# Nội Dung tìm hiểu:

## Big Data

### Khái niệm

Big Data là các tập dữ liệu có khối lượng lớn và phức tạp. Độ lớn đến mức các phần mềm xử lý dữ liệu truyền thống không có khả năng thu thập, quản lý và xử lý dữ liệu trong một khoảng thời gian hợp lý.

Big Data không chỉ đề cập đến kích thước của dữ liệu mà còn bao gồm tốc độ, sự đa dạng và tính xác thực của dữ liệu.

Những tập dữ liệu lớn này có thể bao gồm các dữ liệu có cấu trúc, không có cấu trúc và bán cấu trúc, mỗi tập có thể được khai thác để tìm hiểu insights

### Đặc điểm

Big Data không chỉ đơn thuần là dữ liệu lớn mà còn bao gồm nhiều đặc điểm và tính chất quan trọng giúp hiểu rõ hơn về cách quản lý, phân tích và khai thác dữ liệu. Đặc điểm nổi bật như:

* Dữ liệu được tạo ra và truyền tải với tốc độ rất nhanh từ các nguồn trực tuyến, mạng xã hội, cảm biến và hệ thống giao dịch
* Khối lượng dữ liệu thường là hàng terabyte (TB), petabyte (PB) hoặc thậm chí exabyte (EB)
* Dữ liệu trong Big Data đến từ nhiều nguồn khác nhau và ở nhiều định dạng khác nhau, bao gồm dữ liệu có cấu trúc (structured), dữ liệu bán cấu trúc (semi-structured) và dữ liệu phi cấu trúc (unstructured).
* Giá trị là mục tiêu cuối cùng của việc khai thác Big Data (Dữ liệu lớn có thể mang lại những hiểu biết và thông tin giá trị giúp các tổ chức và doanh nghiệp đưa ra các quyết định chiến lược, tối ưu hóa hoạt động và tạo ra lợi thế cạnh tranh )
* Lưu lượng dữ liệu có thể thay đổi đáng kể theo thời gian, làm cho việc quản lý và phân tích dữ liệu trở nên phức tạp hơn

### Ứng dụng thực tế

Big data và phân tích có thể được áp dụng trong nhiều vấn đề kinh doanh và nhiều trường hợp sử dụng khác nhau. Dữ liệu lớn (Big Data) trên thực tế đang được ứng dụng vào rất nhiều lĩnh vực của nền kinh tế, tạo những chuyển biến ấn tượng, giúp tăng hiệu quả và năng suất của doanh nghiệp.

1. Ngân hàng

A close-up of a blue and green background

Description automatically generated

Trong hệ thống ngân hàng, Big Data đã và đang được ứng dụng hiệu quả thể hiện vai trò quan trọng của mình trong mọi hoạt động của ngân hàng: từ thu tiền mặt đến quản lý tài chính. Ngân hàng ứng dụng Big Data như thế nào:

* Sử dụng các kỹ thuật phân cụm giúp đưa ra quyết định quan trọng. Hệ thống phân tích có thể xác định các địa điểm chi nhánh nơi tập trung nhiều nhu cầu của khách hàng tiềm năng, để đề xuất lập chi nhánh mới.
* Kết hợp nhiều quy tắc được áp dụng trong các lĩnh vực ngân hàng để dự đoán lượng tiền mặt cần thiết sẵn sàng cung ứng ở một chi nhánh tại thời điểm cụ thể hàng năm.
* Khoa học dữ liệu hiện đang là nền tảng của hệ thống ngân hàng kĩ thuật số.
* Machine learning và AI đang được nhiều ngân hàng sử dụng để phát hiện các hoạt động gian lận và báo cáo cho các chuyên viên liên quan.
* Khoa học dữ liệu hỗ trợ xử lý, lưu trữ và phân tích lượng dữ liệu khổng lồ từ các hoạt động hàng ngày và giúp đảm bảo an ninh cho ngân hàng.

1. Thương mại điện tử

A computer screen with icons and symbols

Description automatically generated

Thương mại điện tử không chỉ tận hưởng những lợi ích của việc điều hành trực tuyến mà còn phải đối mặt với nhiều thách thức để đạt được các mục tiêu kinh doanh. Lý do là bởi các doanh nghiệp dù là nhỏ hay lớn, khi đã tham gia vào thị trường này đều cần đầu tư mạnh để cải tiến công nghệ. Big Data có thể tạo lợi thế cạnh tranh cho doanh nghiệp bằng cách cung cấp thông tin chuyên sâu và các bản báo cáo phân tích xu hướng tiêu dùng.

Thương mại điện tử ứng dụng Big Data:

* Có thể thu thập dữ liệu và yêu cầu của khách hàng ngay cả trước khi khách thực sự bắt đầu giao dịch.
* Tạo ra một mô hình tiếp thị hiệu suất cao.
* Nhà quản lý trang thương mại điện tử có thể xác định các sản phẩm được xem nhiều nhất và tối ưu thời gian hiển thị của các trang sản phẩm này.
* Đánh giá hành vi của khách hàng và đề xuất các sản phẩm tương tự. Điều này làm tăng khả năng bán hàng, từ đó tạo ra doanh thu cao hơn.
* Nếu bất kỳ sản phẩm nào được thêm vào giỏ hàng nhưng cuối cùng không được khách hàng mua, Big Data có thể tự động gửi code khuyến mại cho khách hàng cụ thể đó.
* Các ứng dụng Big Data còn có thể tạo một báo cáo tùy chỉnh theo các tiêu chí: độ tuổi, giới tính, địa điểm của khách truy cập, v.v.
* Xác định các yêu cầu của khách hàng, những gì họ muốn và tập trung vào việc cung cấp dịch vụ tốt nhất để thực hiện nhu cầu của họ.
* Phân tích hành vi, sự quan tâm của khách hàng và theo xu hướng của họ để tạo ra các sản phẩm hướng đến khách hàng.
* Cung cấp các sản phẩm tốt hơn với chi phí thấp hơn.
* Có thể thu thập nhiều dữ liệu về hành vi khách hàng để thiết kế mô hình tiếp thị tối ưu dành được tùy biến theo đối tượng hoặc nhóm đối tượng, tăng khả năng bán hàng.
* Tìm ra sự tương đồng giữa khách hàng và nhu cầu của họ. Từ đó, việc nhắm mục tiêu các chiến dịch quảng cáo có thể được tiến hành dễ dàng hơn dựa trên những phân tích đã có trước đó.

1. Y tế



Khoa học dữ liệu đang dần khẳng định vai trò khá quan trọng trong việc cải thiện sức khỏe con người ngày nay. Big Data không chỉ được ứng dụng để xác định phương hướng điều trị mà giúp cải thiện quá trình chăm sóc sức khỏe.

Ngành y tế ứng dụng Big Data:

* Cho phép người quản lý ca dự đoán các bác sĩ cần thiết vào những thời điểm cụ thể
* Theo dõi tình trạng bệnh nhân bằng để theo dõi hồ sơ sức khỏe điện tử.
* Sử dụng các thiết bị kỹ thuật số có thể đeo, hệ thống Big Data có thể theo dõi bệnh nhân và gửi báo cáo cho các bác sĩ liên quan.
* Big Data có thể đánh giá các triệu chứng và xác định nhiều bệnh ở giai đoạn đầu.
* Có thể lưu giữ các hồ sơ nhạy cảm được bảo mật và lưu trữ lượng dữ liệu khổng lồ một cách hiệu quả.
* Các ứng dụng Big Data cũng có thể báo trước khu vực có nguy cơ bùng phát dịch như: sốt xuất huyết hoặc sốt rét.

1. Digital Marketing

A diagram of a big data

Description automatically generated

Digital Marketing là chìa khóa để cánh cửa thành công cho bất kỳ doanh nghiệp nào. Giờ đây, không chỉ các công ty lớn có thể điều hành các hoạt động quảng cáo tiếp thị mà cả các doanh nhân nhỏ cũng có thể chạy các chiến dịch quảng cáo thành công trên các nền tảng truyền thông xã hội và quảng bá sản phẩm của họ. Big Data đã tiếp sức cho Digital Marketing phát triển thực sự mạnh mẽ, và nó đã trở thành một phần không thể thiếu của bất kỳ doanh nghiệp nào.

Digital Marketing ứng dụng Big Data:

* Phân tích thị trường, đối thủ cạnh tranh và đánh giá mục tiêu kinh doanh. Điều này giúp cho doanh nghiệp xác định rõ hơn, đâu là cơ hội tốt để tiếp tục tiến hành các kế hoạch kinh doanh tiếp theo.
* Có thể xác định người dùng trên các phương tiện truyền thông xã hội và nhắm mục tiêu cho họ dựa trên nhân khẩu học, giới tính, thu nhập, tuổi tác và sở thích.
* Tạo báo cáo sau mỗi chiến dịch quảng cáo bao gồm hiệu suất, sự tham gia của khán giả và những gì có thể được thực hiện để tạo kết quả tốt hơn.
* Khoa học dữ liệu được sử dụng cho các khách hàng nhắm mục tiêu và nuôi dưỡng chu trình khách hàng.
* Tập trung vào các chủ đề được tìm kiếm cao và tư vấn cho các chủ doanh nghiệp thực hiện chúng trên chiến lược nội dung để xếp hạng trang web doanh nghiệp trên cao hơn trên google (SEO).
* Có thể tạo đối tượng tương tự bằng cách sử dụng cơ sở dữ liệu đối tượng hiện có để nhắm mục tiêu các khách hàng tương tự và kiếm được lợi nhuận.

## Phương pháp, thuật toán sử lý BigData

Trong xử lý Big Data, có nhiều phương pháp và thuật toán khác nhau tùy thuộc vào mục tiêu và dữ liệu cần xử lý. Ở đây chúng tôi xin trình bày những thuật toán phổ biến:

### Thuật toán phân lớp Naïve Bayes

#### Mô tả

Bộ phân lớp Naive bayes hay bộ phân lớp Bayes (simple byes classifier) hoạt động như sau:

1. Gọi D là tập dữ liệu huấn luyện, trong đó mỗi phần tử dữ liệu X được biểu diễn bằng một vector chứa n giá trị thuộc tính A1, A2,...,An = {x1,x2,...,xn}
2. Giả sử có m lớp C1, C2,..,Cm. Cho một phần tử dữ liệu X, bộ phân lớp sẽ gán nhãn cho X là lớp có xác suất hậu nghiệm lớn nhất. Cụ thể, bộ phân lớp Bayes sẽ dự đoán X thuộc vào lớp Ci nếu và chỉ nếu:  
   P(Ci|X) > P(Cj|X) (1<= i, j <=m, i != j)  
   Giá trị này sẽ tính dựa trên định lý Bayes.
3. Để tìm xác suất lớn nhất, ta nhận thấy các giá trị P(X) là giống nhau với mọi lớp nên không cần tính. Do đó ta chỉ cần tìm giá trị lớn nhất của P(X|Ci) \* P(Ci). Chú ý rằng P(Ci) được ước lượng bằng |Di|/|D|, trong đó Di là tập các phần tử dữ liệu thuộc lớp Ci. Nếu xác suất tiền nghiệm P(Ci) cũng không xác định được thì ta coi chúng bằng nhau P(C1) = P(C2) = ... = P(Cm), khi đó ta chỉ cần tìm giá trị P(X|Ci) lớn nhất.
4. Khi số lượng các thuộc tính mô tả dữ liệu là lớn thì chi phí tính toàn P(X|Ci) là rất lớn, dó đó có thể giảm độ phức tạp của thuật toán Naive Bayes giả thiết các thuộc tính độc lập nhau. Khi đó ta có thể tính:  
   P(X|Ci) = P(x1|Ci)...P(xn|Ci)

#### Ưu điểm

Giả định độc lập: hoạt động tốt cho nhiều bài toán/miền sữ liệu và ứng dụng.Đơn giản nhưng đủ tốt để giải quyết nhiều bài toán như phân lớp văn bản, lọc spam,..ho phép kết hợp tri thức tiền nghiệm (prior knowledge) và dữ liệu quan sát được (obserwed data).Tốt khi có sự chệnh lệch số lượng giữa các lớp phân loại.Huấn luyện mô hình (ước lượng tham số) dễ và nhanh.

Ví dụ:

Phân các bệnh nhân thành 2 lớp ung thư và không ung thư. Giả sử xác suất để một người bị ung thư là 0.008 tức là P(cancer) = 0.008; và P(nocancer) = 0.992. Xác suất để bệnh nhân ung thư có kết quả xét nghiệm dương tính là 0.98 và xác suất để bệnh nhân không ung thư có kết quả dương tính là 0.03 tức là P(+/cancer) = 0.98, P(+/nocancer) = 0.03. Bây giờ giả sử một bệnh nhân có kết quả xét nghiệm dương tính.

Ta có:  
P(+/canncer)P(cancer) = 0.98 \* 0.008 = 0.0078  
P(+/nocancer)P(nocancer) = 0.03 \* 0.992 = 0.0298  
Như vậy, P(+/nocancer)P(nocancer) >> P(+/cancer)P(cancer).  
Do đó ta xét đoán rằng, bệnh nhân là không ung thư.

#### Công cụ phổ biến

Các công cụ phổ biến của thuật toán phân lớp Naïve Bayes:

* Scikit-Learn (Python)
* Weka (Java)
* Apache Mahout,R, …..

### Thuật toán học máy (Machine learning)

#### Mô tả

Quy trình triển khai thuật toán học máy thường bao gồm 6 bước như sau:

1. Thu thập dữ liệu (Gathering data/Data collection)
2. Tiền xử lý dữ liệu (Data preprocessing)

* Trích xuất dữ liệu – Data extraction
* Làm sạch dữ liệu – Data cleaning
* Chuyển đổi dữ liệu – Data transformation
* Chuẩn hóa dữ liệu – Data normalization
* Trích xuất đặc trưng – Feature extraction

1. Phân tích dữ liệu (Data analysis)
2. Xây dựng mô hình máy học (Model building)
3. Huấn luyện mô hình (Model training)
4. Đánh giá mô hình (Model evaluation)

#### Ưu điểm

* Xác định xu hướng dữ liệu dễ dàng
* Khả năng tự động hóa cao xử lý đa dạng dữ liệu

#### Các công cụ phổ biến

Các công cụ phổ biến như:

* Apache Spark
* Amazon SageMaker
* RapidMiner

### Phương pháp sử lý luồng dữ liệu

#### Mô tả

[Stream Processing](https://viblo.asia/p/khai-niem-ve-stream-processing-GrLZDexelk0) là quá trình xử lý dữ liệu gần như ngay lập tức sau khi nó được tạo ra. Điều này đặc biệt hữu ích khi cần xử lý và phân tích dữ liệu theo thời gian thực.

#### Ưu điểm

Stream Processing cung cấp sự linh hoạt trong việc phân tích và xử lý dữ liệu theo thời gian thực

#### Các công cụ phổ biến

Apache Kafka, Apache Flink, Storm và Google Cloud Dataflow

Ví dụ:

* Chăm sóc sức khỏe: giám sát liên tục dữ liệu thiết bị
* An ninh mạng, phát hiện bất thường: xử lý nhật ký web hoặc khủng bố mạng
* Giao thông vận tải: tối ưu hóa lộ trình và mức tiêu thụ nhiên liệu

### Phương pháp sử lý theo lô

#### Mô tả

Phương pháp sử lí luồng theo lô là việc xử lý một tập dữ liệu lớn mà không cần tương tác người dùng trong quá trình xử lý. Điều này thường được thực hiện với những tác vụ đòi hỏi thời gian xử lý lớn, chẳng hạn như phân tích dữ liệu lớn, xử lý hóa đơn hàng loạt, hoặc thậm chí là xử lý giao dịch ngân hàng hàng loạt.

#### Ưu điểm

Batch Processing phù hợp với các tác vụ đòi hỏi xử lý lượng dữ liệu lớn và không cần kết quả ngay lập tức

#### Các công cụ phổ biến

Hadoop, Spark, Hive và MapReduce

## Công cụ sử lý BigData

Trong phần trước, chúng tôi đã trình bày về BigData và tầm quan trọng của nó trong đời sống thực tiễn, sau đây chúng tôi xin trình bày về các công cụ hàng đầu về việc sử lí BigData.

### Apache Hadoop

Apache Hadoop là một trong những công cụ được sử dụng phổ biến nhất. Hadoop là một bộ khung mã nguồn mở từ Apache và chạy trên phần cứng. Nó được sử dụng để lưu trữ quá trình và phân tích dữ liệu. Hadoop được viết bằng Java.

Apache Hadoop cho phép xử lý dữ liệu song song khi nó hoạt động trên nhiều máy cùng một lúc. Nó sử dụng cấu trúc cụm. Cụm là một nhóm các hệ thống được kết nối qua mạng LAN.

Nó bao gồm 3 phần:

* Hệ thống tệp phân tán Hadoop (HDFS) – Đây là lớp lưu trữ của Hadoop.
* Map-Reduce – Đây là lớp xử lý dữ liệu của Hadoop.
* YARN – Đây là lớp quản lý tài nguyên của Hadoop

Mọi công cụ được phát triển đi kèm với một số nhược điểm. Và Hadoop có một số nhược điểm sau đây:

* Hadoop không hỗ trợ xử lý thời gian thực. Nó chỉ hỗ trợ xử lý hàng loạt.
* Hadoop không thể thực hiện các phép tính trong bộ nhớ.

### Apache Spark

Apache Spark có thể được coi là sự kế thừa của Hadoop khi nó khắc phục được những nhược điểm của Hadoop. Spark, không giống như Hadoop, hỗ trợ cả thời gian thực cũng như xử lý hàng loạt. Nó là một hệ thống phân cụm mục đích chung.

