

MapReduce con Apache Hadoop

zerjioang

Definición del problema

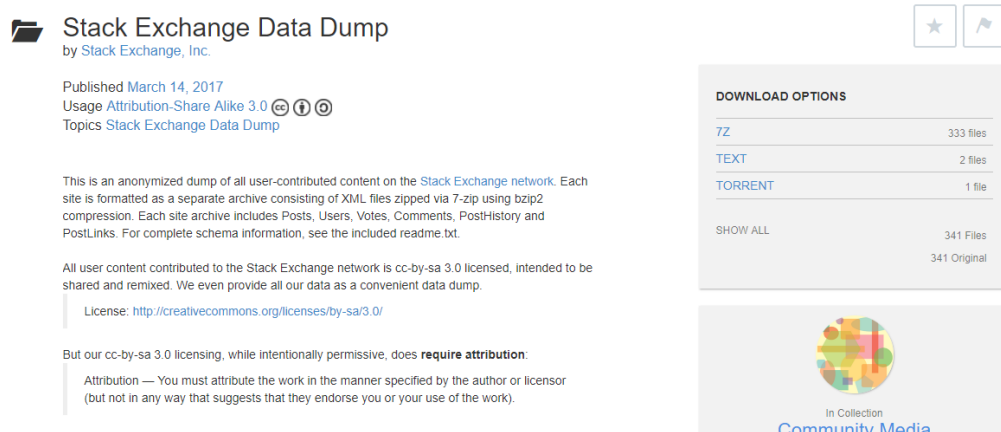
El problema a tratar como objeto de esta práctica es tener una visión global de los usuarios de StackOverflow y sobre todo de conocer que usuarios son los que tienen un mayor conocimiento sobre el tema elegido. Para este ejercicio se ha elegido el tema UNIX, por lo tanto, descubriremos que usuarios de stackoverflow (unix), tiene un mayor conocimiento del tema.

Para ello haciendo uso de herramientas de los propios DUMPS en formato XML que StackOverflow pone a disposición del público en general, podremos obtener una buena fuente de datos para el propósito de este ejercicio. Una vez descargados los datos comprobaremos mediante el algoritmo mapreduce, que usuarios han hecho una mayor cantidad de post y que puntuación han recibido. Por lo tanto, entendemos que para este ejercicio, un post de calidad es aquel que tiene 10 puntos o más. Una vez conocidos que posts tienen 10 puntos o más, se obtendrá el listado de usuarios más activos, dando el resultado en formato lista. Para que el resultado sea más manejable, se mostrara por pantalla el TOP10 de dichos usuarios.

Conjunto de datos

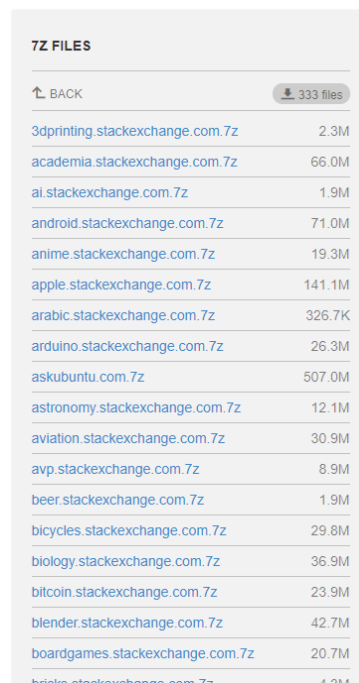
EL conjunto de datos necesario para este ejercicio es el proporcionado por la web stackoverflow.com.

Accediendo a la URL <https://archive.org/details/stackexchange> dispondremos de los siguientes recursos:



The screenshot shows the 'Stack Exchange Data Dump' page on Archive.org. It includes a folder icon, the title 'Stack Exchange Data Dump' by Stack Exchange, Inc., and a star icon. The page is published on March 14, 2017, and is licensed under Attribution-Share Alike 3.0. It lists topics related to the data dump. A description states it is an anonymized dump of all user-contributed content on the Stack Exchange network, formatted as XML files zipped via 7-zip using bzip2 compression. It mentions that each site archive includes Posts, Users, Votes, Comments, PostHistory, and PostLinks. A license link is provided: <http://creativecommons.org/licenses/by-sa/3.0/>. A note about CC-BY-SA 3.0 licensing is also present. On the right, there are 'DOWNLOAD OPTIONS' for 7Z (333 files), TEXT (2 files), and TORRENT (1 file), with a 'SHOW ALL' link. At the bottom right, it says 'In Collection Community Media' with a colorful logo.

