Máster en Ingeniería del Software: Cloud, Datos y Gestión TI

Big Data 2023

Entregable 1 - Map Reduce

Realizado por José Antonio Zamudio Amaya

UNIVERSIDAD D SEVILLA

Índice de contenidos

1. Propuesta

La pandemia de COVID-19 ha afectado a todo el mundo y sigue siendo un importante problema sanitario. Con la creciente cantidad de datos que se están recopilando, es importante analizar los datos para comprender el impacto del virus en diferentes países/regiones. En esta propuesta, sugerimos un proyecto MapReduce utilizando HDFS y Hadoop para analizar el número diario de casos confirmados, muertes y casos recuperados para cada país/región.

El proyecto incluirá los siguientes pasos:

1. Almacenar el conjunto de datos COVID en HDFS.
2. Escribir un programa MapReduce para procesar el conjunto de datos y obtener el número total de casos confirmados, muertes y casos recuperados para cada país/región en cada día.
3. Almacenar el resultado en HDFS.
4. Escribe otro programa MapReduce para calcular el número medio diario de casos confirmados, muertes y casos recuperados para cada país/región.
5. Almacene el resultado final en HDFS.

Programa MapReduce:

1. Mapper: Analiza cada línea del conjunto de datos COVID y emite pares clave-valor donde la clave es el país/región y la fecha, y el valor es una matriz que contiene los casos confirmados, las muertes y los casos recuperados.
2. Reducer: Agrega los valores de cada clave (país/región y fecha) sumando los casos confirmados, las muertes y los casos recuperados.

La salida final se almacenará en un archivo en HDFS que contenga el número de casos confirmados, muertes y casos recuperados para cada país/región. Estos datos podrían utilizarse para analizar la propagación de COVID-19 a lo largo del tiempo y para comparar el impacto del virus en diferentes países/regiones.

2. Dataset

El conjunto de datos COVID-19 proporcionado contiene información diaria sobre el número de casos confirmados, muertes y recuperaciones relacionados con la enfermedad COVID-19. Cada registro en el conjunto de datos incluye información sobre la provincia/estado, país/región, latitud, longitud, fecha, número de casos confirmados, número de muertes, número de recuperaciones, número de casos activos y región de la OMS.

3. Código

En esta sección se muestran los fragmentos de código utilizados para las funciones junto a explicaciones detalladas de su funcionamiento.

3.1. Mapper

```
#!/usr/bin/python3.6

import sys
import csv

# Input format:
Province/State,Country/Region,Lat,Long,Date,Confirmed,Deaths,Recovered,A
ctive,WHO Region

# Output format: key-value pairs where the key is the country/region and
the date, and the value is an array containing the confirmed cases,
deaths, and recovered cases


# Parse each line of the CSV file and emit key-value pairs
reader = csv.reader(sys.stdin)
next(reader) # Skip header row
for row in reader:
      country_region = row[1]
      confirmed = int(row[5])
      deaths = int(row[6])
      recovered = int(row[7])
      value = [confirmed, deaths, recovered]
      print('%s;%s' % (country_region, value))
```

1. La primera línea "#!/usr/bin/python3.6" indica que se está utilizando Python 3.6 para este programa.
2. Las siguientes líneas importan los módulos necesarios: "sys" y "csv". El módulo "sys" proporciona acceso a algunos objetos utilizados o mantenidos por el intérprete y para interactuar con el intérprete de Python. El módulo "csv" proporciona clases para trabajar con archivos CSV.
3. Los comentarios explican el formato de entrada y salida. El archivo CSV de entrada tiene las columnas Province/State, Country/Region, Lat, Long, Date, Confirmed, Deaths, Recovered, Active, WHO Region. El formato de salida es pares clave-valor donde la clave es el país/región y la fecha, y el valor es una matriz que contiene los casos confirmados, muertes y casos recuperados.
4. Luego, se usa el módulo "csv" para leer cada línea del archivo CSV de entrada y crear un objeto de "lector de CSV".