Nó cũng hỗ trợ tính toán trong bộ nhớ, khiến Apache Spark nhanh hơn 100 lần so với Hadoop. Điều này được thực hiện bằng cách giảm số lượng thao tác đọc/ ghi vào đĩa. Nó cung cấp sự linh hoạt hơn so với Hadoop vì nó hoạt động với các kho dữ liệu khác nhau như HDFS, OpenStack và Apache Cassandra.

Nó cung cấp các API cấp cao trong Java, Python, Scala và R. Spark cũng cung cấp một bộ công cụ cấp cao đáng kể bao gồm Spark SQL để xử lý dữ liệu có cấu trúc, MLlib cho Machine Learning, GraphX để xử lý tập dữ liệu đồ thị và Spark Streaming. Nó cũng bao gồm 80 toán tử cấp cao để thực hiện truy vấn hiệu quả.

### Apache Cassandra

Apache Cassandra là hệ cơ sở dữ liệu phân tán, kết hợp những gì tinh tuý nhất của Google Bigtable và Amazon DynamoDB. Ngôn ngữ phát triển Cassandra là Java. Đây là một trong những công cụ dữ liệu lớn tốt nhất có thể chứa tất cả các loại tập dữ liệu cụ thể có cấu trúc, bán cấu trúc và không cấu trúc.

Cassandra được thiết kế có thể chạy trong phần cứng giá rẻ, và cung cấp write throughput khá là cao (latency tầm 0.5ms), trong khi read throughput thì thấp hơn (latency tầm 2.5ms).

### MongoDB

MongoDB là một công cụ phân tích dữ liệu nguồn mở, cơ sở dữ liệu NoQuery cung cấp các khả năng đa nền tảng. Đây là công cụ dành cho doanh nghiệp cần dữ liệu nhanh chóng và thời gian thực để đưa ra quyết định.

MongoDB là công cụ hoàn hảo cho những người muốn các giải pháp dựa trên dữ liệu. Nó thân thiện với người dùng vì nó cung cấp cài đặt và bảo trì dễ dàng hơn. MongoDB là công cụ đáng tin cậy và tiết kiệm chi phí.

Nó được viết bằng C, C ++ và JavaScript. Đây là một trong những cơ sở dữ liệu phổ biến nhất cho Big Data vì nó tạo điều kiện thuận lợi cho việc quản lý dữ liệu phi cấu trúc hoặc dữ liệu thay đổi thường xuyên.

MongoDB sử dụng các lược đồ động. Do đó, bạn có thể chuẩn bị dữ liệu nhanh chóng. Điều này cho phép giảm chi phí tổng thể. Nó thực thi trên ngăn xếp phần mềm MEAN, các ứng dụng NET và, nền tảng Java. Nó cũng linh hoạt trong cơ sở hạ tầng đám mây.

### RapidMiner

Rapid Miner là một nền tảng phần mềm khoa học dữ liệu cung cấp một môi trường tích hợp để chuẩn bị dữ liệu, học máy, học sâu, khai thác văn bản và phân tích dự đoán. Đây là một trong những hệ thống mã nguồn mở hàng đầu cho khai thác dữ liệu.

Chương trình được viết hoàn toàn bằng ngôn ngữ lập trình Java. Chương trình cung cấp một tùy chọn để thử xung quanh với một số lượng lớn các toán tử tùy ý có thể lồng được chi tiết trong các tệp XML và được thực hiện với sự can thiệp của người dùng đồ họa của người khai thác nhanh.

Những công cụ Big Data kể trên không chỉ giúp bạn lưu trữ số lượng lớn dữ liệu mà còn giúp xử lý dữ liệu được lưu trữ một cách nhanh hơn và cung cấp cho bạn kết quả tốt hơn. Đa số các công cụ Big Data đã có sẵn trên thị trường. Bạn chỉ cần chọn công cụ phù hợp với dự án của bạn.

### Apache Storm

Apache Storm là hệ thống tính toán phân tán mã nguồn mở thời gian thực miễn phí. Nếu như Hadoop xử lý dữ liệu hàng loạt (Batch Processing) thì Apache Storm thực hiện xử lý dữ liệu luồng (Unbounded streams of data) một cách đáng tin cậy.

Ưu điểm lớn nhất của Apache Storm là dễ triển khai và có thể tương tác với bất kỳ ngôn ngữ lập trình nào.

Mặt khác, nó đảm bảo việc xử lý từng bộ dữ liệu. Tốc độ xử lý của nó rất nhanh và một tiêu chuẩn có thể quan sát được là tới một triệu tuple được xử lý mỗi giây trên mỗi nút.

## Apache Spark

### Định Nghĩa

Apache Spark là một framework xử lý dữ liệu phân tán mạnh mẽ và linh hoạt. Với kiến trúc phân tán, nó cho phép xử lý các tác vụ dữ liệu lớn và tính toán phân tán trên nhiều node.

Spark được phát triển bắt đầu từ năm 2009 tại AMPLab của Đại học California, Berkeley và sau đó được chuyển giao cho Quỹ phần mềm Apache để quản lý và phát triển tiếp.

Spark được coi là một công cụ xử lý dữ liệu nhanh hơn gấp 10 lần so với các công cụ khác. Nó cung cấp khả năng tính toán phân tán trên các cụm máy tính với hiệu suất cao hơn so với Apache Hadoop và đặc biệt là MapReduce. Spark cũng hỗ trợ nhiều ngôn ngữ lập trình như Scala, Java, Python, giúp cho người dùng dễ dàng phát triển ứng dụng trên nền tảng này.

Triết lý thiết kế của Spark xoay quanh bốn đặc điểm chính:

* Tốc độ
* Dễ sử dụng
* Tính mô-đun
* Khả năng mở rộng Chúng ta hãy xem điều này có ý nghĩa gì đối với framework.

Về tốc độ:

Spark đã theo đuổi mục tiêu tốc độ theo nhiều cách.

Đầu tiên, việc triển khai nội bộ của nó được hưởng lợi rất nhiều từ những bước tiến lớn gần đây của ngành công nghiệp phần cứng trong việc cải thiện giá cả và hiệu suất của CPU và bộ nhớ. Các máy chủ hàng hóa ngày nay có giá rẻ, với hàng trăm gigabyte bộ nhớ, nhiều lõi và hệ điều hành dựa trên Unix cơ bản tận dụng lợi thế của đa luồng hiệu quả và xử lý paral lel. Framework được tối ưu hóa để tận dụng tất cả các yếu tố này.

Thứ hai, Spark xây dựng các tính toán truy vấn của nó dưới dạng đồ thị tuần hoàn có hướng (DAG); Trình lập lịch DAG và trình tối ưu hóa truy vấn của nó xây dựng một biểu đồ tính toán hiệu quả thường có thể được phân tách thành các tác vụ được thực thi song song giữa các công nhân trên cụm. Và thứ ba, công cụ thực thi vật lý của nó, Tungsten, sử dụng tạo mã toàn giai đoạn để tạo mã nhỏ gọn để thực thi.

Về tính dễ sử dụng:

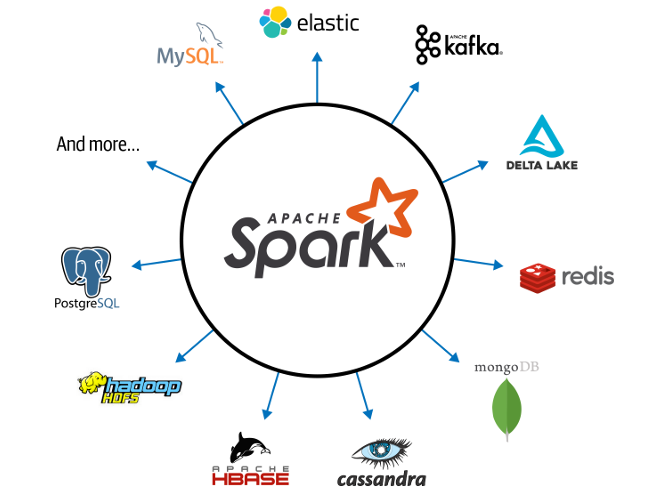
Spark đạt được sự đơn giản bằng cách cung cấp một sự trừu tượng cơ bản của một cấu trúc dữ liệu logic đơn giản được gọi là Tập dữ liệu phân tán có khả năng phục hồi (RDD) mà tất cả các trừu tượng dữ liệu có cấu trúc cấp cao khác, chẳng hạn như DataFrames và Datasets, được cấu trúc. Bằng cách cung cấp một tập hợp các biến đổi và hành động dưới dạng hoạt động, Spark cung cấp một mô hình lập trình đơn giản mà bạn có thể sử dụng để xây dựng các ứng dụng dữ liệu lớn bằng các ngôn ngữ iar nổi tiếng.

Về tính mô dun:

Các hoạt động của Spark có thể được áp dụng trên nhiều loại khối lượng công việc và được thể hiện bằng bất kỳ ngôn ngữ lập trình nào được hỗ trợ: Scala, Java, Python, SQL và R. Spark cung cấp các thư viện hợp nhất với các API được ghi chép đầy đủ bao gồm các mod ules sau làm thành phần cốt lõi: Spark SQL, Spark Structured Streaming, Spark MLlib và GraphX, kết hợp tất cả khối lượng công việc chạy trong một công cụ. Chúng ta sẽ xem xét kỹ hơn tất cả những điều này trong phần tiếp theo. Bạn có thể viết một ứng dụng Spark duy nhất có thể làm tất cả — không cần các công cụ riêng biệt cho các khối lượng công việc khác nhau, không cần phải học các API riêng biệt. Với Spark, bạn có được một công cụ xử lý thống nhất cho khối lượng công việc của mình.

Về tính mở rộng:

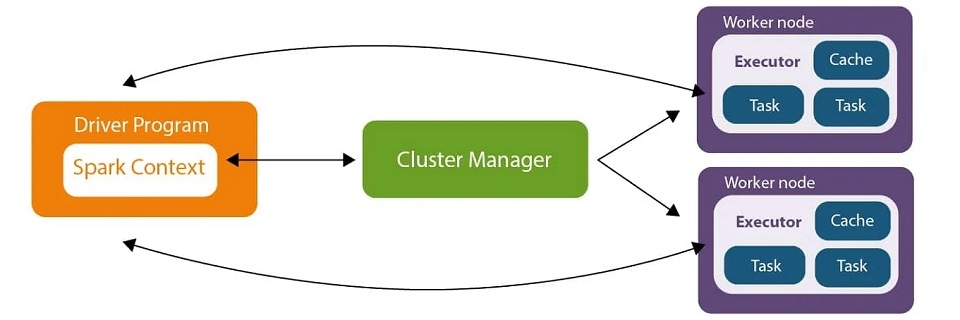
Spark tập trung vào công cụ tính toán song song, nhanh hơn là lưu trữ. Không giống như Apache Hadoop, bao gồm cả lưu trữ và tính toán, Spark tách rời cả hai. Điều đó có nghĩa là bạn có thể sử dụng Spark để đọc dữ liệu được lưu trữ trong vô số nguồn — Apache Hadoop, Apache Cassandra, Apache HBase, MongoDB, Apache Hive, RDBMS, v.v. — và xử lý tất cả trong bộ nhớ. DataFrameReader và DataFrame Writers của Spark cũng có thể được mở rộng để đọc dữ liệu từ các nguồn khác, chẳng hạn như Apache Kafka, Kinesis, Azure Storage và Amazon S3, vào sự trừu tượng hóa dữ liệu logic mà nó có thể hoạt động.



### Kiến trúc của Apache Spark

Kiến trúc của Apache Spark bao gồm các thành phần chính như:

* RDDs
* Driver program
* Spark Context
* Cluster Manager
* Executors



Các thành phần này hoạt động cùng nhau để quản lý và thực thi quy trình xử lý dữ liệu trên một môi trường phân tán. Điều này giúp Apache Spark đạt được hiệu suất cao, khả năng mở rộng và độ tin cậy trong việc xử lý dữ liệu lớn và tính toán phân tán

#### Resilient Distributed Datasets (RDD)

##### Định nghĩa

Resilient Distributed Datasets (RDDs) là một cấu trúc dữ liệu cơ bản và quan trọng trong Apache Spark, cho phép xử lý dữ liệu phân tán trên các cụm máy tính một cách linh hoạt và hiệu quả. Đây là một phần chính của nền tảng Apache Spark, giúp đơn giản hóa việc xử lý dữ liệu lớn bằng cách tự động xử lý phân tán và chịu sự cố.

##### Vai trò

RDD có vai trò to lớn trong Spark:

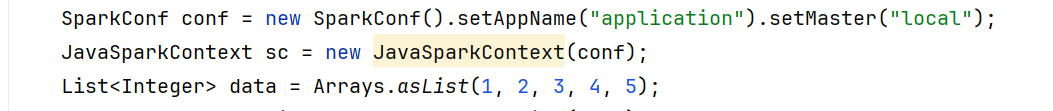
* Lưu trữ dữ liệu phân tán
* Hỗ trợ khả năng chịu lỗi
* Tính toán song song
* Tích hợp bộ nhớ (in memory) để tính toán nhanh hơn.

##### Cách khởi tạo

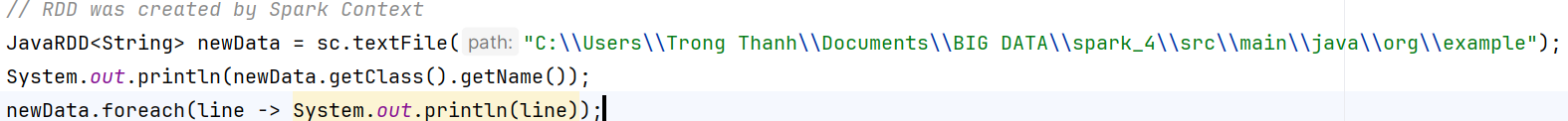
Có ba cách để tạo RDD:

* Song song hóa một tập hợp hiện có trong chương trình trình điều khiển của bạn.
* Tham chiếu một tập dữ liệu trong hệ thống lưu trữ bên ngoài, chẳng hạn như hệ thống tệp được chia sẻ, HDFS, Database, HBase hoặc bất kỳ nguồn dữ liệu nào cung cấp Hadoop InputFormat.
* RDD cũng có thể được tạo ra từ 1 RDD khác.

Các bộ sưu tập song song được tạo bằng cách gọi phương thức JavaSparkContext's parallelizetrên một Collectionchương trình trình điều khiển hiện có của bạn. Các phần tử của bộ sưu tập được sao chép để tạo thành một tập dữ liệu phân tán có thể được vận hành song song. Ví dụ, đây là cách tạo một bộ sưu tập song song chứa các số từ 1 đến 5:

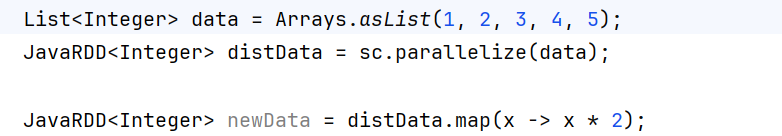


RDD tệp văn bản có thể được tạo bằng phương thức SparkContext's textFile. Phương thức này lấy URI cho tệp (hoặc đường dẫn cục bộ trên máy hoặc URI hdfs://, s3a://, v.v.) và đọc nó như một tập hợp các dòng. Sau đây là một ví dụ về lệnh gọi:



Spark có thể tạo các tập dữ liệu phân tán từ bất kỳ nguồn lưu trữ nào được Hadoop hỗ trợ, bao gồm hệ thống tệp cục bộ, HDFS, Cassandra, HBase, [Amazon S3](http://wiki.apache.org/hadoop/AmazonS3) , v.v. Spark hỗ trợ các tệp văn bản, [SequenceFiles](https://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapred/SequenceFileInputFormat.html) và bất kỳ Hadoop [InputFormat](http://hadoop.apache.org/docs/stable/api/org/apache/hadoop/mapred/InputFormat.html) nào khác .

RDD được tạo ra từ một RDD khác, ví dụ như:



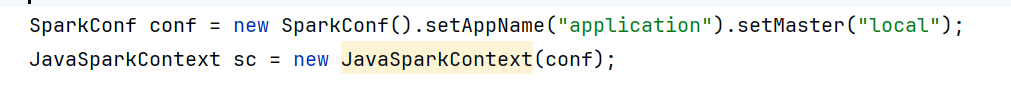
#### Driver program

##### Định nghĩa

Driver Program là thành phần chính trong ứng dụng Spark. Nó chịu trách nhiệm khởi tạo SparkContext, quản lý tiến trình thực thi của ứng dụng, phân phối công việc đến các Executor (thực thi) và theo dõi tiến độ của các tác vụ.

##### Vai trò

* Driver Program chịu trách nghiệm khởi tạo Spark Context:



* Phân chia các công việc thành các task
* Kiểm soát, theo dõi tiến trình
* Thu thập kết quả và trả về cho người dùng

#### Spark Context

##### Định nghĩa

Spark Context là điểm vào (entry point) chính để tương tác với các chức năng của Spark. Nó chịu trách nhiệm kết nối ứng dụng với cụm Spark, quản lý tài nguyên, và cung cấp các API để thực hiện các phép toán trên dữ liệu.

