Práctica Hadoop Sergio Yunta Martín

Módulo 4 datahack edición 63 Máster Big Data and Architecture

1. Parte Práctica

En esta parte se explican los resultados de la parte práctica, la explicación de cómo replicar y ejecutar los scripts se encuentra en el fichero README.md

En resumen, se han creado tres tablas en hive, una por cada fichero del dataset, siguiendo las instrucciones del mismo. Se han cargado los ficheros desde local.

1.1 Ejercicio 1

1. ¿Cuál es la película con más opiniones?

```
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 7.28 sec   HDFS Read: 24604756 HDFS Write: 175679 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 3.84 sec   HDFS Read: 181322 HDFS Write: 33 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 120 msec
OK
2858    American Beauty (1999)  3428
Time taken: 58.204 seconds, Fetched: 1 row(s)
WARN: The method class org.apache.commons.logging.impl.SLF4JLogFactory#release() was invoked.
WARN: Please see http://www.slf4j.org/codes.html#release for an explanation.
```

El código se encuentra en el fichero *queries/1_most_popular_movie.hql*

Como vemos, la película con más opiniones es *American Beauty (1999).*

2. ¿Qué 10 usuarios son los más activos a la hora de puntuar películas?

User_id / Num_reviews

```
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 7.53 sec   HDFS Read: 24601052 HDFS Write: 131414 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 3.7 sec   HDFS Read: 136421 HDFS Write: 99 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 230 msec
OK
4169    2314
1680    1850
4277    1743
1941    1595
1181    1521
889     1518
3618    1344
2063    1323
1150    1302
1015    1286
Time taken: 55.623 seconds, Fetched: 10 row(s)
WARN: The method class org.apache.commons.logging.impl.SLF4JLogFactory#release() was invoked.
WARN: Please see http://www.slf4j.org/codes.html#release for an explanation.
```

El código se encuentra en el fichero *queries/2_active_users.hql*

Esos son los identificadores de los 10 usuarios más populares junto con los comentarios que ha realizado en las diferentes películas, ordenados por esto último.

3. ¿Cuáles son las tres mejores películas según los scores? ¿Y las tres peores?

```
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 7.73 sec   HDFS Read: 24605542 HDFS Write: 198808 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 3.27 sec   HDFS Read: 204502 HDFS Write: 97 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 0 msec
OK
1830    Follow the Bitch (1998) 5.0
3233    Smashing Time (1967)    5.0
3607    One Little Indian (1973)        5.0
Time taken: 61.535 seconds, Fetched: 3 row(s)
WARN: The method class org.apache.commons.logging.impl.SLF4JLogFactory#release() was invoked.
WARN: Please see http://www.slf4j.org/codes.html#release for an explanation.
```

Esta query con las mejores se encuentra en *queries/3_1_best_score_movies.hql*

Estos datos pueden estar sesgados porque son películas que tienen pocas reseñas, se podría hilar más fino para el fin de la práctica esto se ha omitido.

```
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 8.19 sec   HDFS Read: 24605542 HDFS Write: 198808 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 3.61 sec   HDFS Read: 204502 HDFS Write: 143 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 800 msec
OK
3460    Hillbillys in a Haunted House (1967)    1.0
2217    Elstree Calling (1930)  1.0
641     Little Indian, Big City (Un indien dans la ville) (1994)        1.0
Time taken: 60.209 seconds, Fetched: 3 row(s)
WARN: The method class org.apache.commons.logging.impl.SLF4JLogFactory#release() was invoked.
WARN: Please see http://www.slf4j.org/codes.html#release for an explanation.
```

Esta query con las mejores se encuentra en *queries/3_2_worst_score_movies.hql*

4. ¿Hay alguna profesión en la que deberíamos enfocar nuestros esfuerzos en publicidad? ¿Por qué?

```
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 8.32 sec   HDFS Read: 24604849 HDFS Write: 137514 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 3.26 sec   HDFS Read: 142974 HDFS Write: 125 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 580 msec
OK
4169    0       2314
1680    20      1850
4277    16      1743
1941    17      1595
1181    7       1521
889     20      1518
3618    17      1344
2063    4       1323
1150    20      1302
1015    3       1286
Time taken: 71.765 seconds, Fetched: 10 row(s)
WARN: The method class org.apache.commons.logging.impl.SLF4JLogFactory#release() was invoked.
WARN: Please see http://www.slf4j.org/codes.html#release for an explanation.
```

Esta query se encuentra en *queries/4_most_important_occupations.hql*

El razonamiento en este apartado es seguir la idea de los usuarios más activos o populares, si sacamos sus profesiones, tendremos las más importantes para orientar nuestra publicidad. La segunda columna nos indica la profesión, que debemos ir a consultar al README del dataset.

