**Spark SQL Introduction**

**Mục lục**

[**Spark - Introduction**](#_9y1n1nspk9aq) **1**

[**Spark – RDD**](#_tb1qhb7qtye5) **3**

[**Spark SQL - Introduction**](#_t1s9i8b1ijfk) **6**

[**Spark SQL - DataFrames**](#_vg4qqxs23biq) **7**

[**Spark - Installation**](#_fyxtu6hqnxkj) **11**

[Bước 1: Xác minh cài đặt Java](#_dx5kzhitj1e1) 11

[Bước 2: Cài đặt Scala](#_mhee3ybae41t) 11

[Bước 3: Cài đặt Spark](#_5ie01tmxvkwx) 12

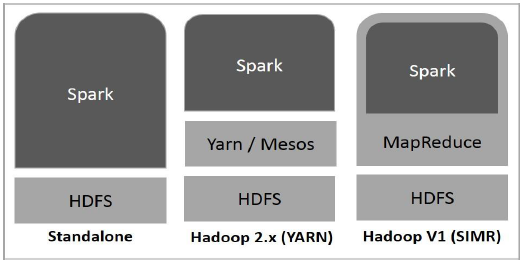
[Bước 5: Kiểm thử Spark](#_yyqjcbfrarmk) 12

# Spark - Introduction

Apache Spark được thiết kế để xử lý dữ liệu lớn một cách nhanh chóng trên các cụm máy tính phân tán. Dựa trên Hadoop MapReduce và nó mở rộng mô hình MapReduce để sử dụng hiệu quả nó cho nhiều loại tính toán hơn, bao gồm các truy vấn tương tác và xử lý luồng.

**Các tính năng của Apache Spark:**

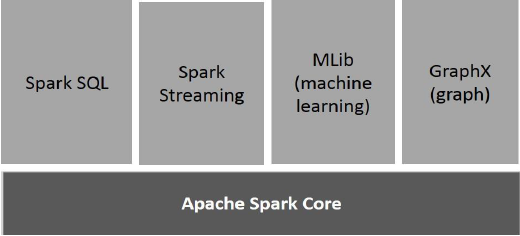
* **Tốc độ** - Spark giúp chạy ứng dụng trong cụm Hadoop, nhanh hơn tới 100 lần trong bộ nhớ RAM và nhanh hơn 10 lần khi chạy trên đĩa. Điều này có thể thực hiện được bằng cách giảm số lượng các thao tác đọc / ghi vào đĩa. Nó lưu trữ dữ liệu xử lý trung gian trong bộ nhớ RAM.
* **Hỗ trợ nhiều ngôn ngữ** - Spark cung cấp các API tích hợp sẵn bằng Java, Scala hoặc Python. Do đó, bạn có thể viết ứng dụng bằng các ngôn ngữ khác nhau. Spark đưa ra 80 toán tử cấp cao để truy vấn tương tác.
* **Phân tích nâng cao** - Spark không chỉ hỗ trợ ‘Map’ và ‘Reduce’. Nó cũng hỗ trợ các truy vấn SQL, Streaming data, Máy học (ML) và các thuật toán Đồ thị.



**Có ba cách triển khai Spark:**

* **Độc lập** – Spark được triển khai độc lập trên HDFS và không gian được phân bổ cho HDFS rõ ràng. Spark và MapReduce sẽ chạy song song với nhau để bao gồm tất cả các công việc spark trên cụm.
* **Hadoop Yarn** - Spark chạy trên Yarn mà không cần cài đặt trước hoặc truy cập root. Nó giúp tích hợp Spark vào hệ sinh thái Hadoop. Và cho phép các thành phần khác chạy trên đầu ngăn xếp.
* **Spark trong MapReduce (SIMR)** - Spark trong MapReduce được sử dụng để khởi chạy công việc spark ngoài việc triển khai độc lập. Với SIMR, người dùng có thể khởi động Spark và sử dụng shell mà không cần bất kỳ quyền truy cập quản trị nào.

