Práctica evaluable de MapReduce

Grupo

Alumno: Adolfo González Blázquez
Email: adolfo.gonzalez@elcorteingles.es
Teléfono: +34 609 964 414

Alumno: José María Álvarez Fernández
Email: josemaria.alvarezfernandez@elcorteingles.es
Teléfono: +34 696 667 194

Alumno: César Colado Rodríguez
email: cesar.colado@elcorteingles.es
Teléfono: +34 661 415 555

Enunciado

Partiendo del notebook *pig-indice-invertido-estudiantes.ipynb* que se proporciona, implementa un índice invertido sobre el dataset de los fórum posts utilizado en sesiones anteriores. Este índice debe contener para cada palabra, un listado de los identificadores de los posts en los que aparece, así como un contador que indique en cuántos posts aparece.

En detalle, los pasos a seguir serían los siguientes:

1. Carga el fichero de los posts forum_node.tsv, recuerda que está separado por tabuladores.
2. Limpia el fichero: elimina del body caracteres que no sean letras o números, pásalo a minúsculas, confirma que el identificador es numérico, entre otras opciones. En el identificador de post, elimina caracteres que no sean numéricos.
3. Separa el body en palabras y júntalas con el identificador.
4. Elimina duplicados (palabras que aparecen más de una vez en un post).
5. Agrupa las palabras iguales.
6. Prepara los resultados. Para cada palabra, hay que mostrar en orden ascendente el identificador de los posts donde aparece y la cuenta de los posts.
7. Almacena los resultados en HDFS.

Una parte de la salida de este programa es el siguiente:

```
(7001187)},13)
(zyrcster's,{(9247)},1)
(zyrcter,{(11610)},1)
(zytrax,{(6028725),(7002663)},2)
(zyx,{(8004310)},1)
(zz,{(1007745),(10011348)},2)
(zzz,{(8385)},1)
(zzzz,{(14790),(30278)},2)
(zzzzz,{(1007093),(5006080)},2)
(zzzzzzzzzzzzzzz,{(8353)},1)
```

Propuesta

Para la ejecución de esta práctica, se ha seguido la siguiente metodología:

- Crear un cluster Hadoop usando Docker, tal y como se describió en clase

- Usar un Notebook de Jupyter para elaborar el ejercicio, que se ha ejecutado dentro del cluster Hadoop descrito anteriormente. Los pasos dados han sido:
    - Establecer el entorno de trabajo, creando directorios y copiando los datasets pertinentes al entorno Hadoop.
    - Crear el fichero con el código Pig
    - Ejecutar el código Pig en el entorno local y dentro del cluster Hadoop.

1. Crear cluster Hadoop

Para crear y ejecutar el cluster Hadoop, se han usado las imágenes Docker presentadas en clase. En concreto, se han usado las siguientes:

- `accaminero/namenode01`
- `swapnillinux/cloudera-hadoop-yarnmaster`
- `swapnillinux/cloudera-hadoop-datanod`

Para facilitar la ejecución, se han creado una serie de scripts (localizados dentro de la carpeta "docker") que se pueden usar para construir el cluster, arrancarlo y lanzar Jupyter.

- `docker/build.sh` – Construye los contenedores necesarios para crear el cluster Hadoop. El script compartirá la carpeta de la práctica con el Docker en el directorio /media/notebooks
- `docker/start.sh` – Si el cluster estuviese parado (una vez creado), este script lo inicia.
- `docker/stop.sh` – Para la ejecución del cluster Hadoop.
- `docker/jupyter.sh` – Inicia el entorno de ejecución de Jupyter dentro del cluster Hadoop. Para acceder a Jupyter – http://localhost:8889

Para ejecutar el Notebook, dentro de la carpeta de la práctica:

- `docker/build.sh`
- `docker/jupyter.sh`
- http://localhost:8889/pig-indiceinvertido-estudiantes.ipynb

2. Código Pig

A continuación, se describe el código Pig creado para ejecutar el algoritmo de índice invertido propuesto en el ejercicio. Dicho código se puede encontrar en el fichero **students-inverted-index.pig**

1. Carga el fichero de los posts forum_node.tsv, recuerda que está separado por tabuladores. Para cargar el fichero, usamos una extensión de *piggybank* que lee desde un fichero CSV. En concreto, los comandos relevantes son:
    - Registrar PiggyBank