Thành phần chính:

* Configuration (cấu hình): Spark Context sử dụng Lớp SparkConf để thiết lập các cấu hình cho ứng dụng, ví dụ như setName(), bộ nhớ, số lõi,..
* Scheduler (bộ lập lịch): phân phối tài nguyên đến các executors

##### Vai trò

Vai trò của Spark Context:

* Tạo kết nối đến với Cluster để yêu cầu quản lý tài nguyên người dùng.
* Khởi tạo RDD
* Phân phối tác vụ đến các Executor
* …
* Environment (môi trường): gồm những thông tin về môi trường, thông tin về phiên bản của Spark, thư viện được sử dụng, …

#### Cluster Manager

##### Định nghĩa

Cluster manager (trình quản lý cụm) là một thành phần trong hệ thống cluster, chịu trách nhiệm quản lý và điều phối các tài nguyên trong cụm máy tính (cluster). Nhiệm vụ chính của cluster manager bao gồm phân bổ tài nguyên, quản lý các node, giám sát các tác vụ và điều phối việc thực hiện công việc giữa các node trong cụm.

##### Vai trò

Vai trò của Cluster manager

* Phân tán tải công việc: Cluster giúp chia nhỏ khối lượng công việc lớn và phân phối chúng cho nhiều máy tính trong cụm, từ đó tối ưu hóa tài nguyên và tăng hiệu suất.
* Tăng cường khả năng mở rộng: Khi nhu cầu xử lý tăng cao, ta có thể dễ dàng thêm nhiều máy tính (node) vào cluster mà không làm gián đoạn hệ thống.
* Tăng độ tin cậy và sẵn sàng cao: Nếu một node trong cluster gặp sự cố, các node khác có thể tiếp quản nhiệm vụ mà không ảnh hưởng đến hoạt động chung. Điều này giúp hệ thống luôn sẵn sàng và giảm thiểu thời gian chết (downtime).
* Tăng tốc độ xử lý: Cluster cho phép xử lý song song nhiều tác vụ cùng lúc, tăng tốc độ xử lý so với việc chạy trên một máy tính đơn lẻ.
* Quản lý dữ liệu phân tán: Cluster thường được sử dụng để quản lý các hệ thống cơ sở dữ liệu phân tán, giúp xử lý dữ liệu lớn (Big Data) hoặc các tác vụ yêu cầu xử lý lượng dữ liệu khổng lồ.

#### Executors

##### Định nghĩa

Executors là các tiến trình (processes) chạy trên các node của cluster

Thành phần của Executors bao gồm task Execution để sử lí các phép tính cụ thể, block manager để quản lý dữ liệu

##### Vai trò

Vai trò của Executors là:

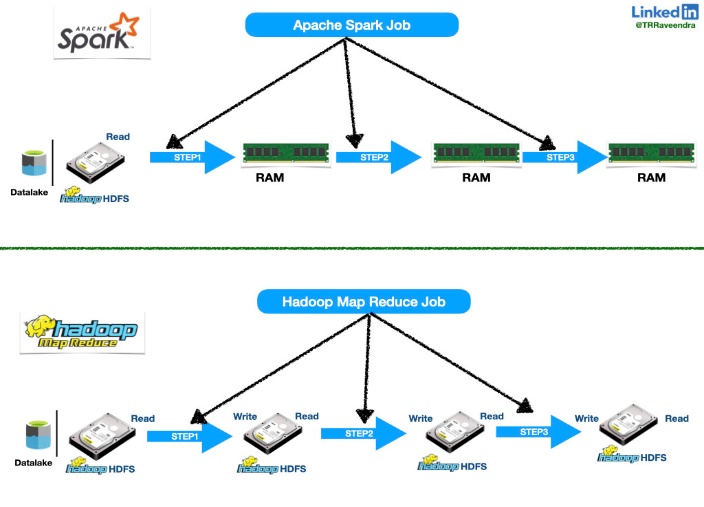
* Thực thi các task
* Quản lý dữ liệu bộ nhớ
* Tương tác với Driver

### Thành phần và thư viện của Spark

#### Spark Core

##### Định nghĩa

Apache Spark Core là thành phần trung tâm của Apache Spark, một nền tảng xử lý dữ liệu phân tán tốc độ cao. Spark Core cung cấp các chức năng cơ bản cho việc xử lý dữ liệu phân tán, đảm bảo khả năng chịu lỗi và làm nền tảng cho các thành phần khác của Spark như Spark SQL, Spark Streaming, MLlib và GraphX.



##### Tính năng

Một trong những tính năng chính của Spark Core là RDD (Resilient Distributed Dataset). RDD là cấu trúc dữ liệu cơ bản trong Spark, đại diện cho một tập dữ liệu phân tán trên nhiều node (máy chủ) trong cluster. RDD hỗ trợ hai loại thao tác chính: Transformation và Action. Các thao tác Transformation như map() và filter() cho phép tạo ra RDD mới từ một RDD hiện có, trong khi các thao tác Action như collect() và count() tính toán kết quả dựa trên RDD và trả về kết quả cho người dùng.

Một đặc điểm quan trọng khác của Spark Core là tính toán trễ (Lazily Evaluated). Các phép biến đổi trên RDD không được thực hiện ngay lập tức mà được lưu lại dưới dạng DAG (Directed Acyclic Graph). Chỉ khi người dùng thực hiện các thao tác Action, Spark mới bắt đầu tiến hành tính toán dựa trên DAG này, giúp tối ưu hóa quá trình xử lý dữ liệu.

Về quản lý cluster, Spark Core có khả năng tích hợp với các hệ thống quản lý như Hadoop YARN, Mesos, hoặc có thể sử dụng cluster riêng của Spark. Điều này cho phép Spark dễ dàng mở rộng quy mô để xử lý lượng dữ liệu lớn một cách hiệu quả. Bên cạnh đó, Spark Core cũng có bộ lập lịch công việc tích hợp, giúp chia nhỏ công việc thành các tác vụ (task) và phân phối chúng đến các node trong cluster một cách tối ưu.

Spark Core hỗ trợ nhiều ngôn ngữ lập trình khác nhau như Scala, Java, Python và R mang lại sự linh hoạt cho các lập trình viên.

#### Spark SQL

##### Định nghĩa

Spark SQL là một mô-đun Spark để xử lý dữ liệu có cấu trúc. Không giống như Spark RDD API cơ bản, các giao diện do Spark SQL cung cấp cung cấp cho Spark nhiều thông tin hơn về cấu trúc của cả dữ liệu và phép tính đang được thực hiện. Về mặt nội bộ, Spark SQL sử dụng thông tin bổ sung này để thực hiện các tối ưu hóa bổ sung. Có một số cách để tương tác với Spark SQL bao gồm SQL và Dataset API. Khi tính toán kết quả, cùng một công cụ thực thi được sử dụng, bất kể bạn đang sử dụng API/ngôn ngữ nào để thể hiện phép tính. Sự hợp nhất này có nghĩa là các nhà phát triển có thể dễ dàng chuyển đổi qua lại giữa các API khác nhau dựa trên cách cung cấp cách tự nhiên nhất để thể hiện một phép biến đổi nhất định.

Một công dụng của Spark SQL là thực thi các truy vấn SQL. Spark SQL cũng có thể được sử dụng để đọc dữ liệu từ một cài đặt Hive hiện có. Khi chạy SQL từ bên trong một ngôn ngữ lập trình khác, kết quả sẽ được trả về dưới dạng Dataset/DataFrame . Bạn cũng có thể tương tác với giao diện SQL bằng dòng lệnh hoặc qua JDBC/ODBC .

Note:

* Hive: là một công cụ cơ sở hạ tầng kho dữ liệu để xử lý dữ liệu có cấu trúc trong Hadoop. Nó nằm trên đỉnh Hadoop để tóm tắt Dữ liệu lớn và giúp truy vấn và phân tích dễ dàng.
* JDBC/ODBC: là các API để truy cập các hệ quản trị cơ sở dữ liệu (DBMS)

##### Datasets và DataFrame

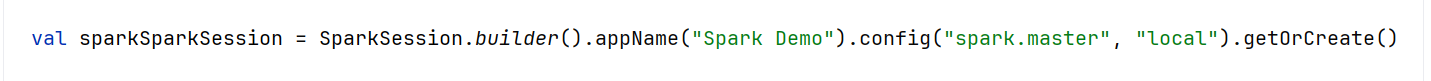
Bộ dữ liệu và khung dữ liệu (Datasets và DataFrame)

Dataset là một tập hợp dữ liệu phân tán. Dataset là một giao diện mới được thêm vào Spark 1.6 cung cấp các lợi ích của RDD (kiểu dữ liệu mạnh, khả năng sử dụng các hàm lambda mạnh mẽ) với các lợi ích của công cụ thực thi được tối ưu hóa của Spark SQL. Dataset có thể được xây dựng từ các đối tượng JVM và sau đó được thao tác bằng các phép biến đổi hàm ( map, flatMap, filter, v.v.). API Dataset có sẵn trong Scala và Java . Python không hỗ trợ API Dataset. Nhưng do bản chất động của Python, nhiều lợi ích của API Dataset đã có sẵn (tức là bạn có thể truy cập trường của một hàng theo tên một cách tự nhiên row.columnName). Trường hợp của R cũng tương tự.

DataFrame là một Dataset được tổ chức thành các cột được đặt tên. Về mặt khái niệm, nó tương đương với một bảng trong cơ sở dữ liệu quan hệ hoặc một khung dữ liệu trong R/Python, nhưng có nhiều tối ưu hóa hơn. DataFrame có thể được xây dựng từ nhiều nguồn khác nhau như: tệp dữ liệu có cấu trúc, bảng trong Hive, cơ sở dữ liệu bên ngoài hoặc RDD hiện có. API DataFrame có sẵn trong Scala, Java, Python và R. Trong Scala và Java, DataFrame được biểu diễn bằng một Dataset của Rows. Trong API Scala , DataFrame chỉ đơn giản là một bí danh kiểu của Dataset[Row]. Trong khi đó, trong API Java , người dùng cần sử dụng Dataset<Row>để biểu diễn DataFrame.

##### SparkSession

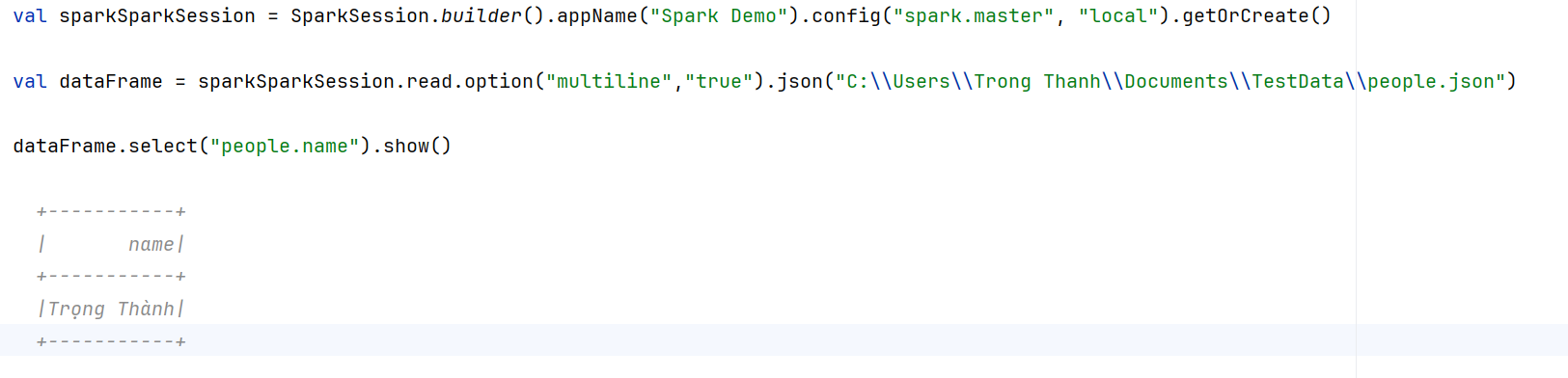
Điểm vào của tất cả các chức năng trong Spark là lớp [SparkSession](https://spark.apache.org/docs/latest/api/java/index.html#org.apache.spark.sql.SparkSession). Để tạo một lớp cơ bản SparkSession, chỉ cần sử dụng SparkSession.builder():



Khởi tạo DataFrames

Với SparkSession, các ứng dụng có thể tạo DataFrame từ dữ liệu [hiện có RDD](https://spark.apache.org/docs/latest/sql-getting-started.html#interoperating-with-rdds) , từ bảng Hive hoặc từ [các nguồn dữ liệu Spark](https://spark.apache.org/docs/latest/sql-data-sources.html) .

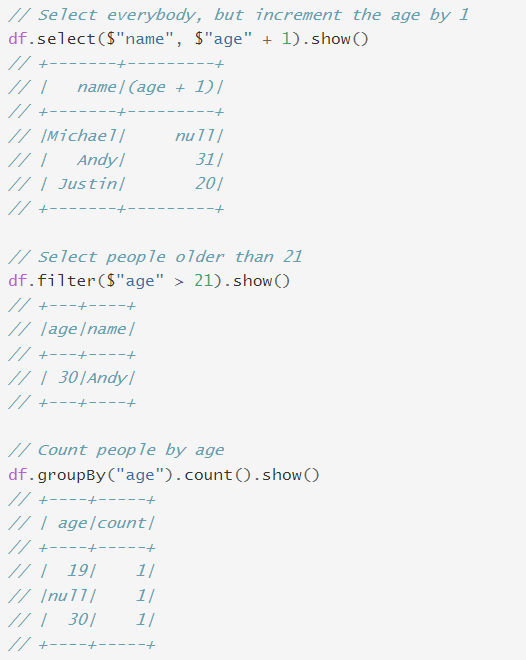
Ví dụ, đoạn mã sau đây tạo DataFrame dựa trên nội dung của tệp JSON



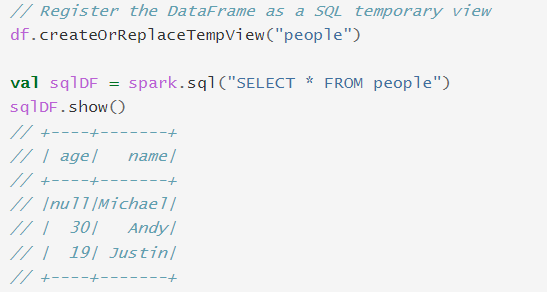
DataFrames cung cấp ngôn ngữ chuyên biệt cho việc xử lý dữ liệu có cấu trúc trong [Scala](https://spark.apache.org/docs/latest/api/scala/org/apache/spark/sql/Dataset.html) , [Java](https://spark.apache.org/docs/latest/api/java/index.html?org/apache/spark/sql/Dataset.html) , [Python](https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/api/pyspark.sql.DataFrame.html) và [R.](https://spark.apache.org/docs/latest/api/R/reference/SparkDataFrame.html)

Như đã đề cập ở trên, trong Spark 2.0, DataFrames chỉ là Dataset của Rows trong Scala và Java API. Các hoạt động này cũng được gọi là "untyped transformations" trái ngược với "typed transformations" đi kèm với các Dataset Scala/Java được typed mạnh.

Một số ví dụ cơ bản về xử lý dữ liệu có cấu trúc bằng cách sử dụng Bộ dữ liệu:



Hàm sql trên SparkSession cho phép các ứng dụng chạy các truy vấn SQL theo chương trình và trả về kết quả dưới dạng DataFrame.



##### Tạo bộ dữ liệu

Bộ dữ liệu tương tự như RDD, tuy nhiên, thay vì sử dụng Java serialization hoặc Kryo, chúng sử dụng [Encoder](https://spark.apache.org/docs/3.5.2/api/scala/org/apache/spark/sql/Encoder.html) chuyên dụng để serialize các đối tượng để xử lý hoặc truyền qua mạng. Trong khi cả bộ mã hóa và serialization chuẩn đều có trách nhiệm biến một đối tượng thành byte, thì bộ mã hóa là mã được tạo động và sử dụng định dạng cho phép Spark thực hiện nhiều thao tác như lọc, sắp xếp và băm mà không cần deserialize các byte trở lại thành một đối tượng.



##### Tương tác với RDD

Spark SQL hỗ trợ hai phương pháp khác nhau để chuyển đổi RDD hiện có thành Dataset. Phương pháp đầu tiên sử dụng phản xạ để suy ra lược đồ của RDD chứa các loại đối tượng cụ thể. Phương pháp dựa trên phản xạ này dẫn đến mã ngắn gọn hơn và hoạt động tốt khi bạn đã biết lược đồ khi viết ứng dụng Spark của mình.

Phương pháp thứ hai để tạo Dataset là thông qua giao diện lập trình cho phép bạn xây dựng lược đồ và sau đó áp dụng nó vào RDD hiện có. Mặc dù phương pháp này chi tiết hơn, nhưng nó cho phép bạn xây dựng Dataset khi các cột và kiểu của chúng không được biết cho đến khi chạy.

##### Nguồn dữ liệu

Spark SQL Công cụ này hỗ trợ nhiều nguồn dữ liệu khác nhau, bao gồm JDBC, ODBC, JSON, HDFS, Hive, ORC và Parquet. Các kho dữ liệu phổ biến khác như Amazon Redshift, Amazon S3, Couchbase, Cassandra, MongoDB, Salesforce.com, Elasticsearch và nhiều kho dữ liệu thuộc hệ sinh thái [Spark Packages](https://spark-packages.org/?q=tags%3A%22Data%20Sources%22).

Ví dụ:

Với Json

Spark SQL có thể tự động suy ra lược đồ của một tập dữ liệu JSON và tải nó dưới dạng Dataset[Row]. Chuyển đổi này có thể được thực hiện bằng cách sử dụng SparkSession.read.json()trên tệp Dataset[String], hoặc tệp JSON.

Lưu ý rằng tệp được cung cấp dưới dạng *tệp json* không phải là tệp JSON thông thường. Mỗi dòng phải chứa một đối tượng JSON hợp lệ, độc lập và riêng biệt. Để biết thêm thông tin, vui lòng xem [định dạng văn bản Dòng JSON, còn được gọi là JSON phân cách bằng dòng mới](http://jsonlines.org/) .

