Big Data

Trabajo Práctico 1 - Grupo 1

De Santi, Matías(51051) - Homovc, Federico(50418) - Pereyra, Cristian(51190) - Pintos, Esteban(51048)

Map Reduce

Para la primera parte del trabajo práctico, se crearon mappers y reducers en Java que cumplían con lo pedido en el enunciado. El objetivo era poder obtener ciertas métricas de un archivo con datos sobre vuelos en los Estados Unidos desde 1987.

Desarrollo

Cuando comenzamos a desarrollar el código, nos dimos cuenta que todas las métricas requerían hacer una junta de datos. En todos los casos se requería juntar los archivos con los datos sobre los vuelos con alguno de los archivos con información extra (datos sobre las aerolíneas, sobre los aviones o sobre los aeropuertos). Si bien los archivos que contenían los datos sobre los vuelos tenían un peso en total de 12GBs, los archivos con la información extra pesaban considerablemente menos (estaban por debajo del mega). Por lo tanto se decidió utilizar Broadcast Join como algoritmo para la junta de los datos.

Dicho algoritmo se implementó de la siguiente manera. Cada maper cargaba los datos de junta en una tabla de hash, utilizando como clave el campo de junta. Para poder cargar los datos en la tabla de hash se sobreescribió el método *Setup* de la clase mapper. Aquellos datos que se encontraban en HBase fueron cargados leyendo la tabla de la base de datos. En cambio, los datos que se encontraban en un archivo csv fueron copiados a cada nodo que ejecutaba el map utilizando *DistributedCache*. De esta manera el archivo se encontraba disponible en todos los nodos mappers y así no existía latencia al momento de abrir y leer el archivo.

Consideraciones sobre las métricas

Demoras en el despegue

Aquellos vuelos que tengan “NA” en el campo de despegue no son considerados para hacer el cálculo. Los resultados de esta métrica se muestran de la siguiente manera: **Mes-Estado promedio-de-demora(minutos)**

Cantidad de vuelos cancelados por aerolínea

Aquellos vuelos para los cuales no exista información sobre la aerolínea no son mostrados en los resultados. Los resultados tienen el siguiente formato: **Aerolínea #vuelos-cancelados**

Cantidad de millas voladas por aerolínea por año

Aquellos vuelos para los cuales no exista información sobre la aerolínea o no tengan información disponible sobre las millas voladas no son mostrados en los resultados. Los resultados tienen el siguiente formato: **Aerolínea-Año #millas-voladas**

Cantidad de horas voladas por determinado tipo de avión

Aquellos vuelos para los cuales no exista información sobre el tipo de avión o no tengan información disponible sobre las horas voladas no son mostrados en los resultados. Los resultados tienen el siguiente formato: Matrícula-del-avión#horas-voladas

Ejecución de las métricas

Para compilar el código, ejecutar ‘*mvn clean install*’. En la carpeta target se creará el archivo *bd-tp1-jar-with-dependencies.jar*. Para poner a correr el job de map-reduce, en el cluster ejecutar *hadoop jar bd-tp1-jar-with-dependencies.jar*. Dependiendo de la métrica que se quiere obtener deben pasarse distintos parámetros.

Todas las métricas requieren de dos parámetros obligatorios: --inFile y --outPath. Como su nombre sugiere, --inPath recibe como argumento el path donde se encuentran los archivos

con información sobre los vuelos. --outPath recibe como argumento el path donde se quiere dejar los resultados.

Las métricas disponibles son las siguientes:

- **--takeOffDelay**: indica que la métrica a ejecutar es la de demoras en el despegue.
- **--flownMiles**: indica que la métrica a ejecutar es la de cantidad de millas voladas.
- **--flightHours**: indica que la métrica a ejecutar es la de cantidad de horas voladas.
- **--cancelledFlights**: indica que la métrica a ejecutar es la de cantidad de vuelos cancelados.

Adicionalmente, algunas métricas requieren parámetros extra:

Cantidad de vuelos cancelados y cantidad de millas voladas

Estas métricas requieren del parámetro **--carriersFile**. Este parámetro recibe un argumento que indica dónde se encuentra el archivo con la información sobre las aerolíneas.

Cantidad de horas voladas

Esta métrica recibe el parámetro **--planeType**. Este parámetro recibe como argumento el tipo del avión para el cual se quiere obtener la métrica. Por ejemplo: **--planeType BOEING**.

