



**BỘ MÔN: BIG DATA APPLICATIONS:**

**REAL-TIME STREAMING**

**\*\*\***

**BÁO CÁO GIỮA KÌ**

**-------**

**SPARK SLIDING WINDOW COMPUTATIONS**

Mã môn học : **231BDAS436177**

Giáo viên hướng dẫn : **ThS. Lê Minh Tân**

Nhóm thực hiện : **Nhóm 12**

TP. Hồ Chí Minh, tháng 10 năm 2023

**DANH SÁCH THÀNH VIÊN NHÓM 12**

***HỌC KỲ I NĂM HỌC 2023-2024***

**Tên đề tài:** Spark sliding window computations

|  |  |  |  |
| --- | --- | --- | --- |
| STT | HỌ TÊN | MSSV | TỶ LỆ CÔNG VIỆC |
| 1 | Văn Hoàng Lương | 20133064 | 100% |
| 2 | Đặng Xuân Bách | 20133023 | 100% |
| 3 | Nguyễn Ngọc Hoài | 20133043 | 100% |
| 4 | Phan Quốc Lưu | 20133065 | 100% |

Nhận xét của giáo viên:

...................................................................................................................................

...................................................................................................................................

...................................................................................................................................

...................................................................................................................................

...................................................................................................................................

...................................................................................................................................

...................................................................................................................................

...................................................................................................................................

...................................................................................................................................

...................................................................................................................................

...................................................................................................................................

..................................................................................................................................

**MỤC LỤC**

[**I.** **PHẦN MỞ ĐẦU** 1](#_Toc147868967)

[**II.** **NỘI DUNG** 1](#_Toc147868968)

[**1.** **Định nghĩa sliding window computations** 1](#_Toc147868969)

[**2.** **Tại sao phải áp dụng spark sliding window computations trong xử lý dữ liệu streaming** 2](#_Toc147868970)

[**3.** **Cách hoạt động của spark sliding window computations trong xử lý dữ liệu** 2](#_Toc147868971)

[**4.** **Ưu và nhược điểm khi sử dụng spark sliding window computations** 3](#_Toc147868972)

[**5.** **Ứng dụng spark sliding window computations vào thực tiễn** 3](#_Toc147868973)

[**III.** **Demo: Áp dụng spark sliding window computations đếm số từ xuất hiện trong mỗi cửa sổ thời gian được truyền** 4](#_Toc147868974)

[**1. Mô tả sơ lược** 4](#_Toc147868975)

[**2. Cách thực hiện** 4](#_Toc147868976)

[**2.1. Khởi động netcat** 4](#_Toc147868977)

[**2.2. Khởi động demo** 4](#_Toc147868978)

[**2.3. Gửi dữ liệu đến socket server** 5](#_Toc147868979)

[**2.4. Quan sát kết quả** 5](#_Toc147868980)

[**3. Giải thích** 5](#_Toc147868981)

[**IV.** **Demo: Áp dụng spark sliding window computations để đếm số lượng tin nhắn** 7](#_Toc147868982)

[**1. Mô tả sơ lược** 7](#_Toc147868983)

[**2. Cách thực hiện** 7](#_Toc147868984)

[**2.1. Thêm Spark dependency** 7](#_Toc147868985)

[**2.2. Chuẩn bị dữ liệu** 8](#_Toc147868986)

[**2.3. Khởi động demo** 9](#_Toc147868987)

[**3. Giải thích** 9](#_Toc147868988)

[**3.1. Tạo SparkSession** 9](#_Toc147868989)

[**3.2. Định nghĩa Schema cho dữ liệu** 9](#_Toc147868990)

[**3.3. Đọc dữ liệu và tạo View tạm thời** 10](#_Toc147868991)

[**3.4. Xử lý dữ liệu với cửa sổ thời gian trượt** 10](#_Toc147868992)

[**TÀI LIỆU THAM KHẢO** 11](#_Toc147868993)

# 

# **PHẦN MỞ ĐẦU**

**Apache Spark** là một nền tảng mạnh mẽ cho xử lý dữ liệu lớn, đã trở thành một trọng tâm quan trọng trong ngành công nghiệp dữ liệu. Được phát triển tại Đại học California, Spark là một hệ thống mã nguồn mở cho phép xử lý dữ liệu phân tán trên các cụm máy tính lớn.

