**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH KHOA CÔNG NGHỆ THÔNG TIN**

****

**TS. NGÔ HỮU DŨNG**

**BÀI GIẢNG THỰC HÀNH**

**NHẬP MÔN DỮ LIỆU LỚN**

Ngô Hồng Thông

2264001

**Ỹ nghĩa của Lab 5:**

1. Thực hành viết chương trình MapReduce bằng Java
   * Làm quen với cấu trúc một ứng dụng MapReduce cơ bản (Mapper, Reducer, Driver).
   * Hiểu cơ chế xử lý dữ liệu theo cặp key–value trong Hadoop.
2. Biên dịch và đóng gói chương trình thành JAR
   * Sử dụng lệnh javac để compile code với thư viện Hadoop.
   * Đóng gói các class thành file JAR để chạy trên Hadoop.
3. Chạy chương trình trên Hadoop ở chế độ pseudo-distributed
   * Thực hành khởi chạy NameNode, DataNode, ResourceManager, NodeManager.
   * Sử dụng lệnh hadoop jar để submit job lên YARN.
4. Xử lý dữ liệu weblogs thực tế (weblogs.txt)
   * Áp dụng WordCount để đếm số lần xuất hiện của các từ trong tập log.
   * Mở rộng thành LineCount (đếm số dòng) và SortByCount (sắp xếp kết quả).
5. So sánh hiệu năng giữa các bài toán và dataset
   * Quan sát sự khác biệt khi chạy WordCount trên dữ liệu nhỏ (students.txt) và dữ liệu lớn (app.log).
   * Hiểu ý nghĩa của HDFS và MapReduce khi xử lý dữ liệu lớn so với cách thủ công

**5.1 Chuẩn bị môi trường**

Để chuẩn bị một môi trường cho bài lab 5

**Tạo thư mục cho lab:**

hadoop@root:~/bigdata-labs/lab5/src$

A black rectangular object with a black border

AI-generated content may be incorrect.

Ở những lab trước em đã setup:  
**Thư mục: $HADOOP\_HOME/etc/hadoop/**

**core-site.xml**

Cấu hình chung của Hadoop.

Thường chứa fs.defaultFS → ví dụ: hdfs://namenode:9000.

**hdfs-site.xml**

Cấu hình HDFS.

Các tham số quan trọng:

dfs.replication → số bản sao mặc định.

dfs.namenode.name.dir → nơi lưu metadata của NameNode.

dfs.datanode.data.dir → nơi lưu block của DataNode.

**mapred-site.xml** (thường copy từ mapred-site.xml.template)

Cấu hình MapReduce.

mapreduce.framework.name = yarn.

**yarn-site.xml**

Cấu hình YARN (ResourceManager, NodeManager).

Ví dụ: yarn.resourcemanager.hostname.

**START lại HDFS và YARN**

Đây là trường hợp em đã mở lại sau nên là HDFS và YARN đã đóng vì thế cần phải khỏi động lại. Nếu mà chạy **JPS** mà ra đẩy đủ các node thì không cần phải chạy lại

**Lệnh :**

**start-dfs.sh**: bật HDFS (lưu trữ dữ liệu)

**start-yarn.sh**: bật YARN (quản lý tài nguyên & job)

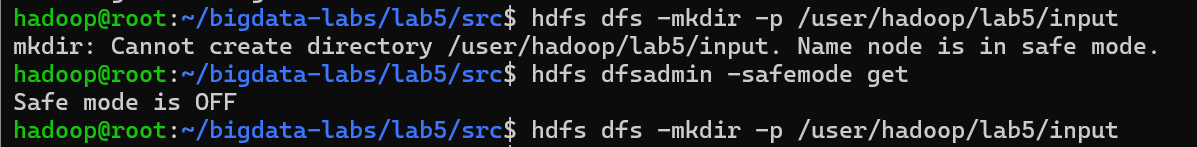
A computer screen with white text

AI-generated content may be incorrect.

