TRƯỜNG ĐH CÔNG NGHỆ THÔNG TIN TP.HCM
Khoa Hệ Thống Thông Tin
Chương trình đào tạo: Kỹ sư Hệ thống Thông tin

Bài Thực hành số 1

Tên môn học: ***Dữ liệu lớn***

I. Tóm tắt bài thực hành

1. Yêu cầu lý thuyết

Sinh viên đã được trang bị kiến thức:

- Các hệ điều hành Linux
- Sử dụng thành thạo hệ điều hành Ubuntu
- Mô hình lập trình MapReduce
- Ngôn ngữ lập trình Python
- …

2. Nội dung

❖ **Ôn tập lại những kiến thức cần thiết**

- Cài đặt hệ điều hành Ubuntu.
- Thành thạo các thao tác, câu lệnh trong Ubuntu.
- Xem lại mô hình lập trình MapReduce đã được học ở lớp lý thuyết.
- Cài đặt một IDE để lập trình trên ngôn ngữ Python.

❖ **Cài đặt framework Apache Hadoop**

❖ **Cài đặt framework Apache Spark**

3. Kết quả cần đạt

✓ Sinh viên cài đặt được hệ điều hành Ubuntu trên máy tính cá nhân (có thể dùng máy ảo hoặc song song với hệ điều hành đang có).

✓ Sử dụng được ngôn ngữ Python để lập trình các thuật toán được yêu cầu ở các buổi học sau.

✓ Hiểu rõ quá trình cài đặt framework Apache Hadoop.

✓ Hiểu rõ quá trình cài đặt framework Apache Spark.

✓ Lập nhóm và cài đặt thành công framework Apache Hadoop và Apache Spark trên máy các thành viên trong nhóm, tạo thành cụm tính toán phân tán.

II. Ôn tập lại những kiến thức đã học

Sinh viên tiến hành cài đặt hệ điều hành Ubuntu vào máy tính cá nhân hoặc máy ảo để thực hành.

Tham khảo tài liệu của các môn học trước, cũng như tìm hiểu thêm trên internet để biết thêm về hệ điều hành Ubuntu, cũng như cú pháp của các câu lệnh sử dụng trong môi trường dòng lệnh của Ubuntu. Sử dụng thường xuyên để có thể thành thạo các thao tác trong hệ điều hành này.

Sử dụng tài liệu lý thuyết của môn học để ôn tập củng cố kiến thức về mô hình lập trình MapReduce.

Download và cài đặt một IDE hỗ trợ lập trình trên ngôn ngữ Python (ví dụ: Jupyter Notebook), tìm hiểu thêm về ngôn ngữ lập trình Python trên internet, cũng như nắm rõ cấu trúc một chương trình, cú pháp các câu lệnh trong ngôn ngữ Python. Tìm hiểu để sử dụng một cách thuần thục IDE để lập trình phần mềm.

III. Cài đặt framework Apache Hadoop

Sinh viên tìm hiểu việc cài đặt framework Apache Hadoop qua sự hướng dẫn của giảng viên thực hành:

Apache Hadoop có thể được cài đặt trên **2000** máy chạy hệ điều hành Linux. **Có thể** chạy được trên **Windows**. Tuy nhiên, hướng dẫn sau (các lệnh minh họa) sẽ thực hiện trên môi trường hệ điều hành Ubuntu.

1. Hướng dẫn cài đặt trên hệ điều hành Ubuntu:

Yêu cầu phần mềm:

- `JDK` (khuyến khích là `Oracle JDK`) phiên bản **11**. Có thể tham khảo cách cài đặt trên internet.
- Gói phần mềm `ssh` cho Linux, được cài đặt bằng câu lệnh trong Terminal như sau: `sudo apt install ssh`
- Gói phần mềm `Apache Hadoop` phiên bản mới nhất được tải về từ đường dẫn: `https://hadoop.apache.org/releases.html`
  Lưu ý: Bản `binary` là bản đã được biên dịch, nên tải bản này, thay vì `source` nguồn, sau khi tải về cần phải biên dịch rất phức tạp.

