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

Dành cho bậc đại học

THÀNH PHỐ HỒ CHÍ MINH - NĂM 2025

LỜI NÓI ĐẦU

Trong thời đại hiện nay, dữ liệu được sinh ra với tốc độ chóng mặt, từ các mạng xã hội, thiết bị IoT, đến các hệ thống giao dịch thương mại điện tử. Việc thu thập, lưu trữ, xử lý và phân tích khối lượng dữ liệu khổng lồ này là thách thức lớn, nhưng cũng mở ra cơ hội tuyệt vời cho các nhà khoa học dữ liệu, kỹ sư dữ liệu và các nhà nghiên cứu.

Mục tiêu của bộ bài giảng thực hành này là giúp sinh viên:

- Hiểu cơ bản về môi trường Linux và kết nối từ xa qua SSH.
- Làm quen với Hadoop và HDFS, từ chế độ standalone đến pseudo-distributed.
- Thực hành viết các chương trình MapReduce để xử lý dữ liệu thực tế.
- Khởi động với Apache Spark, bao gồm RDD, DataFrame, và Spark SQL, tích hợp với Hive.
- Sử dụng Python, Pandas, và các thư viện trực quan hóa (Matplotlib, Seaborn, Plotly) để khai thác dữ liệu và xây dựng dashboard cơ bản.
- Chuẩn bị kỹ năng cần thiết để thực hiện project cuối kỳ, từ phân tích dữ liệu, xử lý song song đến trực quan hóa và báo cáo kết quả.

Bài giảng được xây dựng theo hình thức thực hành từng bước, kết hợp lab hướng dẫn chi tiết và bài tập thực tế. Sinh viên sẽ có cơ hội làm việc trực tiếp trên máy ảo Ubuntu, thao tác với Hadoop và Spark, cũng như trải nghiệm việc triển khai một pipeline dữ liệu từ đầu đến cuối.

Các lab được thiết kế theo hướng linh hoạt, phù hợp cả với sinh viên mới bắt đầu và những bạn muốn thử nghiệm các môi trường nâng cao như pseudo-distributed hoặc fully-distributed. Ngoài ra, sinh viên sẽ được khuyến khích tự học và mở rộng kiến thức thông qua các video hướng dẫn và tài liệu tham khảo đi kèm.

Hy vọng bộ bài giảng này sẽ là cẩm nang thực hành hữu ích, giúp các bạn không chỉ nắm vững kiến thức lý thuyết mà còn có thể ứng dụng vào các dự án thực tế, nghiên cứu, hoặc khóa luận tốt nghiệp.

Chúc các bạn học tập hăng say và khám phá thế giới dữ liệu lớn đầy thú vị!

Thành phố Hồ Chí Minh, ngày 17 tháng 9 năm 2025

Người biên soạn

MỤC LỤC

LAB 1 LINUX & SSH

Mục tiêu

- Cài Ubuntu Server (VM) hoặc dùng image có sẵn.
- Kết nối tới máy ảo bằng SSH từ máy thật.
- Ôn/biết các lệnh Linux cơ bản và thao tác file.

1.1 Tài nguyên cần tải về

- Ubuntu Server 20.04 LTS ISO (Focal): https://releases.ubuntu.com/focal/ubuntu-20.04.6-live-server-amd64.iso
- VirtualBox (nếu dùng VirtualBox): https://www.virtualbox.org/wiki/Downloads
- VMware Workstation (nếu dùng VMware): https://www.vmware.com
- Image sẵn có từ OSBoxes: VDI VirtualBox: Ubuntu Server 20.4; Username: osboxes, Password: osboxes.org

1.2 Tạo máy ảo Ubuntu Server

1.2.1 Tạo VM mới

1. Mở VirtualBox → New → đặt tên (ví dụ server-20.04).
2. Type: Linux, Version: Ubuntu (64-bit).
3. RAM: tối thiểu **2048 MB** (tốt hơn 4096 MB nếu máy host đủ).
4. CPU: 2 cores (tùy host).
5. Trường hợp 1 - dùng image VDI của OSBoxes (nhanh hơn):
    - Giải nén .7z (7zip),
    - Chọn Use existing virtual disk, chọn file VDI.
    - Login mặc định: osboxes / osboxes.org.
    - Sau khi đăng nhập, có thể đổi mật khẩu bằng `passwd`.

    Trường hợp 2 - dùng image ISO để cài từ đầu:
    - Chọn Create a virtual hard disk, định dạng VDI

- Storage: Dynamically allocated, dung lượng > 10GB
- Mount file ISO vào ổ CD/DVD của VM, Start, tiến hành cài Ubuntu Server theo hướng dẫn mặc định.

1.2.2 Thiết lập mạng

- **Bridged Adapter**: máy ảo nhận IP trên cùng mạng LAN với máy thật — dễ SSH.
- **NAT**: an toàn, nhưng nếu dùng NAT bạn cần cấu hình Port Forwarding (VirtualBox → Settings → Network → Adapter (NAT) → Advanced → Port Forwarding). Ví dụ:
  - Host Port: 2222 → Guest Port: 22
  - Sau đó SSH từ máy thật vào: ssh -p 2222 <user>@127.0.0.1

1.3 Cài SSH trên Ubuntu

Đăng nhập vào VM (bằng console VM trong VirtualBox) và chạy:

```
sudo apt update
sudo apt install -y openssh-server
sudo systemctl enable --now ssh
sudo systemctl status ssh   # kiểm tra đang chạy
```

Nếu firewall bật (ufw), cho phép SSH:

```
sudo ufw allow ssh
sudo ufw status
```

1.4 Kiểm tra địa chỉ IP của máy ảo

```
ip a
# hoặc
hostname -I
```

Ghi lại IP (ví dụ 192.168.1.50) — dùng để SSH từ máy thật.

1.5 Kết nối SSH từ máy thật

- **Windows 10/11 PowerShell**:

```
ssh username@<IP_may_ao>
# ví dụ
ssh osboxes@192.168.1.50
```

- **PuTTY (Windows cũ)**: mở PuTTY → Hostname/IP → Port 22 (hoặc 2222 nếu NAT+port-forwarding) → Open.

Như vậy bạn đã giả lập một máy chủ ảo và có thể sử dụng kết nối SSH để truy cập từ xa. Login vào hệ thống máy chủ ảo từ máy thật bằng cách sử dụng SSH và thực hiện các bài tập ở phần tiếp theo.