Cobertura de tests

Para obtener el reporte de la cobertura de tests, ejecutar mvn cobertura:cobertura. Dentro de la carpeta site en target se encontrarán archivos html con el reporte.

Dificultades encontradas

Dado que el lenguaje de programación es conocido, no se presentaron mayores dificultades. La única dificultad que se nos presentó fue con los tipos de los mappers y los reducers. Si uno especifica que el mapper tiene X tipo de retorno y por error uno retorna otro tipo, no se lanza ninguna excepción pero tampoco agrupa de manera correcta en los reducers.

Otra dificultad se nos presentó al momento de hacer los tests. Aquellos reducers que usan datos de archivos csv pudieron ser testeados sin ningún problema ya que el path al archivo es parametrizable y por lo tanto se puede crear un archivo en la carpeta resources y ser utilizado. En cambio, aquellos mappers que utilizaban datos de HBase no pudieron ser

testeados ya que no encontramos manera de poder mockear HBase en memoria para poder correr los tests.

Pig

Para la segunda parte del trabajo práctico, se utilizó Hadoop Pig para poder obtener otras métricas con los datos mencionados en la primer parte, con un adicional con información sobre huracanes en esa época.

Al contrario que con Map Reduce, el desarrollo con Pig no fue tan ameno. En primer lugar tuvimos que lidiar con la sintaxis de Pig, debido a que no tiene gran similitud con ningún lenguaje que hayamos utilizado anteriormente. En segundo lugar, cuando teníamos algún error ya sea de sintaxis o de ejecución, los errores no eran muy claros y fue un gran desafío interpretarlos y solucionarlos.

Sin embargo, notamos que Pig provee una muy buena abstracción por sobre lo que es Map Reduce y por lo tanto uno puede hacer consultas sencillas de leer.

Ejecución de las métricas

Todas las métricas requieren los siguientes parámetros:

- **PIGGYBANK_PATH**: Path donde se encuentra el jar de las funciones definidas en el Piggy Bank.
    - Default: '/home/hadoop/pig-0.11.1/contrib/piggybank/java/piggybank.jar'

- **FLIGHTS_PATH**: Path de la carpeta donde se encuentran los datos de los vuelos:
    - Default: '/user/hadoop/ITBA/TP1/INPUT/SAMPLE/data/'

- **AIRPORTS_HBASE_PATH**: Tabla Hbase donde se encuentran los datos de los aeropuertos:
    - Default: 'hbase://itba_tp1_airports'

- **OUTPUT_PATH**: Path de la carpeta donde se van a encontrar los resultados de las métricas:

- Default: 'metricN/output'. Donde N es el número de la métrica (de la 1 a lal 4)

Consideraciones sobre las métricas

Top 5 de aeropuertos con mayor demora total de despegue por año

Para implementar esta métrica lo que se realizó fue un join entre los vuelos y los aeropuertos quedándonos con los campos de la demora, el aeropuerto y el año. Luego se agrupó por año y aeropuerto y se calculó la suma de la demora como “totaldelay”. Finalmente se agrupó por año lo obtenido en el paso anterior y se calculó el top 5 para cada aeropuerto ordenando los datos de manera descendente y aplicando un límite de 5 resultados.

Los resultados de esta métrica se muestran de la siguiente manera: **Año; Aeropuerto; demora-total (minutos).**

Información sobre demoras para el año 2005

Esta métrica recibe el parámetro opcional **SELECTED_AIRPORT,** donde se le indica el aeropuerto para el cual se quiere calcular la métrica. Si no se encuentra el parámetro, se calculará la misma para todos los aeropuertos.

Para implementar esta métrica también se realizó un join entre los vuelos y los aeropuertos. Luego para cada tupla se creó la fecha en formato “yyyy/M/d” , se marcó con un 1 si fue demorado o 0 en caso contrario y lo mismo si el vuelo fue cancelado o desviado. Por último se agrupó por día y se sumaron todos los valores mencionados anteriormente.