En este ejemplo, el que más sale es el 20 (escritor) seguido de 17 (técnico/ingeniero).

5. ¿Se te ocurre algún otro insight valioso que pudiéramos extraer de los datos procesados? ¿Cómo?

```
MapReduce Jobs Launched:
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 8.49 sec   HDFS Read: 24604802 HDFS Write: 137514 SUCCESS
Stage-Stage-3: Map: 1  Reduce: 1   Cumulative CPU: 3.9 sec   HDFS Read: 142937 HDFS Write: 129 SUCCESS
Total MapReduce CPU Time Spent: 12 seconds 390 msec
OK
4169    50      2314
1680    25      1850
4277    35      1743
1941    35      1595
1181    35      1521
889     45      1518
3618    56      1344
2063    25      1323
1150    25      1302
1015    35      1286
Time taken: 63.009 seconds, Fetched: 10 row(s)
WARN: The method class org.apache.commons.logging.impl.SLF4JLogFactory#release() was invoked.
WARN: Please see http://www.slf4j.org/codes.html#release for an explanation.
```

Esta query se encuentra en *queries/5_most_important_ages.hql*

Se ha seguido la lógica anterior, pero en vez de profesiones, hacerlo con edades, lo cual combinado con lo anterior nos puede permitir refinar el objetivo de nuestra publicidad.

Vemos que el valor más repetido es 35, que nos indica el rango de 35 a 44 años de edad, muy seguido del de 25 que incluye el rango 25 a 34 años. Por lo tanto, deberíamos orientar la publicidad a gente de entre 25 a 44 años.

1.2 Ejercicio 2

Primero necesitamos crear la tabla destino en la base de datos mysql, esto es importante porque si no el comando para exportar va a fallar. Se ha elegido el caso del apartado 4 del ejercicio anterior.

Antes de tirar con sqoop, hay varios pasos:

1. Crear la tabla en la base de mysql en la base de datos : `create table important_ocupations(user_id int, occupation int, num_reviews int);`

2. Ejecutar la consulta que queremos exportar, pero con las instrucciones necesarias para guardarlas en HDFS (y luego exportarlo con sqoop): `INSERT OVERWRITE DIRECTORY '/user/cloudera/queries/most_important_occupations' ROW FORMAT DELIMITED FIELDS TERMINATED BY ','` justo antes de empezar la sentencia SELECT. Esto está en *dump_most_important_occupations.hql.*
3. Ejecutar el comando sqoop, se puede ver en *sqoop_export.sh* : `sqoop export --connect jdbc:mysql://localhost/retail_db --username retail_dba --password cloudera --table important_occupations --export-dir /user/cloudera/queries/most_important_occupations --input-fields-terminated-by ','` Esto insertará todo lo que esté en el directorio de hdfs */user/cloudera/queries/most_important_occupations* en la tabla important_occupations de la base de datos retail_db.

Además, se ha quitado la limitación de usuarios, para poder sacar más métricas sobre estos datos. Es importante exportarlo a una base de datos relacional porque la latencia baja muchísimo, estamos hablando que pasamos de un tiempo de consulta de 1 minuto en hive a, como vemos más abajo, milisegundos. Esto hace que se pueda ver desde una web de una manera mucho más cómoda.

```
|    5937 |         12 |         102 |
|    1441 |         16 |         102 |
|    3148 |         20 |         102 |
+---------+------------+-------------+
6040 rows in set (0.01 sec)

mysql> select * from important_occupations;
```

2. Dimensionamiento clúster Hadoop

El volumen de datos que vamos a recibir son:

	Media Eventos	Tamaño por evento
Fuente 1	10.000 eventos/día	15 KB
Fuente 2	120.000 eventos/día	300 Bytes
	150.000 eventos/día	100 KB
	170.000 eventos/día	800 KB
	2000 eventos/día	1500 KB

Para calcular el tamaño que necesitamos para almacenar los datos de un año tenemos que tener en cuenta dos cosas:

- La replicación por defecto de Hadoop para garantizar que no se pierda información en el caso de que una máquina se caiga es de **3**. Es decir, necesitamos el **triple de tamaño** en el cluster del que necesitaríamos normalmente para almacenar los datos.
- El tamaño de bloque por defecto es de **128MB**, esto es importante porque define el **tamaño mínimo de escritura** (el de lectura también pero no aplica en este caso), es un factor a tener en cuenta.

Con esto en cuenta, podemos empezar con los cálculos, vamos a ver la tabla anterior calculando cuánto tamaño necesitaríamos para almacenar cada evento al día.