1.6 Bài tập thực hành

1.6.1 Thông tin cơ bản

Kiểm tra **user hiện tại**:

```
whoami
```

Kiểm tra **thư mục hiện tại**:

```
pwd
```

Liệt kê nội dung thư mục:

```
ls -l
```

1.6.2 Làm việc với thư mục

Tạo thư mục Lab 1:

```
mkdir -p ~/bigdata-labs/lab1
cd ~/bigdata-labs/lab1
```

Tạo vài thư mục con:

```
mkdir data logs
```

1.6.3 Làm việc với file

Tạo file hello.txt với nội dung:

```
echo "Hello Big Data at IUH" > hello.txt
```

Copy file:

```
cp hello.txt hello_copy.txt
```

Tạo symbolic link:

```
ln -s hello.txt hello_link.txt
```

Xem nội dung file:

```
cat hello.txt
cat hello_copy.txt
cat hello_link.txt
```

Xóa file gốc hello.txt, rồi thử xem nội dung link:

```
rm hello.txt
cat hello_link.txt
```

→ Ghi nhận sự khác biệt giữa hard link và soft link.

1.6.4 Chuyển file giữa máy thật và máy ảo bằng SCP

Từ máy thật, gửi file test.txt vào máy ảo:

```
scp test.txt osboxes@<IP_may_ao>:~/bigdata-labs/lab1/
```

Từ **máy ảo**, copy file hello_copy.txt về máy thật:

```
scp hello_copy.txt <user_may_that>@<IP_may_that>:~/Desktop/
```

1.6.5 Làm việc với log và dữ liệu text

Tạo file log thời gian:

```
date > log_time.txt
```

Xem nội dung bằng:

```
cat log_time.txt
less log_time.txt
head log_time.txt
tail log_time.txt
```

Tạo file students.txt chứa danh sách 10 sinh viên (dùng cat > students.txt, kết thúc bằng Ctrl+D). Hiển thị danh sách theo thứ tự:

```
sort students.txt
```

Hiển thị có số dòng:

```
nl students.txt
```

Upload file weblogs.txt mẫu để dùng cho các lab sau

1. Giảng viên cung cấp file weblogs.txt (tải về máy thật trước).
2. Từ máy thật, gửi file này vào VM:

```
scp weblogs.txt osboxes@<IP_may_ao>:~/bigdata-labs/lab1/
```

3. Trên máy ảo, kiểm tra:

```
ls -lh ~/bigdata-labs/lab1/weblogs.txt
head ~/bigdata-labs/lab1/weblogs.txt
```

Ghi chú: File này sẽ được dùng lại trong **Lab 2 (Hadoop MapReduce)** và các lab tiếp theo.

1.6.6 Đăng xuất SSH

Gõ:

```
exit
```

hoặc nhấn Ctrl+D.

1.7 Tài liệu tham khảo

- Ubuntu Server 20.04 LTS: https://releases.ubuntu.com/focal/
- VirtualBox: https://www.virtualbox.org/wiki/Downloads
- SSH cơ bản: https://www.openssh.com/manual.html
- SCP command: https://linux.die.net/man/1/scp

LAB 2 HADOOP SINGLE NODE SETUP

Mục tiêu

- Cài Java (OpenJDK) và Hadoop (single-node).
- Biết cấu hình biến môi trường cần thiết cho Hadoop.
- Chạy ví dụ MapReduce (ví dụ grep) trong chế độ standalone để kiểm tra cài đặt.
- Chuẩn bị cho Lab 3 (pseudo-distributed).

Bạn đã có Ubuntu Server VM và có thể SSH từ máy thật.

- Bạn đã có một thư mục làm việc, ví dụ ~/bigdata-labs/lab1 chứa dữ liệu mẫu (weblogs.txt, student_scores.csv, ...).

2.1 Tải Hadoop

Bạn có thể dùng phiên bản bạn muốn; ở đây ví dụ dùng **Hadoop 3.3.1** (link mẫu):

https://archive.apache.org/dist/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz

Trên VM:

```
cd ~
wget https://archive.apache.org/dist/hadoop/common/hadoop-
3.3.1/hadoop-3.3.1.tar.gz
tar -xzf hadoop-3.3.1.tar.gz
# sẽ giải nén thành thư mục hadoop-3.3.1 trong home
```

Bạn cũng có thể dùng lệnh scp để copy file nén lên server ảo.

2.2 Cài Java (OpenJDK 11)

```
sudo apt update
sudo apt install -y openjdk-11-jdk
```

Kiểm tra:

```
java -version
# mong thấy "openjdk version "11.x"
```

Tìm đường dẫn JAVA_HOME:

```
export JAVA_HOME=$(dirname $(dirname $(readlink -f $(which
java))))
echo $JAVA_HOME
# ví dụ: /usr/lib/jvm/java-11-openjdk-amd64
```

2.3 Thiết lập biến môi trường cho Hadoop (tạm thời và persistent)

Giả sử Hadoop giải nén tại ~/hadoop-3.3.1. Bạn có thể thêm các dòng sau vào ~/.bashrc để persistent:

Mở ~/.bashrc (hoặc ~/.profile) và thêm:

```
# Hadoop env
export HADOOP_HOME=~/hadoop-3.3.1
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

# JAVA (nếu chưa set)
export JAVA_HOME=$(dirname $(dirname $(readlink -f $(which
java))))
```

Sau khi lưu:

```
source ~/.bashrc
```

Kiểm tra:

```
echo $HADOOP_HOME
hadoop version
# nếu thấy version -> OK
```

2.4 Cấu hình hadoop-env.sh

Mở file:

```
$HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

Tìm dòng # export JAVA_HOME=... và sửa thành:

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
```

Lưu file.

2.5 Kiểm tra lệnh Hadoop (standalone)

Standalone (mặc định) không cần chỉnh file XML hay khởi daemon.

Chạy:

```
$HADOOP_HOME/bin/hadoop
# hoặc
hadoop
```

Bạn sẽ thấy output help/usage. Nếu lệnh báo lỗi "JAVA_HOME not set" hoặc tương tự — quay lại bước 2/4.

2.6 Chạy ví dụ MapReduce (Standalone Mode)

Ví dụ grep có sẵn trong JAR ví dụ. Thực hiện tại thư mục Hadoop home hoặc nơi bạn muốn:

Chuẩn bị input:

```
mkdir -p ~/hadoop-demo/input
# copy một số file mẫu (vd .xml config)
cp $HADOOP_HOME/etc/hadoop/*.xml ~/hadoop-demo/input/
# hoặc copy weblogs.txt từ lab1:
cp ~/bigdata-labs/lab1/weblogs.txt ~/hadoop-demo/input/
cd ~/hadoop-demo
```

Chạy ví dụ grep:

```
$HADOOP_HOME/bin/hadoop jar
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-
3.3.1.jar grep input output 'dfs[a-z.]+'
```

- Lệnh trên: dùng JAR ví dụ để chạy job grep, tìm theo regex dfs[a-z.]+.
- Nếu JAR phiên bản khác, sửa tên JAR cho đúng (dùng ls $HADOOP_HOME/share/hadoop/mapreduce/ để kiểm tra).

Xem output:

```
cat output/*
# hoặc
ls output
```

Ghi chú: Ở chế độ standalone, Hadoop chạy trong 1 tiến trình Java, kết quả viết ra thư mục local output.