**Thành phần của Spark**



* **Apache Spark Core** là nền tảng cho các thành phần còn lại và các thành phần này muốn khởi chạy được thì đều phải thông qua Spark Core do Spark Core đảm nhận vai trò thực hiện công việc tính toán và xử lý trong bộ nhớ RAM đồng thời nó cũng tham chiếu các dữ liệu được lưu trữ tại các hệ thống lưu trữ bên ngoài.
* **Spark SQL** là một thành phần nằm trên Spark Core cung cấp một kiểu data abstraction mới (SchemaRDD) nhằm hỗ trợ cho cả kiểu dữ liệu có cấu trúc (structured data) và dữ liệu nửa cấu trúc (semi-structured data – thường là dữ liệu có cấu trúc nhưng không đồng nhất và cấu trúc của dữ liệu phụ thuộc vào chính nội dung của dữ liệu ấy). Spark SQL hỗ trợ DSL (Domain-specific language) để thực hiện các thao tác trên DataFrames bằng ngôn ngữ Scala, Java hoặc Python và nó cũng hỗ trợ cả ngôn ngữ SQL với giao diện command-line và ODBC/JDBC server.
* **Spark Streaming** tận dụng khả năng lập lịch nhanh chóng của Spark Core để thực hiện phân tích luồng. Nó nhập dữ liệu trong các lô nhỏ và thực hiện các phép biến đổi RDD (Tập dữ liệu phân tán có khả năng phục hồi) trên các lô dữ liệu nhỏ đó.
* **MLlib** là một khung công tác học máy phân tán trên Spark vì kiến ​​trúc Spark dựa trên bộ nhớ phân tán.
* **GraphX** ​​là một khung xử lý đồ thị phân tán trên Spark. Nó cung cấp một API để thể hiện tính toán đồ thị có thể lập mô hình đồ thị do người dùng xác định bằng cách sử dụng API trừu tượng Pregel. Nó cũng cung cấp thời gian chạy được tối ưu hóa cho phần trừu tượng này.

# 2. Spark – RDD

**Tập dữ liệu phân tán có khả năng phục hồi (RDD)**

là một cấu trúc dữ liệu cơ bản của Spark. Nó là một tập hợp phân tán bất biến của các đối tượng. Mỗi tập dữ liệu trong RDD được chia thành các phân vùng logic, có thể được tính toán trên các nút khác nhau của cụm. RDD có thể chứa bất kỳ loại đối tượng Python, Java hoặc Scala nào, bao gồm cả các lớp do người dùng định nghĩa.

Về mặt hình thức, RDD là một tập hợp các bản ghi được phân vùng, chỉ đọc. RDD có thể được tạo thông qua các hoạt động xác định trên dữ liệu trên bộ lưu trữ ổn định hoặc các RDD khác. RDD là một tập hợp các phần tử chịu được lỗi có thể hoạt động song song.

Có hai cách để tạo RDD:

* Song song hóa bộ sưu tập hiện có trong chương trình trình điều khiển.
* Tham chiếu tập dữ liệu trong hệ thống lưu trữ bên ngoài, chẳng hạn như hệ thống tệp chia sẻ, HDFS, HBase hoặc bất kỳ nguồn dữ liệu nào cung cấp định dạng đầu vào Hadoop.

**Chia sẻ dữ liệu chậm trong MapReduce**

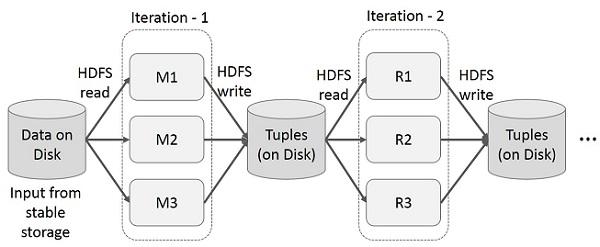
MapReduce được sử dụng rộng rãi để xử lý và tạo các bộ dữ liệu lớn với một thuật toán phân tán, song song trên một cụm. Nó cho phép người dùng viết các phép tính song song, sử dụng một tập hợp các toán tử cấp cao, mà không phải lo lắng về phân phối công việc và khả năng chịu lỗi.

