KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
227 Nguyễn Văn Cừ, Phường 4, Quận 5, TP.HCM
Điện Thoại: (08) 38.354.266 - Fax:(08) 38.350.096

CSC17106 – XỬ LÝ PHÂN TÍCH DỮ LIỆU TRỰC TUYẾN

HƯỚNG DẪN THỰC HÀNH

SPARK CƠ BẢN

I. Thông tin chung

Mã số:	HD03
Thời lượng dự kiến:	3 tiếng
Deadline nộp bài:	-
Hình thức:	-
Hình thức nộp bài:	-
GV phụ trách:	Phạm Minh Tú
Thông tin liên lạc với GV:	pmtu@fit.hcmus.edu.vn

II. Chuẩn đầu ra cần đạt

Bài hướng dẫn này nhằm mục tiêu đạt giúp sinh viên được các mục tiêu sau:

1. Cài đặt Spark trên môi trường window
2. Cấu hình Spark
3. Chạy Spark và các lệnh cơ bản

III. Mô tả

Các bước cài đặt:

1. Cài đặt Java:

Trước tiên, bạn cần cài đặt Java trên máy tính Windows của bạn. Tải xuống và cài đặt JDK (Java Development Kit) từ trang web của Oracle hoặc sử dụng OpenJDK.

2. Cài đặt Hadoop cho Window:

Spark yêu cầu một số thư viện Hadoop. Bạn có thể tải xuống bản Hadoop pre-built cho Windows từ trang web của Apache Hadoop và cài đặt nó. Sau khi cài đặt, bạn cần đặt biến môi trường HADOOP_HOME để Spark có thể tìm thấy nó.

Lưu ý: Nếu sinh viên đã cài đặt được các bước 1, 2 trước đó thì không cần cài lại.

3. Tải và cài đặt Apache Spark:

Truy cập trang web chính thức của Apache Spark để tải xuống phiên bản Spark bạn muốn sử dụng. Chọn phiên bản "Pre-built for Apache Hadoop" và tải xuống tệp ZIP.
Giải nén tệp ZIP vào một thư mục trên máy tính của bạn.

4. Đặt biến môi trường SPARK_HOME:

Đặt biến môi trường SPARK_HOME để Spark biết vị trí cài đặt của nó. Để làm điều này, mở "System Properties" trên Windows và thêm biến môi trường mới với tên SPARK_HOME và giá trị là đường dẫn đến thư mục Spark bạn vừa giải nén.

5. Chỉnh sửa biến môi trường PATH:

Chỉnh sửa biến môi trường PATH của bạn để bao gồm %SPARK_HOME%\bin.

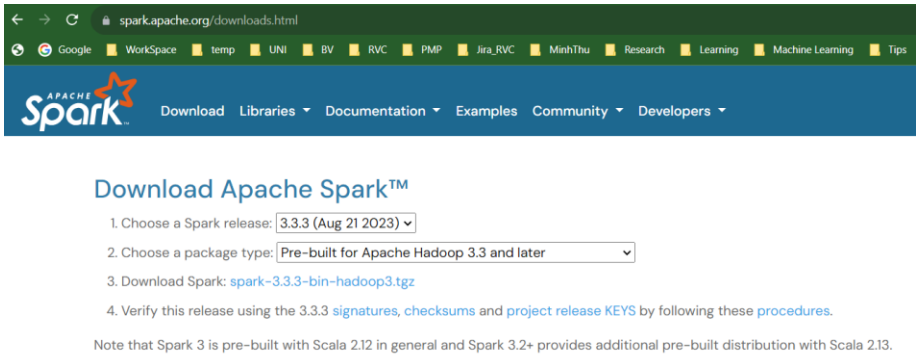
6. Kiểm tra cài đặt Spark:

Mở Command Prompt và chạy Spark Shell bằng lệnh sau để đảm bảo rằng Spark đã được cài đặt đúng cách:

