## การบ้าน MapReduce

### ข้อ1. MapReduce ด้วย Hadoop

ผู้จัดทำเขียน Blog อธิบายเพิ่มเติมไว้สามารถดูได้ที่ : https://medium.com/equinox-blog/ทำ-mapreduce-word-count-บน-ubuntu-server-b2f149c43f77

การทำ MapReduce บน HDFS นั้นสามารถทำได้ตามขั้นตอนดังนี้

**1.Input Data**

ข้อมูล text file ที่จะนำเข้าใน MapReduce


**2.Imprement Mapper**

สร้าง file mapper2.py ไว้สำหรับ map ข้อมูลจาก data source ที่อ่านเข้ามา

```python
#!/usr/bin/env python

from operator import itemgetter
import sys
import collections

if __name__ == '__main__':
    for line in sys.stdin:
        words = line.split()
        words2 = collections.deque(words)
        words2.rotate(1)

        words = list(map(lambda a, b: a + " " + b, list(words2), words))
        del words[0]

        for word in words:
            sys.stdout.write('{0}\t1\n'.format(word))
```

mapper2.py hosted with ❤ by GitHub view raw

หรือใช้ #wget https://raw.githubusercontent.com/Kzis/map_reduce/master/mapper2.py

ได้เลยเนื่องจากผู้จัดทำได้นำไปวางไว้บน Host server เรียบร้อยแล้ว

**3.Imprement Reducer**

สร้าง file reducer2.py ไว้สำหรับ reduce data ที่ได้มาด้วย key หาก key ที่อ่านมามีค่าซ้ำก็ให้ทำการนับเพิ่ม จากนั้นจึง write ออกมา

```python
#!/usr/bin/env python

from operator import itemgetter
import sys
import collections

if __name__ == '__main__':

    word2count = dict()

    for line in sys.stdin:
        line = line.strip()
        word, count = line.split('\t', 1)
        count = int(count)

        if word in word2count:
            word2count[word] = word2count[word] + count
        else:
            word2count[word] = count

    for word in word2count.keys():
        sys.stdout.write('{0}\t{1}\n'.format(word, word2count[word]))
```

reducer2.py hosted with ❤ by GitHub view raw

หรือใช้ #wget https://raw.githubusercontent.com/Kzis/map_reduce/master/reducer2.py

ได้เลยเนื่องจากผู้จัดทำได้นำไปวางไว้บน Host server เรียบร้อยแล้ว

**4.Put file to Hadoop**

นำ file นี้ขึ้นไปวางไว้บน Hadoop ด้วยคำสั่ง #hadoop fs -put text.txt /user/cloudera/text.txt

จากนั้นดู file ที่ put ด้วยคำสั่ง #hadoop fs -ls /user/cloudera

**5.Run MapReduce**

ทำการ Run MapReduce Job ด้วยคำสั่งดังต่อไปนี้

# hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming.jar

–input /user/cloudera/text2.txt –output /user/cloudera/wx2

–mapper mapper2.py –reducer reducer2.py

–file mapper2.py –file reducer2.py

โดยคำสั่งมี Option ดังนี้

-input : text file ที่จำนำมาเป็น data source

-output : path ที่จะ write file output ลงไป

-mapper : file ที่จะทำหน้าที่เป็น mapper

-reducer : file ที่จะทำหน้าที่เป็น reducer

จากนั้นเมื่อทำการ Submit job ก็จะได้ดังภาพ

```
19/04/06 16:11:04 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/prox
y/application_1554565633210_0002/
19/04/06 16:11:04 INFO mapreduce.Job: Running job: job_1554565633210_0002
19/04/06 16:11:20 INFO mapreduce.Job: Job job_1554565633210_0002 running in uber mode : false
19/04/06 16:11:20 INFO mapreduce.Job:  map 0% reduce 0%
```

เมื่อ job success แล้วเราก็สามารถไปดูผลลัพธ์ได้ที่ path /user/cloudera/wx2

ด้วยคำสั่ง #hadoop fs -cat /user/cloudera/wx2/part-00000

```
[root@quickstart mnt]# hadoop fs -cat /user/cloudera/wx2/part-00000
Beer River      1
Deer Bear       1
Car River       2
Bear River      1
Car Bear        1
River Beer      1
Car Car 1
Deer Car        1
River Bear      1
[root@quickstart mnt]#
```

**ข้อ2. MapReduce ด้วย Spark บน Google Colaboratory**

ผู้จัดทำเขียน Blog อธิบายเพิ่มเติมไว้สามารถดูได้ที่ : https://medium.com/equinox-blog/ทำ-mapreduce-wordcount-ด้วย-pyspark-b9bf2550529e

การทำ MapReduce ด้วย Spark บน Google Colaboratory นั้นสามารถทำได้ตามขั้นตอนดังนี้