Spark có khả năng xử lý hàng tỷ dòng dữ liệu trên nhiều máy tính song song, giúp tăng cường hiệu suất xử lý dữ liệu so với các hệ thống truyền thống.

Một số yếu tố quan trọng của Spark bao gồm tích hợp xử lý dữ liệu thời gian, khả năng phân tích dữ liệu phức tạp, hỗ trợ cho **machine learning** và các thư viện mạnh mẽ để xử lý dữ liệu lớn.

Dữ liệu dạng dòng thời gian thường là dữ liệu ghi lại theo thời gian, chẳng hạn như dữ liệu từ cảm biến **IoT**, dữ liệu log hệ thống, dự đoán thời tiết và giao dịch tài chính. Điều này có nghĩa là nó chứa thông tin về sự biến đổi của dữ liệu theo thời gian. Một trong những thách thức lớn trong việc xử lý dữ liệu thời gian là khả năng phân tích và trích xuất thông tin có ý nghĩa từ dữ liệu liên quan đến thời gian. Điều này đặc biệt quan trọng trong các lĩnh vực như dự đoán sự cố, theo dõi xu hướng thời gian thực và đưa ra quyết định dựa trên dữ liệu thời gian.

Tính toán trên cửa sổ trượt (**sliding window computations**) là một kỹ thuật quan trọng trong xử lý dữ liệu thời gian. Nó cho phép chúng ta tạo ra các khung thời gian nhỏ và liên tục để thực hiện tính toán trên dữ liệu trong từng khung thời gian này. Điều này giúp chúng ta theo dõi các biến đổi của dữ liệu theo thời gian và phát hiện sự cố hoặc xu hướng.

1. **NỘI DUNG**

## **Định nghĩa sliding window computations**

**Sliding window computations** là một kỹ thuật quan trọng trong xử lý dữ liệu thời gian, cho phép chúng ta thực hiện tính toán đối với dữ liệu theo từng phạm vi thời gian nhất định. Cửa sổ trượt di chuyển qua dữ liệu dạng dòng thời gian và được sử dụng để tạo ra các khung dữ liệu con (**window**) có kích thước cố định để thực hiện tính toán.

Cửa sổ trượt (**sliding window**): Đây là một khung thời gian có kích thước xác định và được sử dụng để xác định phạm vi dữ liệu mà chúng ta muốn thực hiện tính toán. Cửa sổ trượt di chuyển liên tục qua dữ liệu theo một khoảng thời gian nhất định.

Tính toán (**computations**): Đây là quá trình thực hiện các phép tính, thống kê hoặc các hoạt động xử lý dữ liệu trên dữ liệu nằm trong cửa sổ trượt. Các tính toán này có thể bao gồm việc tính giá trị trung bình, tổng hợp, phát hiện sự cố, hay bất kỳ phân tích nào liên quan đến thời gian.

Trong lĩnh vực tài chính, cửa sổ trượt có vai trò quan trọng trong việc theo dõi giá cổ phiếu. Hãy xem xét trường hợp bạn đang quản lý một quỹ đầu tư và muốn theo dõi giá trị trung bình của một cổ phiếu trong khoảng thời gian 30 ngày để xác định xu hướng giá cổ phiếu. Bạn sẽ sử dụng cửa sổ trượt có kích thước 30 ngày để tạo ra các khung thời gian liên tục và trượt qua dữ liệu giá cổ phiếu hàng ngày. Trong mỗi cửa sổ trượt, bạn tính giá trị trung bình của giá cổ phiếu trong khoảng 30 ngày đó. Như cửa sổ di chuyển qua thời gian, bạn có thể theo dõi sự thay đổi của giá trị trung bình, giúp bạn xác định xu hướng tăng hoặc giảm của giá cổ phiếu. Nếu giá trị trung bình vượt qua một ngưỡng nhất định, bạn có thể quyết định thực hiện các hành động như mua hoặc bán cổ phiếu.

## **Tại sao phải áp dụng spark sliding window computations trong xử lý dữ liệu streaming**

Trong xử lý dữ liệu streaming, việc áp dụng *spark sliding window computations* là một quy trình quan trọng nhằm nắm bắt sự biến động của dữ liệu qua thời gian. Cửa sổ trượt cho phép quan sát thông tin trong các khoảng thời gian liên tiếp, tạo điều kiện cho việc xác định xu hướng và thay đổi của dữ liệu một cách chi tiết và độc lập với thời điểm cụ thể.