Los resultados de esta métrica se muestran de la siguiente manera: **Fecha(yyyy/M/d); #vuelos-demorados; demora-total(minutos); #vuelos-cancelados; #vuelos-desviados; #vuelos-cancelados-por-clima**

Top 5 de aeropuertos con mayor promedio de vuelos demorados por día para cada huracán

Para implementar esta métrica también se realizó un join entre los vuelos y los aeropuertos. La mayor dificultad que se nos presentó fue la de parsear la fecha correctamente, la cual era necesaria para poder distinguir entre los diferentes huracanes. Lo que se realizó fue después de la junta generar el campo date con el formato “yyyy-M-d” y el campo delayed indicando con un 1 si el vuelo tenía un departure delay, o un 0 en caso contrario. Luego se iteró por los resultados anteriores y según las fechas generadas se agregó el campo hurricane y days con el nombre y días de duración del mismo respectivamente. Debido a que habían fechas en las cuales no hubo huracanes, lo que se hizo fue quedarnos únicamente con aquellas donde sí hubo. Por último lo que se realizó fue agrupar los datos por huracán y aeropuerto y calcular el promedio de demora como la suma de las mismas dividido la cantidad de días que duró el huracán. Finalmente agrupando por huracán y ordenando el promedio descendentemente pudimos quedarnos con el top 5 utilizando un limit de 5.

Los resultados de esta métrica se muestran de la siguiente manera: **Huracán; Aeropuerto; Promedio-demora (minutos); Año**

Día con mayor cantidad de cancelaciones por clima en el país por cada huracán

Por último, para esta métrica se realizó algo muy similar a la anterior. Se hizo la junta entre las dos tablas, se parseó la fecha y se indicó con un 1 si el CancellationCode era igual a “B” (código que indica que fue cancelado por problemas climáticos) o 0 si no. Luego como en el caso anterior se agregó el campo hurricane indicando el nombre del huracán y agrupando por el mismo y la fecha, se calculó la suma total de vuelos cancelados. Finalmente como en los casos anteriores se ordenó descendentemente por la suma calculada y nos quedamos con el top 5 utilizando un limit de 5.

Los resultados de esta métrica se muestran de la siguiente manera: **Huracán; Fecha (yyyy-M-d); #vuelos-cancelados**

Problemas encontrados

Durante el desarrollo de las métricas nos encontramos con el siguiente problema. El job de map reduce se lanzaba correctamente pero fallaba en los nodos del cluster diciendo que no

encontraba algunas clases de Java. Luego de consultar con la cátedra, nos dimos cuenta que no se estaban incluyendo algunos jars en el script de pig que hacía que cuando se distribuyera el código, los nodos no supieran de dónde sacar las clases.

Hive

Para la tercer parte del trabajo práctico, se utilizó Hadoop Hive para poder obtener otras métricas con los datos mencionados en la primer parte.

Debido a que usa una sintaxis SQL, no nos trajo dificultad resolver las métricas pedidas, aunque cuando nos encontrábamos con algún error, el mismo era muy poco claro. Sobre todo, la falta de claridad se notó más cuando el error era por sintaxis.

Al igual que Pig, Hive provee una gran abstracción sobre map-reduce aunque deja que el usuario haga algunas sugerencias sobre cómo procesar la query (ver la métrica **Las rutas más voladas)**

Ejecución de las métricas

hive -f script.hql

Los scripts aceptan el parámetro FLIGHT_DATA con la ubicación de los archivos con los datos de los vuelos

Ejemplo:

hive -hiveconf FLIGHT_DATA='/user/hadoop/ITBA/TP1/INPUT/SAMPLE/data' -f script.hql

Adicionalmente, el script metric9.hql acepta AIRPORTS_DATA, con la ubicación del archivo con los datos de los aeropuertos

IMPORTANTE!

El archivo con los datos de los aeropuertos debe estar SOLO en la carpeta. Si hay otros .csv, lanza un error.

Tampoco admite que se le pase el path a un archivo (por ejemplo /user/hadoop/ITBA/TP1/INPUT/SAMPLE/ref/airports.csv)

Si se quisiera usar /user/hadoop/ITBA/TP1/INPUT/SAMPLE/ref/, entonces dicha carpeta debería contener sólamente el archivo airports.csv

Los output se guardan en el HDFS en el siguiente path

/user/hadoop/output/metricX

donde X es el nro de la métrica

Consideraciones sobre las métricas

Para las siguientes métricas se generaron las tablas de vuelos y aeropuertos según fue necesario.

Las rutas más voladas

En un principio se pensó hacer una consulta con una sub consulta dentro de la cláusula WHERE. Esto no es posible en hive y por lo tanto se tuvo que optar por la siguiente solución.