Đối với tệp JSON nhiều dòng thông thường, hãy đặt multiLine tùy chọn thành true.



#### Spark Streamming

##### Định nghĩa

Spark Streaming là phần mở rộng của API Spark cốt lõi cho phép xử lý luồng dữ liệu trực tiếp có khả năng mở rộng, thông lượng cao, chịu lỗi. Dữ liệu có thể được thu thập từ nhiều nguồn như Kafka, Kinesis hoặc socket TCP và có thể được xử lý bằng các thuật toán phức tạp được thể hiện bằng các hàm cấp cao như map, reduce, join và . Cuối cùng, dữ liệu đã xử lý có thể được đẩy ra hệ thống tệp, cơ sở dữ liệu và bảng điều khiển trực tiếp. Trên thực tế, bạn có thể áp dụng các thuật toán [học máy](https://spark.apache.org/docs/latest/ml-guide.html) và [xử lý đồ thị](https://spark.apache.org/docs/latest/graphx-programming-guide.html) window của Spark trên các luồng dữ liệu.



##### Cách hoạt động

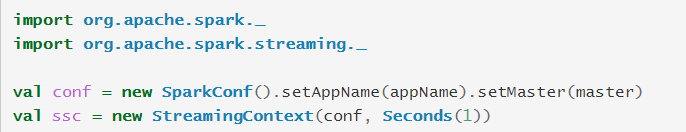
Về mặt nội bộ, nó hoạt động như sau. Spark Streaming nhận luồng dữ liệu đầu vào trực tiếp và chia dữ liệu thành từng đợt, sau đó được xử lý bởi công cụ Spark để tạo luồng kết quả cuối cùng theo từng đợt.



Spark Streaming cung cấp một sự trừu tượng hóa cấp cao được gọi là *luồng rời rạc* hoặc *DStream* , biểu diễn một luồng dữ liệu liên tục. DStream có thể được tạo từ các luồng dữ liệu đầu vào từ các nguồn như Kafka và Kinesis hoặc bằng cách áp dụng các hoạt động cấp cao trên các DStream khác. Về mặt nội bộ, DStream được biểu diễn dưới dạng một chuỗi [RDD](https://spark.apache.org/docs/3.5.2/api/scala/org/apache/spark/rdd/RDD.html) .

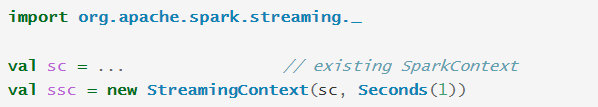
##### StreamingContext

Để khởi tạo chương trình Spark Streaming, cần phải tạo đối tượng StreamingContext , đây là điểm vào chính của mọi chức năng Spark Streaming.



Tham appNamesố là tên cho ứng dụng của bạn hiển thị trên UI cụm. masterlà [URL cụm Spark, Mesos, Kubernetes hoặc YARN](https://spark.apache.org/docs/3.5.2/submitting-applications.html#master-urls) hoặc chuỗi "local[\*]" đặc biệt để chạy ở chế độ cục bộ. Trong thực tế, khi chạy trên cụm, bạn sẽ không muốn mã hóa cứng mastertrong chương trình mà muốn [khởi chạy ứng dụng bằngspark-submit](https://spark.apache.org/docs/3.5.2/submitting-applications.html) và nhận ứng dụng tại đó. Tuy nhiên, đối với thử nghiệm cục bộ và thử nghiệm đơn vị, bạn có thể truyền "local[\*]" để chạy Spark Streaming trong quá trình (phát hiện số lõi trong hệ thống cục bộ). Lưu ý rằng điều này tạo ra một [SparkContext](https://spark.apache.org/docs/3.5.2/api/scala/org/apache/spark/SparkContext.html) (điểm bắt đầu của tất cả các chức năng Spark) có thể được truy cập dưới dạng ssc.sparkContext.

Một StreamingContextđối tượng cũng có thể được tạo ra từ một SparkContextđối tượng hiện có.



##### DStream

Discreted Stream hay DStream là sự trừu tượng cơ bản do Spark Streaming cung cấp. Nó biểu diễn một luồng dữ liệu liên tục, hoặc là luồng dữ liệu đầu vào nhận được từ nguồn hoặc là luồng dữ liệu đã xử lý được tạo ra bằng cách chuyển đổi luồng đầu vào. Về mặt nội bộ, DStream được biểu diễn bằng một chuỗi liên tục các RDD, là sự trừu tượng của Spark đối với một tập dữ liệu phân tán, không thay đổi (xem [Spark Programming Guide](https://spark.apache.org/docs/3.5.2/rdd-programming-guide.html#resilient-distributed-datasets-rdds) để biết thêm chi tiết). Mỗi RDD trong DStream chứa dữ liệu từ một khoảng thời gian nhất định, như thể hiện trong hình sau.



Bất kỳ thao tác nào được áp dụng trên DStream đều được chuyển thành các thao tác trên RDD cơ bản. Ví dụ, trong [ví dụ trước](https://spark.apache.org/docs/3.5.2/streaming-programming-guide.html#a-quick-example) về việc chuyển đổi luồng dòng thành từ, flatMap thao tác được áp dụng trên mỗi RDD trong lines DStream để tạo RDD của words DStream. Điều này được thể hiện trong hình sau.



#### Spark Mlib

##### Định nghĩa

MLlib là thư viện học máy (ML) của Spark. Mục tiêu của nó là làm cho việc học máy thực tế có thể mở rộng và dễ dàng. Ở cấp độ cao, nó cung cấp các công cụ như:

* Thuật toán ML: các thuật toán học tập phổ biến như phân loại, hồi quy, phân cụm và lọc cộng tác
* Đặc điểm hóa: trích xuất đặc điểm, chuyển đổi, giảm chiều và lựa chọn
* Đường ống: công cụ để xây dựng, đánh giá và điều chỉnh Đường ống ML
* Sự bền bỉ: lưu và tải các thuật toán, mô hình và Đường ống
* Tiện ích: đại số tuyến tính, thống kê, xử lý dữ liệu, v.v.

##### Các thuật toán học máy

Trong Spark MLlib, có nhiều thuật toán học máy phục vụ cho các tác vụ khác nhau như hồi quy, phân loại, phân cụm, và giảm chiều dữ liệu. Dưới đây là các thuật toán học máy phổ biến trong MLlib:

Học có giám sát (Supervised Learning):

* Hồi quy tuyến tính
* Hồi quy logistic
* Cây quyết định
* Rừng ngẫu nhiên
* …

Học không giám sát (Unsupervised Learning):

* K-mean
* Gaussian Mixture Models
* …

Xử lý dữ liệu văn bản:

* Word2Vec
* TF-IDF

Ngoài ra còn rất nhiều các thuật toán khác.

#### Spark GraphX

##### Định nghĩa

GraphX ​​là một thành phần mới trong Spark dành cho đồ thị và tính toán song song đồ thị. Ở cấp độ cao, GraphX ​​mở rộng Spark [RDD](https://spark.apache.org/docs/3.5.2/api/scala/org/apache/spark/rdd/RDD.html) bằng cách giới thiệu một dạng trừu tượng [đồ thị](https://spark.apache.org/docs/3.5.2/graphx-programming-guide.html#property_graph) mới : một đồ thị đa hướng có các thuộc tính được gắn vào mỗi đỉnh và cạnh. Để hỗ trợ tính toán đồ thị, GraphX ​​đưa ra một tập hợp các toán tử cơ bản (ví dụ: [subgraph](https://spark.apache.org/docs/3.5.2/graphx-programming-guide.html#structural_operators) , [joinVertices](https://spark.apache.org/docs/3.5.2/graphx-programming-guide.html#join_operators) và [aggregateMessages](https://spark.apache.org/docs/3.5.2/graphx-programming-guide.html#aggregateMessages) ) cũng như một biến thể được tối ưu hóa của [Pregel](https://spark.apache.org/docs/3.5.2/graphx-programming-guide.html#pregel) API. Ngoài ra, GraphX ​​bao gồm một bộ sưu tập ngày càng tăng [các thuật toán](https://spark.apache.org/docs/3.5.2/graphx-programming-guide.html#graph_algorithms) đồ thị và [trình xây dựng](https://spark.apache.org/docs/3.5.2/graphx-programming-guide.html#graph_builders) để đơn giản hóa các tác vụ phân tích đồ thị.

##### Biểu đồ Thuộc tính

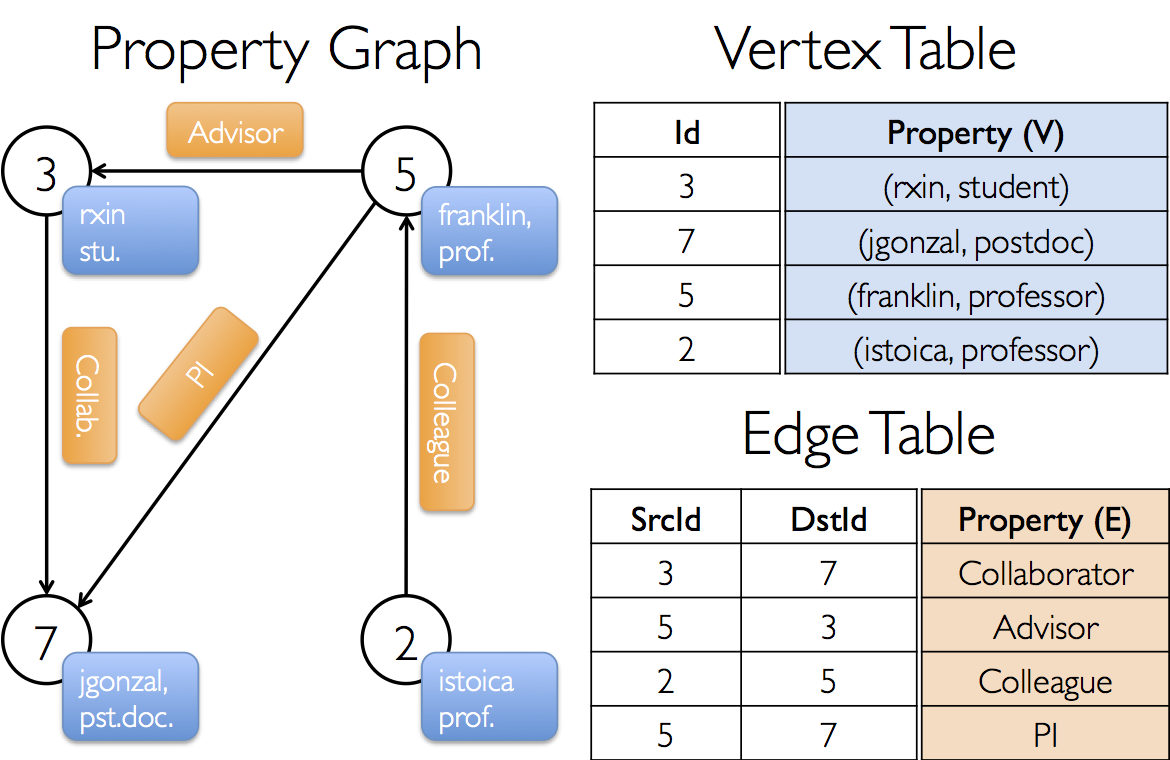
Đồ [thị thuộc tính](https://spark.apache.org/docs/3.5.2/api/scala/org/apache/spark/graphx/Graph.html) là một đồ thị đa hướng có các đối tượng do người dùng định nghĩa được gắn vào mỗi đỉnh và cạnh. Một đồ thị đa hướng là một đồ thị có hướng có nhiều cạnh song song có khả năng chia sẻ cùng một đỉnh nguồn và đích. Khả năng hỗ trợ các cạnh song song giúp đơn giản hóa các tình huống mô hình hóa trong đó có thể có nhiều mối quan hệ (ví dụ: đồng nghiệp và bạn bè) giữa các đỉnh giống nhau. Mỗi đỉnh được khóa bằng một định danh dài 64 bit *duy nhất*VertexId ( ). GraphX ​​không áp đặt bất kỳ ràng buộc sắp xếp nào lên các định danh đỉnh. Tương tự như vậy, các cạnh có các định danh đỉnh nguồn và đích tương ứng.

Đồ thị thuộc tính được tham số hóa qua các kiểu đỉnh ( VD) và cạnh ( ED). Đây là các kiểu đối tượng được liên kết với từng đỉnh và cạnh tương ứng.

GraphX ​​tối ưu hóa việc biểu diễn các kiểu đỉnh và cạnh khi chúng là các kiểu dữ liệu nguyên thủy (ví dụ: int, double, v.v.) giúp giảm dung lượng bộ nhớ bằng cách lưu trữ chúng trong các mảng chuyên dụng.

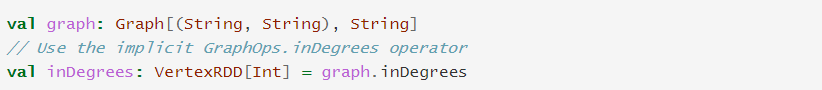
Ví dụ về biểu đồ thuộc tính

Giả sử chúng ta muốn xây dựng một đồ thị thuộc tính bao gồm nhiều cộng tác viên khác nhau trên dự án GraphX. Thuộc tính đỉnh có thể chứa tên người dùng và nghề nghiệp. Chúng ta có thể chú thích các cạnh bằng một chuỗi mô tả mối quan hệ giữa các cộng tác viên:



##### Toán tử đồ thị

Cũng giống như RDD có các phép toán cơ bản như map, filter, và reduceByKey, đồ thị thuộc tính cũng có một tập hợp các toán tử cơ bản lấy các hàm do người dùng định nghĩa và tạo ra các đồ thị mới với các thuộc tính và cấu trúc đã được chuyển đổi. Các toán tử cốt lõi có các triển khai được tối ưu hóa được định nghĩa trong [Graph](https://spark.apache.org/docs/3.5.2/api/scala/org/apache/spark/graphx/Graph$.html)và các toán tử tiện lợi được biểu thị dưới dạng các hợp phần của các toán tử cốt lõi được định nghĩa trong [GraphOps](https://spark.apache.org/docs/3.5.2/api/scala/org/apache/spark/graphx/GraphOps.html). Tuy nhiên, nhờ Scala ngầm định, các toán tử trong GraphOpstự động có sẵn dưới dạng các thành viên của Graph. Ví dụ, chúng ta có thể tính toán bậc trong của mỗi đỉnh (được định nghĩa trong GraphOps) theo công thức sau:



##### Toán tử cấu trúc

Hiện tại GraphX ​​chỉ hỗ trợ một tập hợp đơn giản các toán tử cấu trúc thường dùng và chúng tôi dự kiến ​​sẽ bổ sung thêm trong tương lai.

Toán [reverse](https://spark.apache.org/docs/3.5.2/api/scala/org/apache/spark/graphx/Graph.html#reverse:Graph[VD,ED]) tử trả về một đồ thị mới với tất cả các hướng cạnh bị đảo ngược. Điều này có thể hữu ích khi, ví dụ, cố gắng tính toán PageRank nghịch đảo. Vì phép toán đảo ngược không sửa đổi các thuộc tính đỉnh hoặc cạnh hoặc thay đổi số lượng cạnh, nên nó có thể được triển khai hiệu quả mà không cần di chuyển hoặc sao chép dữ liệu.

Toán [subgraph](https://spark.apache.org/docs/3.5.2/api/scala/org/apache/spark/graphx/Graph.html#subgraph((EdgeTriplet[VD,ED])%E2%87%92Boolean,(VertexId,VD)%E2%87%92Boolean):Graph[VD,ED]) tử lấy các vị từ đỉnh và cạnh và trả về đồ thị chỉ chứa các đỉnh thỏa mãn vị từ đỉnh (đánh giá là đúng) và các cạnh thỏa mãn vị từ cạnh *và kết nối các đỉnh thỏa mãn vị từ đỉnh* . subgraph Toán tử có thể được sử dụng trong một số tình huống để giới hạn đồ thị ở các đỉnh và cạnh quan tâm hoặc loại bỏ các liên kết bị hỏng.

Toán [mask](https://spark.apache.org/docs/3.5.2/api/scala/org/apache/spark/graphx/Graph.html#mask[VD2,ED2](Graph[VD2,ED2])(ClassTag[VD2],ClassTag[ED2]):Graph[VD,ED]) tử xây dựng một đồ thị con bằng cách trả về một đồ thị chứa các đỉnh và cạnh cũng có trong đồ thị đầu vào. Điều này có thể được sử dụng kết hợp với toán subgraphtử để hạn chế một đồ thị dựa trên các thuộc tính trong một đồ thị liên quan khác. Ví dụ, chúng ta có thể chạy các thành phần được kết nối bằng cách sử dụng đồ thị có các đỉnh bị thiếu và sau đó hạn chế câu trả lời cho đồ thị con hợp lệ.

Toán [groupEdges](https://spark.apache.org/docs/3.5.2/api/scala/org/apache/spark/graphx/Graph.html#groupEdges((ED,ED)%E2%87%92ED):Graph[VD,ED]) tử hợp nhất các cạnh song song (tức là các cạnh trùng lặp giữa các cặp đỉnh) trong đa đồ thị. Trong nhiều ứng dụng số, các cạnh song song có thể được *thêm vào* (trọng số của chúng được kết hợp) thành một cạnh duy nhất, do đó làm giảm kích thước của đồ thị.

