**Shape, square

Description automatically generatedTRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

Ảnh có chứa văn bản, mẫu họa

Mô tả được tạo tự động

**BÁO CÁO DỰ ÁN GIỮA KỲ**

**Môn học: BIGDATA SENSTIAL**

**Mã lớp học phần: BDES333877\_22\_1\_04**

**GVHD:ThS. Lê Thị Minh Châu**

**Sinh viên thực hiện: Đỗ Hoàng Thịnh -20133122**

TP Hồ Chí Minh, tháng 10 năm 2022

Text

Description automatically generatedExercise 2:

Hadoop MapReduce

Concepts and Technologies for Distributed Systems and Big Data Processing – SS 2017

**Task 1 Paper Reading**

**a) How do the input keys/values, the intermediate keys/values and the output keys/values relate?**

Answer:

Chúng liên kết với nhau dựa vào 2 hàm:

map (in\_key, in\_value) -> list(out\_key, intermediate\_value)

để phân tích cặp khóa/giá trị đầu vào và tạo ra tập hợp các cặp khóa/giá trị trung gian.

reduce (out\_key, list(intermediate\_value)) -> list(out\_value)

để tổng hợp các giá trị trung gian của một khóa nhất định thành tập hợp các cặp khóa/giá trị đầu ra.

Diagram

Description automatically generated

**b) How does MapReduce deal with node failures?**

Answer:

Khi một node (worker) gặp sự cố, MapReduce sẽ xử lý bằng quy trình thực hiện lại:

- Phát hiện sự cố thông qua kiểm tra heartbeats định kỳ

- Thực hiện lại các tiến trình map đang chạy, và kể cả các tiến trình map đã hoàn thành

- Thực hiện lại các tiến trình reduce đang chạy

- Commit hoàn thành tiến trình qua master

**c) What is the meaning and the implication of locality? How is it used?**

Answer:

Trong Hadoop, locality được định nghĩa là quá trình đưa công đoạn tính toán đến tận các node nơi dữ liệu liên quan được lưu trữ, hoặc ít nhất là gần đó, thay vì việc đưa các dữ liệu liên quan đến nơi chứa các mô-đun tính toán. Việc này giảm thiểu tắc nghẽn băng thông kết nối, tăng khả năng xử lý dữ liệu và thông lượng của hệ thống.

Trong Hadoop, các tập dữ liệu được lưu trữ trong hệ thống HDFS. Các tập dữ liệu này được phân nhỏ và lưu trữ rải rác khắp các datanodes trong Hadoop cluster. Khi một người dùng thực thi MapReduce, NameNode sẽ gửi mô-đun tính toán (đoạn code) đến các datanodes – nơi chứa các dữ liệu liên quan. Master của tiến trình MapReduce sẽ lấy thông tin nơi lưu trữ của dữ liệu đầu vào và thử sắp xếp tiến trình map trên máy có chứa bản sao của dữ liệu đầu vào cho tiến trình đó. Nếu không thành công, Master sẽ thử lại trên một máy khác lân cận trong hệ thống để tiến trình được gần nhất với nơi lưu trữ dữ liệu cần thiết.

Khi thực thi MapReduce trên phần lớn các workers trong Hadoop cluster, hầu hết sẽ là dữ liệu cục bộ, có thể thậm chí không tiêu hao băng thông kết nối.

**d) Which problem is addressed by introducing a *combiner function* to the MapReduce model?**

Answer:

Combiner – hàm gộp – hoặc còn được gọi là Mini-Reducer, sẽ thực hiện công việc tổng hợp đầu ra của hàm Mapper với Key tương tự, trước khi xuất kết quả cho hàm Reducer thực thi. Cho nên, một Combiner sẽ chạy sau hàm Mapper và trước hàm Reducer.

Đối với các tập dữ liệu lớn, hàm Mapper sẽ sản sinh ra một lượng rất lớn dữ liệu trung gian trước khi xuất ra cho hàm Reducer để tiếp tục tính toán. Việc này khiến kết nối rất dễ bị tắt nghẽn vì băng thông quá tải, đây chính là lí do tại sao hàm Combiner được sử dụng, tuy nhiên, việc sử dụng hay không hoàn toàn phụ thuộc vào người lập trình.

