Hands-on II Big Data & Cloud Computing

Magister en Data Science, Universidad del Desarrollo

[redacted]
[redacted]

Objetivos

En esta actividad los alumnos deben responder preguntas asociadas al dataset GDELT utilizado en el Hands-on I, utilizando Apache Spark a través de las interfaces PySpark y Spark SQL.

Tareas a realizar

Parte 1: Exploración del dataset con PySpark y Spark SQL

1. ¿Cuál fue el sitio web con mayor cantidad de eventos registrados el 6 de Octubre del 2020?
    a. Implemente la consulta usando sólo Dataframes y HiveContext en PySpark (al menos los filtros, joins y operaciones agregadas deben ser ejecutados en los Dataframes, no calculados directo en una query SQL) (5 puntos).
    b. Implemente la consulta usando Spark SQL (puede reutilizar las consultas desarrolladas en el Hands-on I). ¿Cómo se compara en términos de tiempo de ejecución con Hive? (1 punto)
2. ¿Cuál es la ciudad con mayor número de apariciones en los eventos registrados el 6 de Octubre del 2020?
    a. Implemente la consulta usando sólo Dataframes y HiveContext en PySpark (al menos los filtros, joins y operaciones agregadas deben ser ejecutados en los Dataframes, no calculados directo en una query SQL) (5 puntos).
    b. Implemente la consulta usando Spark SQL (puede reutilizar las consultas desarrolladas en el Hands-on I). ¿Cómo se compara en términos de tiempo de ejecución con Hive? (1 punto)
3. ¿Cuál fue el evento generado durante Noviembre del 2020 con mayor cantidad de menciones en los medios?
    a. Implemente la consulta usando sólo Dataframes y HiveContext en PySpark (al menos los filtros, joins y operaciones agregadas deben ser ejecutados en los Dataframes, no calculados directo en una query SQL) (10 puntos).
    b. Implemente la consulta usando Spark SQL (puede reutilizar las consultas desarrolladas en el Hands-on I). ¿Cómo se compara en términos de tiempo de ejecución con Hive? (2 punto)
4. ¿Cuáles son los 10 sitios web que generaron en promedio la mayor cantidad de menciones poco confiables (<40% de confianza) durante Noviembre del 2020?
    a. Implemente la consulta usando sólo Dataframes y HiveContext en PySpark (al menos los filtros, joins y operaciones agregadas deben ser ejecutados en los Dataframes, no calculados directo en una query SQL) (10 puntos).
    b. Implemente la consulta usando Spark SQL (puede reutilizar las consultas desarrolladas en el Hands-on I). ¿Cómo se compara en términos de tiempo de ejecución con Hive? (2 punto)
5. ¿Cuáles son los 10 sitios web que generaron en promedio la mayor cantidad de menciones muy confiables (>80% de confianza) durante Septiembre, Octubre, Noviembre y Diciembre del 2020? (promedio de todas las apariciones durante los 4 meses)

a. Implemente la consulta usando sólo Dataframes y HiveContext en PySpark (al menos los filtros, joins y operaciones agregadas deben ser ejecutados en los Dataframes, no calculados directo en una query SQL) (20 puntos).

b. Implemente la consulta usando Spark SQL (puede reutilizar las consultas desarrolladas en el Hands-on I). ¿Cómo se compara en términos de tiempo de ejecución con Hive? (4 puntos)

6. Diseñe una consulta que le parezca relevante y que involucre cruzar los datasets export y mentions usando la llave GlobalEventID (primeros campos en ambos).

    a. Implemente la consulta con las herramientas de PySpark o Spark SQL. Si escogió Spark SQL, ¿cómo se compara en términos de tiempo de ejecución con Hive? (3 puntos)

Parte 2: Performance de ejecución con Spark

1. Elija una de las consultas de la parte I y realice un análisis de performance usando Spark UI y el cálculo de eficiencia de la ejecución (3 puntos).
2. Modifique los parámetros de ejecución para mejorar el performance, ¿cuál fue la configuración que escogió y cuál puede ser la explicación de esto? (3 puntos)
3. Si escogió Spark SQL, ¿cuál fue el máximo speedup obtenido con respecto a la misma consulta en Hive? (speedup=tiempo_hive/tiempo_sparksql-optimizado) (4 puntos)

Entregables

Se pueden realizar las tareas en grupos de 2 o 3 personas. Cada grupo debe entregar un documento detallando el proceso realizado para responder cada pregunta, incluyendo screenshots de los resultados dentro de la plataforma de cómputo (screenshot a la pantalla completa y un zoom a cada parte de interés). Los comandos o el código utilizados en cada pregunta deben ser explicados paso a paso (screenshots de apoyo también pueden ser incluídos de manera opcional).