Sau khi đã bật lại HDFS và YARN thì ta bắt đầu copy lại dữ liệu để làm các bước tiếp theo

**Copy dataset:**

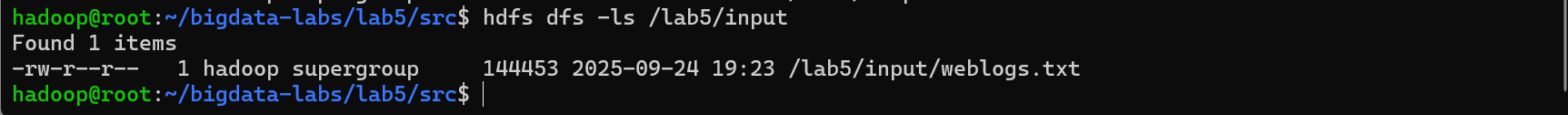
Việc lỗi safe mode là khi mới chạy lại Name node ở trong chế độ safe mode nên cần phải đợi một chút

****

Việc dùng lệnh hdfs dfs để tạo thư mục trong DFS

Còn để copy ta dùng lệnh -put trong hdfs dfs thì để copy file dữ liệu vào hdfs

**Kết quả:**

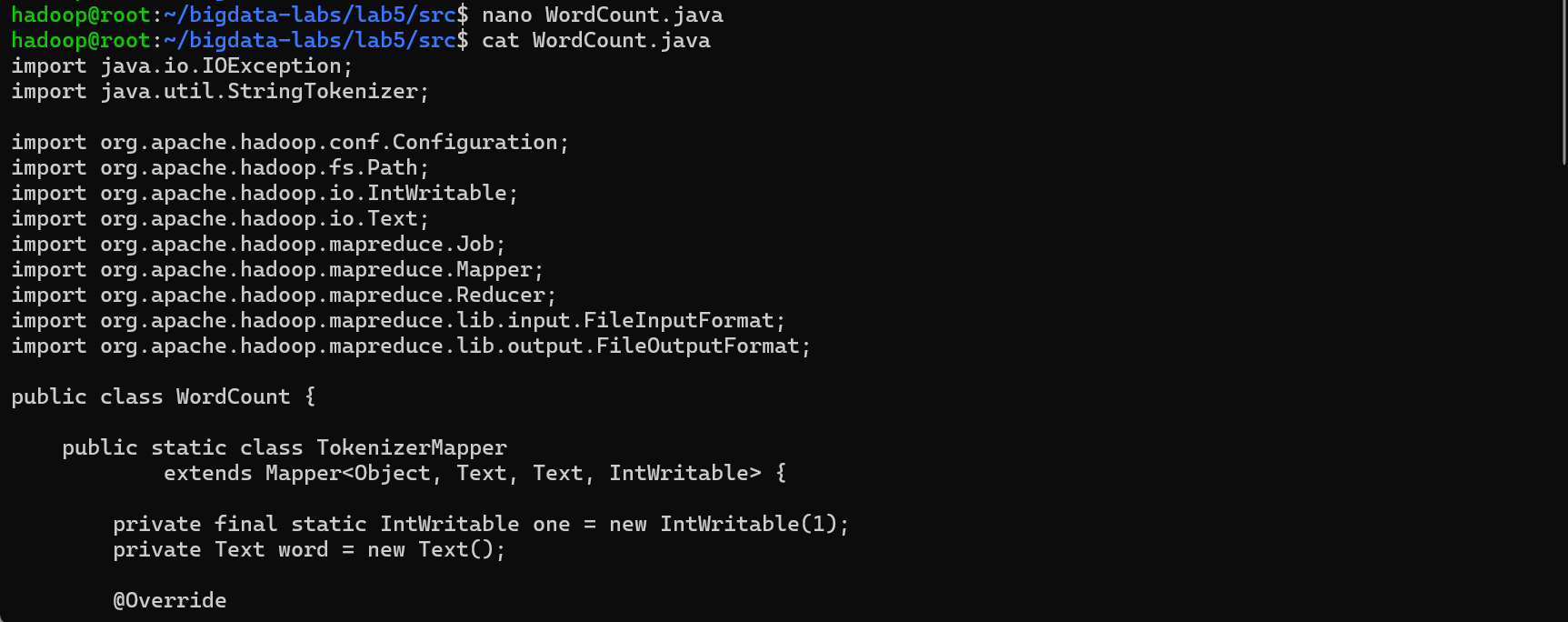
****

**A black screen with white text

AI-generated content may be incorrect.**

**5.2 Viết chương trình WordCount (Java)**

Tạo file WordCount.java:

****

**Luồng xử lý của hàm wordcount**

1. Input: Một hoặc nhiều file văn bản trên HDFS.
   1. Ví dụ: weblogs.txt hoặc input.txt.
2. Mapper:
   1. Đọc từng dòng văn bản.
   2. Tách thành các từ.
   3. Phát ra cặp (từ, 1) cho mỗi từ.
3. Shuffle & Sort (Hadoop runtime):
   1. Tự động gom nhóm tất cả các cặp (từ, 1) có cùng key (từ).
   2. Đưa về cho Reducer.
4. Reducer:
   1. Nhận (từ, [1,1,1,...]).
   2. Cộng dồn lại → (từ, số\_lần\_xuất\_hiện).
5. Output: File kết quả trên HDFS chứa danh sách từ và số lần xuất hiện.

**Ví dụ cho hàm wordcount:**

**Input file:**

Hadoop is fast

Hadoop is scalable

**Mapper output:**

(Hadoop,1) (is,1) (fast,1) (Hadoop,1) (is,1) (scalable,1)

**Reducer output:**

(Hadoop,2) (is,2) (fast,1) (scalable,1)

* Kết quả đạt được sẽ đếm được tổng số lần xuất hiện chung của những từ ngữ có trong file input đầu vào

**5.3 Compile và đóng gói**

Sau khi đã có được file wordcount bằng java thì phải compile và đóng gói lại thì mới chạy được hàm wordcount trong hdfs