Các bước cài đặt cơ bản:

a) Chuẩn bị:

- Giải nén tập tin `hadoop-x.x.x.tar.gz`, đặt ở thư mục tùy ý. Trong thư mục con sau khi giải nén, chỉnh sửa tập tin `etc/hadoop/hadoop-env.sh` để định nghĩa biến môi trường Java như sau:

```
# set to the root of your Java installation
export JAVA_HOME=đường dẫn đến thư mục cài đặt Java
```

  Trong đó *đường dẫn đến thư mục cài đặt Java* có thể được xác định bằng cách thực thi câu lệnh sau trong cửa sổ dòng lệnh Terminal:

```
$(dirname $(dirname $(readlink -f $(which javac))))
```

- Trong thư mục hadoop, chạy lệnh sau trong cửa sổ dòng lệnh Terminal để kiểm tra:

```
$ bin/hadoop
```

  Câu lệnh trên sẽ hiển thị toàn bộ hướng dẫn sử dụng cho câu lệnh `hadoop`.
- Sau các bước chuẩn bị trên, hadoop đã sẵn sàng để được cài đặt ở một trong 3 chế độ sau.
  Lưu ý: chỉ được chọn một trong ba chế độ hoạt động bên dưới để cài đặt.

b) Hoạt động độc lập:

Mặc định, Hadoop được tinh chỉnh để hoạt động ở chế độ độc lập (không phân tán) trên một tiến trình Java đơn lẻ (single Java process). Chế độ này thường được dùng để tìm và khắc phục lỗi cho phần mềm.

Đoạn lệnh dưới đây sẽ thực thi một ví dụ được lập trình sẵn trong Hadoop. Đầu tiên, tạo thư mục input, sao chép tất cả các tập tin .xml từ trong thư mục chứa các cài đặt của hệ thống Hadoop vào thư mục input. Sau đó, chạy chương trình ví dụ đã được lập trình sẵn, trong đó, chương trình này sẽ đọc nội dung các tập tin, cắt ra và đếm các mẫu theo biểu thức chính quy (regular expression) cho sẵn (trong ví dụ này là `'dfs[a-z.]+'`) và kết quả cuối cùng sẽ được xuất ra trong thư mục output.

```
$ mkdir input
$ cp etc/hadoop/*.xml input
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-
examples-x.x.x.jar grep input output 'dfs[a-z.]+'
$ cat output/*
```

c) Hoạt động theo kiểu giả lập phân tán:

Hadoop có thể hoạt động trên một máy đơn dưới chế độ giả lập phân tán trong đó, mỗi dịch vụ của Hadoop chạy trên mỗi tiến trình Java độc lập nhau (separate Java process).

Cài đặt - thêm các nội dung sau vào các tập tin tương ứng trong thư mục của hadoop:

- Trong tập tin `etc/hadoop/core-site.xml`:

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
    <property>
        <name>hadoop.tmp.dir</name>
        <value>/tmp/hadoop-${user.name}</value>
    </property>
</configuration>
```

Ý nghĩa: gán giá trị `hdfs://localhost:9000` cho biến `fs.defaultFS`. Với biến `fs.defaultFS` là biến chứa địa chỉ của NameNode có dạng `hdfs://host:port/`. Biến `hadoop.tmp.dir` lưu đường dẫn đến thư mục tạm chứa các dữ liệu của NameNode, DataNode, SecondaryNameNode, nên thay đổi do giá trị mặc định

(/tmp/hadoop-${user.name}) được gán là thư mục tạm của hệ điều hành, sẽ bị xóa sau khi khởi động lại máy.

❖ Trong tập tin etc/hadoop/hdfs-site.xml:

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

Ý nghĩa: gán giá trị 1 cho biến dfs.replication. Với biến dfs.replication định nghĩa số nhân bản dữ liệu của hệ thống HDFS. Biến dfs.namenode.name.dir lưu đường dẫn thư mục nơi NameNode chứa tên bảng (fsimage), nên gán giá trị khác (mặc định là: file://${hadoop.tmp.dir}/dfs/name). Biến dfs.datanode.data.dir lưu đường dẫn thư mục nơi DataNode chứa các block dữ liệu, nên gán giá trị khác (mặc định là: file://${hadoop.tmp.dir}/dfs/data). Nếu vẫn muốn thư mục NameNode và DataNode vẫn lưu trong thư mục con trong thư mục {hadoop.tmp.dir} thì không cần phải thay đổi (thêm) các cài đặt này.

❖ Thiết lập và kiểm tra kết nối

Kiểm tra xem đã thiết lập kết nối giữa hai máy mà không cần mật khẩu hay chưa, sử dụng lệnh sau trong Terminal

```
$ ssh localhost
```

Nếu chưa được thiết lập (vẫn bị hỏi mật khẩu đăng nhập) thì thực hiện các câu lệnh sau:

```
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

Sau đó, kiểm tra lại kết nối như trên.

❖ Khởi chạy hệ thống

Định dạng hệ thống tập tin (format the filesystem):

```
$ bin/hdfs namenode -format
```

Khởi động NameNode và DataNode:

```
$ sbin/start-dfs.sh
```

Kiểm tra các thành phần đã khởi chạy bằng câu lệnh (dùng Java Virtual Machine Process Status Tool):

```
$ jps
```

Nếu NameNode và DataNode đã khởi động thành công thì kết quả trả về ngoài `jps` sẽ có thêm `namenode`, `datanode`, `secondarynamenode`.

Hadoop sẽ ghi nhật ký tại đường dẫn thư mục được định nghĩa trong biến `$HADOOP_LOG_DIR`, mặc định là `$HADOOP_HOME/logs`.

Mở thử trang giao diện của NameNode, mặc định tại địa chỉ:

```
http://localhost:9870/
```

Để chạy được một chương trình MapReduce trên hệ thống file HDFS, cần tạo thư mục theo yêu cầu như sau:

```
bin/hdfs dfs -mkdir /user
bin/hdfs dfs -mkdir /user/<username>
```

Trong đó `<username>` là tên người dùng máy tính.

Chạy thử hệ thống. Tạo thư mục trên HDFS để chạy thử một chương trình MapReduce có sẵn:

```
$ bin/hdfs dfs -mkdir input
```

Copy các tập tin đầu vào vào các thư mục đã tạo trên hệ thống tập tin phân tán:

```
$ bin/hdfs dfs -put etc/hadoop/*.xml input
```

Khởi chạy chương trình MapReduce đã được lập trình sẵn:

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

Sao chép tập tin kết quả từ hệ thống tập tin phân tán ra ngoài hệ thống tập tin bình thường và hiển thị kết quả:

```
$ bin/hdfs dfs -get output output
$ cat output/*
```

Hoặc đọc trực tiếp kết quả trên hệ thống tập tin phân tán

```
$ bin/hdfs dfs -cat output/*
```

Khi đã hoàn tất, dừng các thành phần đã chạy ở trên bằng câu lệnh:

```
$ sbin/stop-dfs.sh
```

Để kiểm tra các thành phần đã dừng hay chưa, sử dụng câu lệnh `jps` như trên.

