**TRƯỜNG ĐẠI HỌC THỦY LỢI**

KHOA CÔNG NGHỆ THÔNG TIN

A blue and white logo

Description automatically generated with low confidence

**PH ÂN TÍCH DỮ LIỆU LỚN**

**Đề tài:**

**Xử lý dữ liệu với Apache Spark**

Giảng viên hướng dẫn: Ths.Nguyễn Đắc Hiếu

Lớp : 61TH6

Nhóm thực hiện : 5

Sinh viên thưc hiện : Nguyễn Thành Trung

Dương Văn Thành

**MỤC LỤC**

[CHƯƠNG 1: TỔNG QUAN VỀ BIG DATA 3](#_Toc134462378)

[1. Khái niệm: 3](#_Toc134462379)

[2. Đặc trưng: 3](#_Toc134462380)

[3. Các ứng dụng phổ biến của Big Data: 4](#_Toc134462381)

[4. Các công nghệ dành cho Big Data: 7](#_Toc134462390)

[CHƯƠNG 2: SPARK VÀ HADOOP MAPREDUCE 9](#_Toc134462391)

[2.1 Hadoop MapReduce: 9](#_Toc134462392)

[2.1.1 Định nghĩa: 9](#_Toc134462393)

[2.1.2 Các hàm chính của MapReduce: 9](#_Toc134462394)

[2.1.3 Cơ chế hoạt động của MapReduce: 10](#_Toc134462395)

[2.1.4 Ưu và nhược điểm của MapReduce: 12](#_Toc134462396)

[2.2 Apache Spark: 13](#_Toc134462397)

[2.2.1 Tổng quan: 13](#_Toc134462398)

[2.2.2 Tính năng của Apache Spark: 15](#_Toc134462399)

[2.2.3 Các thành phần của Apache Spark: 15](#_Toc134462400)

[2.2.4 Kiến trúc của Apache Spark: 16](#_Toc134462401)

[2.2.5 Resilient Distributed Datasets: 18](#_Toc134462402)

[2.2.6 Lazy Evaluation, Transformations and Actions: 19](#_Toc134462403)

[2.2.7 Directed Acyclic Graph (DAG): 21](#_Toc134462404)

[2.2.8 SparkSQL, Dataframe và Dataset: 23](#_Toc134462405)

[CHƯƠNG 3: XỬ LÝ DỮ LIỆU VỚI APACHE SPARK 26](#_Toc134462406)

[3.1 Tổng quan về dữ liệu: 26](#_Toc134462407)

[3.2 Tiền xử lý dữ liệu: 27](#_Toc134462408)

[3.3 Thăm dò và khai thác thông tin: 29](#_Toc134462409)

[CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 34](#_Toc134462410)

[4.1 Kết luận: 34](#_Toc134462411)

LỜI MỞ ĐẦU

Sự phát triển của xã hội dẫn đến bùng nổ dữ liệu trong những thập niên gần đây. Những sản phẩm công nghệ mới đem lại nhiều tiện ích trong cuộc sống, được ứng dụng ở nhiều lĩnh vực thông tin truyền thông. Hệ thống thông tin điện tử, trực tuyến, các website của những doanh nghiệp tổ chức được phát triển mạnh mẽ góp phần tăng cường mối quan hệ, hợp tác ở nhiều lĩnh vực như văn hóa xã hội, khoa học công nghệ, y tế, giáo dục, giải trí…

Tuy nhiên những phương thức lưu trữ dữ liệu đã bộc lộ rất nhiều hạn chế. Ngày nay khối lượng dữ liệu vô cùng lớn, kích cỡ lên đến hàng trăm terabyte cho đến petabyte chỉ cho một tập hợp dữ liệu.. Những phương pháp đó không cho phép liên kết và phân tích nhiều dạng dữ liệu khác nhau. Khối lượng dữ liệu gia tăng nhanh nhưng tốc độ xử lý dữ liệu (thu nhận, xử lý, đáp trả) mất rất nhiều thời gian trong khi nhu cầu của con người muốn xử lý được ngay dữ liệu tức thời (tính đến bằng mili giây)

Và Big data ra đời đã đánh dấu một trang mới trong lịch sử phát triển công nghệ.. Sự phức tạp và không thể định hình thành một thể thống nhất của dữ liệu. Big data cũng là một nhân tố làm nó trở nên khó đồng bộ để lưu trữ trong một hệ thống cơ sở dữ liệu truyền thống. Dữ liệu được thu thập từ nhiều nguồn khác nhau bao gồm: dữ liệu không giới hạn từ internet, web 2.0, từ các thiết bị nghiên cứu, dữ liệu từ các thiết bị thông minh (hay còn gọi là smart device). Do đó nó mang cấu trúc không cố định. Big data đã thể hiện được sức mạnh và tầm ảnh hưởng đến mọi lĩnh vực trong xã hội.

Trong đề tài này em sẽ trình bày về công nghệ Apache Spark và các ứng dụng mà Spark sử dụng để phân tích dữ liệu lớn.

# CHƯƠNG 1: TỔNG QUAN VỀ BIG DATA

## 1. Khái niệm:

- Theo Wikipedia*:* Dữ liệu lớn ([Tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): Big data) là một thuật ngữ cho việc xử lý một tập hợp dữ liệu rất lớn và phức tạp mà các ứng dụng xử lý dữ liệu truyền thống không xử lý được, bao gồm cả dữ liệu có cấu trúc và không có cấu trúc.

- Theo Gartner: Dữ liệu lớn là những nguồn thông tin có đặc điểm chung khối lượng lớn, tốc độ nhanh và dữ liệu định dạng dưới nhiều hình thức khác nhau,do đó muốn khai thác được phải đòi hỏi phải có hình thức mới để đưa ra quyết định khám phá và tối ưu hóa quy trình.

- Dữ liệu đến từ rất nhiều nguồn khác nhau:

Text

Description automatically generated

*Hình 1. Minh họa nguồn gốc của dữ liệu.*

- Một số lợi ích có thể mang lại như: Cắt giảm chi phí, tiết kiệm thời gian và giúp tối ưu hóa sản phẩm, hỗ trợ con người đưa ra những quyết định đúng và hợp lý hơn.