Para realizar las tareas, debe utilizar un cluster Hadoop con algún proveedor de servicios cloud. En particular, se recomienda el servicio Dataproc de Google Cloud, Azure HDInsight, o EMR de AWS. Si por alguna razón no puede crear una cuenta gratis con algunos de estos proveedores, contáctese con el profesor. Advertencia: no utilice su tarjeta de crédito normal para activar una cuenta, se recomienda usar tarjetas de crédito virtuales con un mínimo de USD habilitado (por ejemplo 10 USD).

Se recomienda que en cada grupo de alumnos al menos un integrante tenga activado un servicio en alguno de los proveedores cloud, y entre todos gestionen el uso de los recursos de manera eficiente (eliminar/detener el cluster si no lo están utilizando).

Fecha de entrega

Martes 18 de Mayo antes de las 23:59 hrs, a través de la sección Tareas de Canvas.

Consultas

A través del foro de Canvas o directamente al profesor al correo o.peredo@udd.cl.

Ejemplo

Como ejemplo, se mostrará el paso a paso para responder una pregunta parecida a las planteadas en las tareas.

¿Cuál fue el evento con mayor cantidad de menciones muy confiables (>60% de confianza) durante Junio del 2018?

Solución:

Para ver la carga de los datos en un tabla Hive, revisar ejemplo del Hands-on I, sección Hive.

PySpark (sólo con HiveContext)

Utilizando la misma consulta SQL desarrollada en el Hands-on I, podemos insertarla dentro del código productivo de PySpark quedando de la siguiente manera:

```
sshuser@hn0-cluste:~/profesor$ cat ejemploPySpark.py

from pyspark import SparkConf, SparkContext

from pyspark.sql import HiveContext

conf = SparkConf().setAppName("test-magister")

sc = SparkContext(conf=conf)

hiveCtx = HiveContext(sc)

query = hiveCtx.sql("select * from (select GlobalEventID, COUNT(Confidence) as cnt from 201806mentions where
Confidence>60 group by GlobalEventID) a order by cnt desc limit 1")

for x in query.collect():

    print(x)
```

El resultado es el siguiente:

```
sshuser@hn0-cluste:~/profesor$ spark-submit ejemploPySpark.py > out.txt 2>/dev/null

sshuser@hn0-cluste:~/profesor$ cat out.txt

Row(GlobalEventID=764935245, cnt=1285)
```

PySpark (sólo con Dataframes)

Ahora la misma consulta, pero utilizando Dataframes para todas las operaciones, dentro del código productivo de PySpark queda de la siguiente manera:

```
sshuser@hn0-cluste:~/profesor$ cat ejemploPySpark2.py

from pyspark import SparkConf, SparkContext

from pyspark.sql import HiveContext

from pyspark.sql.functions import col,desc


conf = SparkConf().setAppName("test-magister")

sc = SparkContext(conf=conf)

hiveCtx = HiveContext(sc)


df = hiveCtx.sql("select * from 201806mentions")

query= df.filter(df.confidence>60).groupBy('globaleventid').count().select('globaleventid',
col('count').alias('cnt')).sort(desc('cnt')).limit(1)
```

```
for x in query.collect():
    print(x)
```

El resultado es el siguiente:

```
sshuser@hn0-cluste:~/profesor$ spark-submit ejemploPySpark2.py > out.txt 2>/dev/null
sshuser@hn0-cluste:~/profesor$ cat out.txt
Row(GlobalEventID=764935245, cnt=1285)
```

Spark SQL

La ejecución de la consulta usando Spark SQL se realiza a través del cliente spark-sql, guardando en un archivo la consulta y ejecutando de la siguiente manera:

```
sshuser@hn0-cluste:~/profesor$ cat query.hql
select * from (select GlobalEventID, COUNT(Confidence) as cnt from 201806mentions where Confidence>60 group by GlobalEventID) a order by cnt desc limit 1
sshuser@hn0-cluste:~/profesor$ spark-sql -S -f query.hql > out.txt
sshuser@hn0-cluste:~/profesor$ cat out.txt
764935245       1285
```

Búsqueda de GlobalEventID en archivos “export”