      ```
      REGISTER /usr/lib/pig/piggybank.jar;
      ```

    - Cargar el fichero usando CSVExcelStorage, separando los campos por tabulador:

      ```
      load 'forum_node.tsv' using
      org.apache.pig.piggybank.storage.CSVExcelStorage('\t',
      'YES_MULTILINE', 'NOCHANGE', 'SKIP_INPUT_HEADER')
      ```

2. Limpia el fichero: elimina del body caracteres que no sean letras o números, pásalo a minúsculas, confirma que el identificador es numérico, entre otras opciones. En el identificador de post, elimina caracteres que no sean numéricos.
    - Para cada línea de datos cargada en el paso anterior, ejecutamos las siguientes sustituciones:

      ```
      cleandata = foreach data generate
      REPLACE(pid, '[a-zA-Z]+', '') as post_id,
      LOWER(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(body,
      '\\\\n\\\\r', ''), '\\\\r', ''), '\\\\n', ''), '<*>', ''), '[^a-
      zA-Z0-9\'\\s]+', ' ')) AS clean_body;
      ```

    - Confirmamos que los IDs son números usando las funciones de PiggyBank:

      ```
      cleandata_filtered = filter cleandata by
      org.apache.pig.piggybank.evaluation.IsNumeric(post_id);
      ```

3. Separa el body en palabras y júntalas con el identificador.
    - Usamos una función custom de PiggyBank para asegurarnos que el identificador es un Integer, y sacamos del body las palabras:

      ```
      words_data = FOREACH cleandata_filtered GENERATE
      StringToInt(post_id) as post_id_int:int,
      FLATTEN(TOKENIZE(clean_body)) as word;
      ```

    - Además, eliminamos las entradas vacías:

      ```
      words_data_filtered = filter words_data by SIZE(word) > 0;
      ```

4. Agrupamos por palabra y elimina duplicados (palabras que aparecen más de una vez en un post).
    - Agrupamos:

      ```
      word_groups = GROUP words_data_filtered BY word;
      ```

- Eliminamos duplicados y recontamos:

```
index = FOREACH word_groups {
    pairs = DISTINCT $1.$0;
    cnt = COUNT(pairs);
    GENERATE $0 as word, pairs as index_bag, cnt as count;
};
```

5. Prepara los resultados. Para cada palabra, hay que mostrar en orden ascendente el identificador de los posts donde aparece y la cuenta de los posts.
    - Ordenamos los posts por su identificador:

```
sorted_index = foreach index {
    sorted_bag = order index_bag by $0;
    generate word, sorted_bag, count;
}
```

6. Almacena los resultados en HDFS.
    - Por último, almacenamos los resultados en HDFS:

```
STORE sorted_index INTO 'inverted_index';
```

3. Ejecución del código Pig y resultados

Una vez creado el código descrito anteriormente, procedemos a ejecutarlo.

1. **Ejecución en local** dentro del contenedor Hadoop:

```
$ pig -f students-inverted-index.pig -x local
```

2. **Comprobamos los resultados** obtenidos. Para ello cogemos las líneas de salida desde el número 60000 al 60010, por ejemplo:

```
$ tail -60000 ./inverted_index/part-r-00000  | head -10

googlecode      {(79),(10617),(28844),(45630),(1006011),(1034790),(300
1942),(5001673),(5002485),(6001946),(6002641),(6002651),(6002660),(600
2690),(6003146),(6003978),(6004231),(6004298),(6005746),(6006014),(600
7830),(6010307),(6015242),(6016801),(6022761),(6025168),(6027118),(700
6210)}   28
googlemail      {(66818),(66848),(67057),(6005844),(6006822),(6008744)
} 6
googlemale      {(10011506)}    1
googleplex      {(2009842)}     1
googleplus      {(6002796)}     1
googolplex      {(2009965)}     1
goosebumps      {(5002139)}     1
gorchynski      {(12002258),(12002259),(12002270),(12002296),(12002300
),(12002345),(12002440)} 7
gordeychuk      {(12003403)}    1
gorilla834      {(6019443),(6019538)}  2
```

