



CSC17106 – XỬ LÝ PHÂN TÍCH DỮ LIỆU TRỰC TUYẾN

HƯỚNG DẪN THỰC HÀNH

PYSPARK STREAMING

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 về Spark Streaming
2. Giới thiệu netcat window và cài đặt
3. Xây dựng ứng dụng PySpark Streaming đơn giản

III. Mô tả

1. Giới thiệu về Spark Streaming

Apache Spark là một framework xử lý dữ liệu phân tán mạnh mẽ, được phát triển để xử lý dữ liệu lớn nhanh chóng và hiệu quả.

Spark Streaming là một phần của Apache Spark, cho phép xử lý dữ liệu dòng liên kết trong thời gian thực.

Ví dụ một vài trường hợp dùng spark streaming

Xử lý log và giám sát hệ thống thời gian thực: PySpark Streaming có thể được sử dụng để thu thập, phân tích và giám sát log hệ thống và ứng dụng thời gian thực. Bằng cách xử lý log ngay khi chúng được tạo ra, bạn có thể theo dõi hiệu suất hệ thống và phát hiện sớm các vấn đề hoặc lỗi.

Phân tích dữ liệu thời gian thực từ các nguồn sensor: PySpark Streaming cho phép bạn xử lý dữ liệu từ các cảm biến và nguồn dữ liệu IoT (Internet of Things) trong thời gian thực. Điều này có thể được ứng dụng trong các lĩnh vực như quản lý năng lượng, giám sát môi trường, và quản lý tài nguyên.

Xử lý dữ liệu trực tiếp từ Kafka: PySpark Streaming tích hợp tốt với Apache Kafka, cho phép bạn xử lý dữ liệu dòng liên kết từ hàng triệu nguồn một cách hiệu quả. Điều này thường được sử dụng trong việc xây dựng hệ thống xử lý sự kiện thời gian thực và hệ thống tập trung dữ liệu.

Xử lý dữ liệu trực tiếp từ nguồn web và truyền hình: PySpark Streaming cũng có thể được sử dụng để xử lý dữ liệu từ nguồn web và truyền hình trực tiếp. Điều này có thể áp dụng trong các tình huống như phân tích trực tiếp các dữ liệu xã hội, tin tức thời sự, hoặc phát hiện sự kiện trực tiếp từ các nguồn truyền hình.

Xây dựng hệ thống gợi ý thời gian thực: PySpark Streaming có thể được sử dụng để xây dựng hệ thống gợi ý dựa trên dữ liệu thời gian thực. Ví dụ, hệ thống có thể gợi ý sản phẩm, nội dung, hoặc dịch vụ dựa trên hành vi của người dùng trong thời gian thực.

Giám sát mạng và bảo mật: PySpark Streaming có thể được sử dụng để giám sát mạng và phát hiện các hoạt động không bình thường hoặc tấn công bảo mật trong thời gian thực. Điều này giúp cải thiện bảo mật hệ thống và mạng.

Xây dựng hệ thống quảng cáo thời gian thực: PySpark Streaming có thể được sử dụng để xây dựng hệ thống quảng cáo thời gian thực, giúp các doanh nghiệp hiển thị quảng cáo đối tượng dựa trên hành vi và sở thích của người dùng trong thời gian thực.

2. Xây dựng ứng dụng xử lý dữ liệu thời gian thực đơn giản

Giả sử dữ liệu text được truyền từ socket có port 9999, spark streaming xử lý bằng cách đếm tần suất xuất hiện của các từ từ text stream

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *

# Create the SparkSession
spark =
SparkSession.builder.appName("sparkstream").getOrCreate()

spark.sparkContext.setLogLevel('WARN')

# Define input sources
lines = (spark.readStream.format("socket").option("host",
"localhost").option("port", 9999).load())

# Transform data
words = lines.select(split(col("value"), "\\s").alias("word"))

# Get the count of published words
counts = words.groupBy("word").count()

# Define the checkpoint directory
checkpointDir = "C:/checkpoint/"
```

```
# Start streaming defining the necessary configurations
streamingQuery = (counts.writeStream.format("console")
    .outputMode("complete").trigger(processingTime="1 second")
    .option("checkpointLocation", checkpointDir).start())

streamingQuery.awaitTermination()
```

Giải thích:

```
spark = SparkSession.builder.appName("sparkstream").getOrCreate()
```

Khởi tạo ứng dụng spark với tên ứng dụng "sparkstream"

Trong quá trình chạy spark, có nhiều thông tin (INFO) được thông báo dẫn đến khó phát triển, để tắt bớt thông tin này, chúng ta có thể dùng lệnh sau:

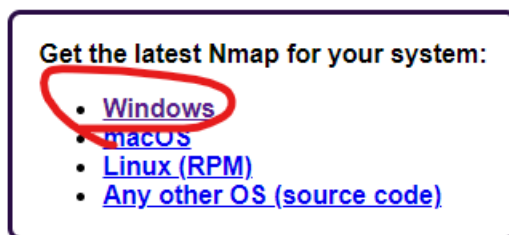
```
spark.sparkContext.setLogLevel('WARN')
```

Chỉ hiển thị thông tin WARN và ERROR, các thông tin còn lại không xuất ra màn hình console

```
lines =
(spark.readStream.format("socket").option("host", "localhost").option
("port", 9999).load())
```

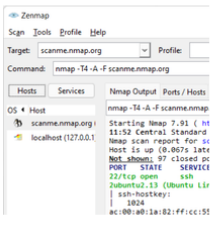
Dòng lệnh trên đang định nghĩa input data từ socket với port 9999. Để giả lập dữ liệu từ port này ta có thể dùng công cụ netcat cho Window.

Cài đặt netcat tại đây: <https://nmap.org/download.html>



Sau đó tải file bên dưới và cài đặt theo từng bước hướng dẫn từ chương trình cài đặt.

Microsoft Windows binaries



Please read the [Windows section](#) of the Install Guide for limitations and installation instructions for the Windows version of Nmap. It's provided as an executable self-installer which includes Nmap's dependencies and the Zenmap GUI. We support Nmap on Windows 7 and newer, as well as Windows Server 2008 R2 and newer. We also maintain a [guide for users who must run Nmap on earlier Windows releases](#).

Note: The version of Npcap included in our installers may not always be the latest version. If you experience problems or just want the latest and greatest version, download and install [the latest Npcap release](#).

Latest stable release self-installer: [nmap-7.94-setup.exe](#)
Latest Npcap release self-installer: [npcap-1.7.6.exe](#)

We have written [post-install usage instructions](#). Please [notify us](#) if you encounter any problems or have suggestions for the installer.

Linux RPM Source and Binaries

Many popular Linux distributions (Redhat, Mandrake, Suse, etc) use the [RPM](#) package management system for quick and easy binary package installation. We have written a detailed [guide to installing our RPM packages](#), though these simple commands usually do the trick:

```
rpm -vhu https://nmap.org/dist/nmap-7.94-1.x86_64.rpm
rpm -vhu https://nmap.org/dist/zenmap-7.94-1.noarch.rpm
rpm -vhu https://nmap.org/dist/ncat-7.94-1.x86_64.rpm
rpm -vhu https://nmap.org/dist/nping-0.7.94-1.x86_64.rpm
```

Sau khi cài đặt bạn có thể dùng lệnh để input từ netcat: `ncat -lv -p 9999`

```
C:\Users\PC2023>ncat -lv -p 9999
Ncat: Version 7.94 ( https://nmap.org/ncat )
Ncat: Listening on [::]:9999
Ncat: Listening on 0.0.0.0:9999
```

Tại thời điểm chạy lệnh này, thông tin hiển thị cho thấy chương trình đang chờ kết nối từ các app tại port 9999, sau khi kết nối thành công thì có thể input tại CMD.

Ví dụ chạy chương trình pyspark streaming đề cập trên theo lệnh sau từ CMD:

```
Microsoft Windows [Version 10.0.22621.2283]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PC2023>spark-submit D:\workspace\spark\demo_spark.py
```

Lúc này cửa sổ ncat sẽ thông báo như sau:

```
C:\Users\PC2023>ncat -lv -p 9999
Ncat: Version 7.94 ( https://nmap.org/ncat )
Ncat: Listening on [::]:9999
Ncat: Listening on 0.0.0.0:9999
Ncat: Connection from 127.0.0.1:5531.
```

Báo hiệu đã có ứng dụng spark kết nối, bây giờ ta có thể truyền bất cứ chuỗi nào:

```
C:\Users\PC2023>ncat -lv -p 9999
Ncat: Version 7.94 ( https://nmap.org/ncat )
Ncat: Listening on [::]:9999
Ncat: Listening on 0.0.0.0:9999
Ncat: Connection from 127.0.0.1:5531.
hello world
tu
hello
world
I am tu
```

Chương trình spark theo thời gian sẽ tính toán đếm số lần xuất hiện các word (từ) trên các chuỗi vừa nhập từ ncat theo thời gian thực.