**Tạo thư mục build:**

mkdir -p ~/bigdata-labs/lab5/build

**Compile:**

cd ~/bigdata-labs/lab5/src

javac -classpath $(hadoop classpath) -d ../build WordCount.java Đóng gói JAR:

**cd ~/bigdata-labs/lab5/build**

jar cf wc.jar \*.class

**A screen shot of a computer

AI-generated content may be incorrect.**

**5.4 Chạy MapReduce Job**

Sau khi đã có hàm wordcount thì nếu chưa put file dữ liệu vào input thì ta sẽ put dữ liệu là web.log vào input để làm data cho chạy mapreduce.

Bước tiếp theo nếu đã có dữ liệu thì chạy mapreduce trên file đó

**Lệnh:**

hdfs dfs -put ~/bigdata-labs/lab5/src/web.logs.txt /lab5/input

hadoop jar ~/bigdata-labs/lab5/build/wc.jar WordCount /lab5/input /lab5/output\_test\_20251001

**Giải thích lệnh:**

Chạy chương trình MapReduce Wordcount được đóng gói trong wc.jar, lấy dữ liệu đầu vào từ thư mục /lab5/input trên HDFS, xử lý và lưu kết quả vào /lab5/output trên HDFS.

**Kết quả:  
A screen shot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Kết quả output dữ liệu đầu ra:  
A black screen with white text

AI-generated content may be incorrect.**

**Kết quả khi chạy thêm time;  
A screen shot of a computer

AI-generated content may be incorrect.**

**Kết quả đọc file dữ liệu đầu ra:**

**Lệnh :** hdfs dfs -cat /lab5/output/part-r-00000

**A screenshot of a computer

AI-generated content may be incorrect.**

**Thống kê theo method**

"GET": xuất hiện 2000 lần → nghĩa là trong log có 2000 request GET.

