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

PYSPARK 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. Giới thiệu PySpark
2. RDD (Resilient Distributed Datasets)
3. PySpark DataFrame

III. Mô tả

PySpark: là một thư viện mã nguồn mở cho ngôn ngữ Python được phát triển bởi Apache Software Foundation. Nó được thiết kế để hỗ trợ xử lý và phân tích dữ liệu lớn (Big Data) thông qua mô hình tính toán phân tán. PySpark là một phần quan trọng của hệ sinh thái Apache Spark, một nền tảng phân tích dữ liệu mạnh mẽ và hiệu quả cho các ứng dụng xử lý dữ liệu phức tạp.

Với sự kết hợp giữa sức mạnh của Python và tính phân tán của Apache Spark, PySpark cho phép bạn thực hiện xử lý dữ liệu và tính toán song song trên các cụm máy tính, giúp tăng hiệu suất và khả năng mở rộng.

Tại sao cần dùng PySpark?

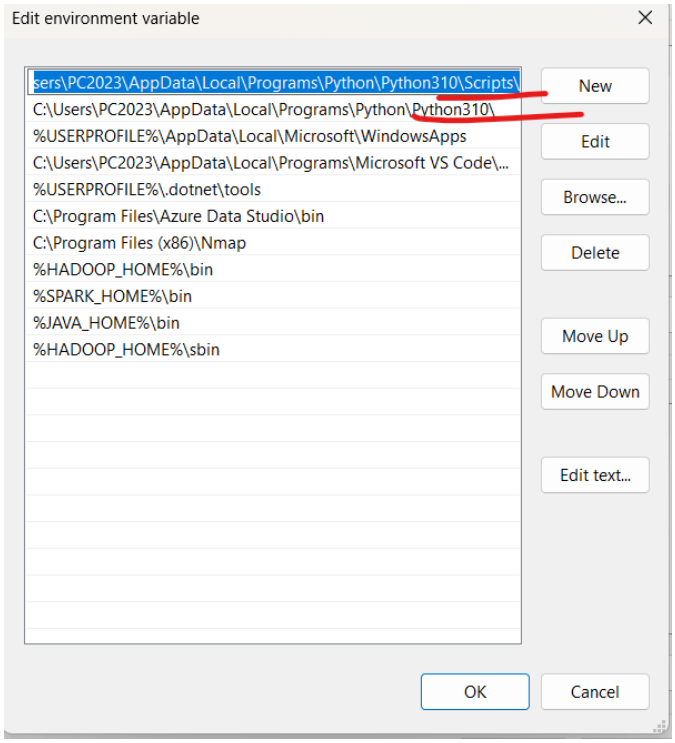
Xử lý dữ liệu lớn: PySpark giúp bạn xử lý và phân tích dữ liệu có kích thước lớn mà Python truyền thống không thể xử lý.
Hiệu suất cao: PySpark tận dụng tính phân tán của Apache Spark, giúp tăng hiệu suất tính toán.
Hỗ trợ nhiều nguồn dữ liệu: PySpark hỗ trợ đọc và ghi dữ liệu từ nhiều nguồn như HDFS, Cassandra, Hive, và nhiều hệ thống lưu trữ dữ liệu khác.
Thư viện phong phú: PySpark cung cấp nhiều thư viện cho việc xử lý dữ liệu, máy học (machine learning), xử lý đồ thị (graph processing), và xử lý luồng dữ liệu (stream processing).

Dễ học: Dựa trên Python, PySpark dễ học và có cú pháp đơn giản, phù hợp cho người mới học lập trình dữ liệu lớn.

Cài đặt:

Để sử dụng PySpark, bạn cần cài đặt nó trên máy tính của mình. Cài đặt có thể thực hiện thông qua quản lý gói (package manager) của Python. Bạn cũng cần cài đặt Apache Spark trước khi sử dụng PySpark.

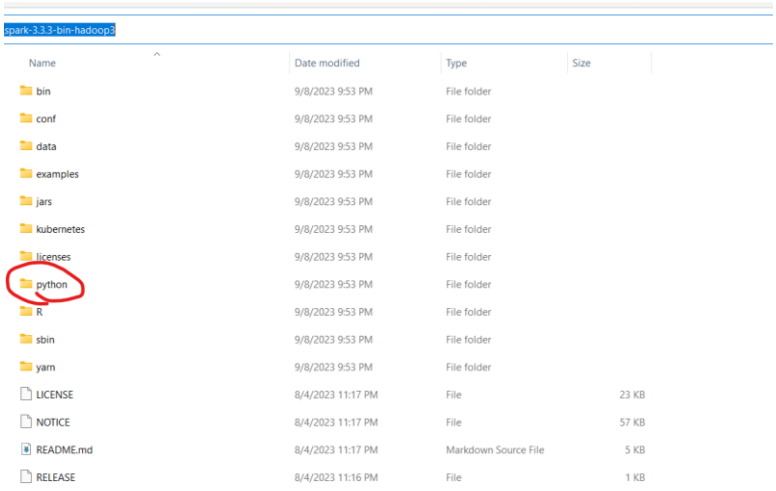
Nếu dùng Spark 3, hãy cài đặt python version 3.9 để tránh lỗi thư viện về sau này. Sau khi cài đặt python thì khai báo biến môi trường như hình dưới đây.