## 2. Đặc trưng:

- Big Data được mô tả bởi những đặc trưng sau:

* Volume (Dung lượng): Số lượng dữ liệu được tạo ra và lưu trữ. Kích thước của dữ liệu xác định giá trị và tiềm năng insight- và liệu nó có thể thực sự được coi là dữ liệu lớn hay không.
* Variety (Tính đa dạng): Các dạng và kiểu của dữ liệu. Dữ liệu được thu thập từ nhiều nguồn khác nhau và các kiểu dữ liệu cũng có rất nhiều cấu trúc khác nhau.
* Velocity (Vận tốc): Trong trường hợp này nghĩa là tốc độ các dữ liệu được tạo ra và xử lý để đáp ứng các nhu cầu và thách thức trên con đường tăng trưởng và phát triển.
* Veracity (Tính xác thực): Chất lượng của dữ liệu thu được có thể khác nhau rất nhiều, ảnh hưởng đến sự phân tích chính xác.
* Value (Giá trị): Giá trị thông tin mang lại.

Diagram

Description automatically generated

*Hình 2: Minh họa đặc trưng của Big Data*

## 3. Các ứng dụng phổ biến của Big Data:

### 3.1. Ngành tài chính - ngân hàng:

- Trong ngành tài chính ngân hàng, Big Data được sử dụng để phân tích và xác định các địa điểm tập trung nhiều nhu cầu của khách hàng tiềm năng để đề xuất lập chi nhánh mới, dự đoán lượng tiền mặt cần thiết sẵn sàng cung ứng ở một chi nhánh tại thời điểm cụ thể, nâng cao hệ thống ngân hàng kĩ thuật số, phát hiện các hoạt động gian lận và báo cáo cho các chuyên viên liên quan giúp đảm bảo an ninh cho ngân hàng.

### 3.2. Y tế, chăm sóc sức khỏe:

- Trong ngành y tế, Big Data không chỉ được ứng dụng để xác định phương hướng điều trị mà còn giúp cải thiện quá trình chăm sóc sức khỏe. Big Data giúp dự đoán thời điểm cần sự có mặt của bác sĩ, giúp theo dõi tình trạng bệnh nhân bằng để theo dõi hồ sơ sức khỏe điện tử, theo dõi bệnh nhân và gửi báo cáo cho các bác sĩ liên quan. Ngoài ra, Big Data còn có thể đánh giá các triệu chứng và xác định nhiều bệnh ở giai đoạn đầu, lưu giữ các hồ sơ nhạy cảm được bảo mật và lưu trữ lượng dữ liệu khổng lồ một cách hiệu quả. Đặc biệt, nó cũng có thể dự báo trước các khu vực có nguy cơ bùng phát dịch bệnh nguy hiểm.

### 3.3. Thương mại điện tử:

- Đối với ngành thương mại điện tử, doanh nghiệp nào sở hữu và ứng dụng tốt Big Data sẽ tạo ra lợi thế cạnh tranh lớn trên thị trường. Big Data giúp nhà quản lý xác định được sản phẩm nào được xem nhiều nhất để tối ưu thời gian hiển thị, tự gửi mã ưu đãi cho những sản phẩm khách hàng bỏ vào giỏ hàng nhưng không mua. Đặc biệt, Big Data có thể phân tích hành vi, sở thích, sự quan tâm của khách hàng, giúp nhà quản lý hiểu hơn về khách hàng để cung cấp các sản phẩm theo đúng xu hướng, nhu cầu thị trường.

### 3.4. Ngành bán lẻ:

- Ngành bán lẻ ứng dụng Big Data trong việc xây dựng mô hình chi tiêu của khách hàng, trợ giúp dự đoán cung - cầu cho các sản phẩm. Kết hợp với các dữ liệu về thời điểm giao dịch, dữ liệu truyền thông xã hội, dự báo thời tiết để xác định chính xác nhất sản phẩm phù hợp để luôn sẵn sàng cung ứng cho khách hàng. Nhà quản lý có thể dựa vào dữ liệu về thói quen mua hàng, sở thích của khách hàng để xác định vị trí, cách bố trí sản phẩm trên kệ hàng và đưa ra các chiến lược kinh doanh mới để cải thiện.

### 3.5. Digital Marketing:

- Digital Marketing ngày càng trở thành một công cụ quan trọng cho các doanh nghiệp hiện nay. Khi ứng dụng Big Data, doanh nghiệp sử dụng Digital Marketing có thể xác định đối tượng mục tiêu trên các trang mạng xã hội dựa trên thông tin nhân khẩu học, giới tính, tuổi tác, sở thích. Bên cạnh đó, có thể cá nhân hóa các hoạt động tìm kiếm trên Google, Email Marketing, hiển thị quảng cáo phù hợp và tạo báo cáo chi tiết sau mỗi chiến dịch quảng cáo.

### 3.6. Ngành công nghiệp:

- Phát triển phần mềm sản phẩm: Các công ty sử dụng Big Data để xây dựng các mô hình dự đoán cho các sản phẩm và dịch vụ mới. Họ phân loại các thuộc tính chính của sản phẩm/dịch vụ trong quá khứ và hiện tại, sau đó mô hình hóa mối quan hệ giữa các thuộc tính và phát triển thành phần mềm hoàn chỉnh.

- Tăng trải nghiệm khách hàng: Big Data cho phép bạn thu thập dữ liệu từ các phương tiện truyền thông xã hội, lịch sử web, nhật ký cuộc gọi và các nguồn khác. Nhờ đó, doanh nghiệp hiểu rõ khách hàng của mình hơn qua chính hành vi thực tế của khách hàng.

- Máy học (Machine Learning): Máy học là một bước tiến của công nghệ hiện đại và Big Data góp phần lớn trong sự phát triển của máy học. Hiện nay máy móc có thể tự học từ dữ liệu lịch sử thay vì con người phải lập trình, các dữ liệu lịch sử này được thu thập đầy đủ vào Big Data.

- Thúc đẩy sự cải tiến mới: Big data cung cấp dữ liệu về thông tin sản phẩm, lịch sử phát triển của mọi lĩnh vực, ngành nghề. Qua đó, nhà quản lý có thể xác định được những điểm cần cải tiến, phát triển để công nghệ mới đem lại sự sáng tạo, mới mẻ, hữu ích cho các ngành công nghiệp.