- Cài đặt thành phần quản lý tài nguyên (YARN) trên hệ thống giả lập phân tán

  Trong tập tin `etc/hadoop/mapred-site.xml`:

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapreduce.application.classpath</name>
        <value>$HADOOP_MAPRED_HOME/share/hadoop/mapr
educe/*:$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/l
ib/*</value>
    </property>
</configuration>
```

Ý nghĩa: gán giá trị `yarn` cho biến `mapreduce.framework.name`. Với biến `mapreduce.framework.name` là biến cài đặt framework thực thi MapReduce job, mặc định là `local`. Biến `mapreduce.application.classpath` giữ thông tin đường dẫn đến vị trí lưu trữ các thư viện và chương trình kèm theo trong quá trình thực thi ứng dụng MapReduce, biến này được sử dụng để hỗ trợ cho quá trình triển khai bộ đệm phân tán (Distributed Cache Deploy), giúp nhà quản trị có thể thêm vào các thư viện cần thiết cho ứng dụng hay tùy chỉnh cho ứng dụng có thể chạy trên nhiều phiên bản MapReduce khác nhau.

Trong tập tin `etc/hadoop/yarn-site.xml`:

```
<configuration>
    <property>
      <name>yarn.nodemanager.aux-services</name>
      <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.env-whitelist</name>
        <value>JAVA_HOME,HADOOP_COMMON_HOME,
HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_D
ISTCACHE,HADOOP_YARN_HOME,HADOOP_HOME,PATH,LANG,TZ,H
ADOOP_MAPRED_HOME</value>
    </property>
</configuration>
```

Ý nghĩa: gán giá trị `mapreduce_shuffle` cho biến `yarn.nodemanager.aux-services`. Với biến `yarn.nodemanager.aux-services` là biến khai báo các thành phần phụ của hệ thống, mà ở đây đã tinh chỉnh để sử dụng thành phần xáo trộn (shuffle) dữ liệu. Biến `yarn.nodemanager.env-whitelist`

chứa giá trị là những biến môi trường mà các container (đại diện cho một tài nguyên được cấp phát trong cụm) có thể ghi đè thay vì sử dụng giá trị mặc định của NodeManager.

Khởi động ResourceManager và NodeManager, bằng câu lệnh:

```
$ sbin/start-yarn.sh
```

Dùng Java Virtual Machine Process Status Tool để kiểm tra các thành phần đã khởi chạy. Nếu thành công, kết quả trả về sẽ bao gồm các thành phần `nodemanager` và `resourcemanager`.

Mở thử trang giao diện của ResourceManager, mặc định tại địa chỉ:

```
http://localhost:8088/
```

Chạy một MapReduce job thử nghiệm như ở trên.

Khi đã hoàn tất, tắt các thành phần đã chạy bằng câu lệnh:

```
$ sbin/stop-yarn.sh
```

Để kiểm tra các thành phần đã dừng hay chưa, sử dụng câu lệnh `jps` như trên.

d) Hệ thống phân tán thực

Tất cả các máy trong hệ thống sẽ đóng vai trò là các **máy con**, là một nút tính toán trong cụm máy phân tán. Và trong đó, chọn một máy kiêm vai trò là **máy chính**, dùng để quản lý, phân phối tài nguyên, triển khai chương trình…

***Lưu ý:* tên người dùng (`user`) ở tất cả các máy phải giống nhau để có thể thiết lập kết nối.**

- Thu thập thông tin địa chỉ IP của tất cả các máy trong cụm và thêm vào tập tin `/etc/hosts` của các máy.

```
192.168.32.102    TenMayChinh
192.168.32.103    TenMayCon01
192.168.32.104    TenMayCon02
192.168.32.105    TenMayCon03
…
```

Trong tập tin `/etc/hosts` loại bỏ các dòng tương tự như sau để tránh lỗi kết nối:

```
127.0.0.1    localhost
127.0.1.1    TenMay
```

- Tại máy chính, tạo khóa (key) để thiết lập các kết nối nếu chưa có:

```
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

Sao chép khóa đến tất cả các máy con trong cụm:

```
$ ssh-copy-id -i ~/.ssh/id_rsa.pub hduser@TenMayCon
```