2.7 Thử ví dụ WordCount

Bạn có thể thử wordcount trên file logs:

```
# tạo input từ file logs
mkdir -p input_wc
```

```
cp ~/bigdata-labs/lab1/weblogs.txt input_wc/

# chạy wordcount
$HADOOP_HOME/bin/hadoop jar
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-
3.3.1.jar wordcount input_wc output_wc

# kiểm tra kết quả
cat output_wc/part-r-00000 | head
```

2.8 Pseudo-Distributed Mode

2.8.1 Cấu hình Hadoop (pseudo-distributed)

Sửa các file trong $HADOOP_HOME/etc/hadoop/:

core-site.xml

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

hdfs-site.xml

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///home/osboxes/hadoop_data/dfs/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///home/osboxes/hadoop_data/dfs/datanode</value>
  </property>
</configuration>
```

mapred-site.xml

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
```

```
</configuration>
```

yarn-site.xml

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

2.8.2 Khởi động Hadoop

Format NameNode (chỉ lần đầu):

```
hdfs namenode -format
```

Start HDFS:

```
start-dfs.sh
```

Start YARN:

```
start-yarn.sh
```

Kiểm tra daemon:

```
jps
```

Kỳ vọng: NameNode, DataNode, ResourceManager, NodeManager.

2.8.3 Test WordCount

Tạo thư mục input trên HDFS:

```
hdfs dfs -mkdir -p /user/osboxes/input
hdfs dfs -put $HADOOP_HOME/etc/hadoop/*.xml /user/osboxes/input
```

Chạy WordCount:

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-examples-*.jar wordcount /user/osboxes/input
/user/osboxes/output
```

Xem kết quả:

```
hdfs dfs -cat /user/osboxes/output/part-r-00000 | head
```

2.9 Tài liệu tham khảo

- Hadoop official docs: https://hadoop.apache.org/docs/

- Hadoop MapReduce examples: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
- Java OpenJDK: https://openjdk.org/

LAB 3 HDFS & DATA INGESTION

Mục tiêu

- Hiểu khái niệm HDFS và sự khác biệt với hệ thống file local.
- Thực hành quản lý file/directory trên HDFS.
- Upload dữ liệu thực vào HDFS.
- Chạy WordCount/grep trên dữ liệu lớn hơn.
- Làm quen với log và kết quả lưu trên HDFS.

3.1 Chuẩn bị

- Máy ảo Ubuntu Server đã cài Hadoop (Lab 2, pseudo-distributed).
- Dữ liệu mẫu (đã có weblogs.txt, có thể cung cấp thêm student_scores.csv qua link/scp).
- HDFS đang chạy (NameNode + DataNode + YARN).

```
jps
# mong thấy: NameNode, DataNode, ResourceManager, NodeManager
```

3.2 Các lệnh cơ bản với HDFS

So sánh local vs HDFS:

```
# Local
ls ~/bigdata-labs/lab1

# HDFS (ls root user folder)
hdfs dfs -ls /

# Tạo thư mục trên HDFS
hdfs dfs -mkdir -p /user/osboxes/lab3/input
```

Các thao tác:

```
# Copy file từ local lên HDFS
hdfs dfs -put ~/bigdata-labs/lab1/weblogs.txt
/user/osboxes/lab3/input/

# Copy ngược về local
hdfs dfs -get /user/osboxes/lab3/input/weblogs.txt
~/Desktop/weblogs_copy.txt
```

```
# Xem nội dung file
hdfs dfs -cat /user/osboxes/lab3/input/weblogs.txt | head

# Kiểm tra dung lượng
hdfs dfs -du -h /user/osboxes/lab3/input

# Xóa file/thư mục
hdfs dfs -rm /user/osboxes/lab3/input/weblogs.txt
hdfs dfs -rm -r /user/osboxes/lab3/output_wc
```

3.3 Thực hành WordCount trên HDFS

```
# Upload lại file
hdfs dfs -put ~/bigdata-labs/lab1/weblogs.txt
/user/osboxes/lab3/input/

# Chạy wordcount
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-examples-*.jar \
  wordcount /user/osboxes/lab3/input
/user/osboxes/lab3/output_wc

# Xem kết quả
hdfs dfs -ls /user/osboxes/lab3/output_wc
hdfs dfs -cat /user/osboxes/lab3/output_wc/part-r-00000 | head
```

3.4 Thực hành Grep trên HDFS

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-examples-*.jar \
  grep /user/osboxes/lab3/input /user/osboxes/lab3/output_grep
'error|fail'

hdfs dfs -cat /user/osboxes/lab3/output_grep/part-r-00000 |
head
```

3.5 Bài tập thực hành

3.5.1 Tạo thư mục /user/osboxes/lab3/logs trên HDFS

```
hdfs dfs -mkdir -p /user/osboxes/lab3/logs
hdfs dfs -ls /user/osboxes/lab3
```

3.5.2 Upload file log_time.txt.

```
hdfs dfs -put ~/bigdata-labs/lab1/log_time.txt /user/osboxes/lab3/logs/
```

```
hdfs dfs -ls /user/osboxes/lab3/logs
```

3.5.3 Xem nội dung trên HDFS

```
hdfs dfs -cat /user/osboxes/lab3/logs/log_time.txt | head -5
```

3.5.4 Upload và chạy WordCount

1. Upload file students.txt từ Lab 1 lên HDFS:

```
hdfs dfs -put ~/bigdata-labs/lab1/students.txt
/user/osboxes/lab3/
```

2. Chạy WordCount trên local filesystem:

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-examples-*.jar wordcount ~/bigdata-
labs/lab1/students.txt ~/bigdata-labs/lab1/output_local
```

3. Chạy WordCount trên HDFS:

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-examples-*.jar wordcount
/user/osboxes/lab3/students.txt /user/osboxes/lab3/output_hdfs
```

4. Xem kết quả:

```
hdfs dfs -cat /user/osboxes/lab3/output_hdfs/part-r-00000
```

3.5.5 So sánh hiệu năng

- Ghi nhận thời gian thực thi WordCount trên local vs HDFS (có thể dùng time để đo).
- Nhận xét sự khác biệt (dù nhỏ, nhưng HDFS mới là nền tảng để chạy trên cluster ở Lab sau).

3.5.6 Kiểm tra dung lượng file trên HDFS

```
hdfs dfs -du -h /user/osboxes/lab3/
```

3.6 Tài liệu tham khảo

- HDFS Commands Guide: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html
- Hadoop file permissions: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsPermissionsGuide.html

LAB 4 MAPREDUCE

Mục tiêu

- Hiểu sâu hơn về cơ chế MapReduce.
- Thực hành chạy các job MapReduce mẫu với dữ liệu thực tế hơn.
- Làm quen với việc quản lý input/output trên HDFS.

4.1 Chuẩn bị

- Đảm bảo Hadoop đang chạy (NameNode, DataNode, ResourceManager, NodeManager).
- Có dữ liệu từ Lab 1 và Lab 3.