200: xuất hiện 2000 lần → số lượng request thành công (HTTP status code 200).

HTTP/1.1": xuất hiện 2000 lần → tất cả request dùng giao thức HTTP/1.1.

**Thống kê theo engpoint**

/ → trang chủ, được truy cập 410 lần.

/cart → trang giỏ hàng, 393 lần.

/checkout → trang thanh toán, 398 lần.

/login → trang đăng nhập, 401 lần.

**Thông kế theo địa chỉ IP**

192.168.1.1 -> 101 lần

192.168.1.10 -> 110 lần

192.168.1.11 -> 104 lần

192.168.1.9 -> 99 lần

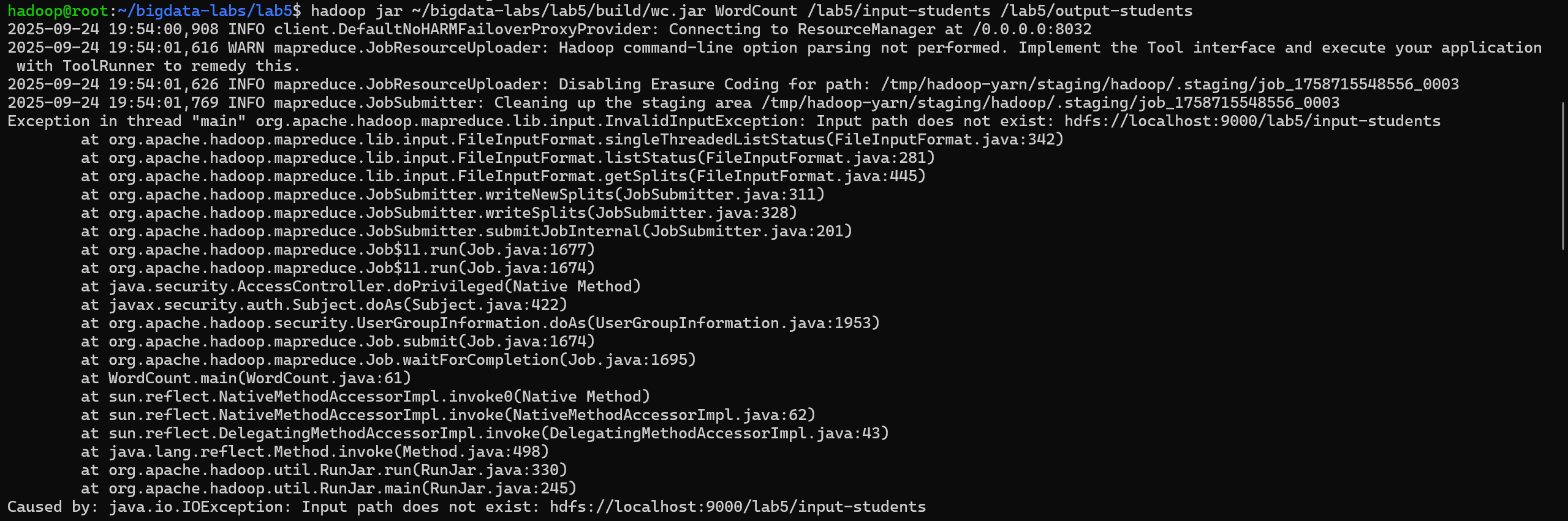
**Thống kế theo ngày**

Ngày 10/07/2024 có 75 request.

