МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС

«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»

Лабораторна робота № 1

з курсу «Оброблення надвеликих масивів даних»

Тема: «Introduction to the Hadoop Environment»

Виконав:

студент I курсу

групи ДА-71мн

Колінько Анжела

Київ – 2017

Варіант 2

1. Hadoop

1) Установка Hadoop

core-site.xml:

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>fs.defaultFS</name>

<value>hdfs://localhost:9000</value>

</property>

</configuration>

hdfs-site.xml:

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

<property>

<name>dfs.replication</name>

<value>1</value>

</property>

</configuration>

В hadoop-env.sh

export JAVA\_HOME=/usr/lib/jvm/java-7-openjdk-amd64

2) Запуск





3) Виконаємо скрипти для підрахунку кількості слів в файлах A.txt і B.txt:



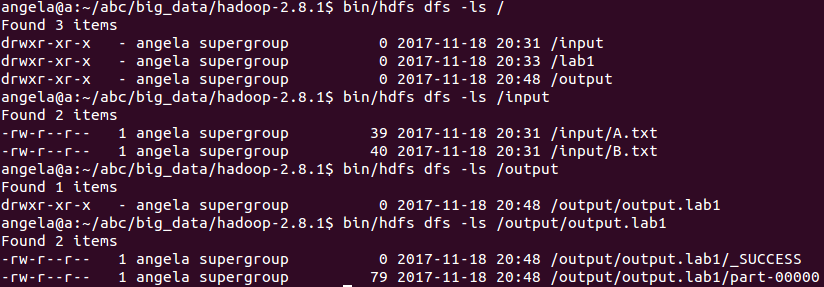
(команда bin/hadoop jar share/hadoop/tools/lib/hadoop-\*streaming\*.jar \

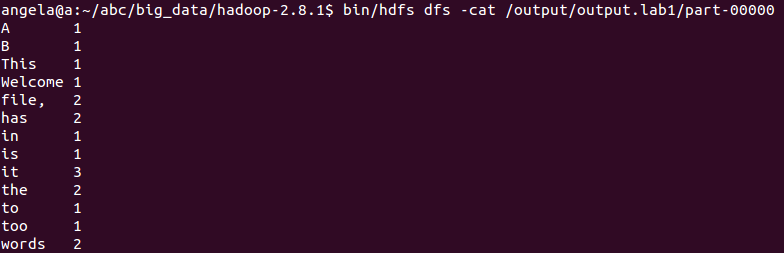
-files ../lab1/mapper.py,../lab1/reducer.py -mapper ../lab1/mapper.py \

-reducer ../lab1/reducer.py \

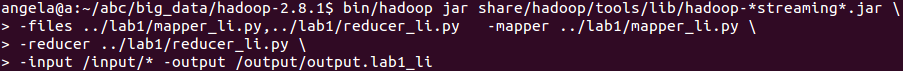
-input /input/\* -output /output/output.lab1)

4) Отримаємо:

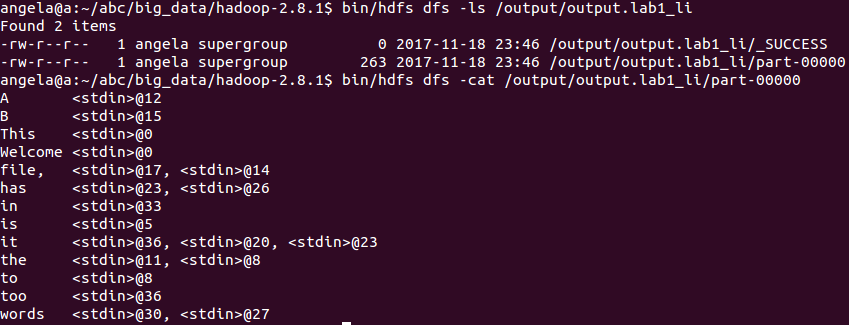




5) Виконаємо скрипти для визначення позицій слів в файлах:



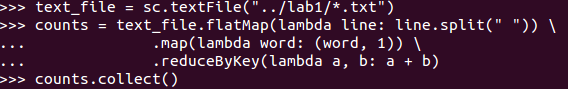
6) Отримаємо:



2. Spark

1) Wordcount:

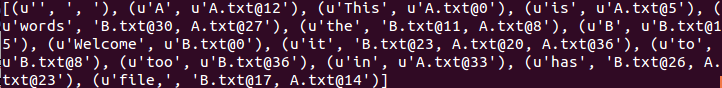
Запустимо ./bin/pyspark і виконаємо наступне:



Отримаємо:



2) Виконаємо скрипти для визначення позицій слів у файлах. Отримаємо:



Додаток 1. mapper.py

#!/usr/bin/env python

import sys

# input comes from STDIN (standard input)

for line in sys.stdin:

# remove leading and trailing whitespace

line = line.strip()

# split the line into words

words = line.split()

# increase counters

for word in words:

# write the results to STDOUT (standard output);

# what we output here will be the input for the

# Reduce step, i.e. the input for reducer.py

#

# tab-delimited; the trivial word count is 1

print '%s\t%s' % (word, 1)

Додаток 2. reducer.py

#!/usr/bin/env python

import sys

# input comes from STDIN (standard input)

for line in sys.stdin:

# remove leading and trailing whitespace

line = line.strip()

# split the line into words

words = line.split()

# increase counters

for word in words:

# write the results to STDOUT (standard output);

# what we output here will be the input for the

# Reduce step, i.e. the input for reducer.py

#

# tab-delimited; the trivial word count is 1

print '%s\t%s' % (word, 1)

Додаток 3. mapper\_li.py

#!/usr/bin/env python

"""A more advanced Mapper, using Python iterators and generators."""

import sys

offset=0;

filename='';

def read\_input(file):

for line in file:

global filename

filename=file.name

# split the line into words

yield line.split()

def main(separator='\t'):

# input comes from STDIN (standard input)

data = read\_input(sys.stdin)

for words in data:

# write the results to STDOUT (standard output);

# what we output here will be the input for the

# Reduce step, i.e. the input for reducer.py

#

# tab-delimited; the trivial word count is 1

for word in words:

global filename

global offset

outstr=filename+'@'+str(offset)

print '%s%s%s' % (word, separator, outstr)

offset+=len(word)+1;#1 for space character;

if \_\_name\_\_ == "\_\_main\_\_":

main()

Додаток 4. reducer\_li.py

#!/usr/bin/env python

"""A more advanced Reducer, using Python iterators and generators."""

from itertools import groupby

from operator import itemgetter

import sys

def read\_mapper\_output(file, separator='\t'):

for line in file:

yield line.rstrip().split(separator, 1)

def main(separator='\t'):

# input comes from STDIN (standard input)

data = read\_mapper\_output(sys.stdin, separator=separator)

# groupby groups multiple word-count pairs by word,

# and creates an iterator that returns consecutive keys and their group:

# current\_word - string containing a word (the key)

# group - iterator yielding all ["&lt;current\_word&gt;", "&lt;count&gt;"] items

for current\_word, group in groupby(data, itemgetter(0)):

try:

output\_str=", ".join(word\_map[1] for word\_map in group)

print "%s%s%s" % (current\_word, separator, output\_str)

except ValueError:

# count was not a number, so silently discard this item

pass

if \_\_name\_\_ == "\_\_main\_\_":

main()

Додаток 5.spark\_li.py (скрипти для виконання в PySpark shell)

def fmapper(pair):

f, text = pair

f = f[len(f)-5:]

offset=0

outstr=''

for lines in text.splitlines():

for word in lines.split():

outstr+=word+'\t'+f+'@'+str(offset)+'\_\_\_'

offset+=len(word)+1 #1 for space character;

return outstr.split('\_\_\_')

def mmaper(llist):

l2map=llist.split('\t')

word=l2map[0]

entry=l2map[-1]

#print 'word=%s, entry=%s' % (word,entry)

return [word,entry]

rdd = sc.wholeTextFiles("../lab1/input").flatMap(fmapper).map(mmaper).reduceByKey(lambda a, b: str(a) +', '+str(b))

rdd.collect()