



**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:

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

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

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

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

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

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

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

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

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

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

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

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

# **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, Berkeley, 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.

## **Tổng quan về Apache Spark**

**Apache Spark** là một *framework* xử lý dữ liệu phân tán mã nguồn mở, đã trở thành một công cụ quan trọng trong việc xử lý và phân tích dữ liệu lớn.

**Mô hình xử lý dữ liệu phân tán**: Một trong những điểm mạnh đáng chú ý của Spark là khả năng phân tán tính toán trên nhiều máy tính trong một cụm. Điều này cho phép Spark xử lý lượng dữ liệu lớn mà không gặp sự hạn chế của một máy tính duy nhất.

**Xử lý dữ liệu thời gian**: Spark được thiết kế để hỗ trợ xử lý dữ liệu dạng dòng thời gian (time series data) một cách hiệu quả. Điều này làm cho Spark trở thành một lựa chọn lý tưởng cho các ứng dụng liên quan đến thời gian như theo dõi sự cố thời gian thực, phân tích log hệ thống và dự đoán thời tiết.

**Hiệu suất cao**: Spark được tối ưu hóa để làm việc với dữ liệu lớn mà không cần phải sao lưu dữ liệu trung gian sau mỗi bước tính toán. Điều này giúp giảm thiểu việc truyền dữ liệu qua mạng và tăng cường hiệu suất xử lý.

**Tính phân tán (Distributed Computing):** Spark được xây dựng trên kiến thức về tính phân tán. Nó cho phép chia công việc thành các nhiệm vụ nhỏ gửi đến các máy tính khác nhau trong cụm, giúp tận dụng tối đa tài nguyên và tăng tốc độ xử lý dữ liệu lớn.

**Các thư viện và API đa dạng:** Spark cung cấp nhiều thư viện và API cho nhiều mục đích, từ xử lý dữ liệu batch đến xử lý dữ liệu straming, từ machine learning đến xử lý đồ thị. Điều này làm cho Spark phù hợp với một loạt các ứng dụng và tình huống sử dụng.

**Hỗ trợ cộng đồng lớn:** Spark có một cộng đồng phát triển rộng lớn, với nhiều tài liệu, khóa học và hỗ trợ từ cộng đồng. Điều này giúp đơn giản hóa việc học và triển khai Spark trong các dự án thực tế.

Trên tất cả, Spark là một nền tảng mạnh mẽ cho xử lý dữ liệu lớn và dữ liệu thời gian, mang lại tính phân tán, hiệu suất và khả năng xử lý đáng kể.

## **Spark Sliding Window Computations**

**Spark** là một nền tảng mạnh mẽ cho việc thực hiện tính toán trên cửa sổ trượt (**Sliding Window Computations**) trong xử lý dữ liệu thời gian. Dưới đây là cách Spark có thể được sử dụng để thực hiện điều này:

**Spark Streaming:** Spark Streaming là một thành phần quan trọng của Spark được sử dụng để xử lý dữ liệu dạng dòng thời gian với tính phân tán. Nó cho phép bạn chia dữ liệu thành các *micro-batch* nhỏ và thực hiện tính toán trên chúng. Điều này giúp bạn tạo ra cửa sổ trượt và xử lý dữ liệu trong từng cửa sổ này.

**Structured Streaming:** Structured Streaming là một phần mở rộng của Spark Streaming, cung cấp một giao diện đơn giản và tổ chức cho việc xây dựng các ứng dụng xử lý dữ liệu thời gian. Structured Streaming cho phép bạn định nghĩa các truy vấn dựa trên dữ liệu dạng dòng thời gian và tự động cập nhật kết quả khi dữ liệu mới đến. Điều này giúp bạn xây dựng các ứng dụng với cửa sổ trượt một cách dễ dàng hơn.

