OUMOUSS EL MEHDI (D&K)

Internet of Things – MapReduce Session Lab

1. Install Hadoop

http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SingleCluster.html
Hadoop installation was done successfully.

2. MapReduce Tutorial

http://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
The WordCount example has been compiled and executed very well.

3. Hadoop MapReduce application for "Customers who bought this item also bought"

Input: (Books_DataSet)
We have a file that contains per line a transaction

book,book1,book2 book1,book2,book4 book,book2,book3

3.1 First solution:

The idea is quite simple, we just have to output the different possible combinations among the items of a transaction.

Mapper output:

book, book1 book,book2 book1,book book1,book2 book2,book book2,book1	book1,book2 book1,book4 book2,book1 book2,book4 book4,book1 book4,book2	book, book2 book,book3 book2,book book2,book3 book3,book book3,book2

Shuffle: Reducer Input
<book, [book1, book2, book3]>
<book1, [book, book2, book4]>
<book2, [book, book1, book3, book4]>
<book3, [book, book2]>
<book4, [book1, book2]>

Reducer Output: identity
The reduce basically has just to output the input as it is.

3.2 Second solution:

Another solution that came to my mind is as follow :
Instead of outputting many key-value pairs, why not make the mapper output only an item as key and the rest of the items that came with as the value.

Mapper
item : Text boughtWith : Text MAP : void FOR EACH bi IN transaction : b1, b2, …, bn boughtWith = Transaction/bi OUTPUT (bi, boughtWith)

Reducer
REDUCE : void hSet : HashSet<Text> For Each val In values For Each item In val Add item In hSet OUTPUT(key, hSet)

Example:

Mapper Output: <Key, Value> <B, [B1,B2]> <B1, [B,B2]> <B2, [B,B1]> …
Shuffle: <B, [[B1, B2], [B2, B3]]> …
Reducer Output: <B, [B1, B2, B3]> …

3.3 Comparaison:

Sol. 1 <key, value>	**Sol. 2 <key, [values]>**
16/10/04 02:51:22 INFO mapreduce.Job: Counters: 50 File System Counters FILE: Number of bytes read=250 FILE: Number of bytes written=346255 Job Counters Total time spent by all maps in occupied slots (ms)=13091 Total time spent by all reduces in occupied slots (ms)=3847 Total time spent by all map tasks (ms)=13091 Total time spent by all reduce tasks (ms)=3847 Total vcore-seconds taken by all map tasks=13091 Total vcore-seconds taken by all reduce tasks=3847 Total megabyte-seconds taken by all map tasks=13405184 Total megabyte-seconds taken by all reduce tasks=3939328 Map-Reduce Framework Map input records=3 Map output records=18 Map output bytes=208 Map output materialized bytes=256 Reduce shuffle bytes=256 Reduce input records=18 Reduce output records=5 Spilled Records=36 Shuffled Maps =2 Failed Shuffles=0 Merged Map outputs=2 GC time elapsed (ms)=455 CPU time spent (ms)=2410 Physical memory (bytes) snapshot=693559296 Virtual memory (bytes) snapshot=5737598976 Total committed heap usage (bytes)=516947968	16/10/04 03:11:33 INFO mapreduce.Job: Counters: 50 File System Counters FILE: Number of bytes read=180 FILE: Number of bytes written=346115 Job Counters Total time spent by all maps in occupied slots (ms)=13010 Total time spent by all reduces in occupied slots (ms)=3665 Total time spent by all map tasks (ms)=13010 Total time spent by all reduce tasks (ms)=3665 Total vcore-seconds taken by all map tasks=13010 Total vcore-seconds taken by all reduce tasks=3665 Total megabyte-seconds taken by all map tasks=13322240 Total megabyte-seconds taken by all reduce tasks=3752960 Map-Reduce Framework Map input records=3 Map output records=9 Map output bytes=156 Map output materialized bytes=186 Reduce shuffle bytes=186 Reduce input records=9 Reduce output records=5 Spilled Records=18 Shuffled Maps =2 Failed Shuffles=0 Merged Map outputs=2 GC time elapsed (ms)=337 CPU time spent (ms)=2120 Physical memory (bytes) snapshot=703123456 Virtual memory (bytes) snapshot=5727625216 Total committed heap usage (bytes)=532152320

NB: The input used in this experiment is very small (3 lines), it would be very interesting to see how the results would be if we used a bigger dataset.

OUMOUSS EL MEHDI (D&K)

3.4 Execution

Here Attached to this report, is the implementation of the two solution. Next is the steps to follow in order to execute each one of the MapReduce programs. (Same content as the read.me file)

1. Start your hadoop shell
$ start-dfs.sh
$ start-yarn.sh

2. Upload the data into the hdfs
$ cd /usr/local/hadoop/
$ bin/hdfs dfs -mkdir /Iot_Lab
$ bin/hdfs dfs -put '/home/~/RecommenderSystemF/inputdata' /Iot_Lab

3. Execute
$ cd /usr/local/hadoop
/usr/local/hadoop$ bin/hadoop jar /home/~/RecommenderSystemF/RecommenderSystemJ.jar RecommenderSystem /Iot_Lab/Iot_Lab_InputData output

4. Download the output from hdfs (see: /home/~/RecommenderSystemF/part-r-00000)

More Details:

- Compilation cmd: (you should be in the RecommenderSystemF to execute)

$ javac -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-2.7.1.jar:
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.7.1.jar:
$HADOOP_HOME/share/hadoop/common/lib/commons-cli-1.2.jar -d
/home/~/RecommenderSystemF *.java

- hadoop version

Hadoop 2.7.1
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r
15ecc87ccf4a0228f35af08fc56de536e6ce657a
Compiled by jenkins on 2015-06-29T06:04Z
Compiled with protoc 2.5.0
From source with checksum fc0a1a23fc1868e4d5ee7fa2b28a58a
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-2.7.1.jar