	Tamaño diario (MB)
Fuente 1	146,48 MB
Fuente 2	34,33 MB
	14.648,44 MB
	132.812,5 MB
	2.929,69 MB

Resumiendo:

Total diario (MB)	150.571,44 MB
Total Año (MB)	54.958.576,2 MB
Total Año (TB)	52,41 TB

Por lo tanto, cada año se generan **52,41 TB** de datos, pero siguiendo lo que hemos visto en clase y lo mencionado anteriormente, vamos a suponer una replicación típica de 3. Por lo tanto, al año necesitamos $52,41 \cdot 3 = 157,24\,TB$ de almacenamiento, lo que serían redondeando **158 TB / 79 discos / 4 máquinas**.

Pero esto no es todo, estas 4 máquinas serían únicamente para almacenar todos los datos que vamos a recibir, es decir, *DataNodes*, pero para poder tener un clúster hadoop funcional, necesitamos al menos otra máquina que funcione como *NameNode* por lo que ya sumarían 5 máquinas.

Si ya que nos hemos visto forzados a tener 4 *DataNodes* queremos tener Alta Disponibilidad podemos añadir un segundo *NameNode* como *Secondary NameNode* y conseguir así Alta Disponibilidad.

En conclusión, necesitaríamos **6 máquinas** con alta disponibilidad. Si prescindimos de ella, el mínimo indispensable serían 5 máquinas.

3. Estimación arquitecturas Hadoop para distintos casos de uso

Repasemos qué herramientas del ecosistema Hadoop aplican a cada caso uno por uno.

Herramienta de BI (p.ej.: Microstrategy).

Para este primer apartado deberíamos extraer los datos que queremos mostrar en los dashboards. Aquí depende mucho de los requisitos de latencia del dato.

- Si no tiene que ser un dashboard en tiempo real, es decir, no nos importa que los datos tarden un poco en actualizarse, podemos usar **Hive** para poder realizar las consultas y en base a los resultados de las mismas generar las gráficas necesarias.
- Por otro lado, si es muy importante que se refresque en tiempo real, tenemos otra opción, **Impala,** la pega con esta opción es que las consultas se realizan en memoria, se agilizan mucho pero se necesitan muchos más recursos.

Por tanto, para este caso en concreto, no creo que sea tan necesario ese consumo de recursos por lo que podríamos elegir una tecnología como **Hive**. [1] [3]

Microstrategy tenía un servicio llamado Hadoop Gateway que se deprecó en 2021, por lo que se ha descartado.

Web de consultas sobre pedidos realizados.

En este caso el tiempo de consulta es más importante. Además, me parece muy buena oportunidad para, esperando un gran volumen de pedidos, modelarlos de tal manera que se puedan almacenar en una base de datos NoSQL. Si tiramos por esta opción, tenemos una buena opción en el ecosistema Hadoop como **HBase** [4]. Esto nos permitiría acceder a un gran volumen rápidamente tanto a nivel de lectura como escritura, permitiendo tiempos de respuesta razonables en la web.

Generación de informes SQL usando R que se ejecutan mensualmente.

Para generar informes en SQL y R necesitamos algo que tenga opción de usar varios lenguajes de programación. Una opción que se presenta es **Spark,** puede ejecutar consultas SQL y R gracias a **Spark SQL y SparkR.** Esta tecnología se integra muy bien con hadoop, de hecho muchos lo consideran dentro del ecosistema [6] y además, el CEO va a estar muy contento porque es una tecnología que se usa muchísimo y estaríamos al día en lo que se usa en este ámbito. Esto nos permitiría sacar informes sobre muchos datos de manera rápida y ágil.

Recopilación de información de redes sociales.

En este caso está claro que si necesitamos un sistema de recopilación de datos tenemos que buscar entre los servicios disponibles en el ámbito de ingesta de datos y ETLs. Una posible opción es usar **Flume** [7] que parece diseñado para conectar con fuentes remotas y trabajar en streaming, algo importante si hablamos de redes sociales, ya que se generan datos constantemente.

Según las diapositivas de teoría [3] había pensado en **Pig** pero no parece preparado para conectar con fuentes externas, o al menos no he conseguido encontrar mucha información al respecto.

Estas son mis decisiones para cada uno de los casos, he intentado que sean variadas e investigar más herramientas dentro del ecosistema hadoop.

Referencias

[1] Ecosistema hadoop | datahack
[2] ¿Qué es HBase? | IBM
[3] Diapositivas de teoría
[4] Spark SQL & DataFrames | Apache Spark
[5] SparkR (R on Spark) - Spark 3.5.5 Documentation
[6] Apache Spark and Hadoop HDFS: Working Together
[7] Apache Flume