Teniendo el ID del evento, procedemos a buscarlo en los archivos “export”, también convertidos a formato texto plano de la misma manera que los archivos “mentions”:

```
sshuser@hn0-cluste:~$ hadoop fs -cat wasb://hands-on@magisterudd.blob.core.windows.net/gdelt_gz/2018/06/201806*export* | grep 764935245
764935245       20180617        201806  2018    2018.4575       AFGINSTAL       TALIBAN AFG     TAL
INS                     1190    190     19      4       -10.0   24      4       24      -6.88433758066071
4       Jalalabad, Nangarhar, Afghanistan       AF      AF18    9992834.4265 70.4515 -3377673        0
4       Jalalabad, Nangarhar, Afghanistan       AF      AF1899928    34.4265 70.4515 -3377673
20180617113000  https://www.journal-news.com/news/world/the-latest-suicide-bombing-afghanistan-kills/yVnwhBtVPHGxtzEucP31SJ/
```

Herramientas

Línea de comandos Linux

`cd`: cambiar de directorio.

`ls`: listar archivos en un directorio.

`cat`: imprimir contenido de un archivo en la salida standard (pantalla).

`head`: imprimir las primeras líneas de un archivo en la salida standard.

`tail`: imprimir las últimas líneas de un archivo en la salida standard.

`more`: imprimir contenido de un archivo por partes de manera incremental.

`wc`: contar bytes, caracteres, palabras o filas de un archivo.

`sort`: ordenar un archivo según orden lexicográfico o numérico.

`uniq`: eliminar o mostrar las líneas repetidas de un archivo.

`awk`: lenguaje de programación para procesamiento y detección de patrones en texto.

`grep`: herramienta para detección y transformación de patrones usando expresiones regulares.

`sed`: herramienta para detección y transformación de patrones usando expresiones regulares.

Ejemplos de comandos "one-liner" en Linux

Conteo de filas:

```
cat file.txt | wc -l
```

Conteo de columnas (con separador ','):

```
cat file.txt | awk -F ',' '{print NF}' | sort | uniq -c
```

Conteo de columnas (con separador espacio):

```
cat file.txt | awk '{print NF}' | sort | uniq -c
```

Imprimir la segunda columna de todas las filas (con separador ','):

```
cat file.txt | awk -F ',' '{print $2}'
```

Conteo de filas con la misma llave (ejemplo de fila con llave en la primera columna: id1,val1,val2,val3):

```
cat file.txt | awk -F ',' '{map[$1]+=1}END{for(key in map){print key,map[key]}}'
```

Suma de los valores en la primera columna (con separador ','):

```
cat file.txt | awk -F ',' '{sum+=$1}END{print sum}'
```

Filtrar filas con la palabra error (sólo minúsculas):

```
cat file.txt | grep 'error'
```

Seleccionar filas con la palabra error (minúsculas o mayúsculas):

```
cat file.txt | grep '[Ee][Rr][Rr][Oo][Rr]'
```

Remover filas con la palabra error (sólo minúsculas):

```
cat file.txt | grep -v 'error'
```

Remover filas con la palabra error (minúsculas o mayúsculas), simplificado:

```
cat file.txt | grep -iv 'error'
```

Calcular el promedio de los valores en la segunda columna (con separador ','):

```
cat file.txt | awk -F ',' '{sum+=$2; counter+=1}END{print sum/counter}'
```

Calcular la desviación estándar de los valores en la segunda columna (con separador ','):

```
cat file.txt | awk -F ',' '{sum+=$2; counter+=1; squaresum+=($2*$2)}END{print
sqrt( (squaresum/counter) - (sum/counter)^2 )}'
```

Comandos en el filesystem Hadoop

`hadoop fs -ls ruta`: listar archivos en un directorio en el filesystem distribuído.

`hadoop fs –cat ruta/archivo`: imprimir contenido de un archivo (texto plano) en el filesystem distribuído en la salida standard (pantalla).

`hadoop fs –text ruta/archivo`: imprimir contenido de un archivo (comprimido o serializado) en el filesystem distribuído en la salida standard (pantalla).

`hadoop fs –copyFromLocal archivo ruta/` : copiar un archivo en el filesystem local (linux) hacia el filesystem distribuído.

`hadoop fs –copyToLocal ruta/archivo .` : copiar un archivo desde el filesystem distribuído hacia el filesystem local (Linux), en la carpeta actual.

Referencias

https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-perf

"Spark - The Definitive Guide", 1st Edition, Bill Chambers and Matei Zaharia (2018)

"Processing Big Data with Azure HDInsight", 1st Edition, Vinit Yadav (2017)