### 3.7. Lĩnh vực giáo dục đào tạo:

- Ngành giáo dục sử dụng Big Data để ước tính số lượng học sinh, sinh viên tuyển sinh hằng năm, quản lý hồ sơ học sinh, sinh viên, truy xuất nhanh thông tin cần thiết khi gặp các vấn đề phát sinh. Ngoài ra, Big Data cũng giúp ước tính được nhu cầu tuyển dụng cho các ngành nghề hàng năm để đề ra các phương án đào tạo nhằm đáp ứng nguồn nhân lực cho xã hội.

### 3.8. Cơ sở hạ tầng IT hỗ trợ Big Data:

- Đặc điểm của Big Data là lượng dữ liệu khổng lồ và tăng cao qua nhiều năm, do đó cần có hệ thống lưu trữ và máy chủ được thiết kế đặc biệt cho Big Data. Ngoài ra còn có phần mềm thu thập, quản lý và tích hợp dữ liệu. Doanh nghiệp nếu sử dụng Big Data lâu dài thì nên đầu tư cơ sở hạ tầng IT vững vàng, chất lượng để đảm bảo nguồn dữ liệu không bị mất đi hoặc không cập nhật kịp. Phần lớn các cơ sở hạ tầng này sẽ tập trung một chỗ, vì các công ty muốn tiếp tục tận dụng các khoản đầu tư vào trung tâm dữ liệu của mình. Nhưng ngày càng có nhiều tổ chức dựa vào các dịch vụ điện toán đám mây để xử lý nhiều yêu cầu big data của họ.

## 4. Các công nghệ dành cho Big Data:

- Hệ sinh thái Hadoop: Hadoop là một Apache framework mã nguồn mở được viết bằng Java, cho phép xử lý phân tán các tập dữ liệu lớn trên các cụm máy tính (clusters of computers) thông qua mô hình lập trình đơn giản. Hadoop được thiết kế để mở rộng quy mô từ một máy chủ đơn sang hàng ngàn máy tính khác có tính toán và lưu trữ cục bộ (local computation and storage).

- Apache Spark: Đây là một công cụ tính toán nhanh và có mục đích chung để xử lý quy mô lớn dữ liệu. Nó có thể xử lý dữ liệu nhanh hơn 100 lần so với MapReduce.

- Apache Kafka: Apache Kafka là một hệ thống nhắn tin phân tán lượng thông tin cao thường được sử dụng với Hadoop. Nó bổ sung cho hệ sinh thái Big Data.

- Data lakes: Data lakes là các kho lưu trữ chứa khối lượng dữ liệu thô rất lớn ở định dạng gốc, được thiết kế để giúp người dùng dễ dàng truy cập vào một lượng lớn dữ liệu khi có nhu cầu. Các yếu tố giúp tăng trưởng data lakes là những phong trào kỹ thuật số và sự phát triển của IoT.

- NoSQL Databases: Các cơ sở dữ liệu SQL thông thường được thiết kế cho các transaction đáng tin cậy và các truy vấn ngẫu nhiên. Cơ sở dữ liệu NoSQL nêu ra những hạn chế, và lưu trữ và quản lý dữ liệu theo những cách cho phép tốc độ hoạt động cao và sự linh hoạt tuyệt vời. Không giống như các cơ sở dữ liệu SQL, nhiều cơ sở dữ liệu NoSQL có thể được mở rộng theo chiều ngang trên hàng trăm hoặc hàng ngàn máy chủ.

- In-memory databases: Cơ sở dữ liệu trong bộ nhớ (IMDB) là một hệ thống quản lý cơ sở dữ liệu chủ yếu dựa vào bộ nhớ chính (Ram), thay vì HDD. Cơ sở dữ liệu trong bộ nhớ nhanh hơn các cơ sở dữ liệu được tối ưu hóa trong đĩa, một điểm quan trọng để sử dụng phân tích big data và tạo ra các kho dữ liệu và các siêu dữ liệu.

# CHƯƠNG 2: SPARK VÀ HADOOP MAPREDUCE

## 2.1 Hadoop MapReduce:

### 2.1.1 Định nghĩa:

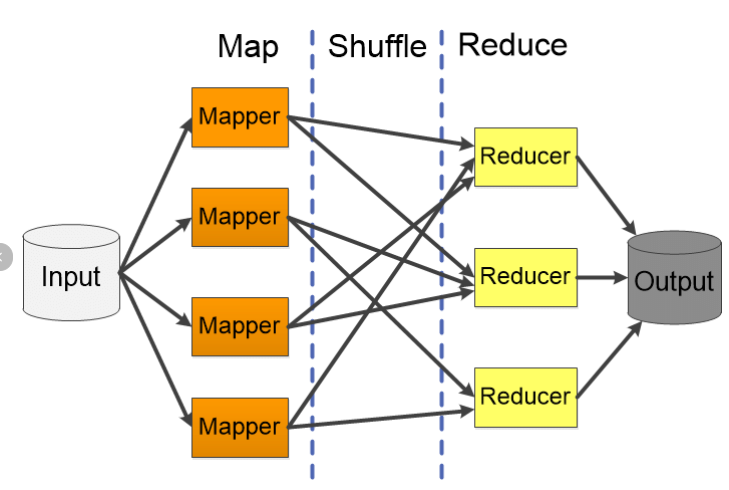
- [MapReduce](https://bizflycloud.vn/tin-tuc/mapreduce-la-gi-20220524152927617.htm) là một mô hình được Google thiết kế độc quyền với khả năng lập trình xử lý một lượng lớn các dữ liệu song song đồng thời phân tán các thuật toán trên cùng một máy tính. Mặc dù MapReduce ban đầu là một công nghệ độc quyền của Google nhưng trong thời gian gần đây, MapReduce đang dần trở thành một trong những thuật ngữ tổng quát hoá.