Với `hduser` là tên người dùng chung ở tất cả các máy.
Lần lượt kết nối đến tất cả các máy trong cụm để lưu lại `fingerprint` vào tập tin `known_hosts` của người dùng tại máy chính.

```
$ ssh TenMayChinh
$ exit
$ ssh TenMayCon
$ exit
```

- Trên tất cả các máy, điều chỉnh các thiết lập trong các tập tin `etc/hadoop/core-site.xml`, `etc/hadoop/hdfs-site.xml`, `etc/hadoop/mapred-site.xml`, `etc/hadoop/yarn-site.xml` như trong phần cài đặt hệ thống giả lập phân tán.
  Lưu ý, trong tập tin `etc/hadoop/core-site.xml` thay đổi thông tin biến `fs.defaultFS` thành địa chỉ máy chứa NameNode (VD: `hdfs://TenMayChinh:9000`). Trong tập tin `etc/hadoop/hdfs-site.xml` có thể tăng số lượng nhân bản dữ liệu được quy định bằng biến `dfs.replication` và thêm vào một biến `dfs.permissions.enabled` (biến quy định việc có kiểm tra quyền trong HDFS hay không) với giá trị là `false`.
- Tại máy chính, điều chỉnh tập tin `etc/hadoop/workers` bằng cách thêm vào tên các máy con, tên mỗi máy con nằm trên một dòng, xóa bỏ dòng localhost có sẵn.

```
TenMayChinh
TenMayCon01
…
```

- Tại máy chính, tiến hành các bước khởi chạy hệ thống và khởi chạy thành phần quản lý tài nguyên (YARN) như trong phần cài đặt hệ thống giả lập phân tán.
  Khi chạy thành công, kiểm tra bằng câu lệnh `jps`, ở máy chính sẽ bao gồm các thành phần `namenode`, `datanode`, `secondarynamenode`, `nodemanager`, `resourcemanager` và ở máy con sẽ bao gồm các thành phần `datanode`, `nodemanager`.

e) Một số lưu ý khác:

Kể từ phiên bản Apache Hadoop 3.0.0 trở đi, một số port mặc định của các dịch vụ đã được thay đổi như trong bảng sau:

Port cũ	Port mới
50470	9871

50070	9870
8020	9820
50091	9869
50090	9868
50020	9867
50010	9866
50475	9865
50075	9864
16000	9600

2. Hướng dẫn cài đặt trên hệ điều hành Windows:

Apache Hadoop không chạy được chế độ phân tán thực trên hệ điều hành Windows, do đó, hướng dẫn sau sẽ cài đặt hệ thống dưới chế độ giả lập phân tán.

Yêu cầu phần mềm:

- `Oracle JDK` phiên bản 8 hoặc 11 được download tại trang chủ của Oracle: `http://www.oracle.com/technetwork/java/javase/downloads/index.html`.
- Gói phần mềm `Apache Hadoop` phiên bản mới nhất được tải về từ đường dẫn: `https://hadoop.apache.org/releases.html`
  Lưu ý: Bản `binary` là bản đã được biên dịch, nên tải bản này, thay vì `source` nguồn, sau khi tải về cần phải biên dịch rất phức tạp.
- Phần thực thi của Hadoop được biên dịch riêng cho Windows: `https://github.com/steveloughran/winutils`. Download thư mục `hadoop-x.x.x/bin` tương ứng với phiên bản Apache Hadoop đã tải ở trên.

Các bước cài đặt cơ bản:

a) Chuẩn bị:

- Giải nén tập tin `hadoop-x.x.x.tar.gz`, đặt ở thư mục tùy ý. Thư mục này nên có đường dẫn ngắn gọn, không chứa khoảng trắng, ví dụ: `C:\Hadoop`. Tiếp tục sao chép các tập tin của *Phần thực thi của Hadoop được biên dịch riêng cho Windows* vào thư mục `bin` trong thư mục Hadoop.
- Cài đặt `Oracle JDK`.