Dependiendo de la categoría elegida a procesar, se elegirá un link u otro para realizar la descarga de datos, de forma que mediante el siguiente desplegable, se pueda descargar todo el conjunto de datos que nos interesa. Tal y como se muestra en la imagen, tenemos multitud de categorías disponibles.



The screenshot shows a list of 7Z files available for download. The list is titled '7Z FILES' and includes a 'BACK' link and a download icon with '333 files'. The files are listed with their names and sizes.

File Name	Size
3dprinting.stackexchange.com.7z	2.3M
academia.stackexchange.com.7z	66.0M
ai.stackexchange.com.7z	1.9M
android.stackexchange.com.7z	71.0M
anime.stackexchange.com.7z	19.3M
apple.stackexchange.com.7z	141.1M
arabic.stackexchange.com.7z	326.7K
arduino.stackexchange.com.7z	26.3M
askubuntu.com.7z	507.0M
astronomy.stackexchange.com.7z	12.1M
aviation.stackexchange.com.7z	30.9M
avp.stackexchange.com.7z	8.9M
beer.stackexchange.com.7z	1.9M
bicycles.stackexchange.com.7z	29.8M
biology.stackexchange.com.7z	36.9M
bitcoin.stackexchange.com.7z	23.9M
blender.stackexchange.com.7z	42.7M
boardgames.stackexchange.com.7z	20.7M
bricks.stackexchange.com.7z	4.3M

En este ejercicio, hemos seleccionado el fichero `unix.stackexchange.com.7z` el cual se puede obtener mediante el enlace:

<https://archive.org/download/stackexchange/unix.stackexchange.com.7z>

Una vez descargado el fichero, se tiene que descomprimir con el siguiente comando, guardando en el directorio out los ficheros XML resultantes.

```
7za e $(ls *.7z) -o./out
```

La descarga y descompresión se puede realizar automáticamente usando el fichero `downloader.py` y ejecutándolo desde Cloudera con el siguiente comando:

```
python2 ./downloader.py
```

Dicho comando mostrara el progreso de la descarga por pantalla y generara un fichero 7z con los datos descargados en el directorio actual. En este ejemplo se ha usado el fichero **chemistry** para ilustrar dicho proceso.

```
[training@localhost StackOverflow MapReduce]$ python2 ./downloader.py
Starting MapReduce on selected dataset

Downloading data...

Downloading: chemistry.stackexchange.com.7z Bytes: 44572347
^    17620992  [39.53%]
```

Programación

Basándose en los ejemplos provistos por Cloudera, la programación de esta práctica se compone de 2 partes:

Fichero **mapper.py**

El objetivo de este fichero es procesar el conjunto de datos Posts.xml y enviar al reducer la información en forma de tupla de lo que interesa. Para ello solamente se enviarán mediante la salida estándar del sistema aquella información de los posts que tenga un score superior a 10 puntos.

```
import sys
from xml.etree import ElementTree as xml

main_root = 'row'
tag = 'OwnerUserId'
score_tag = 'Score'
user_id = None
min_score = 10

#parse stdin xml data
root = xml.parse(sys.stdin).getroot()

#get all rows
rows = root.findall(main_root)

#iterate
print " [Status] MAP"

for row in rows:
    uid = row.get(tag)
    score = row.get(score_tag)
    if uid != None and score != None and int(score) > min_score:
        #print item as key value pair separated by comma
        print "{0},1".format(uid)
```

