viu Universidad Internacional de Valencia

Actividad 1
Programación en Hadoop

Procesamiento de datos masivos

Òscar Garibo

Introducción:

El procesamiento masivo de datos no siempre se puede realizar con tecnologías tradicionales. En muchas ocasiones se tienen que utilizar tecnologías Big Data como Hadoop MapReduce y Spark.

Objetivo:

Conocer el modelo de procesamiento MapReduce y las principales herramientas Big Data.

Desarrollar programas Big Data utilizando el framework Hadoop MapReduce.

Trabajo previo:

Lectura del material docente de la parte específica que se encuentra disponible desde el comienzo del curso en la carpeta: Recursos y materiales>1. Materiales docentes:

Visualización de las videoconferencias teóricas (VC), es decir las sesiones de clases.

Metodología:

En las videoconferencias teóricas (VC) se expondrá al alumno conocimientos, material e indicaciones suficientes para que pueda elaborar una unidad didáctica basada en el aprendizaje y enseñanza por competencias en matemáticas e informática. Las actividades se centrarán en poner en práctica y asentar los conocimientos adquiridos en las videoconferencias teóricas relacionadas al tema.

Actividades a elaborar:

Desarrollo de dos programas Big Data utilizando el framework Hadoop MapReduce o con Hadoop Streaming (puede usar java o python) y análisis de un programa con defectos. En los ejercicios 1 y 2 se valorará positivamente la optimización de las soluciones, por ejemplo minimizando el número de información que se transmite por el cluster.

1. (7,0 ptos) Dado un dataset que contenga entradas con la forma “persona;tienda;gasto”, crea un programa llamado mediaGastadoPersonaTienda que para cada persona indique su gasto medio por tienda, siguiendo el formato persona;tienda;gastomedio. Ejemplo:

Entrada	Salida
Alice;Tienda1;50	Alice;Tienda1;75
Alice;Tienda2;20	Alice;Tienda2;20
Bob;Tienda1;30	Bob;Tienda1;25
Alice;Tienda1;100	
Bob;Tienda1;20	

2. (7,0 ptos) Para este ejercicio utilizarán el fichero de entrada cite75_99.txt que puede ser descargado del National Bureau of Economic Research (NBER) de EEUU (http://www.nber.org/patents/).
Una descripción detallada de este fichero puede encontrarse en:
Hall, B. H., A. B. Jaffe, and M. Trajtenberg (2001). "The NBER Patent Citation Data File: Lessons, Insights and Methodological Tools." NBER Working Paper 8498.

Este fichero contiene citas de patentes emitidas entre 1975 y 1990 en los EEUU. Es un fichero CSV (*comma-separated values*) con más de 16,5 millones de filas, y las primeras líneas son como sigue:

```
"CITING","CITED"
3858241,956203
3858241,1324234
3858241,3398406
3858241,3557384
3858241,3634889
3858242,1515701
3858242,3319261
3858242,3668705
.....
```

La primera línea contiene una cabecera con la descripción de las columnas. Cada una de las otras líneas indica una cita que la patente con el número de la primera columna ha hecho a la patente con el número en la segunda. Por ejemplo, la segunda fila indica que la patente nº 3858241 ("citing" o *citante*) hace una cita a la patente nº 956203 ("cited" o citada).

El fichero está ordenado por las patentes citantes. Así podemos ver que la patente nº 3858241 cita a otras 5 patentes.

Deben implementar un programa MapReduce escrito en Java o Python (a elegir) que, para cada patente de cite75_99.txt, obtenga la lista de las que la citan:

- Posible implementación:
  - El mapper obtiene cada línea del fichero de entrada, separar los campos y los invierte (para obtener como clave intermedia la patente citada y como valor intermedio la patente que la cita), por ejemplo:
    3858245, 3755824 → 3755824 3858245
  - El reducer, para cada patente recibe como valor una lista de las que la citan, ordena esa lista numéricamente y la convierte en un *string* de números separados por coma
    3755824 {3858245 3858247. . . } → 3755824 3858245, 3858247...
- **Formato de salida**: *patente patente1,patente2...* (la separación entre la clave y los valores debe ser un **tabulado**)

- La salida debe de estar **ordenada** por la clave (patentes citadas) y los valores también deben estar ordenados por patentes citantes.
- Los valores se deben guardar separados por coma, sin espacios en blanco entre ellos.
- Deben tener en cuenta la cabecera, para que no aparezca en la salida (el fichero de entrada no debe modificarse de ninguna manera)

3. (6,0 ptos) Dado el siguiente programa Big Data que tiene defectos de diseño, se debe entender cuál es el defecto y hacer un informe que contenga: (1) Nombre y apellidos, (2) Tiempo empleado por el alumno en entender cuál es el defecto, y (3) Descripción del defecto. Todos los archivos del programa se encuentran en la carpeta: “PersonasQueCompranEnMuchasTiendas”.