- Chỉnh sửa tập tin `etc/hadoop/hadoop-env.cmd` để định nghĩa biến môi trường Java như sau:

```
@rem The java implementation to use.  Required.
set JAVA_HOME=đường dẫn đến thư mục cài đặt Java
```

Trong đó: *đường dẫn đến thư mục cài đặt Java* là đường dẫn đến thư mục đã cài đặt `Oracle JDK` ở trên. Đường dẫn này không được chứa khoảng trắng và tuân theo chuẩn đặt tên tập tin 8.3 (**8.3 filename** hay còn được gọi là **short filename** hoặc **SFN**). Ví dụ: `C:\Progra~1\Java\jdk1.8.0_161`.

b) Tạo các biến môi trường:

Tạo các biến môi trường của riêng người dùng, như sau:

- `JAVA_HOME`: đường dẫn đến thư mục cài đặt Java. Ví dụ: `C:\Program Files\Java\jdk1.8.0_161`.
- `HADOOP_HOME`: đường dẫn đến thư mục đã giải nén Apache Hadoop ở trên. Ví dụ: `C:\Hadoop`.

Chỉnh sửa biến môi trường hệ thống, như sau:

- `PATH`: thêm vào sau biến này các thông tin: `%JAVA_HOME%\bin;%HADOOP_HOME%\bin`.

c) Tinh chỉnh các cài đặt:

Chỉnh sửa các tập tin `core-site.xml`, `hdfs-site.xml`, `yarn-site.xml`, `mapred-site.xml` như hướng dẫn ở phần `III.1.c` ở trên.

d) Khởi chạy hệ thống:

Mở cửa sổ dòng lệnh với quyền quản trị tại thư mục đã giải nén Apache Hadoop.

Định dạng hệ thống tập tin (format the filesystem) ở lần chạy đầu tiên, thực thi tập lệnh như sau:

```
bin\hdfs.cmd namenode -format
```

Khởi động NameNode và DataNode:

```
sbin\start-dfs.cmd
```

Lúc này, sẽ có hai cửa sổ dòng lệnh được bật lên tương ứng với hai thành phần hệ thống là `namenode` và `datanode`, theo dõi chi tiết các thông báo trên hai cửa sổ này để biết tình trạng hoạt động của hai thành phần.

Sau đó, khởi động ResourceManager và NodeManager, bằng câu lệnh:

```
sbin\start-yarn.cmd
```

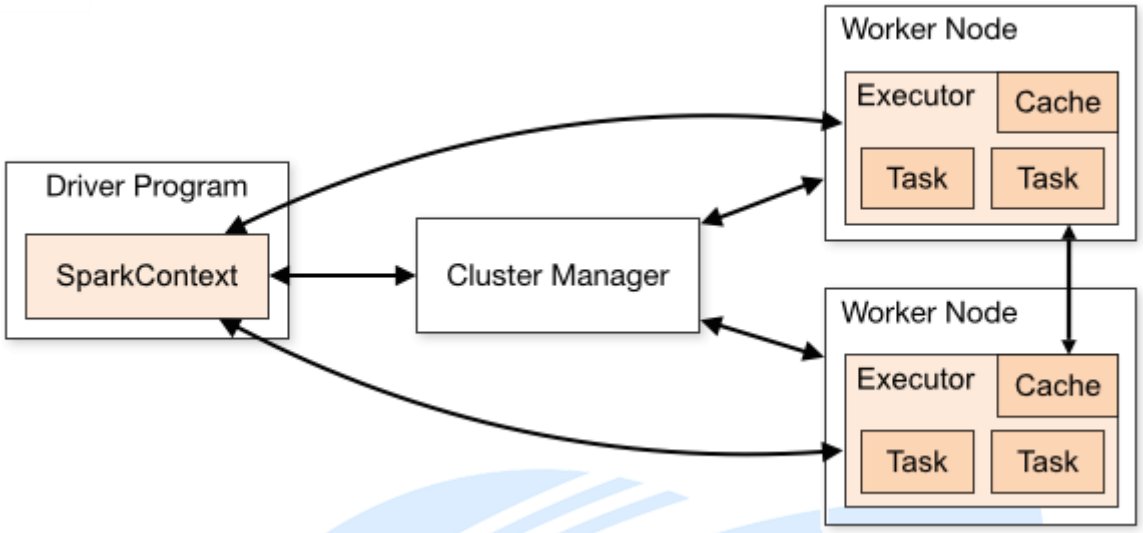
Tương tự như trên, sẽ tiếp tục có hai cửa sổ dòng lệnh được bật lên tương ứng với hai thành phần hệ thống là `nodemanager` và `resourcemanager` cần được quan tâm, theo dõi.

