Demo 6: A taste of Hadoop

8.5.2014

We will run simple MapReduce program word count (“Hadoop’s Hello world”) by using Agora’s Hadoop cluster.

Outline:

1. SSH-connection to allures server
2. SSH-connection to namenode
3. Upload data into HDFS
4. Run word count
5. View results
6. Add case insensitivity
7. Remove stop words
8. Test with other input files

**1) SSH-connection to allures server**

Start PuTTY and take SSH-connection to **allures.it.jyu.fi**

Use your university user account to login.

**2) SSH-connection to namenode**

To form SSH connection to namenode (172.16.173.230) use command

$ ssh <my user account>@172.16.173.230

**3) Upload data into HDFS**

Download text files to your home directory

$ wget http://www.gutenberg.org/cache/epub/78/pg78.txt -O tarzan.txt

$ wget http://www.gutenberg.org/files/521/521-0.txt -O crusoe.txt

Create input directory to HDFS

$ /n/hadoop/hadoop-2.2.0/bin/hadoop fs -mkdir <my input directory>

Upload text files into HDFS

$ /n/hadoop/hadoop-2.2.0/bin/hadoop fs -put tarzan.txt crusoe.txt <my input directory>

View files in HDFS in <my input directory>

$ /n/hadoop/hadoop-2.2.0/bin/hadoop fs -ls <my input directory>

**4) Run word count**

Word count MapReduce program counts a number of occurences for each word in a text file(s). Source code for word count program: <http://users.jyu.fi/~jookriha/WordCount.java>

The TokenizerMapper function firstly products pairs of words and ones. Example: (Tarzan, 1), (is, 1), (the, 1)… Secondly these are grouped by words and finally IntSumReducer function sums all ones for each word.

Run word count example jar.

$ /n/hadoop/hadoop-2.2.0/bin/hadoop jar /n/hadoop/hadoop-2.2.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar wordcount <my input directory> <my output directory>

**5) View results**

View output directory in HDFS. Directory should include \_SUCCESS and part-r-00000 files. Word counts for each word are stored in the part-r-00000 file.

$ /n/hadoop/hadoop-2.2.0/bin/hadoop fs -ls <my output directory>

Usually output files in HDFS are divided as multiple files. Files can be merged to one output file by using getmerge command.

$ /n/hadoop/hadoop-2.2.0/bin/hadoop fs -getmerge <my output directory> <my output file>

Now you should have <my output file> on your home directory. View output file.

$ less <my output file>

Find all the occurences of the word “tarzan”. After opening file with less command use forward search command

$ /<my word>

Press n for next match and N for previous match. Exit press q. Output file can be sorted with command

$ sort -rnk <column> <my output file> > <my sorted file>

**6) Add case insensitivity**

Extract source codes from jar file

$ jar xf /n/hadoop/hadoop-2.2.0/share/hadoop/mapreduce/sources/hadoop-mapreduce-examples-2.2.0-sources.jar

Copy WordCount.java to your home directory and rename it as MyWordCount.java

$ cp org/apache/hadoop/examples/WordCount.java MyWordCount.java

Open MyWordCount.java file in vi text editor using command

$ vi MyWordCount.java

To do modification in vi press Insert.

Remove the package defition (thereby using “default package”) by removing or commenting line: package org.apache.hadoop.examples;.

To compile MyWordCount.java successfully class name has to be same as the java-file name so do modification public class WordCount -> public class MyWordCount. To call rigth class do modification on job.setJarByClass(WordCount.class); -> job.setJarByClass(MyWordCount.class);

Modify TokenizerMapper function to add case insensitivity. Use toLowerCase method.

After modifications press Esc and then command :x to save and exit.

Compile MyWordCount.java and make mywordcount.jar.

$ export CLASSPATH=`find /n/hadoop/hadoop-2.2.0/share/hadoop/ -name '\*.jar' | paste -sd :`

$ javac MyWordCount.java -Xlint

$ jar cf mywordcount.jar MyWordCount\*.class;

Run wordcount.jar. Use different output directory than before.

$ /n/hadoop/hadoop-2.2.0/bin/hadoop jar mywordcount.jar MyWordCount <my input directory> <my output directory>

View results. Find again all the occurences of the word “tarzan”.

**7) Remove stop words**

Add remove stop words handling to word count. Use some short list of stop words. Use contains method to compare String and List of Stings. Do modifications again in TokenizerMapper function. You can try to do this in Eclipse firstly with some example data.

**8) Test with other input files**

Ulysses:

http://www.gutenberg.org/cache/epub/4300/pg4300.txt

The Adventures of Sherlock Holmes:

http://www.gutenberg.org/cache/epub/1661/pg1661.txt

Alice's Adventures in Wonderland:

http://www.gutenberg.org/cache/epub/11/pg11.txt

Frankenstein or the Modern Prometheus:

http://www.gutenberg.org/cache/epub/84/pg84.txt

If you are interested to get Hadoop running on your own computer then check <http://hortonworks.com/products/hortonworks-sandbox/>