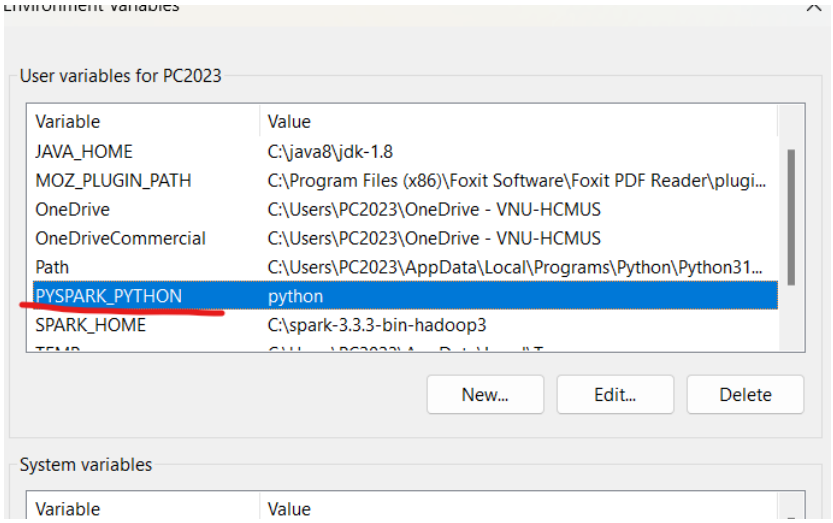
Sau đó có thể cài pyspark thông qua trình quản lý pip

Cú pháp: *pip install pyspark*

Hoặc dùng python có sẵn trong Spark như đã cài đặt bài thực hành trước.



Để dùng python của Spark thì cần khai báo thêm môi trường.



Vì vậy, nếu bạn muốn chạy chương trình pyspark với python hệ thống thì có thể dùng lệnh:

CMD: C:\> python pyspark_file.py

Nếu muốn dùng python của Spark thì thực hiện câu lệnh dưới đây:

CMD: C:\> spark-submit pyspark_file.py

Ví dụ viết một chương trình pyspark như sau:

Chúng ta sẽ đọc dữ liệu từ một tệp CSV, thực hiện một số phép biến đổi và tính toán trên DataFrame, sau đó hiển thị kết quả. Đảm bảo bạn đã cài đặt PySpark trước khi chạy chương trình này.

```
# Import các thư viện cần thiết
from pyspark.sql import SparkSession

# Khởi tạo SparkSession
spark =
SparkSession.builder.appName("pyspark_sample").getOrCreate()
```

```
# Đọc dữ liệu từ tệp CSV và tạo DataFrame
data_path = "path/to/your/data.csv"
df = spark.read.csv(data_path, header=True, inferSchema=True)

# Hiển thị cấu trúc của DataFrame
df.printSchema()

# Hiển thị 5 dòng đầu tiên của DataFrame
df.show(5)

# Thực hiện một số phép biến đổi trên DataFrame
filtered_df = df.filter(df["Age"] > 25)
selected_df = filtered_df.select("Name", "Age")
grouped_df = selected_df.groupBy("Age").count()

# Hiển thị kết quả
print("Kết quả sau khi thực hiện các phép biến đổi:")
grouped_df.show()

# Đóng SparkSession
spark.stop()
```

PySpark sử dụng RDD

```
# Import các thư viện cần thiết
from pyspark import SparkContext

# Khởi tạo SparkContext
sc = SparkContext(appName="RDDExample")

# Đọc dữ liệu từ tệp văn bản và tạo một RDD
data_path = "path/to/your/data.txt"
rdd = sc.textFile(data_path)

# Thực hiện phép biến đổi: chuyển đổi các dòng thành các số và
lọc ra các số chẵn
numbers_rdd = rdd.flatMap(lambda line: line.split(" ")) \
                 .map(lambda x: int(x)) \
                 .filter(lambda x: x % 2 == 0)

# Tính toán trung bình cộng của các số trong RDD
sum_count = numbers_rdd.aggregate((0, 0),
           lambda acc, value: (acc[0] + value, acc[1] + 1),
lambda acc1, acc2: (acc1[0] + acc2[0], acc1[1] + acc2[1]))
average = sum_count[0] / sum_count[1]

# Hiển thị kết quả
print("Trung bình cộng của các số chẵn là:", average)

# Đóng SparkContext
sc.stop()
```

Giải thích dòng lệnh:

```
sum_count = numbers_rdd.aggregate((0, 0),
           lambda acc, value: (acc[0] + value, acc[1] + 1),
lambda acc1, acc2: (acc1[0] + acc2[0], acc1[1] + acc2[1]))
average = sum_count[0] / sum_count[1]
```

Đầu tiên, chúng ta khởi tạo một giá trị ban đầu là (0, 0). Trong trường hợp này, nó gồm hai phần tử: acc[0] để tính tổng của các số và acc[1] để đếm số lượng các số.