Cũng như việc cài đặt trên hệ điều hành Ubuntu đã hướng dẫn ở trên, người dùng có thể truy cập vào các trang giao diện quản lý của NameNode hay ResourceManager để xem thêm các thông tin khác.

IV. Cài đặt framework Apache Spark

Sinh viên tìm hiểu việc cài đặt framework Apache Spark qua sự hướng dẫn của giảng viên thực hành:

1. Chuẩn bị
   - ✓ Tải bản cài đặt Apache Spark từ địa chỉ http://spark.apache.org/downloads.html. Phiên bản mới nhất tính đến ngày 16/09/2022 là 3.3.0.
   - ✓ Spark sử dụng thư viện HDFS và YARN của Hadoop.
   - ✓ Có ba loại tập tin để tải về cài đặt, bao gồm:
     - Tập tin cài đặt đã được đóng gói sẵn dành cho các phiên bản Hadoop thông dụng (Pre-built for …).
     - Tập tin cài đặt được đóng gói không bao gồm các thư viện của Hadoop, dùng với hầu hết các phiên bản Hadoop nào, bằng cách tinh chỉnh các đường dẫn hệ thống (classpath - biến `SPARK_DIST_CLASSPATH`) trong tập tin `conf/spark-env.sh` của Spark (Pre-build with …).
     - Tập tin mã nguồn của Spark, dùng để biên dịch thành chương trình theo từng môi trường cài đặt (Source code).
   - ✓ Spark chạy trên môi trường Unix-like (Linux, Mac OS...) và Windows.
   - ✓ Spark rất dễ cài đặt để chạy độc lập trên một máy, chỉ cần cài đặt biến môi trường cho Java (`JAVA_HOME`).
   - ✓ Yêu cầu:
     - Java 8, Java 11 (nên dùng) hoặc Java 17.
     - Python 3.7 hoặc mới hơn.
     - Kể từ phiên bản 3.2.0, Spark loại bỏ việc hỗ trợ Python 3.6 và Java 8u201.
     - R 3.5 hoặc mới hơn
     - Đối với Scala API, Spark sử dụng Scala 2.12 trở lên.



Mô hình hoạt động của Apache Spark

2. Cài đặt Apache Spark phân tán với trình quản lý phân tán sẵn có (Standalone)
   - ✓ Cài đặt và tùy chỉnh gói `ssh` để máy chính với các máy trong mạng có thể liên kết và trao đổi thông qua các chứng thực (không cần nhập mật khẩu từng máy).
   - ✓ Chỉnh sửa tập tin host (ví dụ: `/etc/hosts`) trong tất cả các máy thuộc mạng để các máy biết tên và địa chỉ IP của nhau.
   - ✓ Dưới đây là các tùy chỉnh cơ bản nhất để thiết lập một hệ thống phân tán Apache Spark sử dụng trình quản lý phân tán sẵn có. Các tập tin tùy chỉnh được sao chép từ các tập tin mẫu (có đuôi `.template`) nằm sẵn trong thư mục `conf` sau đó sửa lại các thông tin.
     - ➢ Lập danh sách tên tất cả các máy trong mạng, mỗi máy trên một hàng và chép vào tập tin `conf/slaves`. Ví dụ:

```
Tên máy 1
Tên máy 2
...
```