5. La línea "next(reader)" se utiliza para saltar la primera fila del archivo CSV que contiene los encabezados de columna.
6. El bucle "for" recorre cada línea del archivo CSV y realiza lo siguiente:
   a. Extrae el nombre del país/ región de la segunda columna (índice 1) y lo guarda en la variable "country_region".
   b. Extrae el número de casos confirmados de la sexta columna (índice 5) y lo convierte en un número entero. Luego, guarda este número en la variable "confirmed".
   c. Extrae el número de muertes de la séptima columna (índice 6) y lo convierte en un número entero. Luego, guarda este número en la variable "deaths".
   d. Extrae el número de casos recuperados de la octava columna (índice 7) y lo convierte en un número entero. Luego, guarda este número en la variable "recovered".
   e. Crea una lista "value" que contiene los valores de "confirmed", "deaths" y "recovered".
   f. Imprime una cadena formateada que contiene la variable "country_region" y "value". La cadena formateada utiliza el signo ";" como separador entre la clave y el valor.

3.2. Reducer

```
#!/usr/bin/python3.6

from operator import add
import sys

current_country = None
current_totals = [0, 0, 0]

for line in sys.stdin:
      line = line.strip()
      country, counts_str = line.split(";")
      counts = [int(x) for x in counts_str[1:-1].split(",")]

      if current_country == country:
      current_totals = map(add, current_totals, counts)
      else:
      if current_country:
            print(current_country + ": " + str(list(current_totals)))
      current_country = country
      current_totals = counts

if current_country:
```

```
print(current_country + ": " + str(list(current_totals)))
```

1. La línea de encabezado se omite con la función next(reader).
2. Se procesa cada línea del archivo de entrada.
3. Se divide cada línea en las columnas de la lista row, y la columna 1 (Country/Region) se asigna a la variable country.
4. La columna 5 (Confirmed), la columna 6 (Deaths) y la columna 7 (Recovered) se asignan a la variable counts y se convierten en números enteros.
5. Los valores counts se agregan a los totales actuales utilizando la función map(add, current_totals, counts).
6. Cuando se cambia el país, se emite el país anterior y sus totales.
7. Al final, se emite el último país y sus totales.
8. Los valores totales se emiten como una lista de tres elementos que contiene el número total de casos confirmados, muertes y recuperaciones.
9. La salida final se presenta en la consola con el nombre del país y la lista de totales de casos confirmados, muertes y recuperaciones.

4. Ejecución

Descargamos los archivos necesarios con el comando wget del repositorio https://github.com/joszamama/big-data y los subimos al directorio hdfs con el comando “hdfs dfs -put /home/maria_dev/mapreduce/filename /”.

Debemos cargar tanto el csv como los scripts de python, y ahora ejecutaremos:

“hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar -input /covid_data.csv -output /output -mapper mapper.py -reducer reducer.py -file mapper.py -file reducer.py”

Este comando ejecuta un trabajo de MapReduce en Hadoop utilizando el archivo de flujo Hadoop (hadoop-streaming.jar). El trabajo toma como entrada un archivo llamado "covid_data.csv" y escribe su salida en una carpeta llamada "output".

El archivo "mapper.py" y "reducer.py" son scripts de Python que se utilizan como el código de mapeo y reducción para el trabajo de MapReduce. Los archivos "mapper.py" y "reducer.py" se especifican con el parámetro "-file" para que Hadoop pueda distribuir estos archivos a través de todos los nodos del clúster.

Esto nos genera un directorio /output con dos ficheros dentro. Veamos el resultado con “hdfs dfs -cat /output/part-00000”

