DATASCI W261: Machine Learning at Scale

Name : Vincent Chu

Email : vslchu@gmail.com

Class Name : W261

Session : 2

Week Number : 1

Date of submission: 5/10/2017

This notebook provides a poor man Hadoop through command-line and python. Please insert the python code by yourself.

***Please note that some minor changes have been made to this script other than the code generation logic of the Map and Reduce Python scripts. This is due to the fact that I am running unix commands with Babun running on top of Windows 7. As an example, I needed to remove carriage return characters in the pGrepCount.sh script that was generated by this notebook.

Map

In [9]:
```
import sys
```

In [10]:
```
%%writefile mapper.py
#!/usr/bin/python
import sys
import re
count = 0
WORD_RE = re.compile(r"[\w']+")
filename = sys.argv[2]
findword = sys.argv[1]
with open (filename, "r") as myfile:
    #Please insert your code

    # Initialize cumulated count of the find word
    cum_findword_count = 0
    for line in myfile:
        # Remove whitespaces at start and end of each line
        line = line.strip()
        # Split the each line into individual words
        words = line.split()

        # Initialize count of the find word in the current line
        findword_count = 0
        for word in words:
            # Strip the incoming word from the file from all common
            # punctuations, symbols and white spaces, and then compare
            # to see if the stripped version matches the findword.
            # Also, both the incoming words and the findword are
            # transformed to all CAPS gfor comparison to avoid any
            # differences only due to case.
            if word.strip("'\",.?!@#$%^&*():;<>|{}[]\/~ \t\n\r").upper() == fi
ndword.upper():
                findword_count += 1
        cum_findword_count += findword_count

# print cumulated count of the findword
print cum_findword_count
```

```
Overwriting mapper.py
```

In [11]:
```
!chmod a+x mapper.py
```

Reduce

In [12]:
```
%%writefile reducer.py
#!/usr/bin/python
import sys
sum = 0
temp_int = 0
for line in sys.stdin:
#Please insert your code

    # convert each line, which contains just a single number,
    # into integer value in preparation of being added to the
    # cumulated sum
    temp_int = int(line.strip())
    sum += temp_int

# print sum of count of each "line" from sys.stdin
sys.stdout.write(str(sum))
```

```
Overwriting reducer.py
```

In [13]:
```
!chmod a+x reducer.py
```

Write script to file

***Please note that I had to add the "python" command explicity to run the mapper.py and reducer.py scripts since I am running Unix commands using Babun installed on Windows 7.

In [14]:
```
%%writefile pGrepCount.sh
ORIGINAL_FILE=$1
FIND_WORD=$2
BLOCK_SIZE=$3
CHUNK_FILE_PREFIX=$ORIGINAL_FILE.split
SORTED_CHUNK_FILES=$CHUNK_FILE_PREFIX*.sorted
usage()
{
    echo Parallel grep
    echo usage: pGrepCount filename word chuncksize
    echo greps file file1 in $ORIGINAL_FILE and counts the number of lines
    echo Note: file1 will be split in chunks up to $ BLOCK_SIZE chunks each
    echo $FIND_WORD each chunk will be grepCounted in parallel
}
#Splitting $ORIGINAL_FILE INTO CHUNKS
split -b $BLOCK_SIZE $ORIGINAL_FILE $CHUNK_FILE_PREFIX
#DISTRIBUTE
for file in $CHUNK_FILE_PREFIX*
do
    #grep -i $FIND_WORD $file|wc -l >$file.intermediateCount &
    python mapper.py $FIND_WORD $file >$file.intermediateCount &
done
wait
#MERGEING INTERMEDIATE COUNT CAN TAKE THE FIRST COLUMN AND TOTOL...
#numOfInstances=$(cat *.intermediateCount | cut -f 1 | paste -sd+ - |bc)
numOfInstances=$(cat *.intermediateCount | python reducer.py)
echo "found [$numOfInstances] [$FIND_WORD] in the file [$ORIGINAL_FILE]"
```

```
Overwriting pGrepCount.sh
```

Run the file

*** Please note that I had to add a command to remove carriage returns from the pGrepCount.sh file since I am running these unix commands using Babun installed on Windows 7. This carriage return issue is something that I previously encountered in W205, so applying the same resolution here.

In [15]:
```
# To remove carriage returns added from Windows system.
# Final script is pGrepCount_v2.sh.
!cat pGrepCount.sh | tr -d "\r" > pGrepCount_v2.sh

# Change access previlege
!chmod a+x pGrepCount_v2.sh
```

Usage: pGrepCount filename word chuncksize

*** Please note that .sh files need to be run with bash given that I am running unix commands using Babun installed on Windows 7.

In [16]:
```
# Run pGrepCount_v2.sh on License.txt
!bash pGrepCount_v2.sh License.txt COPYRIGHT 4k
```

```
found [59] [COPYRIGHT] in the file [License.txt]
```

The Mapper.py logic actually did better than the purely Unix-based poor man's map-reduce, thanks to the logic added to strip all the leading and trailing punctuations and symbols. There are indeed 59 matches of the word COPYRIGHT in the License.txt file. The purely Unix-based version reported only 57, most likely due to the quotes that were attached to 2 of the occurrences of the word COPYRIGHT.

In [ ]:
```

```