     - ➢ Trong tập tin `conf/spark-defaults.conf` thiết lập thông tin spark master như ví dụ sau:

```
spark.master    spark://tên_máy_master:7077
```

     - ➢ Tập tin `conf/spark-env.sh` thiết lập thông tin IP của máy master như sau:

```
SPARK_MASTER_IP=192.168.1.100
```

3. Khởi động và tắt hệ thống
   - ✓ Trong thư mục `SPARK_HOME/sbin` sẽ có những tập tin thực thi sau:
     - ➢ `sbin/start-master.sh` - Khởi chạy thực thể quản lý (master instance) trên máy hiện tại.
     - ➢ `sbin/start-slaves.sh` - Khởi chạy thực thể con (slave instance) trên các máy được liệt kê trong tập tin `conf/slaves`.
     - ➢ `sbin/start-slave.sh` - Khởi chạy thực thể con (slave instance) trên máy hiện tại.
     - ➢ `sbin/start-all.sh` - Khởi chạy thực thể quản lý và tất cả các thực thể con.
     - ➢ `sbin/stop-master.sh` - Dừng thực thể quản lý đã được khởi chạy bởi tập tin thực thi `bin/start-master.sh`.
     - ➢ `sbin/stop-slaves.sh` - Dừng tất cả các thực thể con trên các máy được liệt kê trong tập tin `conf/slaves`.
     - ➢ `sbin/stop-all.sh` - Dừng thực thể quản lý và tất cả các thực thể con.
   - ✓ Chúng ta thường sử dụng tập tin thực thi `start-all.sh` tại máy chính để khởi động toàn bộ hệ thống, cũng như `stop-all.sh` để tắt toàn bộ hệ thống.

4. Chạy thử các ví dụ
   - ✓ Trong gói cài đặt Spark có sẵn một vài chương trình ví dụ viết trên ngôn ngữ Scala, Python, Java hoặc R, được đặt tại đường dẫn `examples/src/main`. Các chương trình này có thể dùng để minh họa hoặc để kiểm tra lỗi cài đặt hệ thống…
   - ✓ Để chạy chương trình ví dụ bằng Java hoặc Scala, sử dụng câu lệnh:

     ```
     bin/run-example <class> [params]
     ```

   - ✓ Khi đó, nó sẽ gọi câu lệnh `spark-submit` để khởi chạy ứng dụng. Ví dụ:

     ```
     ./bin/run-example SparkPi 10
     ```

   - ✓ Có thể chạy chương trình từ trong Scala shell, như:

     ```
     ./bin/spark-shell --master local[2]
     ```

     `--master` là tủy chỉnh đặc biệt trỏ đến máy chủ trong cụm máy xử lý phân tán. Hoặc đặt `local` nếu muốn hệ thống hoạt động với một luồng xử lý, hoặc `local[n]` để hệ thống hoạt động với `n` luồng xử lý.

Chọn tùy chỉnh `--help` để xem thêm hướng dẫn sử dụng các tùy chỉnh khác.

V. Tài liệu tham khảo

Ngoài ra có thể tham khảo thêm chi tiết tại các đường dẫn:

- ✓ Video hướng dẫn cài đặt Apache Hadoop 3.2.1 và Apache Spark 2.4.5 trên nền tảng Lubuntu 19.10: https://youtu.be/Zl9mqMUQEoI
- ✓ Tài liệu gốc về Hadoop: http://hadoop.apache.org/docs/current/
- ✓ Hướng dẫn lập trình với Apache Spark: http://spark.apache.org/docs/latest/programming-guide.html
- ✓ Cách sử dụng spark-submit: http://spark.apache.org/docs/latest/submitting-applications.html
- ✓ Các tùy chỉnh khi cài đặt Spark Standalone: http://spark.apache.org/docs/latest/spark-standalone.html

~ HẾT ~