##### API Pregel

Đồ thị vốn là cấu trúc dữ liệu đệ quy vì các thuộc tính của đỉnh phụ thuộc vào các thuộc tính của các đỉnh lân cận, mà các đỉnh lân cận này lại phụ thuộc vào các thuộc tính của *các* đỉnh lân cận. Do đó, nhiều thuật toán đồ thị quan trọng lặp đi lặp lại tính toán lại các thuộc tính của từng đỉnh cho đến khi đạt đến điều kiện điểm cố định. Một loạt các phép trừu tượng song song đồ thị đã được đề xuất để thể hiện các thuật toán lặp đi lặp lại này. GraphX ​​trình bày một biến thể của API Pregel.

### Nguồn dữ liệu

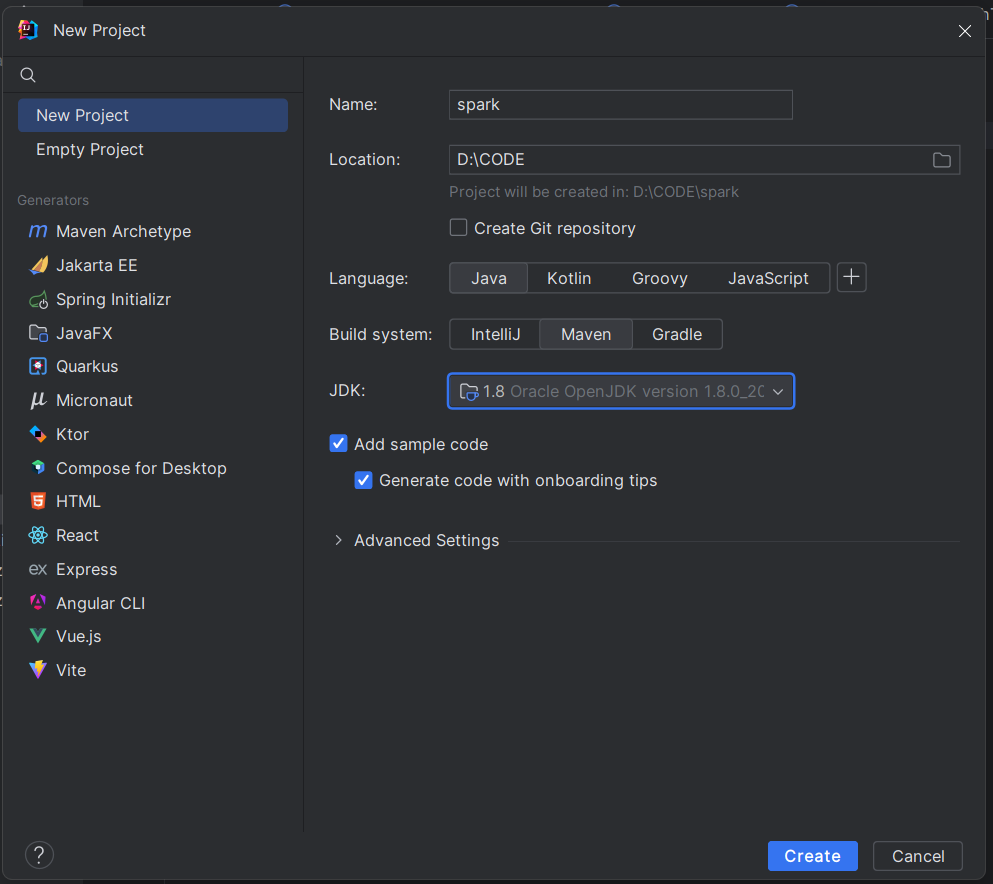
Apache Spark có thể kết nối đến các nguồn dữ liệu khác nhau như:

* Hệ thống tệp JSON
* Cơ sở dữ liệu NoSQL thông qua Cassandra, MongoDB
* Đến các cơ sở dữ liệu thông qua các dirver ODBC (Object Database Connectivity) như JDBC của Java, …
* Hệ thống tệp phân tán của Hadoop là HDFS (Hadoop Distributed File System)
* Hệ thống cloud như Microsoft Azure, Google Cloud Storage, …

### Cài đặt Spark

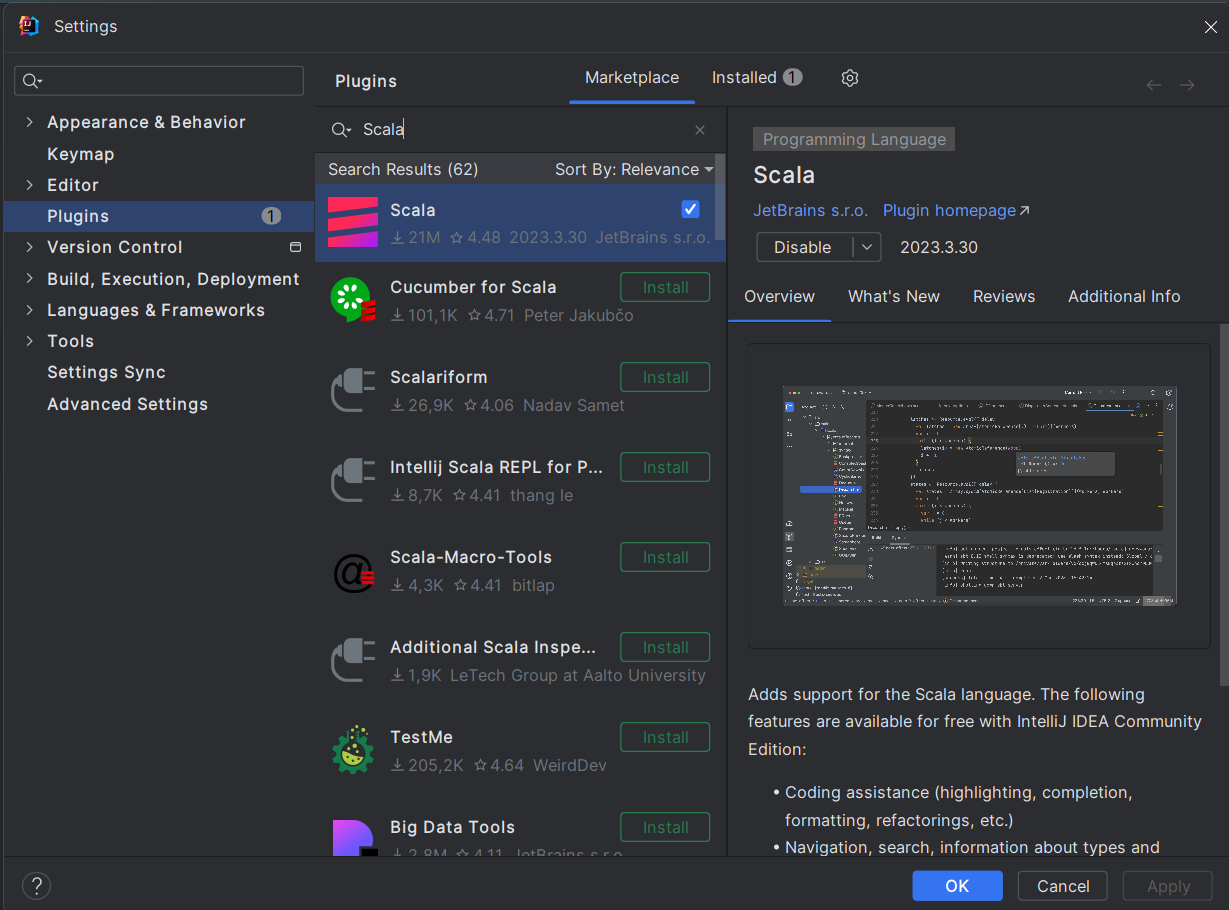
Có 2 cách để cài đặt Apache Spark sử dụng chế độ Local và Cluster, ở đây chúng tôi xin trình bày cách cài đặt, cấu hình Apache Spark trên môi trường phát triển tích hợp (IDE) cụ thể là trên Intelij IDEA ở Local mode

1. Tạo project mới với maven



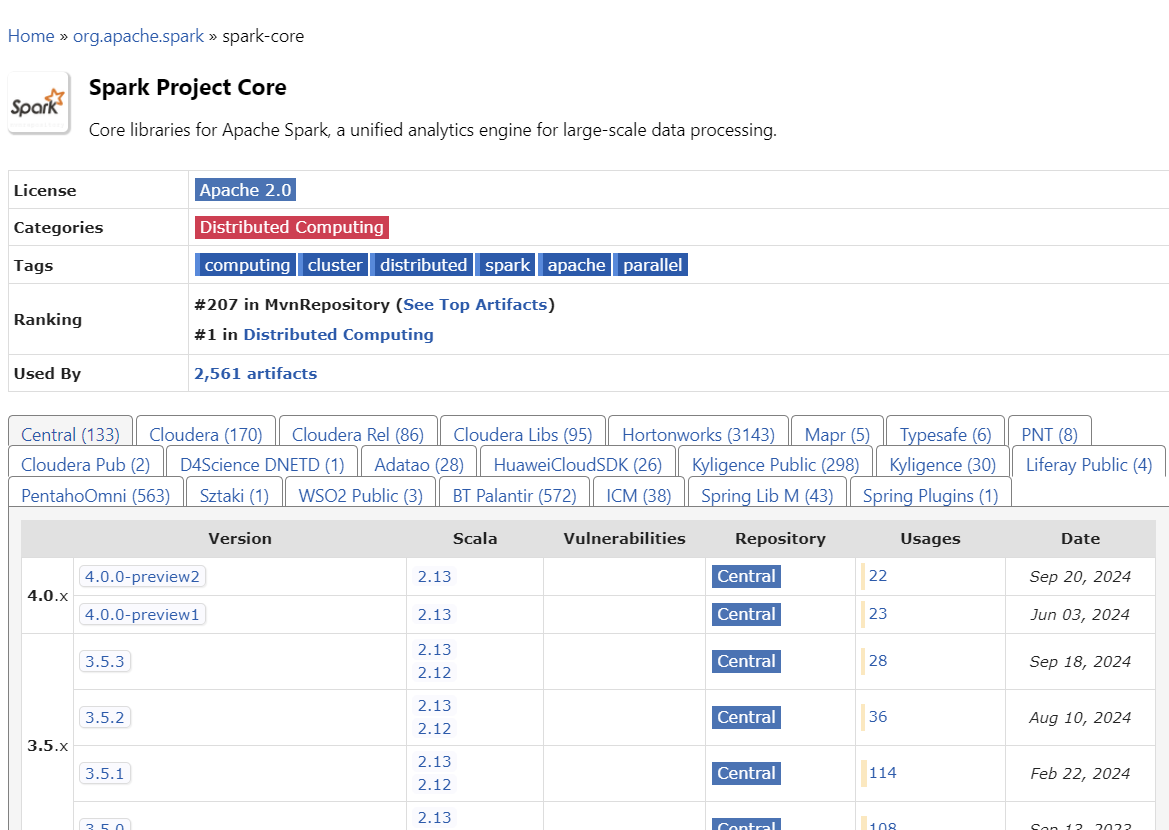
ở đây chúng tôi sử dụng JDK 8 (Java Deverloment Kit), ngoài ra có thể sử dụng JDK 11, JDK 17, một vài phiên bản khác sẽ không thể sử dụng để tạo trương trình với Spark

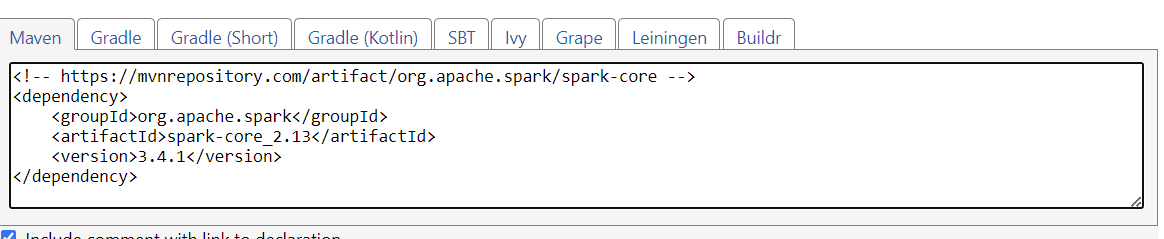
1. Vào phần cài đặt của IntelliJ -> Chọn Plugins -> Download Scala



1. Cài đặt Spark

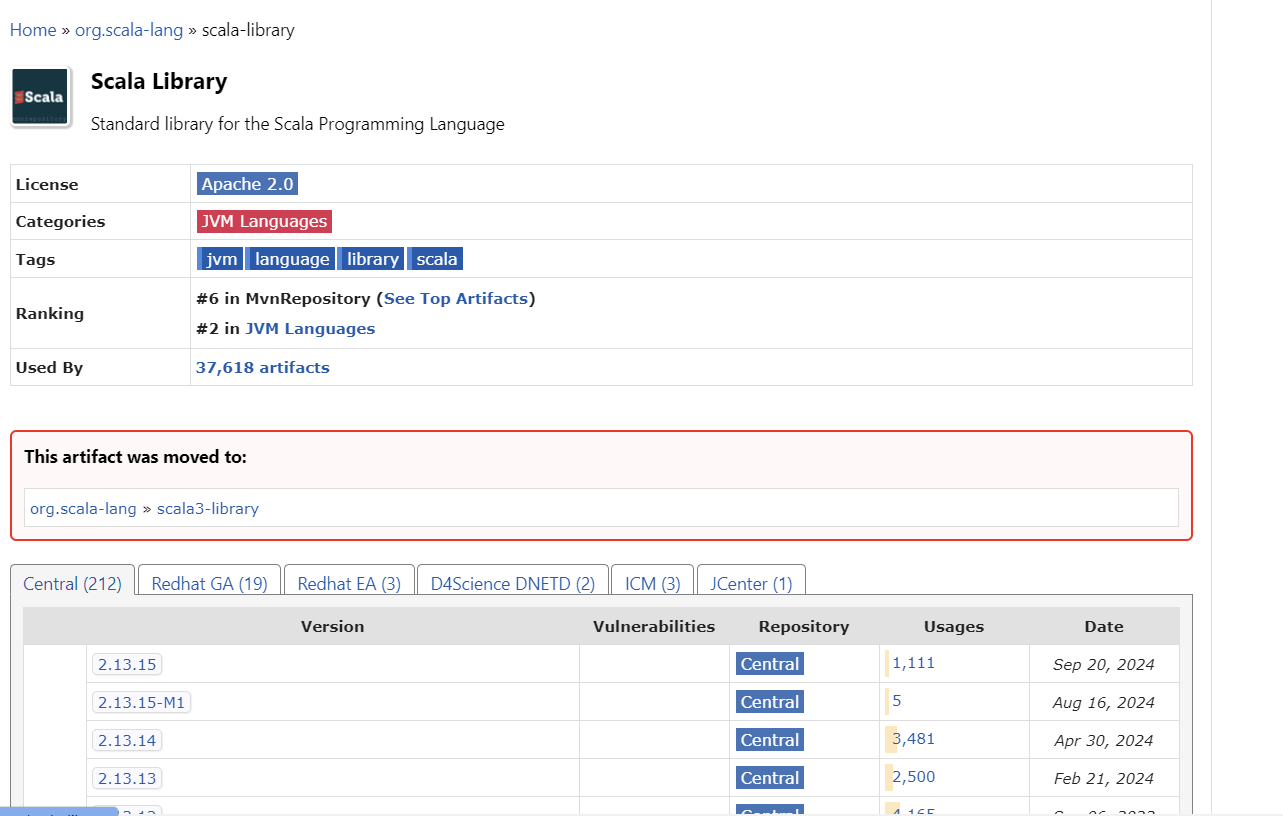
Vào trang web <https://mvnrepository.com/> và tìm kiểm thư viện Apache Spark là Spark Project Core