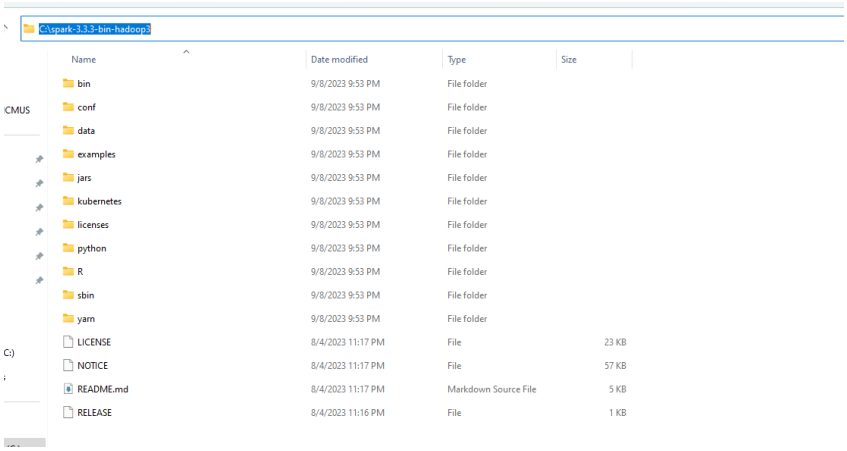
```
C:\Users\PC2023>spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://DESKTOP-N032419:4040
Spark context available as 'sc' (master = local[*], app id = local-1694611811171).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.3.3
      /_/

Using Scala version 2.12.15 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_371)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Vài thông tin quan trọng:

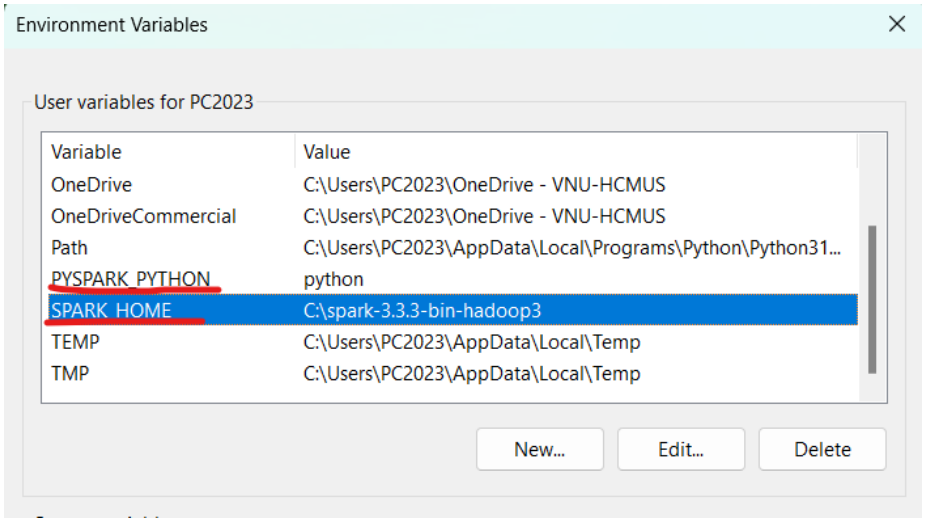
Link tải apache spark: https://spark.apache.org/downloads.html



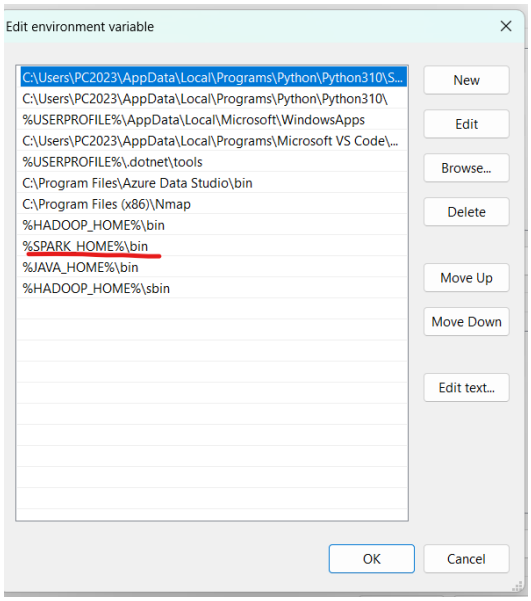
Giải nén và lưu Spark tại đường dẫn:



Đặt biến môi trường.



Chỉnh sửa biến PATH



Cấu trúc dữ liệu cơ bản của Spark

Resilient Distributed Dataset (RDD):

RDD là cấu trúc dữ liệu cơ bản của Spark. Đây là một tập hợp bất biến và được phân tán các phần tử có thể chia thành nhiều phân vùng để xử lý song song. RDD hỗ trợ các phép biến đổi (transformations) và hành động (actions) để thực hiện các phép tính phức tạp trên dữ liệu.