**Spark SQL:** Spark SQL cung cấp một cách tiếp cận SQL cho việc xử lý dữ liệu dạng dòng thời gian. Bạn có thể sử dụng Spark SQL để thực hiện các truy vấn SQL trên dữ liệu thời gian và tạo cửa sổ trượt để áp dụng các hàm tính toán.

**Thư viện của ngôn ngữ lập trình:** Spark hỗ trợ nhiều ngôn ngữ lập trình như Scala, Java, Python, và R. Bạn có thể sử dụng các thư viện và API của Spark trong ngôn ngữ lập trình bạn đã chọn để xây dựng ứng dụng xử lý dữ liệu thời gian với cửa sổ trượt.

**Spark Streaming API:** API này cho phép bạn xác định và xử lý dữ liệu dạng dòng thời gian với tính phân tán. Bạn có thể sử dụng nó để tạo ra DStream (Discretized Stream) và thực hiện tính toán trên từng batch dữ liệu.

**Structured Streaming API:** Được xây dựng trên cơ sở của Spark SQL, API này cho phép bạn xác định truy vấn trực tiếp trên dữ liệu dạng dòng thời gian và tự động cập nhật kết quả khi dữ liệu mới đến. Điều này giúp bạn dễ dàng xây dựng ứng dụng với cửa sổ trượt.

Nhờ vào các API và thư viện này, Spark cung cấp một phạm vi rộng cho việc xử lý dữ liệu thời gian và tính toán trên cửa sổ trượt, giúp bạn xây dựng các ứng dụng mạnh mẽ trong việc theo dõi, phân tích và đưa ra quyết định dựa trên dữ liệu thời gian.

1. **Ưu và nhược điểm khi sử dụng Spark slinding window computations**

Ưu điểm:

* **Phân tích thời gian thực:** Spark sliding window 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:** Sliding window có thể đòi hỏi nhiều tài nguyên, đặc biệt khi xử lý các 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 só 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 đếm số từ xuất hiện trong mỗi cửa sổ thời gian được truyền**

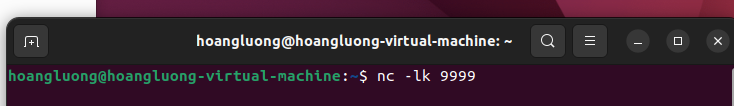
## **1. Mô tả sơ lược**

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.

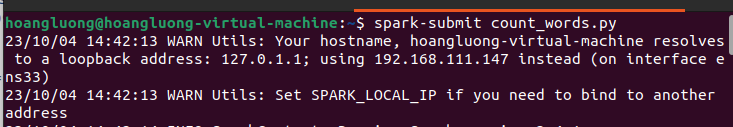
Cửa sổ trượt này cung cấp cái nhìn động về tần suất xuất hiện của từng từ trong khoảng thời gian nhất định. Cửa sổ di chuyển qua và cập nhật kết quả đếm từ. Kết quả cuối cùng được in ra màn hình.

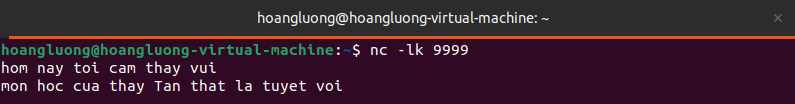
**2. Cách thực hiện**

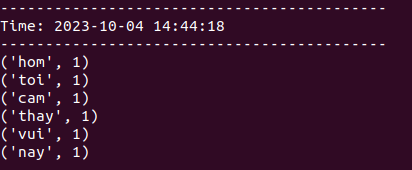
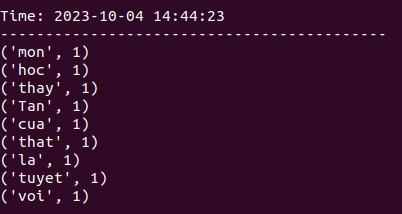
**2.1. Chuẩn bị dữ liệu đầu vào**