Fichero **reducer.py**

Este fichero se encarga de procesar los datos recibidos por el mapper, de forma que por cada tupla que recibe del mapper, este la procesa y la envía por pantalla para mostrarlo como resultado final.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import sys

print " [Status] REDUCE"

lastReadedUserId = None
lastReadedCount = 0
mapper_separator = ","
fixed_data_length = 2

for line in sys.stdin:
```

```

# read standard input line
data_mapped = line.strip().split mapper_separator
#check data length, always must be len = 2
if len(data_mapped) != fixed_data_length:
    # skip received data
    continue
# save received data in each variable
user_id, post_count = data_mapped
if lastReadedUserId != None and lastReadedUserId != user_id:
    print lastReadedUserId, "\t", lastReadedCount
    lastReadedCount = 0
lastReadedUserId = user_id
lastReadedCount += int(post_count)

if lastReadedUserId:
    print lastReadedUserId, "\t", lastReadedCount

```

Estudio de ejecución

Para la ejecución de esta prueba se dispone del fichero `runme.sh`, el cual sirve para iniciar el ejercicio. Dicho fichero contiene lo siguiente:

```

#!/bin/bash

echo "Removing previous files..."
rm *.7z
rm -rf ./out
echo "Installing required dependencies.."
sudo yum install p7zip
echo "Downloading exercise dataset..."
python2 ./downloader.py
echo "Extracting data..."
7za e $(ls *.7z) -o./out
echo "MOST ACTIVE USERS"
echo "User ID   Post number"
cat ./out/Posts.xml | python2 ./mapper.py | sort | python2
./reducer.py | sort --key 2 --numeric-sort --reverse | head -n 10
echo "Top 10 most active users on"
echo "StackOverflow selected category"

```

El resultado de ejecutar dicho script es el siguiente:

```

[training@localhost StackOverflow MapReduce]$ ./runme.sh
Removing previous files...
Installing required dependencies..
Downloading exercise dataset...
Extracting data...
MOST ACTIVE USERS
User ID   Post number
885       836
22565     392
7453      313
None      195
22222     171
38906     128
73        121

```

```
10762      113
86440      89
25985      89
Top 10 most active users on
StackOverflow selected category
[training@localhost StackOverflow MapReduce]$
```

Como se puede apreciar, el usuario con ID 885 es el que más posts de calidad ha puesto sobre temas Unix dado que han sido valorados con al menos una puntuación superior a 10.

Este ID corresponde a <https://unix.stackexchange.com/users/885/gilles> y podemos ver su perfil tal y como se publica en unix.stackexchange.com

StackExchange

sign up log in tour help

user:885

UNIX & LINUX

[/ questions](#) [/ tags](#) [/ users](#) [/ badges](#) [/ unanswered](#) [/ ask question](#)

Profile Activity

Meta User Network Profile



416,939 REPUTATION

85

773

1260

Gilles top 0.01% overall

Moderator on French Language, Computer Science and Emacs. I'm also a unix amateur, and a developer on embedded systems with a computer science background and security leanings by trade.

Avatar picture slightly adapted from a photo by Luridiformis (Zonda Grattus).

7,899 answers

67 questions

~49.6m people reached

stackexchange.com/users/...

Member for 6 years, 9 months

45,281 profile views

Last seen 13 hours ago

Communities (162)

Unix & Linux

416.9k

Meta Stack Exchange

67.2k

Stack Overflow

64k

Top Tags (1,265)

/ shell

SCORE 7,181

POSTS 941

POSTS % 12

/ bash

SCORE 7,022

POSTS 1,173

/ linux

SCORE 5,276

POSTS 992

Ejecucion en Hadoop

Una vez comprobado que se ejecuta correctamente, se procede a ejecutarlo en Hadoop.

Para ello lo primero que hay que hacer es copiar todos los ficheros necesarios, al HDFS usando el comando siguiente desde el directorio donde tenemos los ficheros mapper y reducer.