Sau đó, chúng ta sử dụng phép biến đổi lambda acc, value: (acc[0] + value, acc[1] + 1) để thực hiện hai thao tác trên acc (accumulator) và value (giá trị từ RDD). Đây là cách nó hoạt động:

acc[0] + value: Tổng các số tính được từ lần lặp trước đến lần lặp hiện tại.
acc[1] + 1: Số lượng các số đã tính được từ lần lặp trước đến lần lặp hiện tại.
Khi dòng mã này thực hiện xong trên tất cả các phần tử trong RDD, nó sẽ kết hợp tất cả các acc lại với nhau bằng cách sử dụng phép biến đổi thứ hai: lambda acc1, acc2: (acc1[0] + acc2[0], acc1[1] + acc2[1]). Điều này giúp tổng hợp kết quả từ các phần tử con trên các partition (phân vùng) khác nhau của RDD.

Kết quả cuối cùng là một cặp giá trị (tổng, số lượng). Để tính trung bình cộng, chúng ta chỉ cần chia tổng cho số lượng: average = sum_count[0] / sum_count[1].

PySpark sử dụng cấu trúc dữ liệu dataframe

```
# Import các thư viện cần thiết
from pyspark.sql import SparkSession
from pyspark.sql.functions import col

# Khởi tạo SparkSession
spark =
SparkSession.builder.appName("DataFrameExample2").getOrCreate()

# Đọc dữ liệu từ tệp CSV và tạo DataFrame
data_path = "path/to/your/employee_data.csv"
employee_df = spark.read.csv(data_path, header=True,
inferSchema=True)

# Hiển thị cấu trúc của DataFrame
employee_df.printSchema()

# Hiển thị 5 dòng đầu tiên của DataFrame
employee_df.show(5)

# Thực hiện một số phép biến đổi trên DataFrame
# 1. Lọc ra những người có lương lớn hơn 50000
filtered_df = employee_df.filter(col("Salary") > 50000)

# 2. Chọn các cột cụ thể để hiển thị
selected_df = filtered_df.select("Name", "Age", "Salary")
```

```
# 3. Sắp xếp theo lương giảm dần
sorted_df = selected_df.orderBy("Salary", ascending=False)

# Hiển thị kết quả
print("Kết quả sau khi thực hiện các phép biến đổi:")
sorted_df.show()

# Đóng SparkSession
spark.stop()
```

DataFrame là một khái niệm quan trọng trong PySpark, đóng vai trò như một bảng dữ liệu có cấu trúc, tương tự như DataFrame trong các thư viện khác như Pandas. DataFrame cho phép bạn thực hiện các thao tác dễ dàng trên dữ liệu có cấu trúc.

Trong ví dụ trên, chúng ta đọc dữ liệu từ một tệp CSV và tạo một DataFrame bằng cách sử dụng spark.read.csv. Dòng header=True cho biết rằng tệp CSV có dòng tiêu đề, và inferSchema=True giúp PySpark tự động xác định kiểu dữ liệu của các cột.

Chúng ta sử dụng các phép biến đổi DataFrame như filter, select, và orderBy để xử lý dữ liệu:

filter(col("Salary") > 50000) lọc ra những người có lương lớn hơn 50,000.
select("Name", "Age", "Salary") chọn ra chỉ các cột "Name", "Age", và "Salary".
orderBy("Salary", ascending=False) sắp xếp các dòng theo cột "Salary" theo thứ tự giảm dần.
Cuối cùng, chúng ta hiển thị kết quả bằng cách sử dụng phương thức show() của DataFrame.

DataFrame là một công cụ mạnh mẽ trong PySpark cho phép bạn xử lý và phân tích dữ liệu có cấu trúc một cách hiệu quả trên các dự án xử lý dữ liệu lớn.

IV. Tài liệu tham khảo

1. https://spark.apache.org/docs/latest/api/python/reference/index.html
2. https://spark.apache.org/docs/latest/api/python/index.html
3. https://api-docs.databricks.com/python/pyspark/latest/index.html
4. https://hyukjin-spark.readthedocs.io/en/latest/
5. https://www.palantir.com/docs/foundry/code-workbook/pyspark-reference/

V. Bài tập

1. Đọc và Hiển Thị Dữ Liệu: Đọc dữ liệu từ một tệp CSV chứa thông tin về các sản phẩm (tên, giá, số lượng) và hiển thị năm dòng đầu tiên của dữ liệu.
2. Lọc và Sắp Xếp Dữ Liệu: Sử dụng PySpark để lọc ra các sản phẩm có giá trên 100 và sau đó sắp xếp chúng theo thứ tự giảm dần của giá.
3. Tính Tổng và Trung Bình: Tính tổng và trung bình giá của các sản phẩm sau khi đã lọc theo yêu cầu bài tập số 2.
4. Tạo Thêm Cột: Thêm một cột mới vào DataFrame, ví dụ: "Tổng Giá" bằng cách nhân giá và số lượng của mỗi sản phẩm.

5. Nhóm Dữ Liệu: Nhóm sản phẩm theo danh mục và tính tổng số lượng sản phẩm trong mỗi danh mục.