Inicialmente se crea una tabla con las siguiente columnas: año, origen, destino y total. En cada tupla se guarda el año, la ruta y la cantidad de vuelos que recorrieron esa ruta. Es importante destacar que la ruta FDO-LAX no se consideró como la misma ruta que LAX-FDO.

Una vez que se tiene esta tabla se realiza la siguiente consulta:

```
SELECT year, origin, dest, total
    FROM tmp_table
    DISTRIBUTE BY year
    SORT BY year, total desc
```

Esta consulta simplemente selecciona las columnas ordenándolas por año y el total de vuelos para esa ruta. Sin embargo, la clave de esta consulta es el DISTRIBUTE BY. Esto hace que a cada reducer tenga todos los datos para una clave dada. En este caso, por ejemplo, el reducer 1 podría tener las tuplas correspondientes al año 1988.

Sobre esa consulta, se hace una nueva consulta que tiene la siguiente forma:

```
SELECT *, rank(year) as row_number
```

Lo que se hace en este caso es asignarle un ranking a cada tupla. La función rank es una UDF que simplemente asigna un nro distinto a cada tupla con el mismo año. Por lo tanto, el resultado de esta query podría ser el siguiente

Año	Origen	Destino	Total	Ranking
1998	FDO	MDQ	200	1

1998	MDQ	FDO	175	2
1998	LAX	FDO	166	3

Como la tabla estaba ordenada por año y total, nos aseguramos que todas las tuplas con el mismo año sean consecutivas y que además la que tiene mayor total esté arriba de todo.

Finalmente se hace una última consulta donde se le pide todos aquellas tuplas que tengan un ranking menor o igual a 10.

Los resultados de esta métrica se muestran de la siguiente manera: **Año Origen Destino #total-rutas ranking**

Cantidad de vuelos diarios durante el mes de septiembre 2001

Para realizar esta métrica creamos una tabla temporal con los vuelos del mes 9 y el año 2001, quedándonos con la fecha en formato “yyyy-M-d” y el campo que indica si el vuelo fue cancelado. Luego agrupando por fecha contamos la cantidad de vuelos y sumamos la cantidad de demoras que hubo.

Los resultados de esta métrica se muestran de la siguiente manera:

Fecha(yyy-M-d) #vuelos #vuelos-cancelados

La hora en que salió el último vuelo el 11 de septiembre de 2001

Para realizar esta métrica agrupamos la tabla de vuelos por aeropuerto, y para cada uno calculamos la hora máxima de despegue.

Para este caso, como la hora está en formato *hhmm* se decidió convertirla a un entero y luego obtener el máximo de todos las horas, agrupando por aeropuerto. De esta manera la métrica se resuelve de manera muy sencilla.

Los resultados de esta métrica se muestran de la siguiente manera:

originIATA #hora(hhmm)

Para cada día de 2001, el promedio de demoras de despegue

Para realizar esta métrica generamos una tabla temporal con los vuelos del año 2001

con su demora de despegue. Luego agrupamos por fecha y calculamos el promedio de las demoras con la función de agregación avg.

Los resultados de esta métrica se muestran de la siguiente manera:

fecha(yyyy-M-d) #promedio-demoras

Problemas encontrados

Un problema con el que nos encontramos durante el desarrollo fue que si se crea una tabla y luego se hace un LOAD de ciertos archivos, Hive borra dichos archivos del HDFS. Por lo tanto, corríamos la métrica una vez y al querer correrla de vuelta, obteníamos un error ya que los archivos no se encontraban.

Conclusiones

La generación de las diferentes métricas utilizando las diferentes tecnologías de Hadoop nos permitió poder entrar mucho más en detalle en cómo se implementan diferentes funcionalidades en modo Batch. Pudimos entender como funciona cada una, los diferentes problemas que existen y las ventajas de cada una. Debido a nuestro amplio conocimiento de Java, Java Map Reduce fue lo más simple, debido también a que hay mucha más documentación y soporte al respecto. También hay que destacar la simpleza para poder testear los mappers y reducers.

Consideramos que la abstracción que proveen Hive y Pig por sobre map reduce es algo muy interesante. Sin embargo, notamos que la ejecución de las métricas utilizando Hadoop Map-Reduce es más eficiente que aquellas que se realizaron con Hive y Pig.