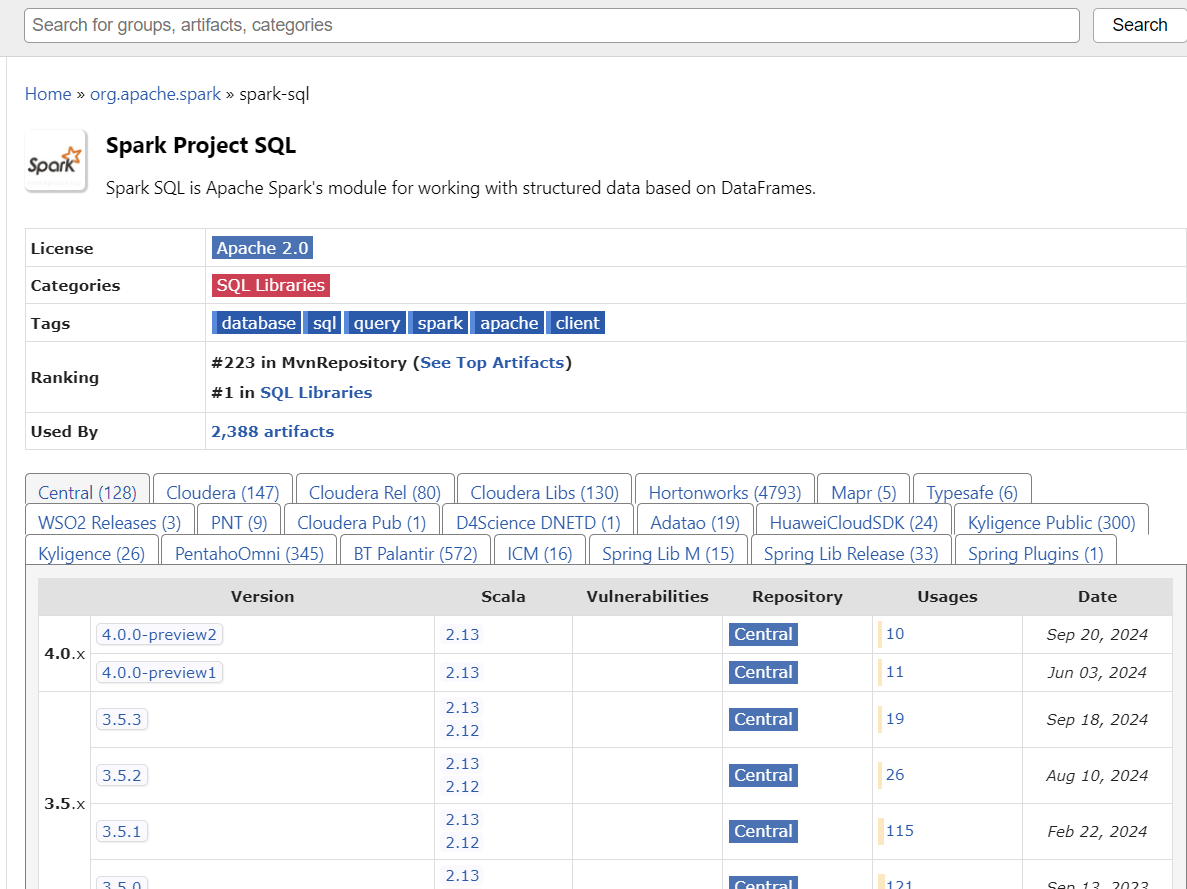




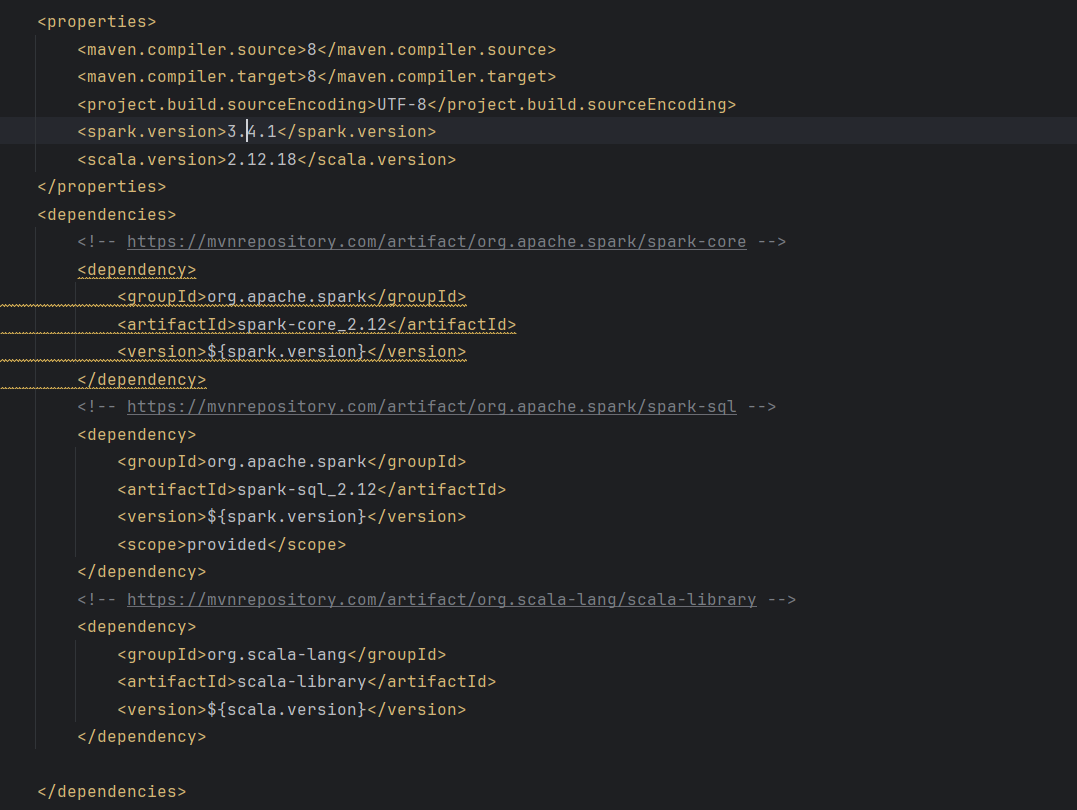
Tùy chọn một thư viện thích hợp để cài đặt, copy đoạn code xml để vào trong file pom.xml trong project

Làm tương tự với Spark Sql và thư viện Scala Library





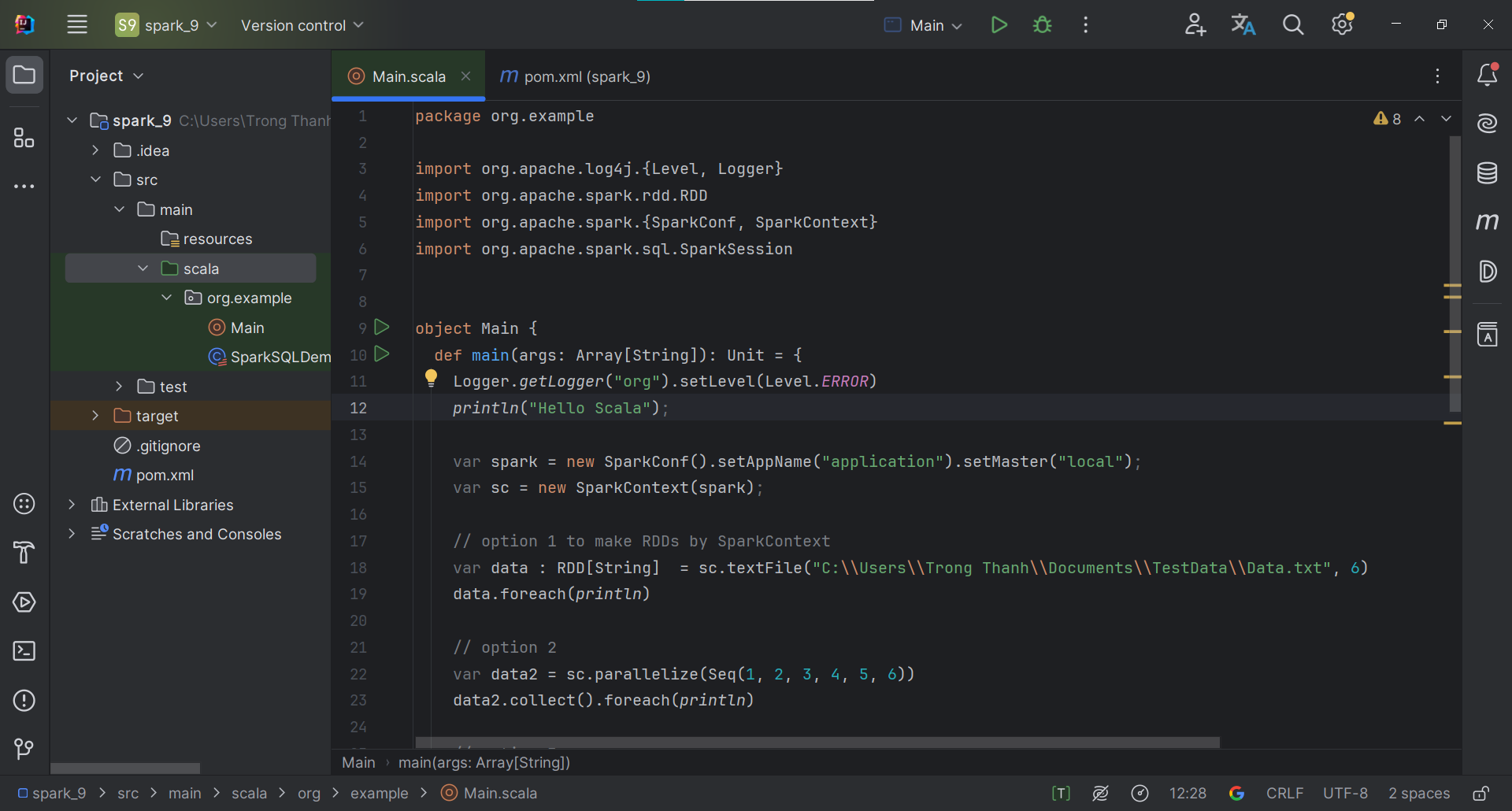
Trong file pom.xml ta sẽ có đoạn cấu hình như sau:



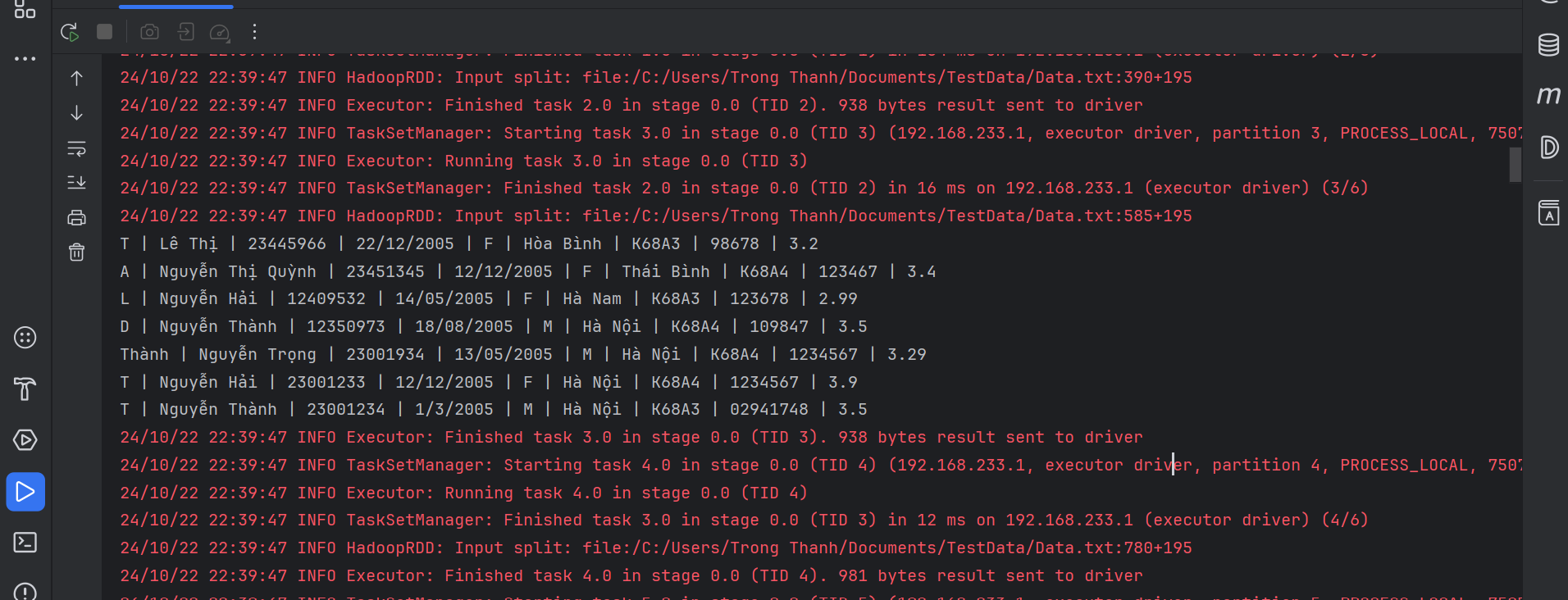
Để trương trình với maven có thể chạy được thì ta thêm đoạn code sau đây để xác định được file nguồn:



1. Sau đó click vào biểu tượng  trong file cấu hình pom để IDE cài đặt các thư viện đã nêu ở trên
2. Trương trình đầu tiên

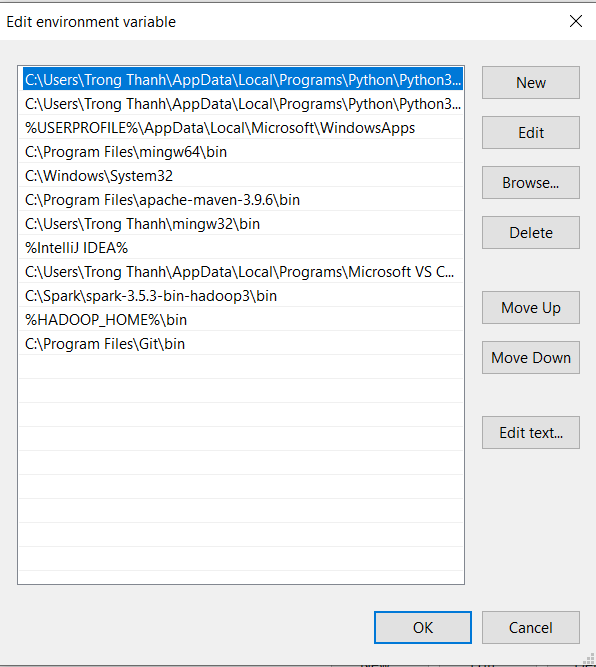


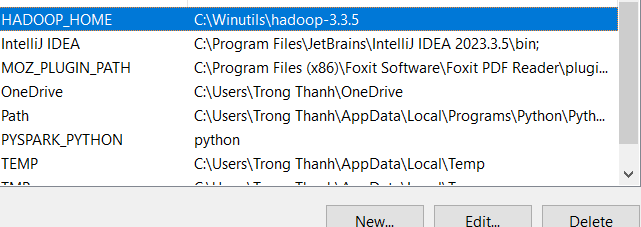
Kết quả:

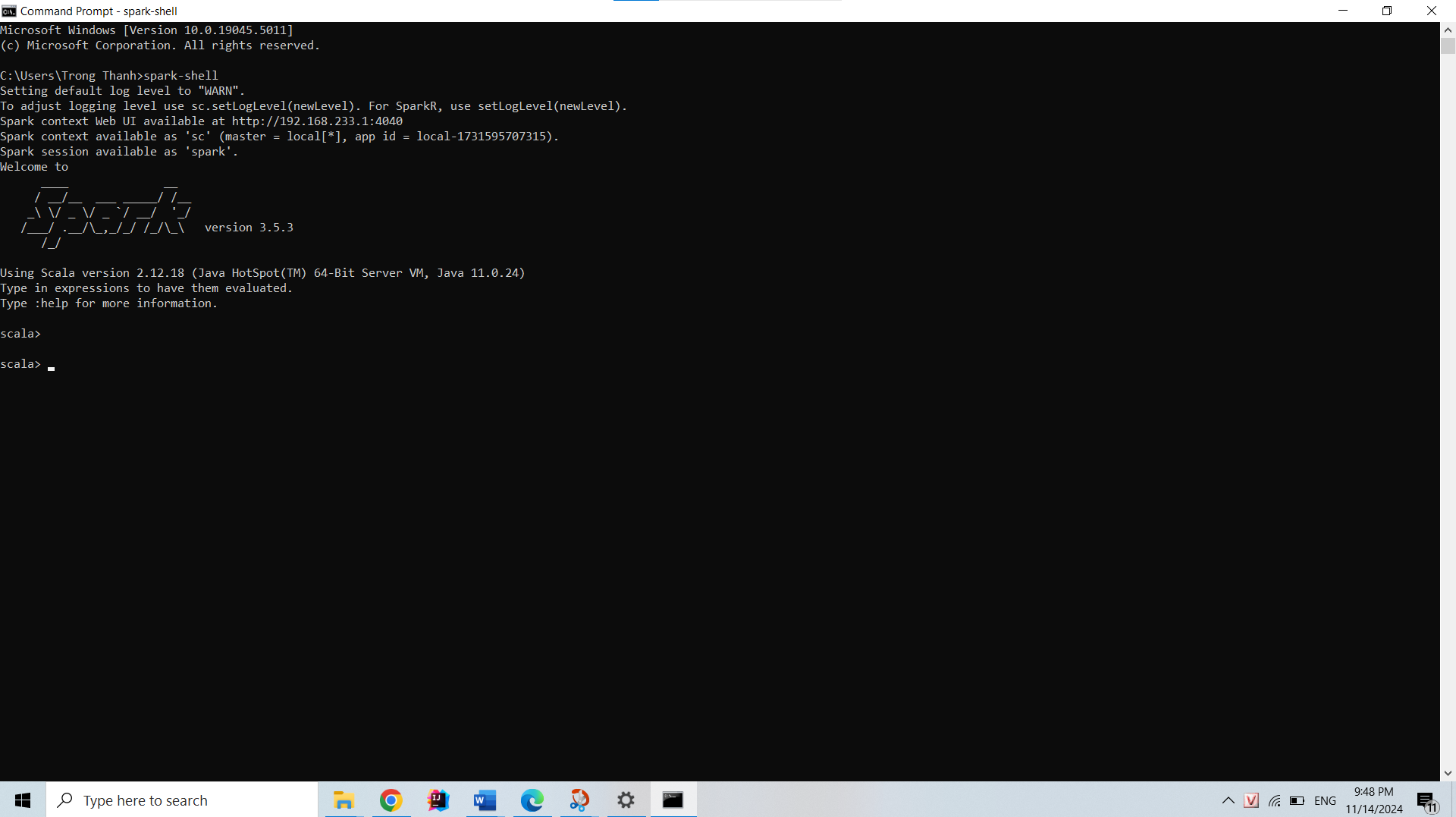


Cách 2 cài đặt Spark sử dụng chế độ local trực tiếp trên máy

* Bước 1 cài jdk-11, có nhiều phiên bản khác tuy nhiên những phiên bản khác khi chạy có thể gặp lỗi cao.
* Cài Spark tùy phiên bản bạn muốn sử dụng.
* Cài biến môi trường cho Spark và Jdk-11.