Ngày 15/07/2024 có 79 request

Ngày 1/07/2024 có 61 request

**Chạy wordcount trên tập dữ liệu student.txt**

****

**Kết quả đọc file dữ liệu đầu ra:**

**A screen shot of a computer

AI-generated content may be incorrect.**

* Dữ liệu của tập student ít hơn và ít phức tạp hơn

**5.7 MapReduce nâng cao – SortByCount**

Để chuẩn bị cho bài nâng cao em đã cho vào bộ dữ liệu ~2G để kiểm tra hiệu năng trên bộ dữ liệu lớn. Đây là bộ data app.log của một Backend đùng để lưu trữ lại logic khi chạy code

**A screenshot of a computer

AI-generated content may be incorrect.**

**A computer screen with white and blue text

AI-generated content may be incorrect.**

**Trước tiên hãy build trước hàm SortByCount**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Luồng xử lý của hàm SortByCount**

1. Input: File kết quả từ chương trình WordCount (ví dụ part-r-00000).
2. Mapper:
   1. Đọc từng dòng (gồm từ + số đếm, cách nhau bởi tab).
   2. Đảo ngược cặp key-value từ (word, count) → (count, word).
3. Shuffle & Sort (Hadoop runtime):
   1. Hadoop gom nhóm các cặp theo **key mới (count)**.
   2. Sắp xếp theo key (mặc định tăng dần, có thể chỉnh Comparator để giảm dần).
4. Reducer:
   1. Nhận key (count) và danh sách các từ có cùng giá trị count.
   2. Phát ra (count, word) cho từng từ.
5. Output: File kết quả trên HDFS chứa danh sách từ và số lần xuất hiện theo thứ tự.

**Ví dụ của hàm SortByCount**

1. Input file (kết quả từ WordCount):

Hadoop 2

is 2

fast 1

scalable 1

1. Mapper output:

(2, Hadoop)

(2, is)

(1, fast)

(1, scalable)

1. Reducer output (sắp xếp theo count):

(1, fast)

(1, scalable)

(2, Hadoop)

(2, is)

**Buildl và compile hàm SortByCount để có thể chạy trên Hadoop**

javac -classpath $(hadoop classpath) -d ../build SortByCount.java

**Sau khi chạy xong sẽ có hàm trong build**

cd ~/bigdata-labs/lab5/build

**Đóng gói thành jar**

jar cf midterm.jar \*.class

**Kết quả**

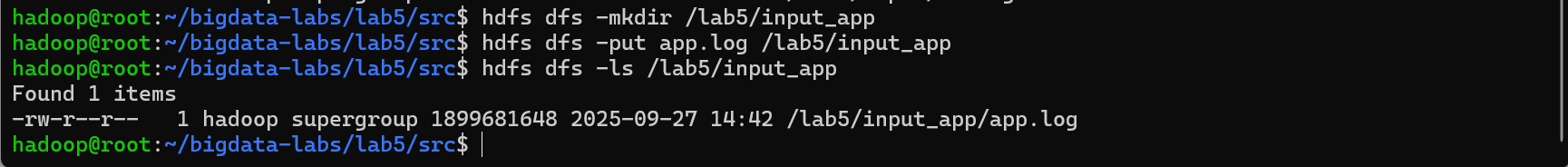
Có cả 2 hàm WordCount và SortByCount.