Cách duy nhất để sử dụng lại dữ liệu giữa các lần tính toán ví dụ như giữa hai công việc MapReduce là ghi dữ liệu đó vào hệ thống lưu trữ ổn định bên ngoài, chẳng hạn như HDFS.

Cả hai ứng dụng “Lặp lại” và “Tương tác” đều yêu cầu chia sẻ dữ liệu nhanh hơn trên các công việc song song. Chia sẻ dữ liệu chậm trong MapReduce do sao chép, tuần tự hóa và ghi đọc đĩa. Về hệ thống lưu trữ, hầu hết các ứng dụng Hadoop, chúng dành hơn 90% thời gian để thực hiện các thao tác đọc-ghi HDFS.

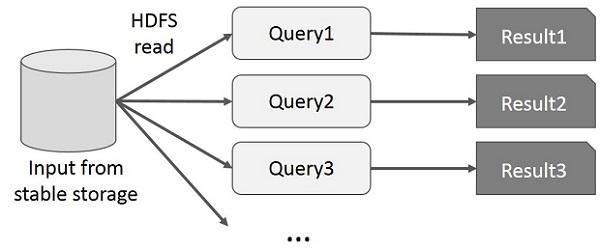
**Các hoạt động lặp lại trên MapReduce**

Hình minh họa sau đây giải thích cách hoạt động của khung hiện tại trong khi thực hiện các hoạt động lặp lại trên MapReduce. Điều này phát sinh chi phí đáng kể do sao chép dữ liệu, I / O đĩa và tuần tự hóa, khiến hệ thống chậm.



**Hoạt động tương tác trên MapReduce**

Người dùng chạy các truy vấn đặc biệt trên cùng một tập con dữ liệu. Mỗi truy vấn sẽ thực hiện I / O đĩa trên bộ nhớ ổn định, điều này có thể chi phối thời gian thực thi ứng dụng.



**Chia sẻ dữ liệu bằng Spark RDD**

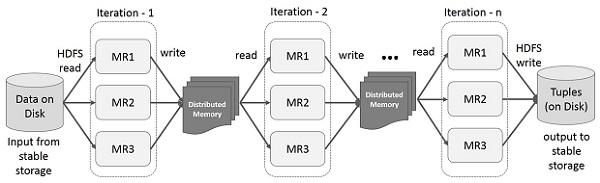
Chia sẻ dữ liệu chậm trong MapReduce do sao chép, tuần tự hóa và IO đĩa. Hầu hết các ứng dụng Hadoop, chúng dành hơn 90% thời gian để thực hiện các thao tác đọc-ghi HDFS.

Chia sẻ dữ liệu trong bộ nhớ RAM nhanh hơn mạng và Đĩa từ 10 đến 100 lần.

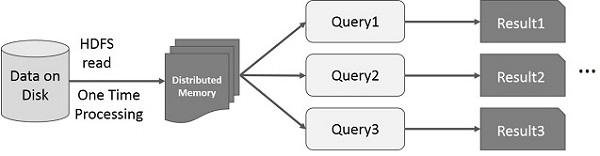
**Hoạt động lặp lại trên Spark RDD**

Spark RDD sẽ lưu trữ các kết quả trung gian trong một bộ nhớ phân tán thay vì Ổ lưu trữ ổn định (Đĩa) và làm cho hệ thống nhanh hơn.

Nếu bộ nhớ phân tán (RAM) không đủ để lưu trữ các kết quả trung gian (Trạng thái công việc), thì nó sẽ lưu các kết quả đó trên đĩa



**Hoạt động tương tác trên Spark RDD**

Nếu các truy vấn khác nhau được chạy lặp lại trên cùng một tập dữ liệu, thì dữ liệu cụ thể này có thể được lưu trong bộ nhớ để có thời gian thực thi tốt hơn. 