* Cài đặt Winutils và cài đặt biến môi trường cho nó.
* Chạy trương trình spark đầu tiên trên bằng cmd:



### Ứng dụng của Spark

Spark có thể được sử dụng trong nhiều kịch bản khác nhau, từ xử lý dữ liệu lớn đến học máy. Các ứng dụng chính của Spark bao gồm:

* Phân tích dữ liệu lớn
* Machine Learnning
* Sử lý thời gian thực
* Sử lý đồ thị
* Và các lĩnh vực khác trong đời sống

#### Phần tích dữ liệu lớn

**Phân tích dữ liệu lớn (Big Data Analytics)** với Apache Spark là một ứng dụng rất phổ biến và mạnh mẽ, đặc biệt trong các hệ thống xử lý dữ liệu khổng lồ phân tán. Spark hỗ trợ các truy vấn SQL, xử lý dữ liệu có cấu trúc và phi cấu trúc, và tối ưu hóa các hoạt động phân tích. Phân tích dữ liệu lớn với Spark được ứng dụng trong nhiều lĩnh vực như thương mại điện tử, tài chính, marketing, và khoa học dữ liệu.

**Cụ thể, phân tích dữ liệu gồm những thành phần chính như sau:**

**1,Spark SQL**

* **Spark SQL** là một thành phần quan trọng của Spark, cung cấp khả năng thực thi các truy vấn SQL trực tiếp trên các tập dữ liệu lớn. Spark SQL hỗ trợ hai mô hình dữ liệu chính:
* DataFrames: Tương tự như bảng trong SQL hoặc DataFrames trong Pandas, DataFrames là một cấu trúc dữ liệu có hàng và cột với khả năng thực thi các truy vấn SQL.
* Dataset API: Dataset là API cấp cao nhất trong Spark, giúp bạn thao tác dữ liệu có kiểu mạnh (strongly typed) và xử lý với hiệu năng cao.

**Ứng dụng:**

* **Báo cáo tài chính**: Truy vấn dữ liệu từ các bảng lớn để tạo ra các báo cáo tài chính, theo dõi doanh thu, và phân tích chi phí.
* **Phân tích khách hàng**: Chạy các truy vấn SQL trên dữ liệu hành vi người dùng để phân tích mô hình hành vi, phân khúc thị trường, và dự đoán xu hướng tiêu dùng.

**Ví dụ:** Nếu bạn có một tập dữ liệu lớn lưu trữ trong hệ thống Hadoop hoặc một cơ sở dữ liệu NoSQL như Cassandra, bạn có thể sử dụng Spark SQL để chạy các truy vấn SQL phân tán trên dữ liệu này. Điều này cho phép thực hiện các truy vấn phức tạp mà vẫn đảm bảo hiệu năng cao nhờ tính năng tối ưu hóa và phân tán dữ liệu của Spark.

2. **Xử lý dữ liệu có cấu trúc và phi cấu trúc**

Spark có khả năng xử lý nhiều loại dữ liệu, từ dữ liệu có cấu trúc như bảng SQL, dữ liệu bán cấu trúc như JSON, đến dữ liệu phi cấu trúc như file văn bản và log. Điều này giúp Spark trở thành công cụ hữu hiệu để phân tích dữ liệu từ nhiều nguồn khác nhau.

**Ứng dụng**

* **Phân tích log**: Dữ liệu log từ các server web hoặc ứng dụng có thể được phân tích để tìm lỗi hoặc các vấn đề hiệu suất.
* **Dữ liệu mạng xã hội**: Phân tích dữ liệu phi cấu trúc từ các mạng xã hội (như các bài viết, tweet) để phát hiện xu hướng hoặc cảm xúc.

**Ví dụ**

* Đọc dữ liệu JSON từ HDFS và thực hiện phân tích theo yêu cầu.
* Phân tích các file log phi cấu trúc để tìm kiếm lỗi hoặc các xu hướng cụ thể.

3. **Tối ưu hóa truy vấn dữ liệu**

Spark sử dụng **Catalyst Optimizer**, một engine tối ưu hóa truy vấn mạnh mẽ, giúp cải thiện hiệu suất của các phép tính và truy vấn SQL trên các tập dữ liệu lớn. Catalyst Optimizer tự động chọn cách tốt nhất để thực hiện các truy vấn dựa trên phân tích dữ liệu.

**Ứng dụng**

* **Tối ưu hóa pipeline ETL**: Tự động cải thiện hiệu suất khi thực hiện các quy trình ETL phức tạp trên các tập dữ liệu lớn.
* **Phân tích dự đoán**: Với các tập dữ liệu lớn, Spark giúp tối ưu hóa các truy vấn phức tạp, từ đó đẩy nhanh các phân tích dự đoán dựa trên dữ liệu lịch sử.

ví dụ

Khi thực hiện một truy vấn phức tạp trên các tập dữ liệu lớn, Spark tự động tối ưu hóa cách dữ liệu được truy cập và tính toán sao cho truy vấn diễn ra nhanh nhất có thể. Việc này bao gồm tối ưu hóa như:

* **Predicate Pushdown**: Chỉ tải về những phần của dữ liệu thực sự cần thiết dựa trên các điều kiện lọc.
* **Join Optimization**: Sắp xếp và ghép nối dữ liệu theo cách tối ưu nhất.

#### Học máy

Học máy (Machine Learning) với Apache Spark là một trong những ứng dụng nổi bật nhất của Spark nhờ khả năng xử lý phân tán và thư viện học máy MLlib mạnh mẽ. MLlib hỗ trợ một loạt các thuật toán học máy có khả năng xử lý lượng dữ liệu khổng lồ, từ các mô hình dự đoán đơn giản đến các hệ thống khuyến nghị phức tạp. Dưới đây là chi tiết về ứng dụng học máy trong Spark:

1, MLlib - Thư viện học máy của Spark

**MLlib** (Machine Learning Library) là một thư viện học máy phân tán trong Spark. Nó cung cấp nhiều thuật toán và công cụ cho các bài toán học máy, bao gồm phân loại, hồi quy, clustering, giảm chiều dữ liệu, và hệ thống khuyến nghị.

**Các tính năng chính của MLlib**:

* **Tập hợp thuật toán phong phú**: Bao gồm phân loại (classification), hồi quy (regression), phân cụm (clustering), hệ thống gợi ý (recommendation system), và khai phá các luật kết hợp (frequent pattern mining).
* **Xử lý dữ liệu trên cụm phân tán**: MLlib chia nhỏ dữ liệu và xử lý phân tán trên nhiều nút, giúp xử lý nhanh với dữ liệu lớn.
* **Tích hợp với DataFrame và Dataset**: Dữ liệu có thể được chuyển đổi và thao tác dưới dạng DataFrame hoặc Dataset, giúp làm việc dễ dàng hơn với dữ liệu có cấu trúc.

2, Thuật toán phổ biến trong Mllib:

a, phân loại: Phân loại là một trong những bài toán phổ biến nhất trong học máy, trong đó mô hình dự đoán nhãn của một đối tượng dựa trên dữ liệu đầu vào.Các thuật toán phổ biến :

* Logistic Regression (Hồi quy Logistic)
* Random Forest
* Gradient-Boosted Trees (GBT)
* Support Vector Machines (SVMs)

Ứng dụng:

* Phân loại email: Dùng để phân loại email vào các danh mục như spam và không spam.
* Chẩn đoán bệnh: Phân loại các hồ sơ y tế để dự đoán bệnh dựa trên các yếu tố y tế và sinh trắc học.

b, Hồi quy(Regression): Hồi quy là quá trình dự đoán giá trị liên tục. Spark MLlib hỗ trợ nhiều thuật toán hồi quy như:

* **Linear Regression** (Hồi quy tuyến tính)
* **Decision Trees** (Cây quyết định)
* **Random Forest Regression**
* **Gradient-Boosted Trees Regression**

Ứng dụng:

* Dự đoán giá nhà: Dùng hồi quy tuyến tính để dự đoán giá nhà dựa trên các thuộc tính như diện tích, vị trí, số phòng ngủ.
* Dự đoán doanh thu bán hàng: Sử dụng hồi quy để dự đoán doanh thu dựa trên các yếu tố như xu hướng tiêu dùng, mùa bán hàng, và dữ liệu lịch sử.

c, phân cụm (Clustering): là quá trình nhóm các đối tượng tương tự lại với nhau thành các nhóm (clusters). Spark MLlib cung cấp các thuật toán phân cụm như:

* **K-Means**: Một trong những thuật toán phân cụm phổ biến nhất, giúp chia dữ liệu thành K nhóm.
* **Gaussian Mixture Models (GMM)**: Sử dụng mô hình phân phối Gaussian để phân cụm dữ liệu.

Ứng dụng:

* Phân đoạn khách hàng: Nhóm các khách hàng có hành vi hoặc đặc điểm tương tự nhau, từ đó tạo ra chiến lược marketing riêng biệt cho từng nhóm.
* Phân tích thị trường: Sử dụng phân cụm để tìm hiểu các phân khúc trong thị trường và phát triển chiến lược kinh doanh.

3. **Xử lý dữ liệu lớn trong học máy**

Spark có thể xử lý dữ liệu lớn trong học máy nhờ khả năng phân tán dữ liệu và tính toán trên nhiều máy. Đối với các bài toán cần xử lý lượng dữ liệu lớn (Big Data), Spark MLlib cho phép thực hiện các phép tính học máy một cách nhanh chóng nhờ vào việc chia nhỏ dữ liệu và phân phối công việc trên nhiều máy trong cụm (cluster).

Ứng dụng:

* Học máy phân tán: Spark được sử dụng để huấn luyện các mô hình học máy trên dữ liệu lớn, ví dụ như trong các hệ thống thương mại điện tử, mạng xã hội, hoặc hệ thống tài chính.
* Phân tích dữ liệu thời gian thực: Kết hợp Spark Streaming với MLlib, người dùng có thể xử lý dữ liệu và áp dụng các mô hình học máy trên luồng dữ liệu thời gian thực.

4. **Pipeline Học Máy (ML Pipeline)**

Spark MLlib hỗ trợ việc xây dựng **pipeline học máy**, giúp đơn giản hóa quá trình từ việc chuẩn bị dữ liệu, huấn luyện mô hình, đến việc đánh giá và triển khai mô hình. ML Pipeline của Spark cho phép xây dựng chuỗi các bước học máy một cách rõ ràng và dễ duy trì.

Ứng dụng:

* Chuỗi xử lý tự động: Dùng pipeline để tự động hóa toàn bộ quy trình học máy từ chuẩn bị dữ liệu đến triển khai mô hình.
* Dự báo nhu cầu sản phẩm: Tạo pipeline học máy để dự đoán nhu cầu tiêu dùng dự

#### Xử lý thời gian thực

**Xử lý thời gian thực (Real-time Processing)** là một trong những ứng dụng mạnh mẽ nhất của **Apache Spark**, đặc biệt với mô-đun **Spark Streaming** (hiện nay là **Structured Streaming** trong các phiên bản mới). Khả năng xử lý thời gian thực cho phép Spark thu thập, xử lý và phân tích dữ liệu ngay khi dữ liệu đến từ các nguồn khác nhau như cảm biến IoT, hệ thống giao dịch tài chính, hoặc các nền tảng truyền thông xã hội.

**Khái niệm về xử lý thời gian thực với Spark**

Trong xử lý thời gian thực, dữ liệu liên tục được gửi đến hệ thống từ các nguồn khác nhau, và yêu cầu hệ thống xử lý ngay lập tức hoặc gần như ngay lập tức để tạo ra thông tin hữu ích. Spark Streaming và Structured Streaming là các thành phần của Spark, cho phép xử lý dữ liệu theo luồng (streaming data) với khả năng phân tán và mở rộng.

**Ứng dụng của xử lý thời gian thực với Spark**

a, phân luồng sự kiện từ cảm biến IoT:

trên 1 hệ thổng IoT có hàng triệu cảm biến như nhiệt độ, độ ẩm, độ cao, trọng lượng,… Apache Spark Streaming có thể thu thập các luồng dữ liệu từ các thiết bị này và xử lý chúng theo thời gian thực, phát hiện các sự kiện bất thường, dự báo xu hướng, hoặc gửi cảnh báo.

Ứng dụng:

* Giám sát môi trường thời gian thực: Dữ liệu từ cảm biến môi trường (nhiệt độ, độ ẩm) được phân tích theo thời gian thực để phát hiện biến đổi khí hậu hoặc tình trạng khẩn cấp.
* Phân tích thiết bị công nghiệp: Dữ liệu từ các thiết bị máy móc được giám sát liên tục để phát hiện lỗi, tối ưu hóa quy trình vận hành và bảo trì dự đoán.

b, Phân tích dữ liệu từ mạng xã hội: Spark Streaming có thể thu thập dữ liệu từ các nền tảng mạng xã hội như Twitter hoặc Facebook, phân tích dữ liệu về các xu hướng, sự kiện hoặc cảm xúc thời gian thực.

Ứng dụng:

* Phân tích cảm xúc người dùng: Thu thập và phân tích các bài đăng trên mạng xã hội để đo lường cảm xúc công chúng theo thời gian thực.

Theo dõi xu hướng: Các công ty marketing có thể sử dụng Spark để phát hiện các xu hướng đang nổi trên Twitter và phản ứng nhanh chóng.

c, **Khả năng mở rộng và độ trễ thấp trong xử lý thời gian thực:**

\*Ứng dụng:

* **Hệ thống giao dịch tài chính**: Các hệ thống giao dịch chứng khoán hoặc ngân hàng cần xử lý hàng triệu giao dịch mỗi giây với độ trễ cực thấp để phản ứng kịp thời với biến động thị trường.
* **Phân tích dữ liệu thời gian thực trong eCommerce**: Hệ thống thương mại điện tử sử dụng Spark Streaming để theo dõi các giao dịch mua sắm, tương tác của khách hàng và phát hiện các hành vi gian lận trong thời gian thực.

d, **Tích hợp với các hệ thống khác:** Spark Streaming có thể dễ dàng tích hợp với các hệ thống khác như Kafka, HDFS, S3, và hệ thống cơ sở dữ liệu NoSQL (Cassandra, HBase).

Ứng dụng:

* **Hệ thống thu thập log server**: Sử dụng Kafka và Spark Streaming để thu thập và phân tích log từ hàng trăm server, phát hiện các lỗi và tấn công bảo mật ngay lập tức.
* **Hệ thống thông báo thời gian thực**: Tạo các cảnh báo dựa trên dữ liệu thời gian thực từ Kafka và gửi thông báo tới người dùng khi phát hiện các sự kiện quan trọng.

e, **Spark Streaming và MLlib cho học máy thời gian thực**

Ứng dụng:

* **Phát hiện gian lận**: Hệ thống có thể học từ các mẫu giao dịch gian lận trong quá khứ và sử dụng Spark Streaming để phát hiện gian lận trong thời gian thực khi giao dịch mới diễn ra.
* **Tối ưu hóa trải nghiệm người dùng**: Cập nhật hệ thống khuyến nghị sản phẩm dựa trên hành vi thời gian thực của khách hàng để đưa ra các đề xuất sản phẩm phù hợp ngay lập tức.

#### Ứng dụng khác

**1, Phân tích dữ liệu (Data Analytics)**

Spark hỗ trợ các hoạt động phân tích dữ liệu lớn, chẳng hạn như việc phân tích log, phân tích dữ liệu người dùng, và xử lý ETL (Extract, Transform, Load). Tích hợp với SQL thông qua **Spark SQL** giúp xử lý truy vấn SQL nhanh chóng trên các bộ dữ liệu lớn.

**2, Xử lý đồ thị (Graph Processing)**

Spark có **GraphX**, một API dành riêng cho việc xử lý đồ thị và tính toán đồ thị phân tán. Điều này rất hữu ích trong các ứng dụng như phân tích mạng xã hội, phân tích đồ thị đường đi ngắn nhất, hoặc hệ thống gợi ý.

**3, Hệ thống khuyến nghị (Recommendation Systems)**

Với khả năng xử lý dữ liệu lớn và thư viện học máy mạnh mẽ, Spark là một lựa chọn tuyệt vời cho việc xây dựng các hệ thống gợi ý (recommendation systems), ví dụ như đề xuất sản phẩm, phim ảnh dựa trên sở thích của người dùng.

## So sánh với Hadoop

### Kiến trúc

Apache Spark:

Kiến trúc in-memory: Spark sử dụng bộ nhớ (RAM) để lưu trữ dữ liệu trong quá trình tính toán, nhờ đó tốc độ xử lý rất nhanh. Nó đọc dữ liệu vào bộ nhớ, thực hiện tính toán và chỉ ghi dữ liệu xuống đĩa khi cần thiết.

Fault-tolerance (Tính chịu lỗi): Spark sử dụng một hệ thống "Directed Acyclic Graph (DAG)" và khái niệm "Resilient Distributed Datasets (RDD)" để phục hồi từ lỗi một cách nhanh chóng.

Hỗ trợ nhiều kiểu xử lý: Spark hỗ trợ các loại xử lý như batch, streaming, interactive, và machine learning trong cùng một framework.

Apache Hadoop:

Kiến trúc dựa trên MapReduce: Hadoop sử dụng mô hình MapReduce để chia nhỏ các nhiệm vụ tính toán. Nó dựa trên việc ghi/đọc dữ liệu từ đĩa cứng giữa các bước xử lý, làm cho nó chậm hơn so với Spark.

Fault-tolerance: Hadoop đảm bảo tính chịu lỗi bằng cách lưu trữ dữ liệu trong HDFS (Hadoop Distributed File System) với cơ chế sao lưu (replication). Nếu một node bị lỗi, dữ liệu từ các bản sao sẽ được sử dụng.

Tập trung vào xử lý batch: Hadoop chủ yếu được thiết kế để xử lý dữ liệu theo lô, và không có khả năng xử lý dữ liệu streaming hay interactive mặc định.