### 2.1.2 Các hàm chính của MapReduce:

- MapReduce có 2 hàm chính là Map () và Reduce (), đây là 2 hàm đã được định nghĩa bởi người dùng và nó cũng chính là 2 giai đoạn liên tiếp trong quá trình xử lý dữ liệu của MapReduce. Nhiệm vụ cụ thể của từng hàm như sau:

* Hàm Map (): có nhiệm vụ nhận Input cho các cặp giá trị/ khóa và output chính là tập những cặp giá trị/khóa trung gian. Sau đó, chỉ cần ghi xuống đĩa cứng và tiến hành thông báo cho các hàm Reduce () để trực tiếp nhận dữ liệu.
* Hàm Reduce (): có nhiệm vụ tiếp nhận từ khóa trung gian và những giá trị tương ứng với lượng từ khóa đó. Sau đó, tiến hành ghép chúng lại để có thể tạo thành một tập khóa khác nhau. Các cặp khóa/giá trị này thường sẽ thông qua một con trỏ vị trí để đưa vào các hàm reduce. Quá trình này sẽ giúp cho lập trình viên quản lý dễ dàng hơn một lượng danh sách cũng như  phân bổ giá trị sao cho  phù hợp nhất với bộ nhớ hệ thống.

- Ở giữa Map và Reduce thì còn 1 bước trung gian đó chính là Shuffle. Sau khi Map hoàn thành  xong công việc của mình thì Shuffle sẽ làm nhiệm vụ chính là thu thập cũng như tổng hợp từ khóa/giá trị trung gian đã được map sinh ra trước đó rồi chuyển qua cho Reduce tiếp tục xử lý.



*Hình 3: Minh họa mô hình MapReduce*

### 2.1.3 Cơ chế hoạt động của MapReduce:

- Các bước hoạt động của MapReduce

* Bước 1: Tiến hành chuẩn bị các dữ liệu đầu vào để cho Map() có thể xử lý.
* Bước 2: Lập trình viên thực thi các mã Map() để xử  lý.
* Bước 3: Tiến hành trộn lẫn các dữ liệu được xuất ra bởi Map() vào trong Reduce Processor
* Bước 4: Tiến hành thực thi tiếp mã Reduce() để có thể xử lý tiếp các dữ liệu cần thiết.
* Bước 5: Thực hiện tạo các dữ liệu xuất ra cuối cùng.

- Luồng dữ liệu của MapReduce: Input Reader, Map Function, Partition Function, Compare Function, Reduce Function, Output Writer

A diagram of a program

Description automatically generated with low confidence

*Hình 4: Minh họa Data Flow của MapReduce*

* Thông qua thư viện MapReduce ứng dụng với từng ngôn ngữ, chương trình có nhiệm vụ phân mảnh tệp dữ liệu đầu vào. Dữ liệu vào được chia thành các phần nhỏ 16 megabytes đến 64 megabytes (MB). Sau đó khởi động việc sao chép chương trình trên các clusters.
* Các máy gồm có: master và worker. Trong đó máy master làm nhiệm vụ điều phối sự hoạt động của quá trình thực hiện MapReduce trên các máy worker, các máy worker làm nhiệm vụ thực hiện Map và Reduce với dữ liệu mà nó nhận được. Bằng cách đặt trạng thái idle máy workers và sau đó gắn cho từng máy task map hoặc reduce.
* Máy master sẽ thực hiện phân phối các tác vụ Map và Reduce vào các worker đang rảnh rỗi. Các tác vụ này được master phân phối cho các máy tính dựa trên vị trí của dữ liệu liên quan trong hệ thống. Máy worker khi nhận được tác vụ Map sẽ đọc dữ liệu mà nó nhận từ phân vùng dữ liệu đã gán cho nó và thực hiện hàm Map. Kết quả đầu ra là các cặp (keyI,valueI) trung gian. Các cặp này được lưu tạm trên bộ nhớ đệm của các máy.
* Sau khi thực hiện xong công việc Map, các máy worker làm nhiệm vụ chia các giá trị trung gian thành R vùng ( tương ứng với R tác vụ Reduce) lưu xuống đĩa và thông báo kết quả, vị trí lưu cho máy master.
* Master sẽ gán giá trị trung gian và vị trí của các dữ liệu đó cho các máy thực hiện công việc Reduce. Các máy reducer làm nhiệm vụ xử lý sắp xếp các key, thực hiện hàm Reduce và đưa ra kết quả cuối cùng.
* Master sẽ kích hoạt thông báo cho chương trình người dùng quá trình MapReduce đã hoàn tất, kết quả đầu ra được lưu trữ trên R tập tin.

### 2.1.4 Ưu và nhược điểm của MapReduce:

- Ưu điểm:

* Khả năng mở rộng tuyến tính: MapReduce có thể chia nhỏ tập dữ liệu lớn thành các phần nhỏ và phân tán xử lý trên nhiều nút, do đó giải quyết được các bài toán với khối lượng dữ liệu lớn và đáp ứng được nhu cầu mở rộng.
* Xử lý song song: MapReduce cho phép xử lý nhiều phần của tập dữ liệu cùng lúc trên các nút khác nhau, do đó tăng tốc độ xử lý dữ liệu.
* Dễ sử dụng: MapReduce có cấu trúc đơn giản, dễ hiểu và dễ sử dụng. Người dùng chỉ cần thực hiện hai công việc chính là viết hàm Map và hàm Reduce.
* Khả năng phục hồi lỗi: MapReduce có khả năng phục hồi lỗi, nếu một nút xử lý bị lỗi, nhiệm vụ đó sẽ được chuyển cho một nút khác xử lý.
* Hỗ trợ nhiều ngôn ngữ lập trình: MapReduce hỗ trợ nhiều ngôn ngữ lập trình như Java, Python, C++...

- Nhược điểm:

* Chậm trong xử lý các tác vụ nhỏ: MapReduce được thiết kế để xử lý các tập dữ liệu lớn, do đó nếu xử lý các tác vụ nhỏ, MapReduce sẽ trở nên chậm.
* Không phù hợp cho các tác vụ thời gian thực: Do MapReduce chia nhỏ tập dữ liệu lớn và xử lý trên nhiều nút, do đó thời gian xử lý của nó không thể dự đoán được và không phù hợp cho các tác vụ thời gian thực.
* Khó khăn trong xử lý dữ liệu có cấu trúc phức tạp: MapReduce không phù hợp để xử lý các tập dữ liệu có cấu trúc phức tạp, vì nó chỉ hỗ trợ xử lý dữ liệu có cấu trúc đơn giản.
* Chi phí cao: MapReduce yêu cầu nhiều tài nguyên phần cứng và phần mềm, do đó chi phí triển khai và vận hành của nó khá cao.
* Khó khăn trong việc lập trình: Việc viết mã MapReduce đòi hỏi kiến ​​thức sâu về lập trình và khá phức tạp, đặc biệt.