```
scala> val info = Array(1, 2, 3, 4)
info: Array[Int] = Array(1, 2, 3, 4)

scala> val distinfo = sc.parallelize(info)
distinfo: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[1] at parallelize at <console>:24

scala> distinfo.reduce((a, b) => a + b)
res2: Int = 10
```

reduce: hành động của RDD

Phương thức **reduce** trong Apache Spark được sử dụng để áp dụng một phép toán gộp lên tất cả các phần tử trong một RDD và trả về một giá trị duy nhất.

Cách hoạt động:

Trong trường hợp này, ((a,b)=> a+b) function này nhận vào hai tham số a và b, và trả về tổng của chúng. Function này có thể được sử dụng trong các tình huống cần truyền một hàm nhỏ như là một đối số cho các phương thức khác như reduce trong Apache Spark, function này được áp dụng tuần tự lên các phần tử, sử dụng a là kết quả tạm thời (initial value hoặc kết quả của lần gộp trước đó) và b là phần tử tiếp theo trong RDD.

Trong Apache Spark, việc tính toán trên nhiều partition (phân vùng) được quản lý và phân công tự động bởi Spark. Spark tự động chia dữ liệu thành các partition và sau đó phân công các tác vụ tính toán đến các executor trên các node trong mạng phân tán. Quy trình này là một phần quan trọng của việc phân phối tính toán trên nhiều máy tính để tận dụng sức mạnh tính toán **song song.**

Dưới đây là một số điểm quan trọng về cách Spark phân công tính toán trên các partition:

Tạo Partition: Khi bạn tạo một RDD từ dữ liệu, Spark sẽ tự động chia dữ liệu thành các partition. Số lượng partition có thể được xác định bởi bạn hoặc được Spark tự động xác định dựa trên kích thước của dữ liệu và cấu hình mặc định của Spark.

Phân công Tác Vụ: Khi bạn áp dụng các phép toán Spark (như map, reduce, filter,...) lên RDD, Spark sẽ phân công các tác vụ đến các executor trên các máy tính trong cluster. Mỗi executor sẽ xử lý một hoặc nhiều partition, và tính toán trên chúng.

Tối ưu hóa Di chuyển Dữ liệu: Spark cố gắng tối ưu hóa việc di chuyển dữ liệu trên mạng bằng cách cố gắng thực hiện tính toán trên các partition mà dữ liệu đã được lưu trữ trên cùng một node (data locality). Điều này giúp giảm tải mạng và tăng hiệu suất tính toán.

Caching: Bạn có thể sử dụng cache hoặc persist để lưu trữ một số partition trong bộ nhớ (RAM) để tránh việc đọc dữ liệu từ đĩa ở mỗi lần tính toán. Điều này có thể cải thiện hiệu suất nếu bạn thường xuyên truy cập các partition cụ thể nhiều lần.