4.2 Tạo dữ liệu mẫu

4.2.1 Upload file weblogs.txt

Giả sử file này nằm ở thư mục ~/bigdata-labs/datasets/weblogs.txt:

```
hdfs dfs -mkdir -p /user/osboxes/lab4/input

hdfs dfs -put ~/bigdata-labs/datasets/weblogs.txt
/user/osboxes/lab4/input/

hdfs dfs -ls /user/osboxes/lab4/input
```

4.2.2 Kiểm tra dữ liệu

```
hdfs dfs -cat /user/osboxes/lab4/input/weblogs.txt | head -10
```

4.3 Thực hành MapReduce

4.3.1 WordCount trên weblogs.txt

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-examples-*.jar wordcount \
/user/osboxes/lab4/input/weblogs.txt
/user/osboxes/lab4/output_wordcount
```

Xem kết quả:

```
hdfs dfs -cat /user/osboxes/lab4/output_wordcount/part-r-00000
| head -20
```

4.3.2 Count số dòng (LineCount)

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-examples-*.jar linecount \
/user/osboxes/lab4/input/weblogs.txt
/user/osboxes/lab4/output_linecount
```

Xem kết quả:

```
hdfs dfs -cat /user/osboxes/lab4/output_linecount/part-r-00000
```

4.3.3 Tìm tần suất xuất hiện URL (giả lập phân tích log)

Chạy WordCount nhưng trên cột URL (sinh viên cần tiền xử lý log bằng awk hoặc cut trước):

Ví dụ: trích cột 7 (URL) từ file weblogs.txt và upload lại:

```
hdfs dfs -cat /user/osboxes/lab4/input/weblogs.txt | awk
'{print $7}' > urls.txt
hdfs dfs -put urls.txt /user/osboxes/lab4/input/urls.txt
```

Chạy WordCount trên urls.txt:

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-examples-*.jar wordcount \
/user/osboxes/lab4/input/urls.txt
/user/osboxes/lab4/output_urls
```

Xem 10 URL được truy cập nhiều nhất:

```
hdfs dfs -cat /user/osboxes/lab4/output_urls/part-r-00000 |
sort -k2 -nr | head -10
```

4.3.4 So sánh nhiều job

- Thử chạy WordCount với students.txt và weblogs.txt để so sánh thời gian.
- Nhận xét sự khác biệt về độ lớn dữ liệu và thời gian chạy.

4.4 Bài tập mở rộng

1. Chạy grep trên HDFS để tìm các dòng chứa từ khóa "error" trong weblogs.txt.

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-examples-*.jar grep \
/user/osboxes/lab4/input/weblogs.txt
/user/osboxes/lab4/output_grep "error"
```

Xem kết quả:

```
hdfs dfs -cat /user/osboxes/lab4/output_grep/part-r-00000 |
head -20
```

2. Thử chạy song song 2 job khác nhau và quan sát ResourceManager UI (nếu bật web).

4.5 Tài liệu tham khảo

- MapReduce Programming Guide: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
- Java MapReduce API reference: https://hadoop.apache.org/docs/stable/api/

LAB 5 MINI PROJECT

Mục tiêu

- Viết chương trình MapReduce bằng Java.
- Compile, đóng gói JAR và chạy trên Hadoop pseudo-distributed.
- Thực hành xử lý dữ liệu weblogs (weblogs.txt).

Yêu cầu nộp bài: **ảnh chụp màn hình code, lệnh compile, chạy, và kết quả.**

5.1 Chuẩn bị môi trường

Giả sử Hadoop đã cài và chạy ở chế độ pseudo-distributed (Lab 2–4).

Tạo thư mục cho lab:

```
mkdir -p ~/bigdata-labs/lab5/src
cd ~/bigdata-labs/lab5/src
```

Copy dataset từ Lab 1:

```
hdfs dfs -mkdir -p /user/osboxes/lab5/input
hdfs dfs -put ~/bigdata-labs/lab1/weblogs.txt
/user/osboxes/lab5/input/
```

5.2 Viết chương trình WordCount (Java)

Tạo file WordCount.java:

```
import java.io.IOException;
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.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;

public class WordCount {

  public static class TokenizerMapper
```

```
       extends Mapper<Object, Text, Text, IntWritable>{

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

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException
{
      StringTokenizer itr = new
StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public 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, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.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);
  }
```

```
}
```

5.3 Compile và đóng gói

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
```

5.4 Chạy MapReduce Job

Trên Hadoop:

```
hadoop jar wc.jar WordCount /user/osboxes/lab5/input
/user/osboxes/lab5/output_wc
```

Xem kết quả:

```
hdfs dfs -ls /user/osboxes/lab5/output_wc
hdfs dfs -cat /user/osboxes/lab5/output_wc/part-r-00000 | head
```

5.5 Bài tập thực hành

1. **Chạy WordCount** trên students.txt (Lab 1) thay vì weblogs.
    - Upload lên HDFS rồi chạy lại job.
2. **Chỉnh sửa code WordCount** thành **LineCount** (đếm số dòng thay vì số từ).
    - Gợi ý: thay vì itr.hasMoreTokens(), mỗi dòng chỉ +1.
3. **So sánh tốc độ**: chạy job trên dataset nhỏ (students.txt) vs dataset lớn (weblogs.txt).

5.6 Nộp bài

- Ảnh chụp:
    - Source code WordCount.java.

- Lệnh compile và đóng gói jar.
- Lệnh chạy job + output trên HDFS.

- File ghi chú nhận xét: so sánh WordCount vs LineCount, local vs HDFS.

5.7 MapReduce nâng cao – SortByCount

5.7.1 Chuẩn bị

Sau khi chạy WordCount xong, bạn đã có:

```
/user/osboxes/lab5/output_wc/part-r-00000
```

Trong đó mỗi dòng là:

```
word    count
```

Upload kết quả WordCount làm input:

```
hdfs dfs -mkdir -p /user/osboxes/lab5/input_sort
hdfs dfs -cp /user/osboxes/lab5/output_wc/part-r-00000
/user/osboxes/lab5/input_sort/
```

5.7.2 Viết chương trình SortByCount.java

Tạo file SortByCount.java trong ~/bigdata-labs/lab5/src:

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
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;

public class SortByCount {

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

    private IntWritable count = new IntWritable();
    private Text word = new Text();
```

```
    public void map(Object key, Text value, Context context)
        throws IOException, InterruptedException {
      String[] parts = value.toString().split("\\t");
      if (parts.length == 2) {
        word.set(parts[0]);
        count.set(Integer.parseInt(parts[1]));
        // đảo key-value: count -> word
        context.write(count, word);
      }
    }
  }

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

    public void reduce(IntWritable key, Iterable<Text> values,
Context context)
        throws IOException, InterruptedException {
      for (Text val : values) {
        context.write(key, val);
      }
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "sort by count");

    job.setJarByClass(SortByCount.class);
    job.setMapperClass(SwapMapper.class);
    job.setReducerClass(DescReducer.class);

    job.setMapOutputKeyClass(IntWritable.class);
    job.setMapOutputValueClass(Text.class);

    job.setOutputKeyClass(IntWritable.class);
    job.setOutputValueClass(Text.class);

    // sort giảm dần

job.setSortComparatorClass(org.apache.hadoop.io.WritableCompara
tor.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

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

5.7.3 Compile và chạy

Compile và đóng gói cùng với WordCount:

```
cd ~/bigdata-labs/lab5/src
javac -classpath $(hadoop classpath) -d ../build
SortByCount.java
cd ~/bigdata-labs/lab5/build
jar cf midterm.jar *.class
```

Chạy **SortByCount**:

```
hadoop jar midterm.jar SortByCount
/user/osboxes/lab5/input_sort /user/osboxes/lab5/output_sort
```

Xem kết quả:

```
hdfs dfs -cat /user/osboxes/lab5/output_sort/part-r-00000 |
head
```

5.8 Bài tập mở rộng

1. Chạy **Job 1 (WordCount)** trên students.txt, sau đó chạy **Job 2 (SortByCount)**.
2. Tìm **top 5 từ xuất hiện nhiều nhất** trong weblogs.txt.
3. Suy nghĩ: nếu dữ liệu lớn hơn (GB), HDFS và MapReduce sẽ giúp ích thế nào so với sort | uniq -c trên Linux?

5.8 Tài liệu tham khảo

- Link dữ liệu mẫu: Download

LAB 6 YARN & Job Management

Mục tiêu

- Hiểu vai trò của YARN trong Hadoop.
- Quan sát daemon ResourceManager / NodeManager.
- Chạy và giám sát job MapReduce trên YARN.
- Tùy chọn: chạy trên fully-distributed cluster nếu có nhiều node.

Điều kiện trước khi thực hiện

- Đã hoàn thành Lab 2 (pseudo-distributed setup) và Lab 3 (HDFS basics).
- Có thư mục dữ liệu mẫu (ví dụ: ~/bigdata-labs/lab1/weblogs.txt, students.txt).
- SSH vào máy ảo hoặc node master trong cluster fully-distributed.

6.1 Khởi động YARN

Pseudo-distributed (single-node):

```
# Start HDFS nếu chưa chạy
start-dfs.sh
# Start YARN
start-yarn.sh
# Kiểm tra daemon
jps
# Kỳ vọng: NameNode, DataNode, ResourceManager, NodeManager
```

Fully-distributed (nhiều node):

- SSH vào node master.
- Start HDFS & YARN:

```
start-dfs.sh
start-yarn.sh
jps
# Master node: NameNode, ResourceManager
# Slave nodes: DataNode, NodeManager
```

- Kiểm tra giao diện web:
  - ResourceManager: http://<master-node>:8088/
  - NameNode: http://<master-node>:9870/

6.2 Tạo thư mục HDFS và upload dữ liệu

```
# Tạo thư mục lab
hdfs dfs -mkdir -p /user/<username>/lab6/input

# Upload dữ liệu
hdfs dfs -put ~/bigdata-labs/lab1/weblogs.txt
/user/<username>/lab6/input/
hdfs dfs -put ~/bigdata-labs/lab1/students.txt
/user/<username>/lab6/input/

# Kiểm tra
hdfs dfs -ls /user/<username>/lab6/input
hdfs dfs -cat /user/<username>/lab6/input/weblogs.txt | head -5
```

Với Fully-distributed: các file sẽ được phân mảnh trên nhiều DataNode, YARN sẽ quản lý phân phối task MapReduce.

6.3 Chạy ví dụ MapReduce trên YARN

Grep job:

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-examples-3.3.1.jar grep \
/user/<username>/lab6/input /user/<username>/lab6/output
'dfs[a-z.]+'
```

WordCount trên students.txt:

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-examples-3.3.1.jar wordcount \
/user/<username>/lab6/input/students.txt
/user/<username>/lab6/output_wc
```

- Xem kết quả:

```
hdfs dfs -cat /user/<username>/lab6/output_wc/part-r-00000 |
head
```

- **Nhận xét:** so sánh tốc độ chạy job trên local filesystem vs HDFS.

Với **Fully-distributed:** job sẽ chia map task trên nhiều NodeManager, output vẫn hợp nhất tại HDFS.

6.4 Giám sát job qua ResourceManager

- Truy cập web UI http://<master-node>:8088/
- Kiểm tra trạng thái job, số map/reduce task, logs.
- Download logs nếu cần để debug job.

6.5 Một số lệnh YARN hữu dụng

```
# Liệt kê job đang chạy
yarn application -list

# Kiểm tra trạng thái job
yarn application -status <application_id>

# Dừng YARN (pseudo hoặc fully-distributed)
stop-yarn.sh
```

6.6 Bài tập thực hành

1. Tạo thư mục /user/<username>/lab6/logs trên HDFS và upload file log_time.txt từ Lab 1.
2. Hiển thị 5 dòng đầu bằng:

```
hdfs dfs -cat /user/<username>/lab6/logs/log_time.txt | head -5
```

3. Chạy WordCount trên students.txt và ghi nhận output.
4. So sánh thời gian chạy WordCount:
    - Trên local filesystem (standalone)
    - Trên HDFS + YARN pseudo-distributed
    - Tùy chọn: fully-distributed nếu có cluster nhiều node.
5. Sử dụng hdfs dfs -du /user/<username>/lab6/input để xem dung lượng file trên HDFS.

Ghi chú chung:

- Nếu sinh viên chỉ có pseudo-distributed: chạy trên 1 node vẫn đủ để thực hành.
- Nếu sinh viên có cluster fully-distributed: sẽ thấy YARN phân phối map/reduce task trên nhiều node, chuẩn bị tốt cho Lab 7–10.

6.7 Tài liệu tham khảo

- Apache Spark official: https://spark.apache.org/docs/latest/

- PySpark API: https://spark.apache.org/docs/latest/api/python/
- Scala Spark API: https://spark.apache.org/docs/latest/api/scala/

LAB 7 HIVE & SQL trên Hadoop

Mục tiêu

- Cài đặt Hive trên Hadoop cluster (pseudo hoặc fully-distributed).
- Biết cách tạo database, table, load dữ liệu từ HDFS.
- Thực hiện các truy vấn SQL cơ bản trên dữ liệu lớn.
- Chuẩn bị dữ liệu và thao tác cho Lab 8 (tương tác với Spark/Hive).

Điều kiện trước khi thực hiện

- Hoàn thành Lab 2–6 (HDFS + YARN + MapReduce).
- Có dữ liệu mẫu: students.txt, weblogs.txt, log_time.txt trong HDFS.

7.1 Cài đặt Hive

```
# Tải Hive (ví dụ Hive 3.1.2)
wget https://archive.apache.org/dist/hive/hive-3.1.2/apache-
hive-3.1.2-bin.tar.gz
tar -xzf apache-hive-3.1.2-bin.tar.gz
mv apache-hive-3.1.2-bin ~/hive-3.1.2