==> Input data được đọc từ HDFS (component phụ trách việc lưu trữ trong Hadoop) → xử lý bằng các thao tác chỉ định → output được ghi vào HDFS → data tiếp tục được load → thao tác tiếp theo được thực hiện → output tiếp tục ghi vào HDFS …  
chuỗi các step [read-process-write] đó được lặp cho đến khi hoàn thành công việc. Vì input được chia thành các block độc lập với nhau, các task map-reduce được thực hiện song song, nên về cơ bản nó hữu ích để xử lí những bộ dữ liệu lớn. Tuy nhiên, MR vẫn còn những tồn tại là quá trình xử lý không thực sự hiệu quả trong trường hợp phải lặp lại nhiều step, vì mỗi step cần thiết phải ghi output vào HDFS trước khi step tiếp theo được thực hiện → việc này tạo ra các vấn đề trong việc lưu trữ và sao chép, tăng độ trễ xử lý do phần lớn thực hiện trên Disk vốn có hiệu suất I/O không cao. Bên cạnh đó là việc develop, debug với MR có phần khó khăn vì code dài dòng.

## 2.2 Apache Spark:

### 2.2.1 Tổng quan:

- Apache Spark bắt đầu tại University of California, Berkeley vào năm 2009 với tên là “Spark research project”, được giới thiệu public lần đầu tiên sau một năm (2010) trong một bài báo có tựa đề [“Spark: Cluster Computing with Working Sets”](https://www.usenix.org/legacy/event/hotcloud10/tech/full_papers/Zaharia.pdf) viết bởi 5 nhà nghiên cứu của AMPlab — UC Berkeley. Vào thời điểm đó, Hadoop MapReduce là công cụ lập trình song song mạnh mẽ và cũng là dự án open-source đầu tiên để xử lý xử lý song song dữ liệu trên các cụm servers với hàng ngàn node. AMPlab đã làm việc với nhiều người dùng MapReduce sớm để hiểu những lợi ích và nhược điểm của mô hình lập trình mới này và do đó có thể tổng hợp danh sách các vấn đề qua một số use cases và bắt đầu thiết kế general-computing platforms. Ngoài ra, nhóm nghiên cứu cũng đã làm việc với Hadoop users tại UC Berkeley để hiểu nhu cầu của họ đối với nền tảng, cụ thể là các nhóm đang học máy quy mô lớn bằng thuật toán lặp cần thực hiện multi passes dữ liệu.

Trong suốt quá trình tìm hiểu, nhóm nghiên cứu nhận ra 2 vấn đề:

1. Cluster Computing có tiềm năng phát triển mạnh mẽ bởi các ứng dụng mới hoàn toàn có thể xây dựng và sử dụng hệ thống, dữ liệu hiện có.
2. Công cụ MapReduce khiến việc xây dựng các ứng dụng lớn trở nên khó khăn và không hiệu quả.

- Từ đó nhóm quyết định phát triển Spark với mục tiêu cốt lõi là giúp việc xây dựng ứng dụng có thể phát triển dễ dàng cùng với khả năng scale-up linh hoạt để xử lý tập dữ liệu lớn và rất lớn, nhóm đã thiết kế một API dựa trên Functional Programming có thể viết ứng dụng một cách ngắn gọn. Sau đó triển khai API này qua một công cụ mới có thể thực hiện chia sẻ dữ liệu trong bộ nhớ hiệu quả, cho phép tái sử dụng qua các bước tính toán. Spark cung cấp tập hợp các Computing Engine và Libraries cho việc xử lý dữ liệu song song trên hệ thống lên tới hàng ngàn Servers, ngoài ra nó còn hỗ trợ nhiều ngôn ngữ lập trình được sử dụng rộng rãi (Python, Java, Scala và R), bao gồm các thư viện cho các tác vụ khác nhau, từ SQL đến Streaming, Machine Learning và graph-parallel computation. Năm 2013, dự án Spark được trao tặng cho Apache Software Foundation, open-source project này sau đó nhận được sự công nhận và đóng góp mạnh mẽ cộng đồng. Cuối cùng, 26/5/2014, Apache release Spark ver1.0 sẵn sàng apply trong môi trường production.

### 2.2.2 Tính năng của Apache Spark:

- Apache Spark có các tính năng đặc trưng sau đây.

* + Tốc độ: Spark có thể chạy trên cụm Hadoop và có thể chạy nhanh hơn 100 lần khi chạy trên bộ nhớ RAM, và nhanh hơn 10 lần khi chạy trên ổ cứng. Bằng việc giảm số thao tác đọc ghi lên đĩa cứng. Nó lưu trữ trực tiếp dữ liệu xử lý lên bộ nhớ.
  + Hỗ trợ đa ngôn ngữ: Spark cung cấp các API có sẵn cho các ngôn ngữ Java, Scala, hoặc Python. Do đó, bạn có thể viết các ứng dụng bằng nhiều các ngôn ngữ khác nhau. Spark đi kèm 80 truy vấn tương tác mức cao.
  + Phân tích nâng cao: Spark không chỉ hỗ trợ ‘Map’ và ‘Reduce’. Nó còn hỗ trợ truy vấn SQL, xử lý theo Stream, học máy, và các thuật toán đồ thị (Graph)

### 2.2.3 Các thành phần của Apache Spark:

A picture containing text, screenshot, font, electric blue

Description automatically generated

*Hình 5: Thành phần của Apache Spark*

- Spark Core: Spark Core là thành phần cốt lõi thực thi cho tác vụ cơ bản làm nền tảng cho các chức năng khác. Nó cung cấp khả năng tính toán trên bộ nhớ và database trong bộ nhớ hệ thống lưu trữ ngoài.

- Spark SQL: là một thành phần nằm trên Spark Core, giới thiệu một khái niệm trừu tượng hóa dữ liệu mới gọi là SchemaRDD, cung cấp hỗ trợ cho dữ liệu có cấu trúc và bán cấu trúc.