Chức năng chính của cửa sổ trượt là cung cấp khả năng tổng hợp và thống kê dữ liệu trong các khoảng thời gian cụ thể, từ đó hỗ trợ quyết định dựa trên các thông tin tổng hợp. Điều này đặc biệt quan trọng khi xử lý dữ liệu liên tục từ nguồn đến nguồn mà không làm gián đoạn quá trình xử lý.

Ngoài ra, cửa sổ trượt cũng giúp phát hiện sự kiện đặc biệt thông qua việc so sánh dữ liệu trong các cửa sổ liên tiếp. Khả năng này giúp nhanh chóng phát hiện sự thay đổi đột ngột hoặc bất thường trong dữ liệu, có ứng dụng quan trọng trong việc giám sát và điều khiển hệ thống thời gian thực.

Tổng kết, sự kết hợp giữa *spark sliding window computations* và xử lý dữ liệu streaming mang lại khả năng hiểu rõ hơn về đặc điểm của dữ liệu theo thời gian, đồng thời giúp tối ưu hóa quy trình xử lý trong môi trường thời gian thực.

## **Cách hoạt động của spark sliding window computations trong xử lý dữ liệu**

Có thể tóm tắt hoạt động của spark sliding window computations như sau:

**Chia thành cửa sổ (window):** Dữ liệu streaming được chia thành các cửa sổ (window) có kích thước và độ trượt được xác định trước. Kích thước của cửa sổ xác định khoảng thời gian mà mỗi cửa sổ bao gồm, trong khi độ trượt quyết định khoảng thời gian giữa các cửa sổ liên tiếp.

**Xác định thời điểm bắt đầu và kết thúc:** Mỗi cửa sổ sẽ có thời điểm bắt đầu và thời điểm kết thúc của nó, di chuyển theo quy tắc độ trượt. Ví dụ, nếu cửa sổ có kích thước 1 phút (*windowDuration*) và độ trượt là 30 giây (*slideDuration*), thì cửa sổ mới sẽ bắt đầu mỗi 30 giây.

**Áp dụng hàm xử lý:** Trong mỗi cửa sổ, một hàm xử lý được áp dụng cho dữ liệu. Điều này có thể là bất kỳ loại xử lý nào mà người lập trình đặt ra, như tính tổng, trung bình, độ lệch chuẩn, hoặc các phép toán phức tạp hơn tùy thuộc vào yêu cầu cụ thể của ứng dụng.

**Cập nhật kết quả:** Kết quả của hàm xử lý được cập nhật liên tục theo thời gian và được gán cho mỗi cửa sổ. Điều này tạo ra một dòng dữ liệu kết quả mới cho mỗi cửa sổ, thường được gửi đến hệ thống đích hoặc lưu trữ để phân tích sau này.

1. **Ưu và nhược điểm khi sử dụng spark sliding window computations**

Ưu điểm:

* **Phân tích thời gian thực:** Spark sliding window computations cho phép phân tích thời gian thực hoặc gần thời gian thực bằng cách xử lý dữ liệu trong khoảng thời gian cụ thể.
* **Xử lý dữ liệu hiệu quả:** Giảm lượng dữ liệu cần xử lý trong một thời điểm nhất định, từ đó dẫn đến việc sử dụng tài nguyên hệ thống hiệu quả hơn.
* **Tính linh hoạt:** Người dùng có thể định nghĩa kích thước và khoảng trượt của các cửa sổ dựa trên yêu cầu cụ thể và tính chất của dữ liệu.
* **Phân tích theo ngữ cảnh:** Cho phép phân tích theo ngữ cảnh, trong đó dữ liệu được phân tích liên quan đến các điểm dữ liệu xung quanh. Điều này quan trong cho các tác vụ như phát hiện bất thường hoặc xác định xu hướng.