```
hadoop fs -put ./
```

Para comprobar que se ha copiado correctamente se puede ejecutar

```
[training@localhost practica]$ hadoop dfs -ls
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Found 1 item
drwxr-xr-x   - training supergroup          0 2017-05-28 11:43 practica
[training@localhost practica]$
```

Para ejecutar MapReduce en hadoop hay que ejecutar el comando

```
hs ./practica/mapper.py ./practica/reducer.py data.xml result
```

Comando, que genera la siguiente salida:

```
[training@localhost xml]$ hs ./practica/mapper.py ./practica/reducer.py data.xml
result
packageJobJar: [./practica/mapper.py, ./practica/reducer.py, /tmp/hadoop-
training/hadoop-unjar4250026219211204914/] [] /tmp/streamjob2221261347470690879.jar
tmpDir=null
17/05/28 11:54:16 WARN mapred.JobClient: Use GenericOptionsParser for parsing the
arguments. Applications should implement Tool for the same.
17/05/28 11:54:16 WARN snappy.LoadSnappy: Snappy native library is available
17/05/28 11:54:16 INFO snappy.LoadSnappy: Snappy native library loaded
17/05/28 11:54:16 INFO mapred.FileInputFormat: Total input paths to process : 1
17/05/28 11:54:16 INFO streaming.StreamJob: getLocalDirs(): [/var/lib/hadoop-
hdfs/cache/training/mapred/local]
17/05/28 11:54:16 INFO streaming.StreamJob: Running job: job 201705261121 0019
17/05/28 11:54:16 INFO streaming.StreamJob: To kill this job, run:
17/05/28 11:54:16 INFO streaming.StreamJob: UNDEF/bin/hadoop job -
Dmapred.job.tracker=0.0.0.0:8021 -kill job 201705261121 0019
17/05/28 11:54:16 INFO streaming.StreamJob: Tracking URL:
http://0.0.0.0:50030/jobdetails.jsp?jobid=job 201705261121 0019
17/05/28 11:54:17 INFO streaming.StreamJob: map 0% reduce 0%
17/05/28 11:54:20 INFO streaming.StreamJob: map 100% reduce 0%
17/05/28 11:54:23 INFO streaming.StreamJob: map 100% reduce 100%
17/05/28 11:54:24 INFO streaming.StreamJob: Job complete: job 201705261121 0019
17/05/28 11:54:24 INFO streaming.StreamJob: Output: result
```

Dentro del directorio creado podemos ver los ficheros que Hadoop ha generado:

```
[training@localhost xml]$ hadoop dfs -ls ./result
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

Found 3 items
-rw-r--r--   1 training supergroup          0 2017-05-28 11:54
result/_SUCCESS
drwxr-xr-x   - training supergroup          0 2017-05-28 11:54
result/_logs
```



```
-rw-r--r-- 1 training supergroup 257 2017-05-28 11:54
result/part-00000
```

Y desde la interfaz web podemos ver que datos nos genera:

Job Counters	HDFS: Number of write operations	0	0	2
	Launched map tasks	0	0	1
	Launched reduce tasks	0	0	1
	Data-local map tasks	0	0	1
	Total time spent by all maps in occupied slots (ms)	0	0	4,163
	Total time spent by all reduces in occupied slots (ms)	0	0	2,751
	Total time spent by all maps waiting after reserving slots (ms)	0	0	0
	Total time spent by all reduces waiting after reserving slots (ms)	0	0	0
Map-Reduce Framework	Map input records	0	0	2,434
	Map output records	0	0	74
	Map output bytes	0	0	483
	Input split bytes	0	0	95
	Combine input records	0	0	0
	Combine output records	0	0	0
	Reduce input groups	0	0	37
	Reduce shuffle bytes	0	0	637
	Reduce input records	0	0	74
	Reduce output records	0	0	37
	Spilled Records	0	0	148
	CPU time spent (ms)	0	0	730
	Physical memory (bytes) snapshot	0	0	200,781,824
	Virtual memory (bytes) snapshot	0	0	776,908,800
	Total committed heap usage (bytes)	0	0	176,492,544
org.apache.hadoop.mapreduce.lib.input.FileInputFormatCounter		BYTES_READ	3,171,110	0 3,171,110

Map Completion Graph - [close](#)



Reduce Completion Graph - [close](#)