### Hiệu năng

Apache Spark:

Tốc độ: Spark có hiệu năng cao hơn so với Hadoop vì nó sử dụng xử lý dữ liệu trong bộ nhớ (in-memory). Điều này giúp giảm thiểu thời gian truy cập và đọc/ghi dữ liệu từ đĩa.

Parallelism: Spark hỗ trợ xử lý song song cao, với khả năng chia nhỏ công việc để thực thi trên nhiều lõi CPU, giúp xử lý dữ liệu nhanh hơn so với Hadoop.

Ưu điểm cho dữ liệu lớn: Đặc biệt khi làm việc với các bài toán phân tích dữ liệu phức tạp và yêu cầu tính toán lặp đi lặp lại (iterative algorithms), Spark vượt trội hơn Hadoop.

In-memory processing: Nhờ xử lý dữ liệu trong bộ nhớ, Spark có tốc độ vượt trội so với Hadoop trong nhiều trường hợp, đặc biệt là với các bài toán lặp lại nhiều lần (iterative algorithms).

Tối ưu hóa DAG: Spark sử dụng DAG cho phép tối ưu hóa luồng công việc (workflow) trước khi thực hiện, giảm thiểu chi phí và cải thiện hiệu suất.

Apache Hadoop:

Disk-based processing: Hadoop sử dụng đĩa cứng để lưu trữ và trao đổi dữ liệu giữa các bước Map và Reduce, dẫn đến hiệu suất chậm hơn, đặc biệt là đối với các bài toán cần nhiều vòng lặp hoặc tính toán phức tạp.

Batch processing: Hadoop hoạt động tốt nhất với các bài toán xử lý dữ liệu theo lô với số lượng lớn, nhưng ít thích hợp cho các bài toán cần xử lý nhanh chóng.

Tốc độ: Hadoop xử lý dữ liệu thông qua quá trình lưu trữ và lấy dữ liệu từ đĩa, nên tốc độ chậm hơn Spark trong các bài toán phức tạp hoặc đòi hỏi nhiều vòng lặp.

Batch Processing: Hadoop thường được dùng cho xử lý dữ liệu theo lô (batch processing), thích hợp cho các tác vụ đơn giản nhưng với khối lượng dữ liệu lớn.

I/O overhead: Do phải liên tục đọc và ghi dữ liệu từ đĩa, hiệu năng của Hadoop thường thấp hơn khi xử lý khối lượng dữ liệu lớn có tính phức tạp.

### Cách dùng

Apache Spark:

Dễ sử dụng hơn: Spark có các API phong phú và dễ hiểu cho các ngôn ngữ phổ biến như Python, Scala, Java, và R, giúp lập trình viên dễ dàng viết và chạy các ứng dụng hơn.

Hỗ trợ đa dạng các loại dữ liệu: Spark hỗ trợ cả dữ liệu theo batch và streaming thông qua các module như Spark Streaming, Spark SQL, và Spark MLlib.

Tương thích với Hadoop: Spark có thể chạy trên các cluster Hadoop sử dụng YARN và HDFS, do đó các doanh nghiệp có thể tận dụng cơ sở hạ tầng Hadoop hiện có để chạy Spark mà không cần đầu tư mới.

**In-memory processing:** Spark có khả năng xử lý dữ liệu trực tiếp trong bộ nhớ RAM, giúp dễ dàng thực hiện các bài toán phức tạp như machine learning hoặc xử lý streaming.

Apache Hadoop:

Phức tạp hơn: Việc phát triển ứng dụng trên Hadoop đòi hỏi lập trình viên phải viết mã phức tạp hơn, đặc biệt là khi phải quản lý các bước xử lý Map và Reduce. Điều này đòi hỏi kiến thức sâu hơn về cách hoạt động của MapReduce.

Hệ sinh thái phong phú: Mặc dù bản thân Hadoop MapReduce không dễ sử dụng, nhưng hệ sinh thái Hadoop bao gồm các công cụ như Hive, Pig, và HBase giúp làm đơn giản hóa việc xử lý dữ liệu.

Tập trung vào batch processing: Hadoop chủ yếu phù hợp với các tác vụ xử lý theo lô (batch), không tối ưu cho các bài toán tương tác (interactive) hoặc thời gian thực (real-time).

### Chi phí

Apache Spark

Tài nguyên phần cứng: Spark yêu cầu nhiều RAM hơn so với Hadoop để tối ưu hóa hiệu suất in-memory, do đó chi phí phần cứng có thể cao hơn.

Chi phí vận hành: Mặc dù chi phí phần cứng cao hơn, Spark giảm thời gian xử lý, do đó có thể tiết kiệm chi phí vận hành lâu dài (như chi phí cho điện và tài nguyên tính toán).

Tính mở rộng: Spark có thể hoạt động trên các cụm (clusters) lớn, Spark cũng có thể hoạt động trên nhiều nền tảng như Mesos, Hadoop YARN, Kubernetes hoặc standalone cluster và khả năng tối ưu hóa hiệu quả sử dụng tài nguyên giúp giảm thiểu chi phí khi mở rộng hệ thống.

Quản lý chi phí: Tuy nhiên, đối với các hệ thống có khả năng mở rộng lớn (distributed systems), việc xử lý nhanh chóng và hiệu quả của Spark giúp tiết kiệm chi phí về lâu dài (giảm thời gian xử lý và tài nguyên).

Apache Hadoop:

Phần cứng yêu cầu thấp hơn: Hadoop thường yêu cầu ít bộ nhớ hơn Spark do nó không xử lý dữ liệu trong RAM, thay vào đó Hadoop có thể hoạt động tốt trên các hệ thống phần cứng với cấu hình thấp hơn

Chi phí vận hành: Hadoop có chi phí phần cứng ban đầu thấp hơn vì không cần yêu cầu bộ nhớ cao, nhưng do tốc độ xử lý chậm hơn, chi phí tính toán trên thực tế có thể tăng cao hơn theo thời gian xử lý, đặc biệt là khi xử lý dữ liệu lớn.

Mở rộng tốt: Hadoop có khả năng mở rộng cao khi xử lý khối lượng dữ liệu rất lớn, tuy nhiên, chi phí khi mở rộng thường liên quan đến việc duy trì và quản lý tài nguyên lưu trữ (đĩa cứng).

Open-source: Cả Spark và Hadoop đều là mã nguồn mở, nên không mất chi phí bản quyền phần mềm. Tuy nhiên, chi phí vận hành (nhân sự, bảo trì, phần cứng) vẫn phải được tính đến.

### Khả năng chịu lỗi

Apache Spark:

RDD (Resilient Distributed Datasets): Spark sử dụng RDD để quản lý dữ liệu phân tán. RDD là một cấu trúc dữ liệu chịu lỗi cho phép Spark khôi phục dữ liệu từ các thao tác trước đó nếu gặp lỗi.

Checkpointing: Đối với các bài toán phức tạp hoặc dài hạn, Spark hỗ trợ checkpoint để lưu trữ trạng thái dữ liệu trên đĩa nhằm phòng tránh việc mất dữ liệu khi có lỗi xảy ra.

Apache Hadoop:

HDFS: Hadoop sử dụng HDFS để lưu trữ dữ liệu phân tán và sử dụng cơ chế sao lưu dữ liệu (data replication) để đảm bảo tính an toàn. Nếu một node gặp sự cố, Hadoop có thể truy cập dữ liệu từ các bản sao khác.

### Ứng dụng

Apache Spark:

Xử lý dữ liệu thời gian thực (real-time): Spark Streaming là lựa chọn tốt cho các bài toán yêu cầu xử lý dữ liệu streaming hoặc phân tích dữ liệu thời gian thực.

Phân tích dữ liệu phức tạp: Spark phù hợp với các bài toán machine learning, phân tích đồ thị, và các mô hình tính toán cần nhiều vòng lặp.

Interactive querying: Spark SQL cho phép truy vấn dữ liệu tương tác với hiệu suất cao.

Hadoop:

Xử lý dữ liệu batch: Hadoop phù hợp với các hệ thống xử lý dữ liệu theo lô lớn, như ETL (Extract, Transform, Load) hoặc phân tích dữ liệu lớn với các bước đơn giản.

Lưu trữ và quản lý dữ liệu: HDFS trong Hadoop thích hợp để lưu trữ lượng lớn dữ liệu không cấu trúc hoặc nửa cấu trúc, như log files, cảm biến IoT, và dữ liệu từ mạng xã hội.

## Tổng kết

### Ưu điểm

* Dễ sử dụng:

Một trong những điểm mạnh quan trọng của Apache Spark là sự dễ tiếp cận. Framework này được thiết kế với mục tiêu giảm độ phức tạp và cung cấp một giao diện người dùng thân thiện. Người dùng chỉ cần có kiến thức cơ bản về database và lập trình Python hoặc Scala, mà không yêu cầu sự hiểu biết sâu rộng về hệ thống hay lập trình phức tạp. Điều này làm cho Apache Spark trở thành một công cụ hữu ích cho mọi người từ các nhà phân tích dữ liệu đến các nhà phát triển ứng dụng.

So với Hadoop, Apache Spark nổi bật với độ ngắn gọn và thuận tiện hơn trong quá trình triển khai và sử dụng. Hadoop yêu cầu người dùng phải xây dựng và quản lý các công đoạn MapReduce, trong khi Apache Spark sử dụng mô hình xử lý dữ liệu linh hoạt hơn, giúp giảm độ phức tạp và tăng hiệu suất.

Apache Spark không chỉ dựa trên ngôn ngữ lập trình đơn lẻ mà còn có thể linh hoạt sử dụng nhiều ngôn ngữ như Scala, Python, Java, và R. Điều này giúp người dùng lựa chọn ngôn ngữ phù hợp với kỹ năng và yêu cầu của dự án mà không gặp nhiều rắc rối.

* Tốc độ nhanh:

Khả năng và tốc độ phân tích thời gian thực của Apache Spark là một trong những đặc điểm ấn tượng, làm cho nó trở thành một lựa chọn hàng đầu cho xử lý dữ liệu thời gian thực và các ứng dụng yêu cầu sự tin cậy và hiệu suất cao.

Apache Spark không chỉ mạnh mẽ trong việc xử lý dữ liệu thời gian thực mà còn khả năng xử lý hàng loạt các sự kiện thời gian thực từ nhiều nguồn khác nhau. Tính linh hoạt của Spark cho phép nó đồng thời xử lý và phân tích dữ liệu, mở ra khả năng kiểm soát thông tin ngay từ khi nó được tạo ra.

Tốc độ xử lý của Apache Spark là ấn tượng, đạt tới hàng triệu sự kiện mỗi giây. Sự hiệu quả này không chỉ đảm bảo rằng dữ liệu được xử lý một cách nhanh chóng mà còn giúp giữ được tính khả dụng của hệ thống trong môi trường yêu cầu phản ứng ngay lập tức.

Ứng dụng Apache Spark trong việc phát hiện gian lận trong các giao dịch ngân hàng là một ví dụ tiêu biểu về khả năng của nó. Khả năng phân tích thời gian thực của Spark kết hợp với tính năng định dạng và nhận biết các mô hình gian lận giúp tạo ra một hệ thống an ninh mạnh mẽ, có khả năng phát hiện và ngăn chặn các hành vi lừa đảo ngay từ khi chúng xuất hiện.

Ưu và nhược điểm của Apache Spark

* Được hỗ trợ bởi các thư viện cấp cao:

Một trong những điểm mạnh đặc biệt của Apache Spark là khả năng tích hợp và sử dụng các thư viện cấp cao như truyền dữ liệu trực tuyến (Spark Streaming), truy vấn SQL (Spark SQL), học máy (MLlib) và xử lý đồ thị (GraphX). Điều này không chỉ giúp nhà phát triển tiết kiệm thời gian và công sức mà còn đảm bảo tính mở rộng và tính tương thích giữa các phần mềm và thư viện.

Thư viện Spark Streaming cho phép xử lý dữ liệu thời gian thực với sự linh hoạt và hiệu suất cao, giúp Spark mở rộng khả năng ứng dụng của mình từ xử lý dữ liệu tĩnh sang xử lý dữ liệu động. Spark SQL cung cấp khả năng truy vấn dữ liệu bằng SQL trực tiếp trên các RDD và DataFrame, tạo điều kiện thuận lợi cho việc tích hợp dữ liệu có cấu trúc vào quy trình xử lý.

MLlib là thư viện máy học tích hợp, mang lại sức mạnh cho Spark trong việc xử lý lượng lớn dữ liệu cho các tác vụ học máy và khám phá dữ liệu. Đồng thời, GraphX giúp thực hiện xử lý đồ thị hiệu quả, mở ra nhiều ứng dụng trong lĩnh vực mạng xã hội, phân tích mạng, và các tác vụ liên quan đến cấu trúc đồ thị.

* Tương thích cao – Hỗ trợ nhiều ngôn ngữ lập trình:

Khả năng tương thích cao và hỗ trợ nhiều ngôn ngữ lập trình là một trong những đặc điểm nổi bật của Apache Spark, làm cho nó trở thành một framework linh hoạt và dễ tích hợp trong nhiều hệ thống và môi trường khác nhau.

Apache Spark được thiết kế để tương thích mạnh mẽ với nhiều định dạng tệp và nguồn dữ liệu khác nhau, tất cả được hỗ trợ bởi cụm Hadoop. Điều này bao gồm việc đọc và ghi dữ liệu vào các hệ thống lưu trữ phổ biến như HDFS (Hadoop Distributed File System), S3, Cassandra, và nhiều nguồn dữ liệu khác. Nhờ đó mở ra khả năng tích hợp Spark vào các hệ thống đã có sẵn mà không gặp phải những thách thức lớn về tương thích dữ liệu.

Một điểm mạnh khác của Apache Spark là khả năng hỗ trợ nhiều ngôn ngữ lập trình. Người dùng có thể lựa chọn sử dụng Scala, Java, Python, hoặc R để phát triển ứng dụng Spark, tùy thuộc vào kỹ năng và sở thích cá nhân. Điều này mang lại sự linh hoạt và thuận tiện cho các nhóm phát triển có thành viên sử dụng các ngôn ngữ lập trình khác nhau.

Một ưu điểm nổi bật là ngôn ngữ Python, được sử dụng rộng rãi trong cộng đồng phân tích dữ liệu. Sự kết hợp giữa hiệu suất cao của Spark và thuận lợi của ngôn ngữ Python giúp người dùng tận dụng lợi ích của cả hai mà không gặp những rắc rối về tương thích.

### Nhược điểm

Một trong những nhược điểm quan trọng của Apache Spark là tài nguyên tiêu tốn lớn. Do việc duy trì các dữ liệu trong bộ nhớ (in-memory) để tối ưu hóa hiệu suất, Spark yêu cầu một lượng tài nguyên bộ nhớ đáng kể, đặc biệt là khi xử lý các tập dữ liệu lớn. Điều này có thể gây khó khăn trong việc quản lý và triển khai trên các hệ thống có tài nguyên hạn chế.

Spark Streaming, mặc dù có khả năng xử lý dữ liệu thời gian thực, nhưng vẫn tồn tại độ trễ. Độ trễ này phụ thuộc vào độ dài của các mini-batches, gây khó khăn cho các ứng dụng yêu cầu phản ứng ngay lập tức và có độ chính xác cao trong xử lý dữ liệu.

Một vấn đề khác của Apache Spark là sự phức tạp trong quản lý và xử lý dữ liệu không cấu trúc. Trong khi Spark tập trung chủ yếu vào dữ liệu có cấu trúc, việc xử lý dữ liệu không cấu trúc như văn bản hoặc JSON có thể đòi hỏi sự tốn kém và đôi khi không hiệu quả.

Cuối cùng, quản lý và giám sát một hệ thống Apache Spark phân tán cũng đòi hỏi kiến thức chuyên sâu và kỹ năng quản trị cao. Việc tối ưu hóa cấu hình, theo dõi hiệu suất, và xử lý lỗi có thể là nhiệm vụ phức tạp, đặc biệt là trong các môi trường lớn.

# Tham khảo

Nguồn tài liệu, học liệu tham khảo:

[1]. Apache Spark. "Spark Documentation Overview and All Components and Architectures." Accessed October 31, 2024. <https://spark.apache.org/docs/3.5.2/>.

[2]. Apache Hadoop. "Hadoop Documentation Overview." Accessed October 31, 2024. <https://hadoop.apache.org/docs/r3.4.0/>.

[3] .Wikipedia. "Big Data and Its Applications in Life." Accessed October 31, 2024. <https://vi.wikipedia.org/wiki/D%E1%BB%AF_li%E1%BB%87u_l%E1%BB%9Bn>.

[4]. Wikipedia. "About Apache Spark and All Definitions." Accessed October 31, 2024. <https://en.wikipedia.org/wiki/Apache_Spark>.

[5]. Wikipedia. "Apache Hadoop." Accessed October 31, 2024. <https://en.wikipedia.org/wiki/Apache_Hadoop>.

[6]. Phuc Ngoc Nghia. "Introduction to Apache Spark." Accessed October 31, 2024. <https://viblo.asia/p/tim-hieu-ve-apache-spark-ByEZkQQW5Q0>.

[7]. Amazon Web Services. "Comparison of Spark and Hadoop." Accessed October 31, 2024. <https://aws.amazon.com/vi/compare/the-difference-between-hadoop-vs-spark>.

[8]. ISB Insight. "Big Data Processing Tools." Accessed October 31, 2024. <https://insight.isb.edu.vn/top-8-cong-cu-big-data-ban-nen-biet/>.

[9]. De Manejar. "Resilient Distributed Datasets." August 17, 2021. Accessed October 31, 2024. <https://demanejar.github.io/posts/spark-rdd/>.

[10]. Nguyet Linh. "What is Apache Spark?" Accessed October 31, 2024. <https://vinahost.vn/apache-spark-la-gi/>.