# Thiết lập biến môi trường
echo 'export HIVE_HOME=~/hive-3.1.2' >> ~/.bashrc
echo 'export PATH=$PATH:$HIVE_HOME/bin' >> ~/.bashrc
source ~/.bashrc

# Khởi tạo thư mục warehouse trên HDFS
hdfs dfs -mkdir -p /user/<username>/warehouse
hdfs dfs -chmod g+w /user/<username>/warehouse
```

Ghi chú: Pseudo-distributed và fully-distributed đều dùng cùng biến HIVE_HOME, HDFS làm warehouse.

7.2 Khởi chạy Hive CLI

```
hive
```

- Nếu lần đầu, Hive tạo metastore embedded (derby).
- Chú ý: fully-distributed có thể dùng MySQL/PostgreSQL làm metastore.

7.3 Tạo Database và Table

Tất cả lệnh SQL dưới đây nhập trực tiếp trong Hive shell

```
-- Tạo database riêng cho lab
CREATE DATABASE IF NOT EXISTS lab7;
USE lab7;

-- Tạo table students (tương ứng với students.txt)
CREATE TABLE students (
    id INT,
    name STRING,
    score INT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

-- Tạo table logs (tương ứng weblogs.txt)
CREATE TABLE weblogs (
    datetime STRING,
    user STRING,
    url STRING,
    status INT
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

7.4 Load dữ liệu từ HDFS

```
-- Copy dữ liệu từ HDFS vào table
LOAD DATA INPATH '/user/<username>/lab1/students.txt' INTO
TABLE students;
LOAD DATA INPATH '/user/<username>/lab1/weblogs.txt' INTO TABLE
weblogs;

-- Kiểm tra dữ liệu
SELECT * FROM students LIMIT 5;
SELECT * FROM weblogs LIMIT 5;
```

Ghi chú: Nếu file trên HDFS chưa có, dùng hdfs dfs -put ~/bigdata-labs/lab1/... để upload.

7.5 Thực hành truy vấn SQL cơ bản

1. Thống kê học sinh theo điểm >= 8

```
SELECT * FROM students WHERE score >= 8;
```

2. Sắp xếp học sinh theo điểm giảm dần

```
SELECT * FROM students ORDER BY score DESC;
```

3. Đếm số request theo trạng thái từ weblogs

```
SELECT status, COUNT(*) as count
FROM weblogs
GROUP BY status;
```

4. Lọc URL chứa từ khóa "login"

```
SELECT * FROM weblogs WHERE url LIKE '%login%';
```

7.6 Một số lệnh Hive hữu dụng

```
-- Kiểm tra database hiện tại
SHOW DATABASES;
SHOW TABLES;

-- Xem schema table
DESCRIBE students;

-- Xem dữ liệu
SELECT * FROM students LIMIT 10;
```

7.7 Bài tập mở rộng

1. Tạo thêm table log_time từ Lab 1 và load dữ liệu vào Hive.
2. Viết truy vấn: đếm số dòng log trong log_time theo từng ngày.
3. Thực hiện JOIN giữa students và log_time nếu có cột phù hợp (ví dụ giả lập student_id).
4. Thử chạy cùng trên pseudo-distributed và fully-distributed, ghi nhận thời gian.
5. Xuất kết quả ra file CSV bằng Hive:

```
INSERT OVERWRITE LOCAL DIRECTORY '/home/<username>/output_students'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
SELECT * FROM students;
```

Ghi chú:

- Lab 7 hướng tới quản lý dữ liệu dạng bảng trên HDFS, kết hợp SQL trên Hive, chuẩn bị cho Lab 8–9 (Spark, trực quan hóa, project cuối).

- Sinh viên có thể dễ dàng chuyển dữ liệu từ MapReduce sang Hive hoặc dùng Hive làm bước trung gian cho dashboard trực quan.

7.8 Tài liệu tham khảo

- Apache Hive official: https://cwiki.apache.org/confluence/display/Hive/Home
- HiveQL reference: https://cwiki.apache.org/confluence/display/Hive/LanguageManual

LAB 8 SPARK SQL & DATAFRAMES

Mục tiêu

- Cài đặt và chạy **Apache Spark** trên pseudo-distributed hoặc fully-distributed Hadoop cluster.
- Tương tác với dữ liệu **Hive table** bằng Spark SQL.
- Thực hiện **phân tích dữ liệu cơ bản**.
- Chuẩn bị dữ liệu trực quan hóa cho Lab 9.

Điều kiện trước khi thực hiện

- Hoàn thành Lab 2–7.
- Có dữ liệu trong HDFS: students.txt, weblogs.txt, log_time.txt.
- Hive đã cài và table đã load dữ liệu.

8.1 Cài đặt Spark

```
# Tải Spark (ví dụ Spark 3.4.1 với Hadoop 3)
wget https://archive.apache.org/dist/spark/spark-3.4.1/spark-
3.4.1-bin-hadoop3.tgz
tar -xzf spark-3.4.1-bin-hadoop3.tgz
mv spark-3.4.1-bin-hadoop3 ~/spark-3.4.1

# Thiết lập biến môi trường
echo 'export SPARK_HOME=~/spark-3.4.1' >> ~/.bashrc
echo 'export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin' >>
~/.bashrc
source ~/.bashrc

# Kiểm tra cài đặt
spark-shell --version
```

Ghi chú: Pseudo-distributed và fully-distributed dùng cùng SPARK_HOME; nếu fully-distributed, đảm bảo Spark biết HADOOP_CONF_DIR.

8.2 Khởi động Spark Shell

```
# Spark Shell Scala
spark-shell

# Hoặc PySpark
pyspark
```

8.3 Spark SQL với Hive

Bạn có thể sử dụng **Scala** hoặc **PySpark** để thực hành Spark SQL với Hive. Chọn một ngôn ngữ và làm toàn bộ phần Lab 8 bằng ngôn ngữ đó.