- Spark Streaming: tận dụng khả năng lập lịch memory-base của Spark Core để thực hiện streaming analytics. Nó lấy dữ liệu theo mini-batches và thực hiện các phép biến đổi RDD (Bộ dữ liệu phân tán có khả năng phục hồi) trên các mini-batches dữ liệu đó.

- MLlib (Machine Learning Library): là một framework machine learning phân tán trên Spark tận dụng khả năng tính toán tốc độ cao nhờ distributed memory-based của kiến ​​trúc Spark.

- GraphX: ​​là một framework xử lý đồ thị phân tán. Nó cung cấp một API để thực hiện tính toán biểu đồ có thể mô hình hóa các biểu đồ do người dùng xác định bằng cách sử dụng API đã được tối ưu sẵn.

### 2.2.4 Kiến trúc của Apache Spark:

- Kiến trúc của Apache spark gồm 2 trình cơ bản, đó là: trình điều khiển (Driver Program) và trình thực thi (Executors). Trong đó:

* Trình điều khiển thực hiện chức năng chuyển đổi mã từ người dùng thành các tác vụ khác nhau. Loại này sẽ phân phối trên các nút xử lý. Trình điều khiển chứa SparkSession.
* Trình thực thi hoạt động trên các nút xử lý. Nó có chức năng thực hiện các nhiệm vụ được giao trên các nút xử lý đó.

- Trong kiến trúc này, Apache Spark tạo ra các lệnh xử lý dữ liệu từ người dùng. Lệnh này ở dạng đồ thị vòng có hướng hoặc tạo ra lớp lập lịch DAG. Chúng có thể xác định các tác vụ được thực thi trên từng nút tương ứng.

A picture containing text, screenshot, font, number

Description automatically generated

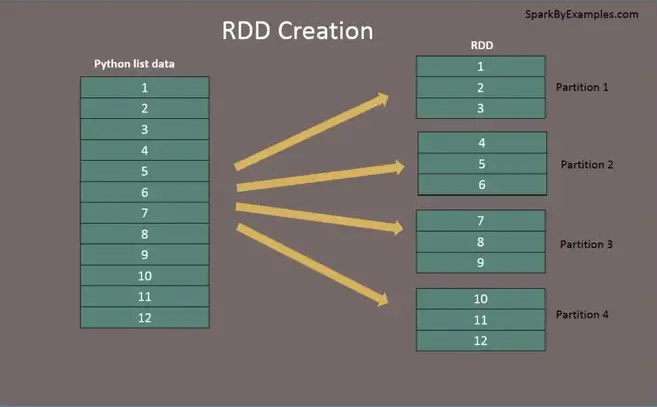
*Hình 6: Mô hình kiến trúc của Apache Spark*

**-** SparkSession: Đại diện cho khả năng tương tác với executors trong 1 chương trình. Spark session chính là entry point của mọi chương trình Spark. Từ SparkSession, có thể tạo RDD/ DataFrame/ DataSet, thực thi SQL… từ đó thực thi tính toán phân tán.

Khi chạy, từ logic của chương trình (chính là code xử lý thông qua việc gọi các API), Driver sẽ sinh ra các task tương ứng và lên lịch chạy các task, sau đó gửi xuống Executor để thực thi. Dữ liệu được lưu trên memory của Executor nên việc thực thi tính toán sẽ nhanh hơn rất nhiều.

### 2.2.5 Resilient Distributed Datasets:

- Resilient Distributed Datasets (RDD) là một cấu trúc dữ liệu cơ bản của Spark. Nó là một tập hợp bất biến phân tán của một đối tượng. Mỗi dataset trong RDD được chia ra thành nhiều phần vùng logical. Có thể được tính toán trên các node khác nhau của một cụm máy chủ (cluster). RDDs có thể chứa bất kỳ kiểu dữ liệu nào của Python, Java, hoặc đối tượng Scala, bao gồm các kiểu dữ liệu do người dùng định nghĩa



*Hình 5: Minh họa cho RDD của Apache Spark*

- Có hai cách để tạo RDDs:

* Tạo từ một tập hợp dữ liệu có sẵn trong ngôn ngữ sử dụng như Java, Python, Scala.
* Lấy từ dataset hệ thống lưu trữ bên ngoài như HDFS, Hbase hoặc các cơ sở dữ liệu quan hệ.

- RDDs được lưu trữ trên bộ nhớ RAM của các node trong cụm máy tính, giúp tăng tốc độ xử lý và giảm chi phí đọc/ghi dữ liệu từ đĩa. Khi một node trong cụm máy tính bị lỗi, các phần tử trong RDD sẽ được tái tạo trên các node khác để đảm bảo tính fault tolerant.

- RDDs có tính chất đàn hồi (resilient), có nghĩa là chúng có khả năng tự phục hồi khi có lỗi xảy ra. Nếu một node trong cụm máy tính gặp sự cố và không thể hoạt động, RDD sẽ tự động chuyển dữ liệu đến các node khác trong cụm máy tính để đảm bảo rằng quá trình xử lý dữ liệu vẫn tiếp tục được thực hiện.

- RDDs cũng cho phép các phép toán song song (parallel operations) để tăng tốc độ xử lý dữ liệu. Các phép toán này được thực hiện trên từng phần của RDDs trên các node khác nhau của cụm máy tính đồng thời, và kết quả được trả về và tổng hợp lại thành một RDD kết quả.

- RDD cung cấp hai loại hoạt động: transformations và actions. Transformations là các hoạt động không tính toán ngay lập tức, mà chỉ tạo ra một RDD mới dựa trên RDD hiện tại. Trong khi đó, actions là các hoạt động tính toán và trả về kết quả cho người dùng hoặc lưu trữ kết quả trở lại trong bộ nhớ hoặc HDFS.