Nhược điểm:

* **Yêu cầu nhiều tài nguyên:** Spark sliding window computations có thể đòi hỏi nhiều tài nguyên, đặc biệt khi xử lý lượng dữ liệu lớn trong thời gian thực.
* **Phức tạp trong quản lý các cửa sổ:** Việc xử lý cửa sổ trượt đưa ra thêm độ phức tạp về việc quản lý trạng thái cửa sổ, định nghĩa biên giới cửa sổ và xử lý dữ liệu đến trễ.
* **Tăng thời gian xử lý:** Tùy thuộc vào kích thước của cửa sổ và khoảng trượt, việc xử lý dữ liệu trong các cửa số trượt có thể làm tăng tổng thời gian xử lý so với việc xử lý theo lô.

1. **Ứng dụng spark sliding window computations vào thực tiễn**

* **IoT và quản lý thiết bị kết nối:** Với sự gia tăng của Internet of Things (IoT), cửa sổ trượt có thể được sử dụng để theo dõi và phân tích dữ liệu từ các thiết bị kết nối trong thời gian thực.
* **Y tế:** Cửa sổ trượt có thể được sử dụng để theo dõi và phân tích dữ liệu y tế thời gian thực, bao gồm theo dõi dấu hiệu sinh học, quản lý dữ liệu điều trị và phát hiện bất thường sớm.
* **Hệ thống an ninh và giám sát:** áp dụng cửa sổ trượt cho phân tích dữ liệu từ hệ thống an ninh và giám sát có thể giúp trong việc phát hiện sự cố và hành vi bất thường ngay lập tức.
* **Game:** Cửa sổ trượt có thể được sử dụng trong ngành công nghiệp game để theo dõi và phân tích hành vi của người chơi trong thời gian thực, cung cấp trải nghiệm tùy chỉnh hơn.

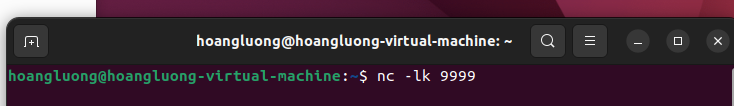
1. **Demo: Áp dụng spark sliding window computations đếm số từ xuất hiện trong mỗi cửa sổ thời gian được truyền**

## **1. Mô tả sơ lược**

Ứng dụng minh họa sử dụng Apache Spark Streaming để đếm số từ xuất hiện trong cửa sổ trượt. Trong ví dụ này, chúng ta sử dụng Python và Spark để tạo một ứng dụng đơn giản. Chương trình lắng nghe dữ liệu từ một nguồn socket cục bộ, sau đó chia dòng thành các từ và đếm số lần xuất hiện của mỗi từ trong một cửa sổ thời gian được xác định.

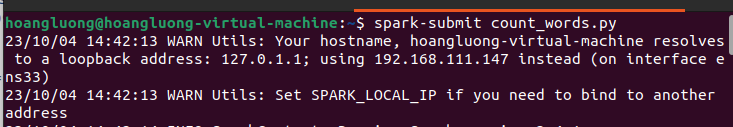
## **2. Cách thực hiện**

### **2.1. Khởi động netcat**

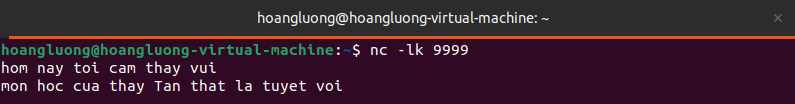
Khởi động netcat để tạo một socket server (*localhost:9999*)

Khởi động netcat

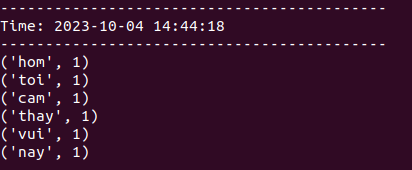
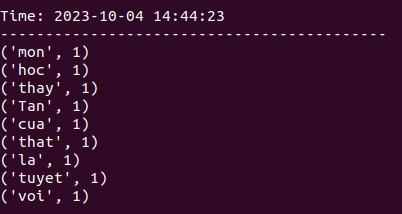
**2.2. Khởi động demo**