```
-----+-----+
|               word|count|
|-----+-----+
| [hello, world, ]| 1|
|               [tu]| 2|
| [i, am, here]| 1|
| [hello, world]| 3|
|               [hello]| 5|
|               [cannot]| 3|
|               [world]| 1|
|               [I, am, tu]| 2|
| [hello, world, he...]| 1|
|-----+-----+

23/09/17 12:16:26 WARN ProcessingTimeExecutor: Current batch is falling behind. The trigger interval is 1000 milliseconds, but
spent 17996 milliseconds
```

Trở lại các lệnh sau:

```
words = lines.select(split(col("value"), "\\s").alias("word"))
```

lines: Đây có thể là một DataFrame hoặc một tập dữ liệu được gọi là "lines". DataFrame là một cấu trúc dữ liệu phân tán trong Spark, giống như một bảng trong cơ sở dữ liệu, với các cột và hàng.

split(col("value"), "\\s"): Đây là một phần quan trọng của câu lệnh. Nó đang sử dụng hàm split để chia dữ liệu trong cột "value" thành các từ riêng biệt. Tham số thứ hai của hàm split là biểu thức chính quy định cách chia tách dữ liệu. Trong trường hợp này, \\s đại diện cho khoảng trắng (space), vì vậy câu lệnh này đang cố gắng chia các giá trị trong cột "value" dựa trên khoảng trắng.

.alias("word"): Phần này đặt tên cho cột kết quả được tạo ra sau khi sử dụng hàm split. Cột mới này sẽ có tên là "word".

words: Cuối cùng, kết quả của câu lệnh sẽ được gán cho biến "words". Điều này có nghĩa là biến "words" sẽ chứa một DataFrame mới với một cột duy nhất được đặt tên là "word", chứa các từ đã tách ra từ cột "value" của "lines" DataFrame.

```
counts = words.groupBy("word").count()
```

words: Đây là một DataFrame chứa các từ đã tách ra từ dữ liệu trong cột "value" của "lines", như đã được giải thích trong câu lệnh trước.

groupBy("word"): Đây là một phần quan trọng của câu lệnh. Nó đang nhóm các dòng của DataFrame "words" dựa trên giá trị trong cột "word". Điều này có nghĩa là nó sẽ tạo các nhóm dựa trên các từ duy nhất trong cột "word". Mỗi nhóm sẽ chứa tất cả các dòng có cùng giá trị trong cột "word".

count(): Sau khi đã nhóm các dòng theo từng từ, hàm count() được sử dụng để đếm số lượng dòng trong mỗi nhóm. Nó sẽ tính toán số lần mỗi từ xuất hiện trong cột "word" của DataFrame "words".

```
streamingQuery = (counts.writeStream.format("console")
    .outputMode("complete").trigger(processingTime="1 second")
    .option("checkpointLocation", checkpointDir).start())
```

counts: Đây là DataFrame chứa các từ duy nhất và số lần xuất hiện tương ứng của chúng trong dữ liệu đang xử lý.

`writeStream`: Phương thức `writeStream` được gọi trên `DataFrame "counts"` để bắt đầu quá trình viết kết quả của xử lý streaming.

`format("console")`: Đây là phần quan trọng để xác định định dạng đầu ra của kết quả. Trong trường hợp này, đầu ra sẽ được xuất ra màn hình console.

`outputMode("complete")`: Xác định chế độ xuất ra. Ở đây, chế độ "complete" được chọn, điều này có nghĩa là toàn bộ kết quả sẽ được xuất ra mỗi lần xử lý hoàn tất.

`trigger(processingTime="1 second")`: Đây là phần quyết định tần suất cập nhật kết quả. Trong trường hợp này, mỗi giây sẽ có một lần cập nhật và xuất kết quả.

`option("checkpointLocation", checkpointDir)`: Đây là tùy chọn để xác định vị trí lưu trữ dữ liệu trạng thái (checkpoint) cho việc xử lý streaming. Điều này quan trọng để đảm bảo tính nhất quán và khả năng phục hồi khi xảy ra sự cố.

`start()`: Cuối cùng, phương thức `start()` được gọi để bắt đầu xử lý streaming và bắt đầu việc xuất kết quả lên console.

