a 'Critic' label and a 'vents' label.

Flume конфиг:

## Flume4\_2

### Configuration 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
Flume4_2.sources.printer.channels = MemChannel
Flume4_2.sinks.output.channels = MemChannel
```

## Проверил статус:

cloudera MANAGER

Clusters ▾ Hosts ▾ Diagnostics ▾ Audits ▾ Charts ▾ Administration ▾

Search Support ▾ studen

Flume4\_2 (GeekBrains Cluster) Actions ▾

30 minutes preceding Jun 21, 2:36 PM UTC

Status Instances Configuration Commands Metric Details Charts Library Audits Quick Links ▾

### Health Tests

Create Trigger

Agent Health  
Healthy Agent: 1. Concerning Agent: 0. Total Agent: 1. Percent healthy: 100.00%. Percent healthy or concerning: 100.00%. [Suppress...](#)

### Status Summary

Agent 1 Good Health  
Hosts 1 Good Health

### Health History

|             |                       |                      |
|-------------|-----------------------|----------------------|
| 1:30 PM     | Agent Health Good     | <a href="#">Show</a> |
| 12:53:09 PM | Agent Health Disabled | <a href="#">Show</a> |
| 12:53:04 PM | Agent Health Unknown  | <a href="#">Show</a> |

### Charts

30m 1h 2h 6h 12h 1d 7d 30d