5. Resultados

```
[maria_dev@sandbox-hdp mapreduce]$ hadoop jar /usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming.jar -input /covid_data.csv -output /outpu
23/03/25 22:14:33 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
packageJobJar: [mapper.py, reducer.py] [/usr/hdp/2.6.5.0-292/hadoop-mapreduce/hadoop-streaming-2.7.3.2.6.5.0-292.jar] /tmp/streamjob892026347
23/03/25 22:14:36 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
23/03/25 22:14:36 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
23/03/25 22:14:36 INFO client.RMProxy: Connecting to ResourceManager at sandbox-hdp.hortonworks.com/172.18.0.2:8032
23/03/25 22:14:36 INFO client.AHSProxy: Connecting to Application History server at sandbox-hdp.hortonworks.com/172.18.0.2:10200
23/03/25 22:14:38 INFO mapred.FileInputFormat: Total input paths to process : 1
23/03/25 22:14:38 INFO mapreduce.JobSubmitter: number of splits:2
23/03/25 22:14:39 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1679771671771_0001
23/03/25 22:14:40 INFO impl.YarnClientImpl: Submitted application application_1679771671771_0001
23/03/25 22:14:40 INFO mapreduce.Job: The url to track the job: http://sandbox-hdp.hortonworks.com:8088/proxy/application_1679771671771_0001/
23/03/25 22:14:40 INFO mapreduce.Job: Running job: job_1679771671771_0001
23/03/25 22:14:52 INFO mapreduce.Job: Job job_1679771671771_0001 running in uber mode : false
23/03/25 22:14:52 INFO mapreduce.Job:  map 0% reduce 0%
23/03/25 22:15:08 INFO mapreduce.Job:  map 1% reduce 0%
23/03/25 22:15:12 INFO mapreduce.Job:  map 100% reduce 0%
23/03/25 22:15:27 INFO mapreduce.Job:  map 100% reduce 100%
23/03/25 22:15:29 INFO mapreduce.Job: Job job_1679771671771_0001 completed successfully
23/03/25 22:15:29 INFO mapreduce.Job: Counters: 49
```

```
Afghanistan: [58414162, 2544820, 18289880]
Albania: [70630218, 1170623, 22652652]
Algeria: [83275239, 2356590, 25670233]
Andorra: [7895545, 69323, 3085649]
Angola: [22165880, 523594, 5895800]
Antarctica: [891, 0, 0]
Antigua and Barbuda: [1112246, 27520, 191813]
Argentina: [2088265538, 43144593, 711610324]
Armenia: [124648195, 2499390, 51634388]
Australia: [180137424, 800414, 8821094]
Austria: [395023309, 5020315, 131744371]
Azerbaijan: [193783927, 2609243, 68764706]
Bahamas: [8104902, 193231, 2484156]
Bahrain: [117391827, 507745, 48612087]
Bangladesh: [568482791, 9257135, 190856309]
Barbados: [5742800, 47999, 692334]
Belarus: [228499009, 1747746, 88191538]
Belgium: [655584626, 13719128, 3797810]
Belize: [9903486, 188515, 2947228]
Benin: [6737379, 58603, 1723986]
Bhutan: [1117658, 931, 294915]
Bolivia: [218552731, 8216753, 69896258]
Bosnia and Herzegovina: [103147606, 4425666, 35933408]
Botswana: [53479803, 652689, 9700789]
Brazil: [8712316058, 236486341, 3412350387]
Brunei: [3112639, 16685, 82751]
```

Los datos proporcionados corresponden a la población total, el número de casos confirmados y el número de muertes por COVID-19 en diferentes países hasta una fecha no especificada en el archivo.

Por ejemplo, en Afganistán la población total es de 58,414,162 personas, se han confirmado 2,544,820 casos de COVID-19 y se han registrado 18,289,880 muertes relacionadas con esta enfermedad.

Se puede observar que los países varían ampliamente en términos de su población total, desde los millones de habitantes de China e India hasta los pocos miles de habitantes de países como Andorra y Antártida. También se puede ver que algunos países tienen una población urbana considerablemente mayor que otros, lo que puede reflejar diferencias en la infraestructura y la calidad de vida en áreas urbanas y rurales.

6. Puntos de mejora y trabajo futuro

1. Actualización de datos: los datos utilizados en este código son de 2021, por lo que una posible mejora sería actualizarlos a la fecha más reciente.
2. Visualización de datos: una posible mejora sería crear gráficos o tablas para visualizar los datos de una manera más fácil de entender para el usuario.
3. Incorporación de más datos: si bien este código ya incluye datos de población, superficie y PIB, se podrían agregar más variables para tener una imagen más completa de cada país.
4. Análisis de datos: una posible mejora sería utilizar técnicas de análisis de datos para encontrar patrones o relaciones entre las diferentes variables y así poder obtener información más valiosa.