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 2

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:

- Sử dụng thành thạo hệ điều hành Ubuntu
- Sử dụng thành thạo một IDE để lập trình với ngôn ngữ Java
- Framework lập trình Apache Hadoop
- Hệ thống tập tin phân tán HDFS (Hadoop FileSystem)
- …

2. Nội dung

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

- Thành thạo các thao tác, câu lệnh trong Ubuntu.
- Xem lại framework Apache Hadoop từ những kiến thức được học trên lớp lý thuyết và thực hành.
- Xem lại kiến thức về hệ thống tập tin phân tán HDFS đã được học trên lớp lý thuyết.

❖ **Các thao tác với hệ thống tập tin phân tán HDFS**

❖ **Lập trình phần mềm xử lý Dữ liệu lớn trên framework Apache Hadoop sử dụng ngôn ngữ Java**

3. Kết quả cần đạt

✓ Sinh viên cần nắm rõ cú pháp và hiểu được ý nghĩa các câu lệnh thường dùng để làm việc với hệ thống tập tin phân tán HDFS.

✓ Sinh viên có đầy đủ kiến thức và kỹ năng để phát triển một phần mềm xử lý Dữ liệu lớn trên framework Apache Hadoop sử dụng ngôn ngữ Java.

II. Ôn tập lại những kiến thức đã học

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 cũng như thực hành của môn học để ôn tập củng cố kiến thức về framework lập trình Apache Hadoop và hệ thống tập tin phân tán HDFS.

III. Các thao tác với hệ thống tập tin phân tán HDFS

Sau khi cài đặt xong Apache Hadoop ở buổi học trước, sinh viên cần khởi động hệ thống tập tin phân tán HDFS và thực hiện theo các hướng dẫn của giảng viên thực hành:

Trong môi trường dòng lệnh, để thao tác với hệ thống HDFS cần sử dụng một trong hai câu lệnh có cú pháp như sau:

```
bin/hadoop fs command [<command_options>] [<args>]
bin/hdfs dfs command [<command_options>[ [<args>]
```

Các câu lệnh cơ bản gồm:

- cat
- copyFromLocal / moveFromLocal / put
- copyToLocal / moveToLocal / get
- cp / mv
- count
- find
- help
- ls / lsr
- mkdir
- rm / rmdir / rmr

Sinh viên cần lưu ý các cú pháp và phân biệt được ý nghĩa của các câu lệnh có chức năng gần giống nhau trong phần hướng dẫn của giảng viên thực hành.

IV. Lập trình phần mềm xử lý Dữ liệu lớn trên framework Apache Hadoop sử dụng ngôn ngữ Java

Sinh viên có thể sử dụng một IDE hỗ trợ ngôn ngữ Java bất kỳ, phù hợp để phát triển các phần mềm xử lý Dữ liệu lớn trên nền tảng của framework Apache Hadoop.

1. Thêm các thư viện Hadoop vào project Java:

Trong project Java, thêm vào các thư viện của Apache Hadoop ở đường dẫn

```
${HADOOP_HOME}/share/hadoop/*/*.jar
${HADOOP_HOME}/share/hadoop/*/lib/*.jar
```

2. Các kiểu dữ liệu đặc biệt trong Apache Hadoop:

Framework MapReduce này chỉ hoạt động trên các cặp `<key, value>`, nghĩa là, framework xem đầu vào (input) cho các công việc ở dạng tập hợp các cặp `<key, value>` và sẽ tạo ra một tập hợp các cặp `<key, value>` là đầu ra (output) của công việc sau khi xử lý, với các kiểu dữ liệu khác nhau. Các lớp `key` và `value` phải được tuần tự hóa (serializable) bởi framework.

Các loại đầu vào và đầu ra của một công việc MapReduce:

```
(input) <k1, v1> -> map -> <k2, v2> -> combine -> <k2, v2>
-> reduce -> <k3, v3> (output)
```

Các kiểu dữ liệu của Apache Hadoop cơ bản, để sử dụng làm `key` hay `value`, bao gồm:

- IntWritable

- `LongWritable`
- `FloatWritable`
- `DoubleWritable`
- `Text`
- `NullWritable` (thường dùng trong trường hợp không có key trong cặp `<key, value>`).

Ngoài ra còn nhiều các loại khác, có thể tham khảo tại link: https://hadoop.apache.org/docs/stable/api/org/apache/hadoop/io/package-summary.html

3. Tạo lớp key hay value tùy chỉnh mới

Để tạo được lớp dữ liệu mới ngoài các lớp kiểu dữ liệu đã có sẵn trong thư viện của Apache Hadoop, người dùng cần phải tạo lớp kế thừa (implement) từ interface **`Writable`** để framework có thể dễ dàng tuần tự hóa (serializable). Bên cạnh đó, các lớp `key` phải kế thừa từ interface **`WritableComparable`** để framework có thể sắp xếp được chúng khi cần gom nhóm theo `key` (`sort`, `shuffle`…), thông qua phương thức so sánh đối tượng được lập trình sẵn.