```
streamingQuery.awaitTermination()
```

Câu lệnh `streamingQuery.awaitTermination()` là một câu lệnh chặn (blocking) trong Apache Spark Streaming. Nó được sử dụng để đợi cho đến khi một streaming query hoặc quá trình xử lý streaming kết thúc hoặc bị ngừng vì một lý do nào đó.

Khi bạn gọi `awaitTermination()` trên một streaming query, mã của bạn sẽ bị chặn, và nó sẽ đợi cho đến khi một trong các điều kiện sau xảy ra:

1. Streaming query hoàn thành: Nếu quá trình xử lý streaming đã xử lý hết tất cả dữ liệu và đã đạt được điều kiện kết thúc, nó sẽ kết thúc một cách tự nhiên.
2. Streaming query bị ngừng vì lỗi: Nếu có lỗi xảy ra trong quá trình xử lý streaming và quá trình dừng lại, `awaitTermination()` sẽ trả về với một ngoại lệ hoặc thông báo lỗi.
3. `awaitTermination()` bị hủy bởi một sự kiện ngoại việc khác: Bạn có thể hủy bỏ chặn bằng cách gọi `streamingQuery.stop()` từ một tiến trình khác hoặc bằng một sự kiện khác trong mã của bạn.

Chức năng chính của `awaitTermination()` là đảm bảo rằng chương trình của bạn không kết thúc cho đến khi quá trình xử lý streaming đã hoàn thành hoặc xảy ra lỗi. Điều này thường được sử dụng để đảm bảo rằng bạn không bị chạy xong và thoát trước khi quá trình xử lý streaming đã hoàn thành công việc của nó.

IV. Tài liệu tham khảo

1. [Apache Spark Streaming Documentation](#)
2. [Apache Spark Streaming Programming Guide](#)
3. [Databricks - Structured Streaming](#)
4. [Learning Spark](#)
5. [Apache Spark Streaming with Python and PySpark](#)



6. [GitHub Spark Streaming Examples](#)
7. [TutorialsPoint - Apache Spark Streaming with PySpark](#)

V. Bài tập

Tính tổng các số nguyên từ Netcat sử dụng PySpark Streaming

Mục tiêu: Xây dựng một ứng dụng PySpark Streaming để đọc và tính tổng các số nguyên được gửi từ Netcat server.

Yêu cầu:

Khởi động Netcat server: Trước hết, hãy mở một cửa sổ terminal và chạy lệnh sau để khởi động Netcat server lắng nghe trên cổng 9999:

```
nc -lk 9999
```

Dữ liệu (các số nguyên) sẽ được gửi đến máy chủ Netcat từ cửa sổ terminal khác.

Xây dựng ứng dụng PySpark Streaming: Bạn cần viết một ứng dụng PySpark Streaming để đọc dữ liệu từ máy chủ Netcat và tính tổng các số nguyên được gửi đến.

Xử lý dữ liệu: Trong ứng dụng PySpark Streaming của bạn, hãy xử lý dữ liệu số nguyên đến từ Netcat và tính tổng của chúng.

In kết quả: Hiển thị tổng các số nguyên lên màn hình console sau mỗi khoảng thời gian nhất định.

Cấu hình checkpoint: Đảm bảo bạn đã cấu hình checkpoint để đảm bảo tính nhất quán và khả năng phục hồi của ứng dụng.

Sử dụng awaitTermination(): Sử dụng awaitTermination() để đợi cho đến khi xử lý streaming kết thúc hoặc bị ngừng vì lỗi.

Gợi ý:

Sử dụng `spark.readStream.format("socket")` để đọc dữ liệu từ máy chủ Netcat vào DataFrame.

Sử dụng các phương thức và hàm của PySpark để xử lý dữ liệu trong DataFrame.

Để đảm bảo bạn đã cấu hình checkpoint, hãy sử dụng `.option("checkpointLocation", "/your/checkpoint/directory")` khi gọi `.writeStream`.

In kết quả tổng ra màn hình bằng cách sử dụng `.outputMode("complete")`.

Bài tập này giúp bạn làm quen với việc sử dụng PySpark Streaming để xử lý và tính toán trực tiếp trên dữ liệu đến từ Netcat.