# CÁC CÔNG CỤ VÀ TÀI LIỆU CẦN CÓ

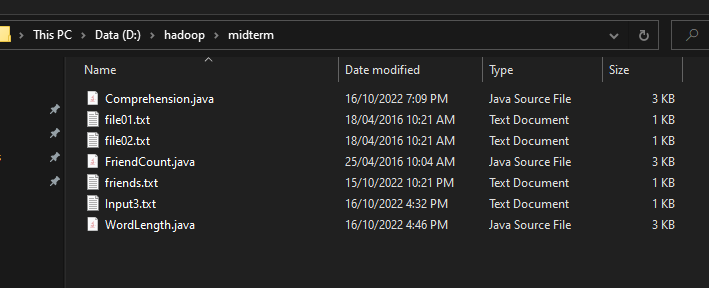
* Unbutu bản 21.0 hoặc Unbutu bản 22.0

Hadoop phiên bản 3.3.2, hoặc 3.3.4  
Login với vai trò root (pass: root) để thực hiện những công việc bên dưới.

Text

Description automatically generated

## Chuẩn bị hết các file cần thực thi ứng với các task vào một folder InputfileTaskMidterm (có thể thay tên khác)



**\*Note:**

FriendCount.java: file chứa code chương trình (Task2)

Friends: chứ dữ liệu (Task2)

WordLength.java: file chứa code chương trình (Task3)

Wordcount.java:file chứa code chương trình (Task2)

textwordlength: chứ dữ liệu text (Task2), (Task3)

Comprehension: file chứa code chương trình (Task4)

file1: chứ dữ liệu text (Task4)

File2: chứ dữ liệu text(Task4)

## Thực hiện Sharing Folder từ máy window vào máy ảo

### Mở thư mục /mnt/hgfs/ để xem folder sau khi sharing

Chạy lệnh*:sudo /usr/bin/vmhgfs-fuse .host:/ /mnt/hgfs -o subtype=vmhgfs-fuse,allow\_other*

### Copy các file có trong thư mục ra home/hadoopthinh

Login vào hadoopthinh và cd đến thư mục midterm

- Gõ ls để kiểm tra các file có trong thư mục midterm

A screenshot of a computer

Description automatically generated with low confidence

- Gõ lệnh *cp \* /home/hadoopthinh* để copy tất cả các file trong thư mục InputfileTaskMidterm ra hadoopthinh

- Gõ $cd để về home và gõ $ls để xem các file sau khi đã copy ra.

Text

Description automatically generated

Chúng ta có thể xem nội dung của các file bằng lệnh: vi “tên file”

Vd: vi FriendCount.java

Text

Description automatically generated

### Khởi động các node ở master và slave để thông dịch các file

- Dùng lệnh *$start-all.sh*

Text

Description automatically generated

# Task 2 Implementation

*First have a look at the MapReduce examples from the lecture. You can download the code from the course website. The ZIP file contains a Maven project, which can be imported into your IDE. An Eclipse project is provided, too. If importing the Eclipse project should not work, please have a look into the informatory section at the end of the exercise to see how to import a Maven project into Eclipse.*

*You can execute the examples, like WordCount, from the Run → Run Configurations menu using the provided run configurations. After the execution has successfully completed, you will find a file \_SUCCESS and a file part-r-xxxxx in the output directory. The latter contains your output.*

*Important note: Make sure you delete the output directory before running your application again. Otherwise you will get an error from Hadoop indicating that the directory already exists.*

Table

Description automatically generated*Taking into account the WordCount example from the lecture, implement FriendCount, which should count the number of friends for each given person. The list of friends is provided in the file input/friendcount/friends. Figure 1 shows the input given by the file and the expected output.*

## \*\*\* File FriendCount

## Check nội dung các file cần thực hiện trong Task 2

### File FriendCount.java: A

Text

Description automatically generated

### File friends: vi friends.txt

Graphical user interface, text

Description automatically generated

## Thông dịch và thực thi chương trình FriendCount

### Tạo thư mục unitTask2

Chứa các file sau khi thông dịch FriendCount.java: *$ mkdir unitTask2*

### Text Description automatically generatedThông dịch file nguồn và tạo file jar *(cần phải có hadoop-core-1.2.1.jar, nếu không có thì download về)*