Mở 1 *terminal* khác và thực hiện chạy demo

Khởi động demo đếm từ

**2.3. Gửi dữ liệu đến socket server**

Gửi dữ liệu đến socket server

**2.4. Quan sát kết quả**

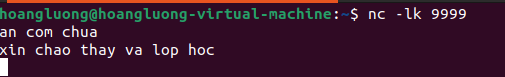
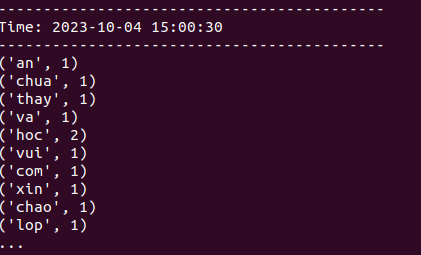
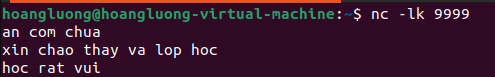
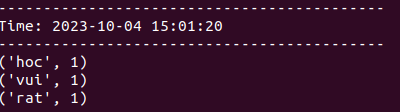
Quan sát kết quả

Quan sát kết quả

**3. Giải thích**

*Spark sliding window computations* được đặc trưng bởi:

+ **WindowDuration:** Thời gian cửa sổ tồn tại, ví dụ ta đặt windowDuration là 60 giây thì spark sẽ tính toán tất cả các dữ liệu trong vòng 60 giây đó.

+ **SlideDuration:** là bước trượt, ví dụ windowDuration tồn tại 60 giây và slideDuration là 10 giây thì mỗi lần trượt, spark sẽ tính toán dữ liệu trong vòng 10 giây và trượt 6 lần trên windowDuration.

Gửi dữ liệu đến server

Quan sát kết quả

Quan sát kết quả

Gửi dữ liệu đến socket server

Sửa windowDuration và slideDuration

Cài đặt windowDuration là 10s và slideDuration là 5s. Khi gửi dữ liệu ở 2.3, ta thấy có 2 dòng dữ liệu (nhập cách nhau > 5s). Quan sát kết quả, ta thấy rằng spark sliding window computations hoạt động đếm số từ ở các dòng dữ liệu đó và in ra kết quả ở 2 thời gian khác nhau.

Sau đó ta quan sát kết quả, nếu ta gửi dữ liệu lên socket server trong vòng 100s thì spark sẽ đếm tất cả các từ xuất hiện theo từng giai đoạn 50s. Khi hết 100s thì sẽ tạo 1 cửa sổ mới và thực hiện tương tự.

1. **Demo: Áp dụng spark sliding window computations để đếm số lượng tin nhắn**

## **1. Mô tả sơ lược**

Ứng dụng demo Spark Structured Streaming với cửa sổ thời gian trượt là một ví dụ thực hành về cách sử dụng Apache Spark để xử lý dữ liệu streaming trong môi trường thời gian thực. Đoạn mã Scala triển khai một luồng xử lý dữ liệu từ folder với tập dữ liệu file txt hai cột chính là "*Date*" (kiểu thời gian) và "*Message*" (kiểu chuỗi). Ứng dụng này sử dụng cửa sổ thời gian trượt 30 giây (tùy chỉnh) để tổng hợp và đếm số lượng tin nhắn trong mỗi cửa sổ.

Bằng cách đọc dữ liệu từ folder và xác định cấu trúc dữ liệu, ứng dụng tạo một temp view cho phép sử dụng SQL trực tiếp trên DataFrame. Các cửa sổ thời gian trượt và watermark được áp dụng để quản lý thời gian và xử lý dữ liệu trễ. Kết quả được xuất ra console và hiển thị số lượng tin nhắn được đếm trong từng cửa sổ thời gian.

Demo này mang lại cách Spark Structured Streaming có thể được tích hợp để tổng hợp và phân tích dữ liệu streaming, với khả năng quản lý dữ liệu không theo thứ tự một cách hiệu quả.

## **2. Cách thực hiện**

### **2.1. Thêm Spark dependency**

Mở tệp built.sbt trong dự án và thêm dependency cho Spark

A screenshot of a computer program

Description automatically generated

Thêm dependenceies trong tệp build.sbt

### **2.2. Chuẩn bị dữ liệu**

Tạo một tệp dữ liệu (file 1.txt) chứa một chuỗi các tin nhắn được gửi đi theo thời gian, có hai cột là Date (đúng định dạng *yyyy-mm-dd hh:mm:ss*) và Message (tin nhắn)

A screenshot of a computer

Description automatically generated

Chuẩn bị dữ liệu (1.txt)