Với Scala:

```
// Scala example
val spark = SparkSession.builder()
    .appName("Lab8")
    .enableHiveSupport()
    .getOrCreate()

spark.sql("USE lab7")

// Load table students
val studentsDF = spark.sql("SELECT * FROM students")
studentsDF.show(5)

// Thống kê điểm trung bình
studentsDF.agg(avg("score")).show()
```

Với PySpark:

```
# PySpark example
from pyspark.sql import SparkSession
spark =
SparkSession.builder.appName("Lab8").enableHiveSupport().getOrC
reate()

studentsDF = spark.sql("SELECT * FROM students")
studentsDF.show(5)

# Thống kê điểm trung bình
studentsDF.groupBy().avg("score").show()
```

8.4 Phân tích dữ liệu từ weblogs

Với Scala:

```
// Số request theo status code
val logsDF = spark.sql("SELECT * FROM weblogs")
logsDF.groupBy("status").count().show()

// Lọc URL chứa 'login'
logsDF.filter(logsDF("url").contains("login")).show(5)
```

Với PySpark

```
logsDF.groupBy("status").count().show()
logsDF.filter(logsDF.url.contains("login")).show(5)
```

8.5 Thao tác cơ bản với DataFrame

1. Thêm cột tính điểm letter grade từ score

Với Scala:

```
import org.apache.spark.sql.functions._
val studentsGradeDF = studentsDF.withColumn("grade",
    when(col("score")>=9,"A")
    .when(col("score")>=8,"B")
    .when(col("score")>=7,"C")
    .otherwise("D"))
studentsGradeDF.show()
```

Với Python:

```
from pyspark.sql.functions import when
studentsDF = studentsDF.withColumn("grade",
    when(studentsDF.score >= 9, "A")
    .when(studentsDF.score >= 8, "B")
    .when(studentsDF.score >= 7, "C")
    .otherwise("D"))
studentsDF.show()
```

2. Lưu DataFrame ra HDFS hoặc local filesystem:

Với Scala:

```
studentsGradeDF.write.mode("overwrite").csv("/user/<username>/l
ab8/students_grade")
```

Với Python:

```
studentsDF.write.mode("overwrite").csv("/user/<username>/lab8/s
tudents_grade")
```

8.6 Chuẩn bị trực quan hóa (Lab 9)

- Xuất dữ liệu Spark DataFrame ra CSV.
- Chuẩn bị dữ liệu students_grade và weblogs_stats cho plot.

- Ghi chú: Lab 9 có thể dùng **Python matplotlib / seaborn / plotly** đọc từ CSV.

8.7 Bài tập mở rộng

1. Tạo DataFrame từ students và weblogs.
2. Thống kê: số sinh viên theo letter grade, số request theo status code.
3. Lọc: học sinh có điểm >= 8; URL chứa "login".
4. Lưu kết quả phân tích ra HDFS để Lab 9 dùng.
5. So sánh hiệu năng pseudo-distributed vs fully-distributed (nếu chọn).

Ghi chú:

- Lab 8 kết hợp Spark SQL + Hive, chuẩn bị dữ liệu để visualization ở Lab 9.
- Giúp sinh viên thực hành data munging, aggregation, filter, join, bước đầu làm data pipeline thực tế.

8.8 Tài liệu tham khảo

- Spark SQL, DataFrame: https://spark.apache.org/docs/latest/sql-programming-guide.html
- Spark & Hive integration: https://spark.apache.org/docs/latest/sql-data-sources-hive-tables.html

LAB 9 Data Visualization Dashboard

Mục tiêu

- Tận dụng dữ liệu từ Lab 8 (Spark + Hive).
- Xuất dữ liệu ra CSV / JSON để trực quan hóa.
- Sử dụng **Python + matplotlib / seaborn / plotly** để tạo dashboard.
- Chuẩn bị tiền đề cho final project (Lab 10).

Điều kiện trước khi thực hiện

- Hoàn thành Lab 8, dữ liệu students_grade và weblogs_stats đã lưu trong HDFS.
- Có Python 3 và thư viện pandas, matplotlib, seaborn, plotly:

```
sudo apt update
sudo apt install -y python3-pip
pip3 install pandas matplotlib seaborn plotly
```

9.1 Lấy dữ liệu từ HDFS

```
# Copy từ HDFS về local
hdfs dfs -get /user/<username>/lab8/students_grade ~/bigdata-
labs/lab9/
hdfs dfs -get /user/<username>/lab8/weblogs_stats ~/bigdata-
labs/lab9/
```

9.2 Tạo DataFrame Python

- Tạo file analysis.py trong VM:

```
cd ~/bigdata-labs/lab9
nano analysis.py
```

- Nội dung gợi ý:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load dữ liệu students
students_df = pd.read_csv('students_grade/part-00000',
header=None, names=['name','score','grade'])
print("Students data:")
```

```
print(students_df.head())

# Load dữ liệu weblogs
weblogs_df = pd.read_csv('weblogs_stats/part-00000',
header=None, names=['ip','url','status'])
print("\nWeblogs data:")
print(weblogs_df.head())

# Thống kê điểm trung bình
print("\nAverage score:", students_df['score'].mean())

# Thống kê số lượng status code
status_counts = weblogs_df['status'].value_counts()
print("\nStatus counts:\n", status_counts)

# Vẽ biểu đồ điểm
sns.histplot(students_df['score'], bins=10)
plt.title("Score distribution")
plt.show()
```

- **Chạy script**

```
python3 analysis.py
```

9.3 Thống kê và trực quan hóa cơ bản

9.3.1 Students Grade Distribution

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.countplot(x='grade', data=students_df)
plt.title("Distribution of Student Grades")
plt.show()
```

9.3.2 Weblog Status Code Count

```
status_counts = weblogs_df['status'].value_counts()
status_counts.plot(kind='bar', title="Weblog Status Code
Count")
plt.show()
```

9.3.3 Top 10 URL truy cập nhiều nhất

```
top_urls = weblogs_df['url'].value_counts().head(10)
```

```
top_urls.plot(kind='barh', title="Top 10 URLs")
plt.show()
```

9.3.4 Dashboard nâng cao (Plotly)

```
import plotly.express as px

# Pie chart phân bố letter grade
fig = px.pie(students_df, names='grade', title='Grade
Distribution')
fig.show()

# Bar chart trạng thái weblog
fig2 = px.bar(status_counts, x=status_counts.index,
y=status_counts.values, title='Weblog Status Counts')
fig2.show()
```

Ghi chú: Plotly cho dashboard tương tác, sinh viên có thể export HTML hoặc dùng Jupyter Notebook.

9.4 Sử dụng Plotly và Jupyter Notebook để tạo dashboard tương tác

Mục tiêu: chạy code trực quan hóa tương tác, dễ chỉnh sửa, xuất HTML.

9.4.1 Cài đặt Jupyter Notebook

Trên VM (Ubuntu Server):

```
sudo apt update
sudo apt install -y python3-pip
pip3 install notebook plotly pandas matplotlib seaborn
```

9.4.2 Khởi chạy Jupyter Notebook trên VM

Chạy:

```
jupyter notebook --no-browser --port=8888
```

- Lệnh trên sẽ in ra **URL truy cập có token**, ví dụ http://localhost:8888/?token=abc123.
- Nếu dùng **NAT + port-forwarding**, forward port 8888 từ host → VM. Ví dụ trong VirtualBox: Host Port 8888 → Guest Port 8888.

9.4.3 Truy cập từ máy thật

- Mở trình duyệt trên máy thật, truy cập:

```
http://<IP_VM>:8888/?token=<token>
```

- Hoặc nếu port-forwarding:

```
http://127.0.0.1:8888/?token=<token>
```

9.4.4 Tạo Notebook và chạy code

1. Trong Jupyter, tạo Notebook mới với kernel Python 3.
2. Sao chép toàn bộ code từ 9.3.1 → 9.3.4 vào các cell.
3. Chạy từng cell để kiểm tra biểu đồ và dashboard.
4. Khi hoàn tất, có thể:
    - Export HTML: File → Download as → HTML (.html)
    - Lưu notebook để nộp hoặc chỉnh sửa sau.