3. Ahora **ejecutamos dentro del cluster Hadoop:**

```
$ pig -f students-inverted-index.pig
```

4. Por último comprobamos que los resultados son los mismos que en la ejecución local:

```
$ hadoop fs -cat inverted_index/part-r-00000 | tail -60000 | head -10
```

```
googlecode      {(79),(10617),(28844),(45630),(1006011),(1034790),(300
1942),(5001673),(5002485),(6001946),(6002641),(6002651),(6002660),(600
2690),(6003146),(6003978),(6004231),(6004298),(6005746),(6006014),(600
7830),(6010307),(6015242),(6016801),(6022761),(6025168),(6027118),(700
6210)}   28
googlemail      {(66818),(66848),(67057),(6005844),(6006822),(6008744)
} 6
googlemale      {(10011506)}    1
googleplex      {(2009842)}     1
googleplus      {(6002796)}     1
googolplex      {(2009965)}     1
goosebumps      {(5002139)}     1
gorchynski      {(12002258),(12002259),(12002270),(12002296),(12002300
),(12002345),(12002440)} 7
gordeychuk      {(12003403)}    1
gorilla834      {(6019443),(6019538)}  2
```

4. Conclusión

Tal y como se puede observar en los códigos de ejemplo de ejecución del programa Pig, y al ejecutar el Notebook adjunto, se completa la transformación de la información de los foros en un índice invertido ordenador por palabra, incluyendo sus ocurrencias en cada post y un conteo de cuantas veces aparece en total en los foros.

5. Código Pig completo

A continuación, adjuntamos el código Pig completo, que también se puede consultar en el fichero **students-inverted-index.pig**

```
/* 1.Carga el fichero de los posts forum_node.tsv, utilizando una extension
de Piggybank para poder quitar la cabecera,
en vez de usar directamente el PigStorage. */
REGISTER /usr/lib/pig/piggybank.jar;
DEFINE StringToInt InvokeForInt('java.lang.Integer.valueOf', 'String');

data =
    load 'forum_node.tsv'
    using org.apache.pig.piggybank.storage.CSVExcelStorage('\t',
'YES_MULTILINE', 'NOCHANGE', 'SKIP_INPUT_HEADER')
    as (pid:chararray, title:chararray, tagnames:chararray,
        author_id:chararray,body:chararray,
        node_type:chararray, parent_id:chararray,
        abs_parent_id:chararray,added_at:chararray,
        score:chararray, state_string:chararray, last_edited_id:chararray,
        last_activity_by_id:chararray, last_activity_at:chararray,
        active_revision_id:chararray, extra:chararray,
        extra_ref_id:chararray, extra_count:chararray, marked:chararray);

/* 2.Limpiamos el fichero quitando los saltos de linea, expresiones html y
la expresión regular que se proponia en el ejercicio. */
cleandata = foreach data generate
    REPLACE(pid, '[a-zA-Z]+', '') as post_id,
    LOWER(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(body, '\\\\n\\\\r', ''),
'\\\\r', ''), '\\\\n', ''), '<*>', ''), '[^a-zA-Z0-9\'\\s]+', ' ')) AS
clean_body;

/* 3.Filtramos los datos de post_id que no son numericos. */
cleandata_filtered = filter cleandata by
org.apache.pig.piggybank.evaluation.IsNumeric(post_id);

/* 4.Creamos tuplas separando el body por espacios y convirtiendo el post_id
en un numerico a través de una función custom, para evitar problemas que
sufrimos con el cast de String a Integer. */
words_data = FOREACH cleandata_filtered GENERATE StringToInt(post_id) as
post_id_int:int, FLATTEN(TOKENIZE(clean_body)) as word;
words_data_filtered = filter words_data by SIZE(word) > 0;

/* 5.Agrupamos por palabra */
word_groups = GROUP words_data_filtered BY word;

/* 6.Por cada grupo de palabras, hacemos un distinct para los post_id,
eliminando los duplicados, contamos el número de post en que aparece
(despues de quitar los duplicados) y generamos una fila con el índice. */
index = FOREACH word_groups {
    pairs = DISTINCT $1.$0;
    cnt = COUNT(pairs);
    GENERATE $0 as word, pairs as index_bag, cnt as count;
};

/* 7.Como se pide que el indice lleve el post_id ordenador, ordenamos la bag
resultante de los posts por su id. */
sorted_index = foreach index {
    sorted_bag = order index_bag by $0;
    generate word, sorted_bag, count;
}

/* 8. Lo guardamos en un fichero. */
STORE sorted_index INTO 'inverted_index';
```