A screenshot of a computer

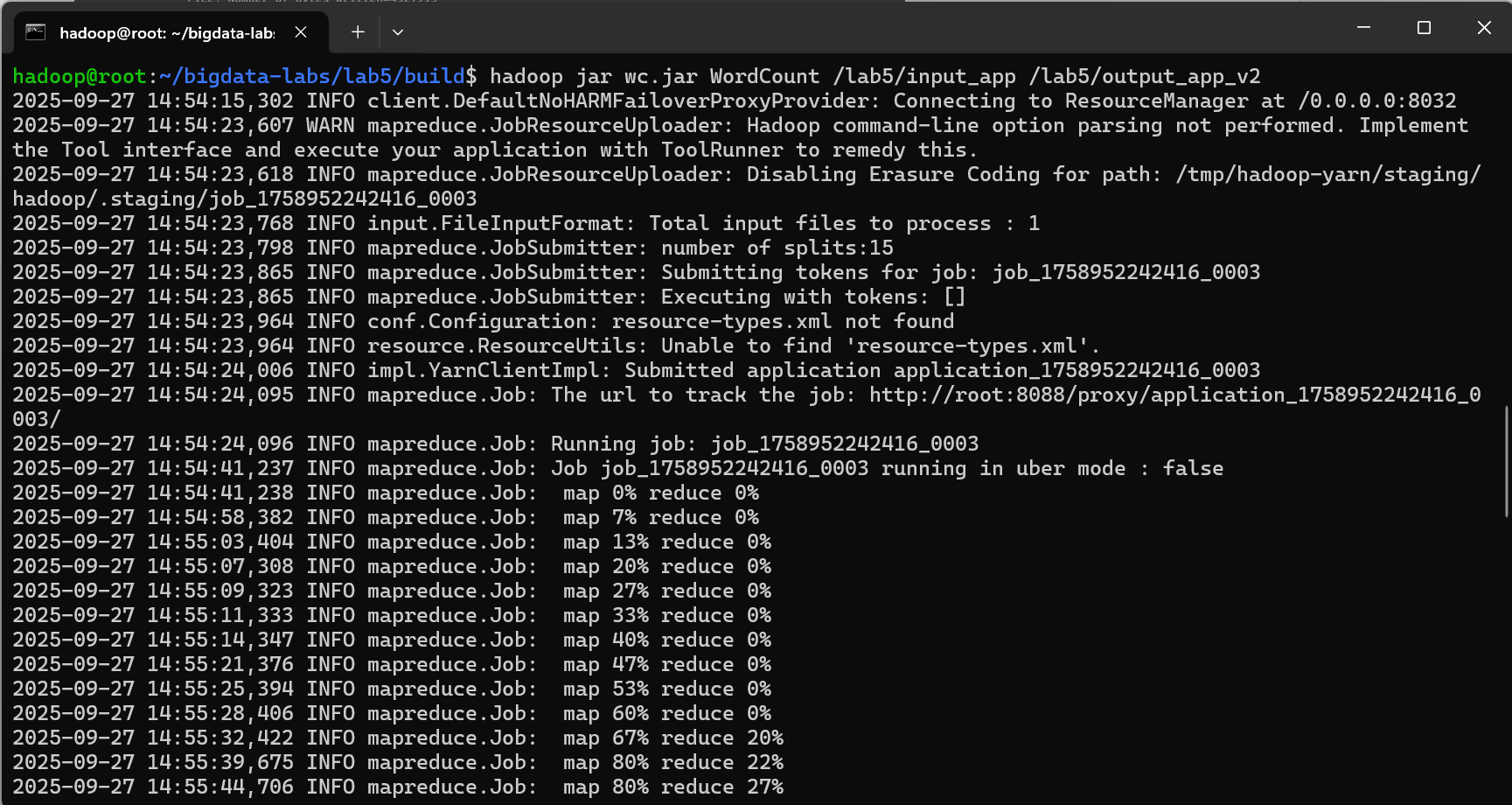
AI-generated content may be incorrect.

**Chạy Worcount và SortByCount trên bộ data bên ngoài với khoảng 10 triệu dòng dữ liệu**

Tạo thư mục và put file dữ liệu vào thư mục để chuẩn bị chạy HDFS



Đâu tiên chạy trươc wordcount trên file app.log (10tr dòng)

****

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Đọc kết quả từ file kết quả**

**A screen shot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Vì là data log nên là sẽ lấy được từng kí tự cũng như là từng chữ theo bài toán wordcount**

**Tiếp theo chạy SortByCount trên bộ dữ liệu đã được chạy SortByCount output đầu ra của hàm CountWord**

hadoop jar midterm.jar SortByCount /lab5/output\_app\_v2 /lab5/output\_app\_v2\_sortbycount