Theo mặc định, mỗi RDD đã chuyển đổi có thể được tính toán lại mỗi khi bạn chạy một hành động trên nó. Tuy nhiên, cũng có thể duy trì một RDD trong bộ nhớ, trong trường hợp đó Spark sẽ giữ các phần tử xung quanh trên cụm để truy cập nhanh hơn nhiều vào lần tiếp theo bạn truy vấn nó. Ngoài ra còn có hỗ trợ cho các RDD lâu dài trên đĩa hoặc được sao chép qua nhiều nút.

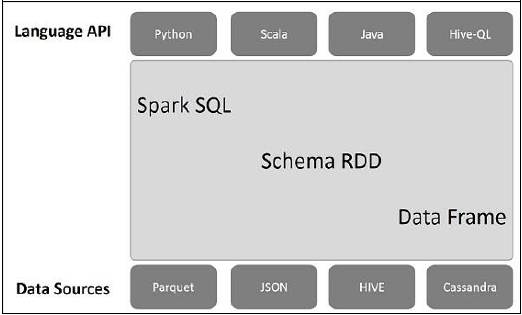
# Spark SQL - Introduction

Spark giới thiệu một mô-đun lập trình để xử lý dữ liệu có cấu trúc được gọi là Spark SQL. Nó cung cấp một chương trình trừu tượng gọi là DataFrame và có thể hoạt động như một công cụ truy vấn SQL phân tán.

**Chức năng của Spark SQL**

* **Tích hợp** - Kết hợp liền mạch các truy vấn SQL với các chương trình Spark. Spark SQL cho phép truy vấn dữ liệu có cấu trúc dưới dạng tập dữ liệu phân tán (RDD) trong Spark, với các API tích hợp bằng python, Scala và Java.
* **Truy cập dữ liệu hợp nhất** - Tải và truy vấn dữ liệu từ nhiều nguồn khác nhau. Schema-RDDs cung cấp một giao diện duy nhất để làm việc hiệu quả với dữ liệu có cấu trúc, bao gồm bảng Apache Hive, tệp parquet và tệp JSON.
* **Khả năng tương thích Hive** - Spark SQL sử dụng lại giao diện người dùng Hive và MetaStore, cung cấp khả năng tương thích hoàn toàn với dữ liệu Hive, truy vấn và UDF hiện có. Chỉ cần cài đặt nó cùng với Hive.
* **Kết nối tiêu chuẩn** - Spark SQL bao gồm một chế độ máy chủ với kết nối JDBC và ODBC tiêu chuẩn công nghiệp.
* **Khả năng mở rộng** - Sử dụng cùng một công cụ cho cả truy vấn tương tác và truy vấn dài. Spark SQL tận dụng lợi thế của mô hình RDD để hỗ trợ khả năng chịu lỗi ở giữa truy vấn, cho phép nó mở rộng thành các công việc lớn.

**Kiến trúc Spark SQL**



Kiến trúc này chứa ba lớp cụ thể là, Language API, Schema RDD và data sources.

# Spark SQL - DataFrames

DataFrame là một tập hợp dữ liệu phân tán, được tổ chức thành các cột. Về mặt khái niệm, nó tương đương với các bảng quan hệ.

Một DataFrame có thể được xây dựng từ một loạt các nguồn khác nhau như bảng Hive, tệp dữ liệu có cấu trúc, cơ sở dữ liệu bên ngoài hoặc RDD hiện có.

**Chức năng của DataFrame**

* Khả năng xử lý dữ liệu có kích thước từ Kilobyte đến Petabyte trên một cụm nút đơn đến cụm lớn.
* Hỗ trợ các định dạng dữ liệu khác nhau (Avro, csv, Cassandra) và hệ thống lưu trữ (HDFS, bảng HIVE, mysql, v.v.).
* Tối ưu hóa hiện đại và tạo mã thông qua trình tối ưu hóa Spark SQL Catalyst
* Có thể dễ dàng tích hợp với tất cả các công cụ và khuôn khổ Dữ liệu lớn thông qua Spark-Core.
* Cung cấp API cho Lập trình Python, Java, Scala và R.