IMPORTANTE: los defectos del programa son defectos del diseño. La sintaxis del programa es correcta, pero la funcionalidad del programa no se ha programado correctamente siguiendo el modelo de procesamiento Big Data. Por ello, puede que los programas los ejecutemos en nuestro ordenador y funcionen correctamente, pero al moverlos al cluster Big Data empiecen a fallar porque están ejecutando varias Mapper, Combiner, Reducer, algunas de ellas puede que se re-ejecuten, acaben antes, etc. Es decir, si ejecutamos dos veces en un cluster de producción el mismo programa con los mismos datos, podría una vez emitir la salida correcta y otra vez una incorrecta. Esto es porque el programa no se diseñó adecuadamente siguiendo el modelo de procesamiento Big Data. Un programa bien diseñado, debería ejecutarse correctamente independientemente de cómo el cluster Big Data decida ejecutarlo. A continuación, se describe el programa y las preguntas que se tienen que responder:

Conjunto de datos: cada fila representa una compra que hizo una persona en una tienda. La fila tiene la siguiente estructura: “persona tienda”.
Por ejemplo “Alice Nunc Corp.” significa que Alice compró en “Nunc Corp.”.
Descripción del programa: el programa tiene que obtener cuáles fueron las personas que compraron en 3 o más tiendas diferentes. Es decir, si se tiene como entrada:

Alice Nunc Corp.
Alice Arcu Aliquam Company
Alice Pharetra Quisque Ac Company
Alice Nunc Corp,
Bob Nunc Corp.

el programa debería emitir Alice porque compró en 3 o más tiendas distintas. Concretamente, en el ejemplo anterior, Alice compró en 3 tiendas: en “Nunc Corp.” (dos compras), “Arcu Aliquam Company”, y “Pharetra Quisque Ac Company”. El programa no emite Bob porque sólo compró en una tienda.

Código: el equipo de desarrollo ha creado los *scripts:*
mapperPersonasQueCompranEnMuchasTiendas.py,
combinerPersonasQueCompranEnMuchasTiendas.py
reducerPersonasQueCompranEnMuchasTiendas.py.

Comando de ejecución:

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-2.4.0.jar -file ./mapperPersonasQueCompranEnMuchasTiendas.py
-mapper ./mapperPersonasQueCompranEnMuchasTiendas.py
-file ./combinerPersonasQueCompranEnMuchasTiendas.py
-combiner combinerPersonasQueCompranEnMuchasTiendas.py
-file ./reducerPersonasQueCompranEnMuchasTiendas.py
-reducer ./reducerPersonasQueCompranEnMuchasTiendas.py
-input casoDePrueba.txt -output ./misalida

(notar que dependiendo de la versión de Hadoop, habría que cambiar el .jar y también las entradas y salidas)

Problema: el equipo de analistas ha observado que el programa no funciona correctamente. Según reportan, han ejecutado el programa con los mismos datos y unas veces proporciona las personas que realmente compraron en 3 o más tiendas, pero en otras ocasiones el programa sólo emite alguna de esas personas. De todos los datos que hay en producción, han reportado que el defecto se puede reproducir con sólo 104 datos que están disponibles en casoDePrueba.txt, en la carpeta PersonasQueCompranEnMuchasTiendas. La salida esperara es Alice, Carol y Dave. Sin embargo, cuando el programa se ejecuta en producción hay ocasiones en las que emite correctamente a esas tres personas, pero en otras ocasiones sólo emite Alice y Dave.

Depuración: el equipo de pruebas ha utilizado una herramienta de localización y de reducción de datos para depurar el programa. Han obtenido lo siguiente:

- El defecto ocurre cuando se ejecutan >1 Combiners
- El defecto se manifiesta en la siguiente configuración con sólo 3 datos: ver imagen reduccion.jpg

Se tiene que analizar el programa para entender el defecto y posteriormente realizar un informe llamado PersonasQueCompranEnMuchasTiendas.pdf que contenga lo siguiente:

1. Nombre y apellidos del estudiante
2. Tiempo empleado en entender el defecto del programa
3. Descripción del defecto:

a) Circunstancias bajo las que falla el programa: se tiene que indicar en qué ejecuciones podría fallar el programa.

b) Motivos por los que falla el programa: se tiene que describir qué es lo que tiene erróneo el programa y que lo hace fallar.

c) Directrices para corregir el defecto: se tiene que indicar a grandes rasgos lo que tendría que cambiar el equipo de desarrollo para eliminar el defecto del programa. No hace falta desarrollar el programa correcto, pero sí hay que indicar qué se tendría que cambiar.

4. ¿Te fue útil la información de depuración (la imagen reduccion.jpg y que el defecto se encontraba en >1 Combiners) para entender el defecto? Sí/No e indicar el por qué.

Sobre la entrega:

- La tarea se entregará en algún formato comprimido (gzip, zip, etc.)
- Esta actividad puede realizarse en grupo de dos personas (preferible) o individual.
- **Cada carpeta en el fichero comprimido debe contener documentos en PDF (con la explicación de la solución y los *print screen* de las ejecuciones), códigos en los lenguajes de programación solicitados, instrucciones para la compilación y ejecución de dichos códigos estándares y los ficheros de entrada y resultados obtenidos, de acuerdo al caso de cada ejercicio.**
  El fichero comprimido debe tener el siguiente formato:
  AG_1-03MBID-Apellido1-Nombre1-Apellido2-Nombre2.gz
- Esta actividad tiene un peso de 20%