9.4.5 Lợi ích

- Có thể chạy lại, chỉnh sửa nhanh, không lo gõ nhầm như trong terminal.
- Dashboard Plotly tương tác trực tiếp: zoom, hover, lọc dữ liệu.
- Có thể kết hợp với dữ liệu HDFS đã lấy từ Lab 8 → Lab 9.

9.5 Bài tập mở rộng

1. Tạo dashboard gồm:
    - Biểu đồ phân bố letter grade
    - Số lượng request theo status code
    - Top 10 URL truy cập nhiều nhất
2. Lưu dashboard dưới dạng PNG / HTML.
3. Tự chọn thêm biểu đồ: ví dụ số request theo giờ, top IP, tỉ lệ học sinh đạt >=8.
4. Ghi chú so sánh dữ liệu trực quan hóa trên pseudo-distributed vs fully-distributed (nếu có).

Chuẩn bị cho Lab 10 (Final Project)

- Dashboard có thể tích hợp vào final project.
- Dữ liệu Hadoop + Spark + Hive + Visualization = pipeline hoàn chỉnh.

9.6 Tài liệu tham khảo

- Pandas: https://pandas.pydata.org/docs/
- Matplotlib: https://matplotlib.org/stable/contents.html
- Seaborn: https://seaborn.pydata.org/
- Plotly: https://plotly.com/python/

LAB 10 FINAL PROJECT: Big Data Pipeline & Dashboard

Mục tiêu

- Tích hợp các kỹ năng: Linux, SSH, HDFS, Hadoop MapReduce, Spark, Hive.
- Xây dựng một **pipeline dữ liệu thực tế** từ ingest → process → analyze → visualize.
- Trình bày kết quả dưới dạng **dashboard** hoặc báo cáo tương tác.

Yêu cầu chuẩn bị

- Hoàn thành Lab 1–9.
- VM Ubuntu với Hadoop + Spark + Hive.
- Dữ liệu mẫu: weblogs.txt, students.txt, student_scores.csv hoặc dữ liệu do sinh viên tự chuẩn bị.

10.1 Chọn đề tài (3 lựa chọn gợi ý)

1. **Phân tích weblogs website**
   - Đếm request theo IP / URL / Status code.
   - Phân tích giờ cao điểm, top URL, lỗi phổ biến.
   - Trực quan hóa dạng bar chart, pie chart, heatmap.
2. **Điểm sinh viên & học tập**
   - Tính trung bình, letter grade, top/bottom performers.
   - So sánh phân bố điểm giữa các lớp / môn.
   - Dashboard tương tác (Plotly / Seaborn).
3. **Dữ liệu kết hợp (Web + Student)**
   - Kết hợp dữ liệu người dùng và tương tác học tập online.
   - Tạo insight: ví dụ ai truy cập nhiều trang tài liệu, ảnh hưởng đến điểm số.

10.2 Workflow đề xuất

Bước 1 – HDFS ingest

```
hdfs dfs -mkdir -p /user/<username>/project/input
hdfs dfs -put ~/bigdata-labs/lab1/weblogs.txt
/user/<username>/project/input/
hdfs dfs -put ~/bigdata-labs/lab1/students.txt
/user/<username>/project/input/
```

Bước 2 – Data processing

- **MapReduce (Hadoop)**

```
hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-
mapreduce-examples-*.jar wordcount
/user/<username>/project/input/weblogs.txt
/user/<username>/project/output/weblogs_wc
```

- **Spark** (Python/Scala)

```
from pyspark.sql import SparkSession

spark =
SparkSession.builder.appName("FinalProject").getOrCreate()
df =
spark.read.text("/user/<username>/project/input/weblogs.txt")
# ví dụ: đếm request theo IP
df.createOrReplaceTempView("weblogs")
spark.sql("SELECT split(value,' ')[0] as ip, count(*) as cnt
FROM weblogs GROUP BY ip").show()
```

- **Hive (tùy chọn)**

```
CREATE DATABASE IF NOT EXISTS project_db;
CREATE EXTERNAL TABLE IF NOT EXISTS project_db.weblogs(ip
STRING, url STRING, status INT) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ' ';
LOAD DATA INPATH '/user/<username>/project/input/weblogs.txt'
INTO TABLE project_db.weblogs;
SELECT status, count(*) FROM project_db.weblogs GROUP BY
status;
```

Bước 3 – Xuất dữ liệu cho visualization

```
hdfs dfs -get /user/<username>/project/output/weblogs_wc
~/bigdata-labs/lab10/
```

Bước 4 – Dashboard / Visualization

- Python: pandas + matplotlib / seaborn / plotly.
- Trực quan hóa các insight: top IP, status code distribution, grade distribution, v.v.

10.3 Bài nộp

1. Pipeline chạy thành công trên Hadoop/Spark/Hive
2. Dashboard / Biểu đồ trực quan
3. Báo cáo ghi chú:
    - Mô tả dữ liệu, pipeline, các bước thực hiện.
    - Kết quả và nhận xét.
    - So sánh tốc độ/hiệu quả trên pseudo-distributed vs fully-distributed (nếu có).

Lưu ý:

- Có thể làm project nhỏ gọn hoặc mở rộng, tùy khả năng.
- Khuyến khích sinh viên tạo data set của riêng mình hoặc dùng data mở.
- Dashboard tương tác là điểm cộng mạnh.

10.4 Tài liệu tham khảo

- Tùy theo đề tài: Spark Streaming, Hive, Visualization, Dashboard, MLlib…
- Gợi ý thêm tutorial tổng hợp:
    - https://www.datacamp.com/
    - https://www.kaggle.com/learn/

TÀI LIỆU THAM KHẢO

Hadoop & MapReduce

1. Hadoop Official Documentation: https://hadoop.apache.org/docs/
2. MapReduce Tutorial: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

Spark & Spark SQL

3. Spark Official: https://spark.apache.org/docs/latest/
4. Spark SQL Guide: https://spark.apache.org/docs/latest/sql-programming-guide.html
5. PySpark API: https://spark.apache.org/docs/latest/api/python/

Hive & HiveQL

6. Hive Manual: https://cwiki.apache.org/confluence/display/Hive/Home
7. HiveQL Reference: https://cwiki.apache.org/confluence/display/Hive/LanguageManual

Python Analytics & Visualization

8. Pandas: https://pandas.pydata.org/docs/
9. Matplotlib & Seaborn: https://matplotlib.org/stable/contents.html, https://seaborn.pydata.org/
10. Plotly: https://plotly.com/python/

Cloud & Big Data Optional

11. AWS EMR tutorial: https://aws.amazon.com/emr/getting-started/
12. Google Cloud Dataproc: https://cloud.google.com/dataproc/docs