**SQLContext**

SQLContext là một class được sử dụng để khởi tạo các chức năng của Spark SQL. Đối tượng lớp SparkContext (sc) là bắt buộc để khởi tạo đối tượng lớp SQLContext.

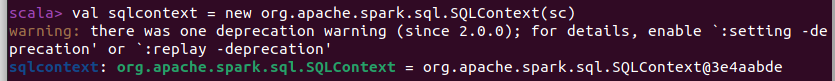
Lệnh sau được sử dụng để khởi tạo SparkContext thông qua spark-shell.

Text

Description automatically generated

Theo mặc định, đối tượng SparkContext được khởi tạo với tên sc khi bắt đầu spark-shell.

Sử dụng lệnh sau để tạo SQLContext.



**Ví dụ:**

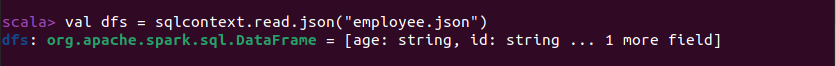
Ví dụ về bản ghi nhân viên trong tệp JSON có tên là employee.json. Sử dụng các lệnh sau để tạo DataFrame (df) và đọc tài liệu JSON có tên là employee.json với nội dung sau:

A screenshot of a computer

Description automatically generated with medium confidence

* **Đọc tài liệu JSON**

Tạo một DataFrame có tên (dfs). Sử dụng lệnh sau để đọc tài liệu JSON có tên là employee.json. Dữ liệu được hiển thị dưới dạng bảng với các trường - id, age.



* **Hiển thị dữ liệu trong DataFrame**

Shape, rectangle

Description automatically generated

* **Sử dụng phương thức printSchema**

Để xem Cấu trúc (Lược đồ) của DataFrame, sử dụng lệnh sau.

Shape

Description automatically generated with low confidence

* **Sử dụng phương thức Select**

Shape

Description automatically generated

* **Sử dụng bộ lọc age**

để tìm nhân viên có tuổi lớn hơn 23 (tuổi> 23).

Text

Description automatically generated

* **Sử dụng phương pháp groupBy**

để đếm số lượng nhân viên ở cùng độ tuổi.