Sử dụng lệnh: *$ javac -classpath hadoop-core-1.2.1.jar -d unitTask2 FriendCount.java*

hadoop-core-1.2.1.jar để chứa các class file cần thiết để thông dịch chương trình, -d xác định thư mục chứa các class file được tạo ra

Sử dụng lệnh: *$ jar -cvf unitTask2.jar -C unitTask2/ .*

Để tạo file unitTask2.jar chứa tất cả (dấu .) file trong thư mục unitTask2 (-C là lấy trong đường dẫn hiện tại)

Text

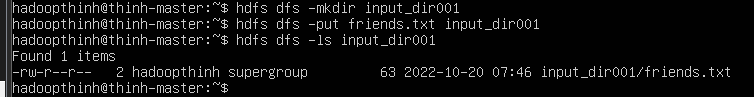
Description automatically generated

### Tạo thư mục input trong HDFS

*$ hdfs dfs -mkdir input\_dir001*

### Đưa dữ liệu friends vào thư mục input\_dir1 trong HDFS

*$ hdfs dfs -put friends.txt input\_dir001*



1. Thực thi chương trình count trong hadoop  
   *$ hadoop jar unitTask2.jar de.tud.stg.FriendCount input\_dir001 output\_dir001* Text

   Description automatically generated

Text

Description automatically generated

### Xuất kết quả

Dùng cat để xem kết quả nằm trong file part-r-0000

*$ hdfs dfs -cat output\_dir001/part-r-00000*

Graphical user interface, text

Description automatically generated

1. Copy kết quả output\_dir001 từ Hadoop ra thư mục bên ngoài

*$ hdfs dfs -get output\_dir001 /home/hadoopthinh*

*Và dùng lệnh ls để xem tất cả các file*

Text

Description automatically generated

Dùng lệnh $ls output\_dir01/ để xem file part-r-00000 đã thành công. Và dùng lệnh $cat output\_dir01/part-r-00000 xem lại kết quả

Text

Description automatically generated

## Task 3 Completion - WordLength

*Complete the following code for WordLength, which should count how many words belong to each of the following four length categories:*

*tiny: 1 letter — small: 2–4 letters*

*medium: 5–9 letters — big: more than 10 letters*

package de.tud.stg;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class WordLength {

public static class TokenizerMapper extends Mapper < Object, Text, Text, IntWritable > {

private final static IntWritable one = new IntWritable(1);

private Text category = new Text();

@Override

protected void map(Object key, Text value, Context context) throws IOException,

InterruptedException {

StringTokenizer tokenizer = new StringTokenizer(value.toString(), ",;\\. \t\n\r\f");

while (tokenizer.hasMoreTokens()) {

String word = tokenizer.nextToken();

int length = word.length();

String c = ((length == 1) ? "tiny" :

(length >= 2 && length <= 4) ? "small" :

(length >= 5 && length <= 9) ? "medium" : "big");

category.set(c);

context.write(category, one);

}

}

}

public static class IntSumReducer extends Reducer < Text, IntWritable, Text, IntWritable > {

private IntWritable result = new IntWritable();

@Override

protected void reduce(Text key, Iterable < IntWritable > values, Context context)

throws IOException,

InterruptedException {

int sum = 0;

for (IntWritable val: values) {

sum += val.get();

}

result.set(sum);

context.write(key, result);

}

}

public static void main(String[] args) throws Exception {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "WordLength");

job.setJarByClass(WordLength.class);

job.setMapperClass(TokenizerMapper.class);

job.setReducerClass(IntSumReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));

FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}

\*\*\* File WordLength

Check nội dung các file cần thực hiện trong Task 3

### File WordLength.java: vi WordLength.java

### Text Description automatically generated

### Text Description automatically generated

### File textwordlength: vi textwordlength

Text

Description automatically generated

## Thông dịch và thực thi chương trình FriendCount

### Tạo thư mục unitTask3

Chứa các file sau khi thông dịch WordLength.java: *$ mkdir unitTask3*

Text

Description automatically generated

### Thông dịch file nguồn và tạo file jar *(cần phải có hadoop-core-1.2.1.jar, nếu không có thì download về)*

Sử dụng lệnh: *$ javac -classpath hadoop-core-1.2.1.jar -d unitTask3* WordLength*.java*

hadoop-core-1.2.1.jar để chứa các class file cần thiết để thông dịch chương trình, -d xác định thư mục chứa các class file được tạo ra