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screen shot of a computer

AI-generated content may be incorrect.**

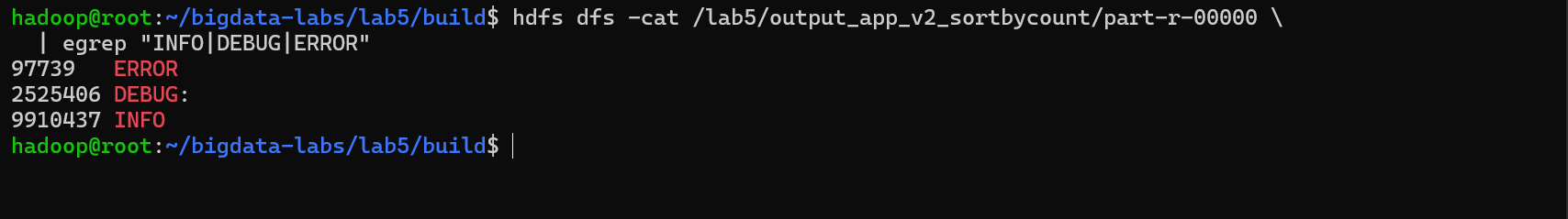
**Đọc kết quả sau khi đã SortByCount**

**A computer screen with white text

AI-generated content may be incorrect.**

**A computer screen with white text

AI-generated content may be incorrect.**

****

**Mục đích của việc em áp dụng WordCount và SortByCount để có thể đếm được số lượng log được ghi ra, là INFO, DEBUG, ERROR**

Tóm lại

1. Thời gian thực chạy job = dựa vào dấu mốc Running job và completed successfully (≈ 57s).
2. Thời gian CPU tiêu tốn = CPU time spent (≈ 34.7s).
3. Thời gian của từng phase:
4. Map slots: 10.5s
5. Reduce slots: 12.4s

**NÂNG CAO CHẠY TRÊN NHIỀU NODE MÀ NHIỀU MÁY ẢO**

**Chuẩn bị 3 máy cùng phiên bản java và hadoop**

1 Máy NodeName (24G)

2 Máy worker (8g)

A screenshot of a computer

AI-generated content may be incorrect.  
  
ở trên ba máy phải setup để có thể nhận biết được từng máy

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

Chỉ khi ping được giữa các máy là bạn đã có thể connect giữa 3 máy

**Cài đạt lại các file hdfs-site và core-site cho worker để không chạy namenode**

Hdfs-site cho namenode

**A screenshot of a computer

AI-generated content may be incorrect.**

Hdfs-site cho worker

**A screenshot of a computer screen

AI-generated content may be incorrect.**

Core-site trên namenode cho cả trên woker

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Setup worker**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Chạy lại hdfs và yarn trên namenode**

stop-dfs.sh/ start-dfs.sh

stop-yarn.sh/ start-yarn.sh

**Jps cho namenode**



**JPS cho woker**

A screenshot of a computer

AI-generated content may be incorrect.

**Sau đó chạy để quản dung lượng sử lý, số lượng Datanode, tình trạng block**

hdfs dfsadmin -report

**A screenshot of a computer

AI-generated content may be incorrect.**

yarn node -list

**A screen shot of a computer

AI-generated content may be incorrect.**

* **Chỉ khi chạy và thấy rằng datanode(2) thì ok**

**Chạy lại WordCount trên file 2g mới fully distributed (3 máy)**

Chạy lại file 2G xem tốc độ và cách chia plit

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**Chạy thêm SortByCount trên dữ liệu kết quả**

**A screenshot of a computer program

AI-generated content may be incorrect.**

Với HDFS + MapReduce:

Dữ liệu được chia block (128MB/256MB) → phân tán nhiều node.

MapReduce chạy song song trên nhiều node → tốc độ nhanh hơn nhiều.

Tự động chống lỗi: node hỏng thì task được chạy lại ở node khác.

Rất phù hợp cho dữ liệu lớn (GB → TB → PB).

Nhược điểm: overhead tạo job, container, JVM warm-up → với file nhỏ sẽ chậm hơn Linux.