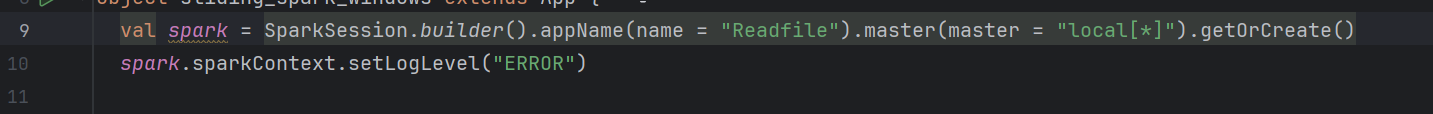
### **2.3. Khởi động demo**

*A screenshot of a computer

Description automatically generatedKết quả trả về*

## **3. Giải thích**

### **3.1. Tạo SparkSession**

Tạo một phiên SparkSession. *setLogLevel*("*ERROR*") được sử dụng để chỉ hiển thị log cấp độ lỗi để giảm sự nhiễu loạn.

Tạo SparkSession

### **3.2. Định nghĩa Schema cho dữ liệu**

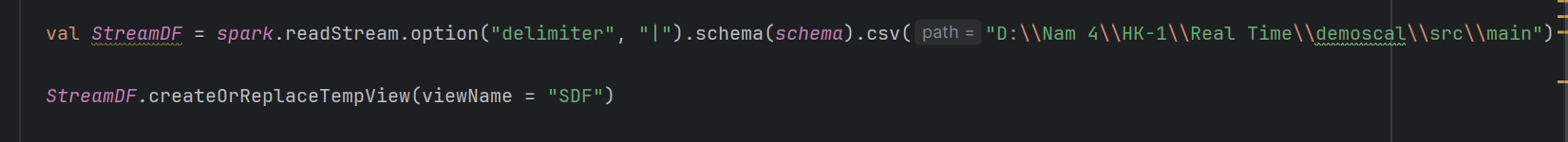
Xác định schema cho dữ liệu streaming, với hai cột là "Date" có kiểu *Timestamp* và "Message" có kiểu *StringType*.

A screen shot of a computer code

Description automatically generated

Định nghĩa schema cho dữ liệu

### **3.3. Đọc dữ liệu và tạo View tạm thời**

Đọc dữ liệu streaming từ file 1.txt trong folder với schema đã được định nghĩa. Cấu hình option("*delimiter*", "|") cho biết rằng dữ liệu trong file 1.txt trong folder được phân tách bằng dấu "|".

Tạo một View tạm thời từ DataFrame để sử dụng với các truy vấn SQL

### **3.4. Xử lý dữ liệu với cửa sổ thời gian trượt**

Xác định watermark cho cửa sổ thời gian trượt giúp loại bỏ các sự kiện cũ hơn khỏi cửa sổ thời gian.

Sử dụng cửa sổ thời gian trượt để tạo các cửa sổ thời gian (windowDuration) 30 giây, với mỗi cửa sổ trượt mỗi 30 giây, và đếm số lượng tin nhắn trong mỗi cửa sổ thời gian này. Kết quả cuối cùng là một *DataFrame* mới có cột "*MessageCount*" thể hiện số lượng tin nhắn trong từng cửa sổ thời gian.

Nếu tập dữ liệu mới có thời gian nằm trong khoảng watermark và thuộc về một cửa sổ thời gian mà ứng dụng đang theo dõi, Spark sẽ tính toán lại kết quả cho cửa sổ đó và xuất kết quả mới lên console.

# **TÀI LIỆU THAM KHẢO**

1. Slide bài giảng ‘Practical data streaming’. Thầy Lê Minh Tân. 2023.
2. Spark: The Definitive Guide. Bill Chambers & Matei Zaharia. 2018.
3. Perform Window Operations during Spark Structured Streaming. Ray han. 2023. <https://www.projectpro.io/recipes/perform-window-operations-during-spark-structured-streaming>
4. Spark Streaming Window Operations – A Quick Guide. Techvidvan. 2023.

<https://techvidvan.com/tutorials/spark-streaming-window-operations/>

1. Spark-course. Talent Origin. 2018.

<https://github.com/TalentOrigin/spark-course>

1. Spark Streaming Log files in a Directory. GK Codelabs. 2020.

<https://www.youtube.com/watch?v=4dWHu-3nosE>