### 2.2.6 Lazy Evaluation, Transformations and Actions:

- Transformations là những thao tác trên RDD để tạo ra một RDD mới. Khi một transformation được thực hiện trên RDD, Spark không thực sự tính toán và trả về kết quả ngay lập tức, mà nó sẽ chỉ đánh dấu (mark) RDD được tạo ra bởi transformation đó để tính toán trên nó sau đó. Điều này được gọi là "lazy evaluation" và giúp giảm thiểu số lượng tính toán không cần thiết, đồng thời tăng tốc độ thực thi các thao tác trên RDD. Một số transformations phổ biến trong Spark gồm:

* map(): thực hiện một hàm trên mỗi phần tử của RDD và trả về một RDD mới chứa kết quả.
* filter(): lọc ra các phần tử thỏa mãn một điều kiện nhất định và trả về một RDD mới chứa các phần tử đó.
* groupByKey(): nhóm các phần tử có cùng khóa lại với nhau và trả về một RDD mới chứa các cặp (khóa, tập hợp các giá trị tương ứng với khóa đó).
* reduceByKey(): áp dụng một hàm giảm (reducer) trên các phần tử có cùng khóa và trả về một RDD mới chứa các cặp (khóa, giá trị sau khi giảm).

- Actions là những thao tác trên RDD để tính toán và trả về kết quả hoặc lưu trữ kết quả vào đĩa. Khi một action được thực hiện trên RDD, Spark sẽ tính toán các transformations được áp dụng trên RDD đó và trả về kết quả cuối cùng. Các action có thể là in ra kết quả trên console, lưu trữ kết quả vào file hoặc đẩy dữ liệu vào hệ thống ngoài. Một số Actions phổ biến trong Spark:

* count(): trả về số lượng phần tử trong RDD.
* collect(): trả về một danh sách chứa tất cả các phần tử trong RDD.
* saveAsTextFile(): lưu trữ RDD vào file văn bản.
* foreach(): áp dụng một hàm trên mỗi phần tử của RDD.

A diagram of a transformation process

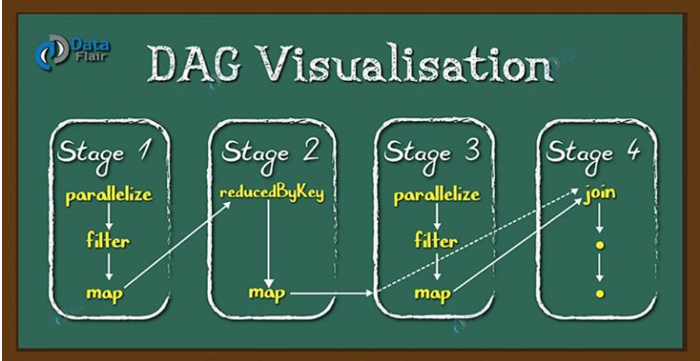
Description automatically generated with low confidence

- Lazy evaluation: là một kỹ thuật được sử dụng trong các ngôn ngữ lập trình để trì hoãn việc tính toán các giá trị đến khi chúng cần thiết. Thay vì tính toán ngay lập tức, các giá trị chỉ được tính toán khi chúng được sử dụng cho một mục đích cụ thể nào đó. Kỹ thuật này thường được sử dụng trong các hệ thống phân tán lớn như Apache Spark để tối ưu hóa hiệu suất tính toán và tài nguyên máy chủ.

* Trong Spark, các phép biến đổi (transformations) như map (), filter(), reduce() sẽ không được thực hiện ngay lập tức khi gọi, mà thay vào đó sẽ tạo ra một kế hoạch dựa trên các phép biến đổi đó và trả về một RDD mới, mà chứa thông tin về các phép biến đổi được áp dụng trên RDD gốc. Khi một phép thực hiện một hành động (action) trên RDD (ví dụ như collect(), take(), count()...), thì các phép biến đổi sẽ được áp dụng và tính toán các giá trị.
* Việc sử dụng lazy evaluation giúp giảm thiểu tình trạng xử lý dữ liệu không cần thiết và tối ưu hóa hiệu suất tính toán, đặc biệt là trong các ứng dụng Big Data. Tuy nhiên, cần lưu ý rằng việc sử dụng quá nhiều phép biến đổi mà không có hành động thực sự có thể gây ra các vấn đề liên quan đến bộ nhớ và hiệu suất.

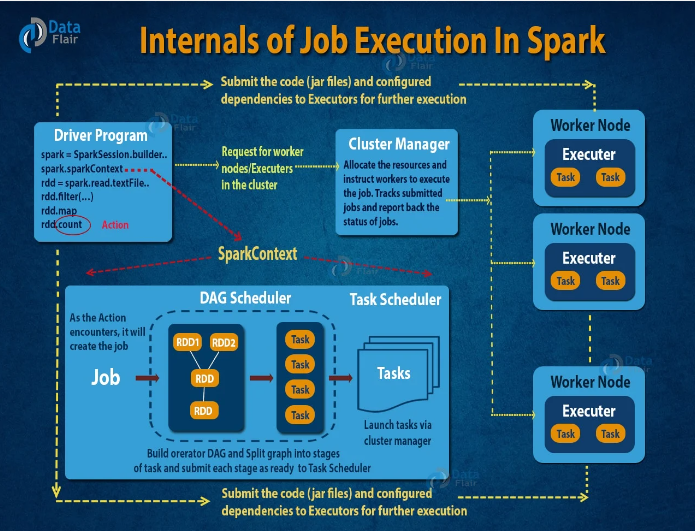
### 2.2.7 **Directed Acyclic Graph (**DAG):

- (Đồ thị theo chu kỳ có hướng) DAG trong Apache Spark là một tập hợp các Đỉnh và Cạnh, trong đó các đỉnh biểu thị các RDD và các cạnh biểu thị Actions sẽ được áp dụng trên RDD. Trong Spark DAG, mọi cạnh đều hướng từ trước đến sau trong chuỗi. Khi kêu gọi Actions, DAG đã tạo sẽ gửi tới DAG Scheduler để phân chia thêm biểu đồ thành các giai đoạn của nhiệm vụ.