Sử dụng lệnh: *$ jar -cvf unitTask3.jar -C unitTask3/ .*

Để tạo file unitTask3.jar chứa tất cả (dấu .) file trong thư mục unitTask3 (-C là lấy trong đường dẫn hiện tại)

Text

Description automatically generated

### Tạo thư mục input trong HDFS

*$ hdfs dfs -mkdir input\_dir003*



### Đưa dữ liệu textwordlength vào thư mục input\_dir03 trong HDFS

*$ hdfs dfs -put textwordlength input\_dir003*

Text

Description automatically generated

1. Thực thi chương trình count trong hadoop  
   *$ hadoop jar unitTask3.jar de.tud.stg.WordLength input\_dir003 output\_dir003* Text

   Description automatically generated

Text

Description automatically generated

### Xuất kết quả

Dùng cat để xem kết quả nằm trong file part-r-0000

*$ hdfs dfs -cat output\_dir003/part-r-00000*

Text

Description automatically generated

### Copy kết quả output\_dir03 từ Hadoop ra thư mục bên ngoài

*$ hdfs dfs -get output\_dir003 /home/hadoopthinh*

*Và dùng lệnh $ls để xem tất cả các file*

Dùng lệnh *$ls output\_dir03/* để xem file part-r-00000 đã thành công. Và dùng lệnh *$cat output\_dir003/part-r-00000* xem lại kết quả

Text

Description automatically generated

# Task 4 Comprehension

*Understand and explain what the following code does. What is the output of the program for the following input?*

*file1.txt: Hello World Bye World*

*file2.txt: Hello Hadoop Goodbye Hadoop*

\*\*\* File Comprehension

Check nội dung các file cần thực hiện trong Task 4

File Comprehension.java: vi Comprehension.javaText

Description automatically generated

### Text Description automatically generated

### File file1,file2: vi file01 , vi file02A picture containing graphical user interface Description automatically generatedText Description automatically generated with medium confidence 3. Tạo thư mục inputtask4 chứa file01, file02

*$ mkdir inputtask4*

Từ trang /home/hadoopthinh copy file01, file02 vào thư mục inputtask4 vừa tạo. Bằng lệnh: *$cp file01 inputtask4 , $cp file02 inputtask4*

## Text Description automatically generated

## Thông dịch và thực thi chương trình Comprehension

### Tạo thư mục unitTask4

Chứa các file sau khi thông dịch Comprehension.java: *$ mkdir unitTask4*

*Text

Description automatically generated*

### Thông dịch file nguồn và tạo file jar *(cần phải có hadoop-core-1.2.1.jar, nếu không có thì download về)*

Sử dụng lệnh: *$ javac -classpath hadoop-core-1.2.1.jar -d unitTask4 Comprehension.java*

hadoop-core-1.2.1.jar để chứa các class file cần thiết để thông dịch chương trình, -d xác định thư mục chứa các class file được tạo ra

Sử dụng lệnh: *$ jar -cvf unitTask4.jar -C unitTask4/ .*

### Tạo thư mục input trong HDFS

*$ hdfs dfs -mkdir input\_dir04*

### Đưa dữ liệu thư mục inputtask4 vào thư mục input\_dir04 trong HDFS

*$ hdfs dfs -put inputtask4 input\_dir04*

1. Thực thi chương trình count trong hadoop  
   *$ hadoop jar unitTask4.jar de.tud.stg.Comprehension input\_dir04/inputtask4 output\_dir0004*

Text

Description automatically generated

Text

Description automatically generated

### Xuất kết quả

Dùng cat để xem kết quả nằm trong file part-r-0000

*$ hdfs dfs -cat output\_dir0004/part-r-00000*

Graphical user interface, text

Description automatically generated

### Copy kết quả output\_dir004 từ Hadoop ra thư mục bên ngoài

*$ hdfs dfs -get output\_dir0004 /home/hadoopthinh*

*Và dùng lệnh $ls để xem tất cả các file*

Dùng lệnh *$ls output\_dir0004/* để xem file part-r-00000 đã thành công. Và dùng lệnh *$cat output\_dir004/part-r-00000* xem lại kết quả

Graphical user interface, text

Description automatically generated