Tạo lớp dữ liệu mới như sau:

```
public class tên_lớp implements WritableComparable<tên_lớp> {
```

Trong lớp mới, chúng ta phải định nghĩa đầy đủ các hàm:

- `Constructors(args tùy ý)` (các hàm khởi tạo)
- `readFields(java.io.DataInput)`
- `write(java.io.DataOutput)`
- `toString()` (framework sẽ tự động dùng hàm này khi ghi dữ liệu ra tập tin)
- `compareTo(chính nó)` (để so sánh các `key` với nhau trong quá trình `shuffle, sort`)

4. Cách biên dịch phần mềm:

✓ Thêm vào file `hadoop_env.sh` biến môi trường `HADOOP_CLASSPATH` dùng để chỉ định thư viện sử dụng trong quá trình biên dịch như sau

```
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
```

✓ Chạy lệnh sau để biên dịch các tập tin java thành các class

```
bin/hadoop com.sun.tools.javac.Main -d <thư mục chứa các file class> <đường dẫn đến các file source java>
```

✓ Dùng lệnh sau để đóng gói tất cả các tập tin class được tạo ở bước trên thành **01** tập tin thực thi (file jar)

```
jar cvf <tên file>.jar -C <đường dẫn thư mục chứa các file class> . <-- có dấu chấm cuối câu lệnh
```

5. Đưa phần mềm lên hệ thống xử lý song song phân tán Apache Hadoop

✓ Khởi động hệ thống xử lý song song phân tán Apache Hadoop

✓ Dùng lệnh sau để khởi chạy phần mềm trên hệ thống

```
bin/hadoop jar <đường dẫn đến file .jar> <package chứa main class> <args>
```

V. Bài tập

1. Bài tập 1:

Tạo cây thư mục trong hệ thống tập tin HDFS theo yêu cầu của GVHD.

2. Bài tập 2:

Tạo project WordCount bằng ngôn ngữ Java, thêm các thư viện cần thiết của Apache Hadoop, có thể tham khảo mã nguồn trên trang chủ.
Biên dịch và chạy thử chương trình.

3. Bài tập 3:

Download dữ liệu bán hàng tại địa chỉ:
http://fimi.uantwerpen.be/data/retail.dat.gz
Tập dữ liệu đầu vào là dữ liệu các giao dịch có dạng:

```
25 52 164 240 274 328 368 448 538 561 630 687 730 775 825 834
39 120 124 205 401 581 704 814 825 834
35 249 674 712 733 759 854 950
39 422 449 704 825 857 895 937 954 964
15 229 262 283 294 352 381 708 738 766 853 883 966 978
26 104 143 320 569 620 798
7 185 214 350 529 658 682 782 809 849 883 947 970 979
227 390
71 192 208 272 279 280 300 333 496 529 530 597 618 674 675 720 855 914 932
```

Với mỗi dòng ghi nhận một giao dịch (một tập các sản phẩm được mua cùng nhau).
Yêu cầu: Lưu dữ liệu vào HDFS và viết 01 chương trình MapReduce có các chức năng:

a) Thống kê sự xuất hiện đồng thời của từng cặp sản phẩm: count (A, B) = số giao dịch chứa đồng thời A và B.
b) Xác suất mua sản phẩm B khi đã mua sản phẩm A (xác suất có điều kiện - conditional probability): prob (B | A) = count (A, B) / count (A).
c) Thống kê sự xuất hiện đồng thời của từng bộ ba sản phẩm: count (A, B, C) = số giao dịch chứa đồng thời A, B và C.
d) Xác suất mua sản phẩm A khi đã mua sản phẩm B và C (xác suất có điều kiện - conditional probability): prob (A | B, C) = count (A, B, C) / count (B, C).

4. Bài tập 4:

Download dữ liệu thông tin về giá cả một số mặt hàng thiết yếu theo thị trường TP.HCM tại: Thông tin giá cả thị trường | Cổng dữ liệu mở Thành Phố Hồ Chí Minh (hochiminhcity.gov.vn).
Yêu cầu: Lưu dữ liệu vào HDFS và viết 01 chương trình MapReduce có các chức năng:

a) Thống kê theo *ngày cập nhật*, số lượng mặt hàng có *giá* cao hơn một ngưỡng giá nhập vào.
b) Giá bán trung bình của từng mặt hàng.
c) Với mỗi mặt hàng, cho biết giá bán cao nhất và thấp nhất

VI. Tài liệu tham khảo

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

- Video minh họa việc biên dịch và chạy chương trình WordCount trên Apache Hadoop: https://youtu.be/JCqf3aG9eFE
- http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HDFSCommands.html
- http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/FileSystemShell.html
- https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html

~ HẾT ~