Sử dụng **Netcat** để tạo một socket server và gửi dữ liệu đến ứng dụng **Spark** (*localhost:9999*)

**2.2. Khởi động ứng dụng**

Mở 1 *terminal* khác và thực hiện chạy ứng dụng

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

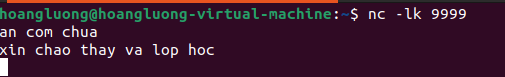
**2.4. Quan sát kết quả**

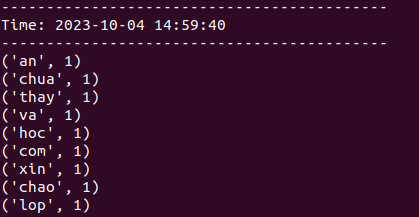
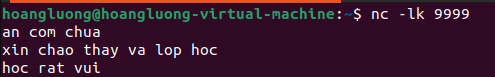
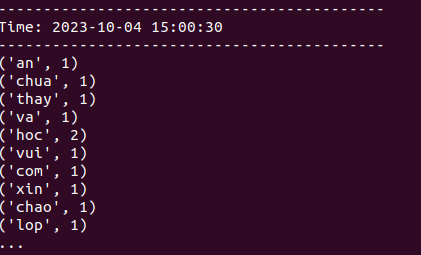
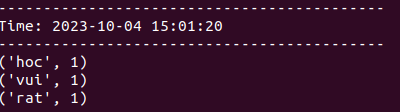
**3. Giải thích**

*Spark sliding windows* 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

Ta sửa *windowDuraion* và *slideDuration*



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

1. **DEMO: Áp dụng spark sliding window 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 tệp 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

### **2.2. Chuẩn bị dữ liệu test**

Tạo một kịch bản data là 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

### **2.3. Run**

Kết quả

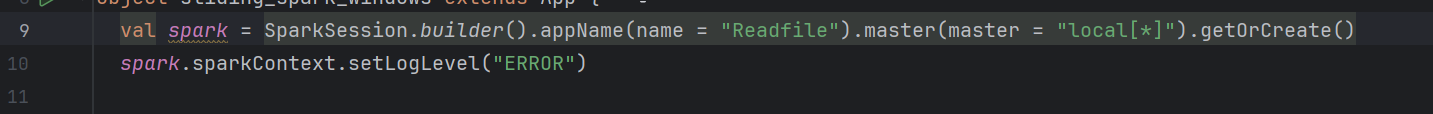
A screenshot of a computer

Description automatically generated

## **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.



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

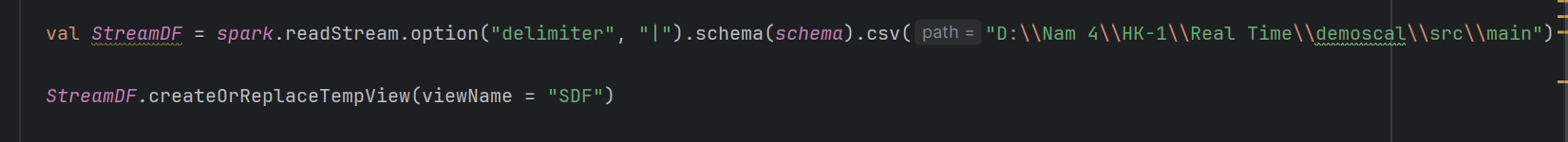
Xác định schema cho dữ liệu streaming, với hai cột là "Date" có kiểu Timestamp và "Message" có kiểu StringType.

A screen shot of a computer code

Description automatically generated

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

Đọc dữ liệu streaming từ tệp TXT trong Folder với schema đã được định nghĩa. Cấu hình option("delimiter", "|") cho biết rằng dữ liệu trong tệp 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 lí dữ liệu với cửa sổ thời gian trượt**

Xác định *Watermark* cho cửa sổ thời gian 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 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>