Text

Description automatically generated

# Spark - Installation

## Bước 1: Xác minh cài đặt Java

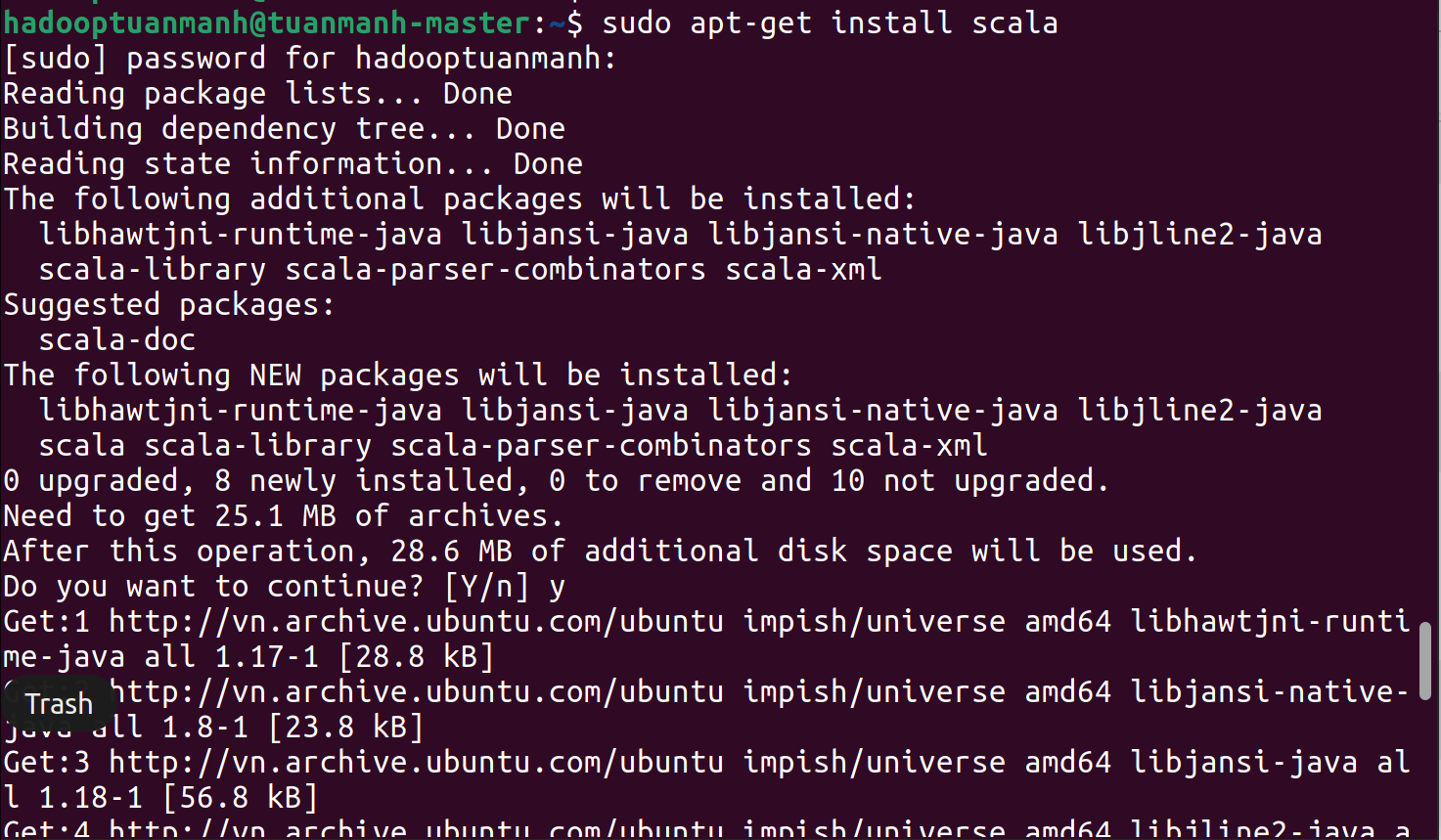
* Sử dụng lệnh sau để xem phiên bản java

Text

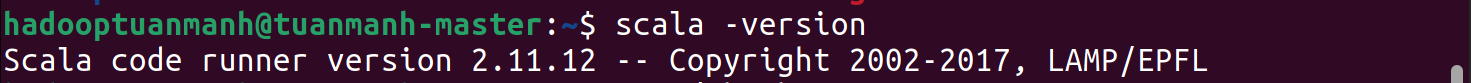
Description automatically generated

## Bước 2: Cài đặt Scala

* Sử dụng lệnh sau để cài đặt Scala



* Kiểm tra phiên bản Scala



* Sử dụng lệnh println để kiểm tra xem Scala đã hoạt động hay chưa.



## Bước 3: Cài đặt Spark

* Sử dụng lệnh wget để cài đặt Spark.

Text

Description automatically generated

* Giải nén Spark

Text

Description automatically generated

* Lệnh sau để di chuyển các tệp phần mềm Spark vào thư mục tương ứng (/ usr / local / spark)



* Đặt biến môi trường cho Spark



## Bước 5: Kiểm thử Spark

* Mở Spark shell, nếu đã xuất hiện như bên dưới là cài Spark thành công.

Text

Description automatically generated

* Dùng lệnh println kiểm tra xem đã thành công chưa.