Repartitioning: Bạn có thể sử dụng phương thức repartition để thay đổi cách chia partition của RDD, ví dụ: chia lại thành nhiều hoặc ít hơn partition. Điều này có thể hữu ích trong việc tối ưu hóa việc phân phối dữ liệu cho các phần công việc cụ thể.

Để minh họa cách Spark xử lý dữ liệu này, hãy xem xét một ví dụ sử dụng phương thức reduce để tính tổng các phần tử trong mảng:

Chia Dữ Liệu thành Partition:
Trong trường hợp này, chúng ta có một RDD với dữ liệu mảng [1, 2, 3, 4]. Spark có thể chia nó thành 2 partition như sau:

- Partition 1: [1, 2]
- Partition 2: [3, 4]

Phân Công Tác Vụ: Spark phân công tác vụ tính toán đến các executor. Trong trường hợp này, chúng ta có thể tưởng tượng có hai executor (executor 1 và executor 2) trong môi trường phân tán.

Tính Toán Trên Từng Partition:
Lambda function hoặc phép toán reduce sẽ được áp dụng trên từng partition riêng lẻ.

Trên Partition 1: [1, 2]
Kết quả tạm thời: 1
Áp dụng lambda: (1, 2) => 1 + 2 = 3

Trên Partition 2: [3, 4]
Kết quả tạm thời: 3
Áp dụng lambda: (3, 4) => 3 + 4 = 7

Tổng Hợp Kết Quả: Sau khi tính toán trên từng partition, Spark sẽ tổng hợp kết quả từ các partition. Trong trường hợp này, kết quả tạm thời từ Partition 1 (3) và kết quả tạm thời từ Partition 2 (7) sẽ được tổng hợp lại bằng cách áp dụng phép toán reduce một lần nữa:

Tổng hợp cuối cùng: (3, 7) => 3 + 7 = 10

Vậy, kết quả cuối cùng của phép toán reduce trên mảng [1, 2, 3, 4] là 10.

Trong Apache Spark, các hoạt động trên RDD (Resilient Distributed Dataset) được chia thành hai loại chính: transformations và actions. Đây là sự phân biệt giữa chúng:

Transformations:

Transformations là các phép biến đổi dữ liệu trên RDD để tạo ra một RDD mới. Khi bạn áp dụng một transformation cho một RDD, nó không thực hiện tính toán ngay lập tức mà chỉ thiết lập một kế hoạch cho các phép biến đổi này.

Các ví dụ về transformations bao gồm **map, filter, reduceByKey, groupByKey, flatMap, distinct**, và nhiều phép biến đổi khác.

Transformations cho phép xây dựng một lược đồ tính toán phức tạp bằng cách kết hợp nhiều transformations lại với nhau.

Actions:

Actions là các phép thực thi trên RDD để thu thập dữ liệu hoặc trả về kết quả. Khi bạn áp dụng một action cho một RDD, Spark bắt đầu tính toán và thực hiện các transformations cần thiết để trả về kết quả cho bạn.

Các ví dụ về actions bao gồm **count, collect, first, take, reduce, saveAsTextFile**, và nhiều phép thực thi khác.

Actions là điểm cuối của một chuỗi tính toán trong Spark và gây ra sự kích hoạt thực sự của tính toán.

Một vài ví dụ:

```
val rdd = sc.parallelize(Array(1, 2, 3, 4, 5))

val mapped_rdd = rdd.map((x) => x * 2) // transformation

val data = mapped_rdd.collect() // actions

// Output: Array(2, 4, 6, 8, 10)
```

```
// Tạo một RDD từ danh sách các số
val rdd = spark.sparkContext.parallelize(Seq(1, 2, 3, 4, 5))

// Sử dụng transformation filter để lọc các số lớn hơn 2
val filteredRdd = rdd.filter(x => x > 2)

// In kết quả
filteredRdd.collect().foreach(println)
```

```
// Tạo một RDD từ danh sách các số
val rdd = spark.sparkContext.parallelize(Seq(1, 2, 3, 4, 5))

// Sử dụng action count để đếm số phần tử trong RDD
val count = rdd.count()

// In kết quả
println("Số phần tử trong RDD là: " + count)
```

```
// Tạo một RDD từ danh sách các số
val rdd = spark.sparkContext.parallelize(Seq(1, 2, 3, 4, 5))
```

```
// Sử dụng action first để lấy phần tử đầu tiên của RDD
val firstElement = rdd.first()

// In kết quả
println("Phần tử đầu tiên trong RDD là: " + firstElement)
```

IV. Tài liệu tham khảo

1. https://spark.apache.org/
2. https://spark.apache.org/docs/latest/
3. https://spark.apache.org/docs/latest/rdd-programming-guide.html
4. https://www.oreilly.com/library/view/learning-spark/9781449359034/
5. https://www.packtpub.com/product/apache-spark-3-a-comprehensive-guide/9781803234981
6. https://www.youtube.com/user/databricks

V. Bài tập cơ bản (dùng spark-shell)

1. Tạo một RDD từ một danh sách các số và sử dụng transformation map để tạo một RDD mới chứa bình phương của mỗi số.
2. Tạo một RDD từ một danh sách các chuỗi và sử dụng transformation filter để lọc ra những chuỗi có độ dài lớn hơn 5 ký tự.
3. Tạo hai RDD riêng biệt từ hai danh sách các số và sau đó sử dụng transformation union để gộp chúng thành một RDD duy nhất.
4. Tạo một RDD từ một danh sách các số và sử dụng action count để đếm tổng số phần tử trong RDD.
5. Tạo một RDD từ một danh sách các chuỗi và sử dụng action first để lấy ra chuỗi đầu tiên trong RDD.
6. Tạo một RDD từ một danh sách các số và sử dụng action reduce để tính tổng của tất cả các số trong RDD.
7. Tạo một RDD từ một danh sách các số và sau đó sử dụng action collect để thu thập tất cả các số trong RDD và in ra chúng.
8. Tạo một RDD từ một danh sách các số và sử dụng transformation map để nhân mỗi số với 2, sau đó sử dụng action reduce để tính tổng của các số đã được nhân.
9. Tạo một RDD từ một danh sách các chuỗi và sử dụng transformation filter để lọc ra các chuỗi có chứa từ "Spark", sau đó sử dụng action count để đếm số lượng chuỗi thỏa điều kiện.

10. Tạo một RDD từ một danh sách các số và sử dụng transformation map để tính bình phương của mỗi số, sau đó sử dụng action reduce để tính tổng bình phương của các số.

VI. Bài tập nâng cao

Performance & Memory

1. Tạo một RDD gồm 1 triệu số nguyên. Hãy đếm số lượng partition mặc định được tạo ra. Dùng lệnh nào để thay đổi số lượng partition?
2. So sánh hiệu suất xử lý khi thực hiện count() trước và sau khi dùng .cache(). Hãy nêu sự khác biệt?
3. Sử dụng glom() để in kích thước của từng partition. Hãy nhận xét kết quả đó?

Key-Value

4. Tạo một RDD gồm các cặp (key, value), trong đó key là một chuỗi, value là số nguyên. Hãy tính tổng giá trị theo từng key.
5. Với dữ liệu key-value, hãy tính trung bình cộng giá trị cho từng key.
6. Tạo một RDD gồm các từ trong nhiều câu văn. Hãy đếm số lần xuất hiện của mỗi từ.

Transformation

7. Sử dụng flatMap để biến một RDD chứa các chuỗi thành RDD chứa từng từ riêng lẻ.
8. Dùng distinct() để loại bỏ các phần tử trùng nhau trong một RDD. Sau đó kết hợp với map để đếm độ dài mỗi phần tử duy nhất.

Kết hợp RDD và filter logic

9. Tạo một RDD gồm các số từ 1 đến 1000. Lọc ra các số chia hết cho cả 3 và 5, rồi tính tổng các số đó.
10. Cho một RDD chứa danh sách tên sinh viên. Hãy lọc ra những sinh viên có họ bắt đầu bằng chữ “Nguyễn” và đếm số lượng.

Suy luận logic

11. Hãy giải thích sự khác biệt giữa reduceByKey và groupByKey. Trong trường hợp nào nên dùng reduceByKey?
12. Giả sử cần tính tổng giá trị các phần tử theo nhóm, nhóm được xác định bởi điều kiện phân loại (ví dụ: chẵn/lẻ). Hãy nêu giải pháp tiếp cận?

Challenge / Mini project

13. Với một đoạn văn bản dài (có thể copy từ một bài báo), hãy viết một pipeline Spark để:

- Chuyển văn bản thành danh sách các từ.
- Lọc bỏ stop words (tự định nghĩa danh sách).
- Đếm số lần xuất hiện của mỗi từ còn lại.
- Sắp xếp giảm dần theo số lần xuất hiện và in ra top 10 từ phổ biến nhất.

14. Hãy viết một pipeline Spark để tính:

- Tổng giá trị các số chia hết cho 7 từ 1 đến 10 triệu.
- So sánh thời gian thực hiện khi chia dữ liệu thành 2, 4, và 8 partition.