**1.Install Java, Spark, and Findspark**

เนื่องจาก Google Colabortory เป็น online editor ที่เป็น ubuntu server จึงต้องทำการ install environment ที่ต้องการจะใช้งานด้วยคำสั่งดังต่อไปนี้

1.1 ทำการ Install Java8

```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

1.2 ทำการ Download Apache Spark 2.4.3

```
!wget -q http://www-eu.apache.org/dist/spark/spark-2.4.3/spark-2.4.3-bin-hadoop2.7.tgz
```

1.3 ทำการ Install Apache Spark

```
!tar xf spark-2.4.3-bin-hadoop2.7.tgz
```

1.4 ทำการ Install Findspark เพื่อเขียน Python สำหรับ Spark

```
!pip install -q findspark
```

**2.Set Environment Variables**

เมื่อทำการ Install ตามขั้นตอนที่ 1 แล้วให้ทำการ Set java home path สำหรับใช้งาน Apache Spark และ Set spark home path สำหรับใช้งาน PySpark ด้วยคำสั่งดังต่อไปนี้

```
import os

os.environ[“JAVA_HOME”] = “/usr/lib/jvm/java-8-openjdk-amd64”

os.environ[“SPARK_HOME”] = “/content/spark-2.4.3-bin-hadoop2.7”
```

**3.Start a SparkSession**

เมื่อสร้าง Environments สำหรับ PySpark เรียบร้อยแล้วให้ทำการสร้าง SparkSession กำหนด Cluster Manager เป็น Local และ สร้าง SparkContext สำหรับ Call API ของ Spark ด้วยคำสั่งดังต่อไปนี้

```
import findspark

findspark.init('/content/spark-2.4.3-bin-hadoop2.7')


from pyspark.sql import SparkSession

from pyspark import SparkContext


spark = SparkSession.builder.master("local").getOrCreate()

sc = SparkContext.getOrCreate()
```

**4.Create function read data source**

ทำการสร้าง file test.csv โดยภายในมีข้อมูลดังนี้

```
1 1 1 2 2 2 3 3 3 4 5 6
a b c 1 2 7 8 9 5 6 b c
```

จากนั้นทำการสร้าง function สำหรับการอ่านข้อมูล text file ที่เป็น data source ดังนี้

```
import csv

line = []

with open('test.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter='\n')
    for row in csv_reader:
        line.append(row[0])

print(line)
```

read-file.py hosted with ❤ by GitHub view raw

**5.Create function convert data to 2gram list**

ทำการสร้าง function สำหรับ convert list data source ที่ทำการอ่านขึ้นมาเป็น list 2 gram word ดังนี้

```python
import collections


def mapper2gram(all_data):
  map_list = []

  for line in all_data:

    words = line.split()
    words2 = collections.deque(words)
    words2.rotate(1)

    words = list(map(lambda a , b: a+" "+b ,list(words2), words))
    del words[0]

    map_list.append(words)
  return map_list

def extract2gram(map_list):
  extract_list = []
  for comp in map_list:
    for inx in comp:
      extract_list.append(inx)
  return extract_list
```

mapper2gram.py hosted with ❤ by GitHub view raw

จากนั้นทำการนำ convert list data source เป็น list 2 gram ด้วยคำสั่ง ดังต่อไปนี้

map_list = mapper2gram(line)

extract_list = extract2gram(map_list)

จะได้ว่า extrac_list เก็บข้อมูลของ data source แบบ 2gram

**6.Create RDDs from list**

ทำการ convert list 2 gram เป็น RDDs โดยการ parallelize ผ่าน Spark context ด้วยคำสั่ง ดังต่อไปนี้

rdd = sc.parallelize(extract_list)

**7.Map key**

ทำการ Tranformation RDDs ตัวใหม่ด้วยคำสั่ง map โดยมี function การ map คือ ค่าที่ส่งเข้าจะถูกแปลงเป็น tuple ของค่าที่ส่งเข้าไปเป็นคีย์ และมีค่าเป็น1 ( คู่อันดับ <K,V> ) ด้วยคำสั่ง ดังต่อไปนี้

words = rdd.map(lambda word: ((word, 1)))

**8.Reduce by key**

ทำการ Actions RDDs ด้วยคำสั่ง reduceByKey โดยมี function reduce คือ การส่ง parameter เข้าไปทีละ 2 ตัว และถ้า key เหมือนกันจะนำ value มารวมกัน

```
counts = words.reduceByKey(lambda a, b: a + b)
print(counts.collect())
```

ก็จะได้ผลลัพธ์ออกมาดังนี้

[('1 1', 2), ('1 2', 2), ('2 2', 2), ('2 3', 1), ('3 3', 2), ('3 4', 1), ('4 5', 1), ('5 6', 2), ('a b', 1), ('b c', 2), ('c 1', 1), ('2 7', 1), ('7 8', 1), ('8 9', 1), ('9 5', 1), ('6 b', 1)]

**Download Source Code ทั้งหมด ได้ที่ :** https://github.com/Kzis/map_reduce