- Cách thức DAG hoạt động:

* Trình thông dịch là lớp đầu tiên, sử dụng trình thông dịch Scala, Spark diễn giải mã với một số sửa đổi.
* Spark tạo biểu đồ toán tử khi bạn nhập mã của mình vào bảng điều khiển Spark.
* Khi chúng tôi gọi một Actions trên Spark RDD ở mức cao, Spark sẽ gửi biểu đồ toán tử tới DAG Scheduler.
* Chia người vận hành thành các giai đoạn của nhiệm vụ trong DAG Scheduler. Một giai đoạn chứa tác vụ dựa trên phân vùng của dữ liệu đầu vào. DAG Scheduler kết hợp các toán tử với nhau. Ví dụ: lịch trình của người vận hành bản đồ trong một giai đoạn duy nhất.
* Các giai đoạn chuyển sang Task Scheduler. Nó khởi chạy tác vụ thông qua cluster manager. Sự phụ thuộc của các giai đoạn không được biết đối với bộ lập lịch tác vụ.
* The Worker thực hiện nhiệm vụ trên node



- Ở cấp độ cao hơn, chúng ta có thể áp dụng hai loại phép biến đổi RDD: phép biến đổi hẹp (ví dụ: map (), filter (), v.v.) và phép biến đổi rộng (ví dụ: reduceByKey()). Chuyển đổi thu hẹp không yêu cầu xáo trộn dữ liệu trên một phân vùng, các chuyển đổi thu hẹp sẽ nhóm thành một giai đoạn duy nhất trong khi chuyển đổi rộng dữ liệu sẽ xáo trộn. Do đó, chuyển đổi rộng dẫn đến ranh giới giai đoạn.

### 2.2.8 SparkSQL, Dataframe:

- SparkSQL:

* Spark SQL là một module trong Apache Spark được sử dụng để xử lý dữ liệu trong hệ thống phân tán thông qua SQL hoặc các câu lệnh tương tự SQL. Nó cho phép người dùng truy vấn dữ liệu lưu trữ trong các nguồn dữ liệu khác nhau bao gồm Hadoop Distributed File System (HDFS), Apache Cassandra, Apache HBase, Amazon S3 và nhiều nguồn dữ liệu khác thông qua cơ chế connector của Spark.
* Spark SQL cung cấp một bộ trình điều khiển (driver) cho phép người dùng truy cập vào các tệp dữ liệu lưu trữ trên HDFS hoặc các hệ thống cơ sở dữ liệu khác thông qua ODBC/JDBC. Bên cạnh đó, Spark SQL cũng hỗ trợ các chức năng xử lý dữ liệu phức tạp như các kết hợp join, subqueries, aggregation, grouping, window functions, cùng với khả năng tùy chỉnh các hàm định nghĩa riêng của người dùng (user-defined functions - UDFs)
* Spark SQL giúp người dùng có thể sử dụng các công cụ và kỹ thuật truy vấn dữ liệu quen thuộc như SQL để truy cập và xử lý các dữ liệu phân tán một cách hiệu quả. Từ đó, Spark SQL đã trở thành một trong những công cụ phổ biến trong các dự án xử lý dữ liệu lớn sử dụng Spark.

A picture containing text, font, line, receipt

Description automatically generated

- Dataframe:

* Dataframe là một cấu trúc dữ liệu phổ biến trong các hệ thống xử lý dữ liệu lớn, được sử dụng để lưu trữ và xử lý các tập dữ liệu có cấu trúc. Dataframe tương tự như bảng trong cơ sở dữ liệu quan hệ, nó có các cột và các hàng. Mỗi cột đại diện cho một thuộc tính, và mỗi hàng đại diện cho một bản ghi.
* Dataframe trong Spark được tổ chức thành các hàng và cột, tương tự như Pandas Dataframe. Mỗi cột có kiểu dữ liệu riêng, và các hàng được đánh số. Các Dataframe có thể được tạo ra từ nhiều nguồn khác nhau, bao gồm các tệp dữ liệu, cơ sở dữ liệu, RDD (Resilient Distributed Datasets), hoặc các API khác.
* Các đặc điểm bao gồm:
* immutable: tính bất biến, dữ liệu của 1 DF sau khi tạo ra sẽ không thay đổi, nếu muốn chỉnh sửa ta cần tạo ra DF mới từ DF ban đầu, thông qua DF api.
* rows: là đối tượng đại diện cho 1 bản ghi dữ liệu. 1 DF = tập các row phân tán
* set of columns has name and an associated type: Ý nói về việc dữ liệu của DF là có cấu trúc, gồm tên là kiểu dữ liệu.

A picture containing text, screenshot, font, diagram

Description automatically generated

# CHƯƠNG 3: XỬ LÝ DỮ LIỆU VỚI APACHE SPARK

## 3.1 Tổng quan về dữ liệu:

A screenshot of a computer

Description automatically generated

- Đây là bộ dữ liệu được sử dụng trong chương thứ hai của cuốn sách gần đây của Aurélien Géron “Học máy thực hành với Scikit-Learn và TensorFlow”. Nó phục vụ như một phần giới thiệu tuyệt vời để triển khai các thuật toán học máy vì nó yêu cầu làm sạch dữ liệu thô sơ, có một danh sách các biến dễ hiểu và nằm ở kích thước tối ưu giữa quá nhỏ và quá cồng kềnh.  
- Dữ liệu chứa thông tin từ cuộc điều tra dân số California năm 1990. Vì vậy, mặc dù nó có thể không giúp bạn dự đoán giá nhà ở hiện tại như bộ dữ liệu Zillow Zestimate, nhưng nó cung cấp một bộ dữ liệu giới thiệu có thể truy cập để dạy mọi người về kiến ​​thức cơ bản của máy học.

- Một số thông tin về các thuộc tính:

* **longitude**: Thước đo khoảng cách về phía tây của một ngôi nhà; một giá trị cao hơn là xa hơn về phía tây
* **latitude**: Thước đo khoảng cách về phía bắc của một ngôi nhà; một giá trị cao hơn là xa hơn về phía bắc
* housingMedianAge: Tuổi trung bình của một ngôi nhà trong một dãy nhà; một số thấp hơn là một tòa nhà mới hơn
* totalRooms: Tổng số phòng trong một dãy nhà
* totalBedrooms: Tổng số phòng ngủ trong một dãy nhà
* population: Tổng số người cư trú trong một khu phố
* **households**: Tổng số hộ gia đình, nhóm người cư trú trong một đơn vị nhà, cho một khu phố
* **medianIncome**: Thu nhập trung bình của các hộ gia đình trong một dãy nhà (được tính bằng hàng chục nghìn đô la Mỹ)
* medianHouseValue: Giá trị nhà trung bình cho các hộ gia đình trong một khu nhà (được tính bằng Đô la Mỹ)
* OceanProximity: Vị trí của ngôi nhà có biển/đại dương

## 3.2 Tiền xử lý dữ liệu:

A screenshot of a computer

Description automatically generated with medium confidence

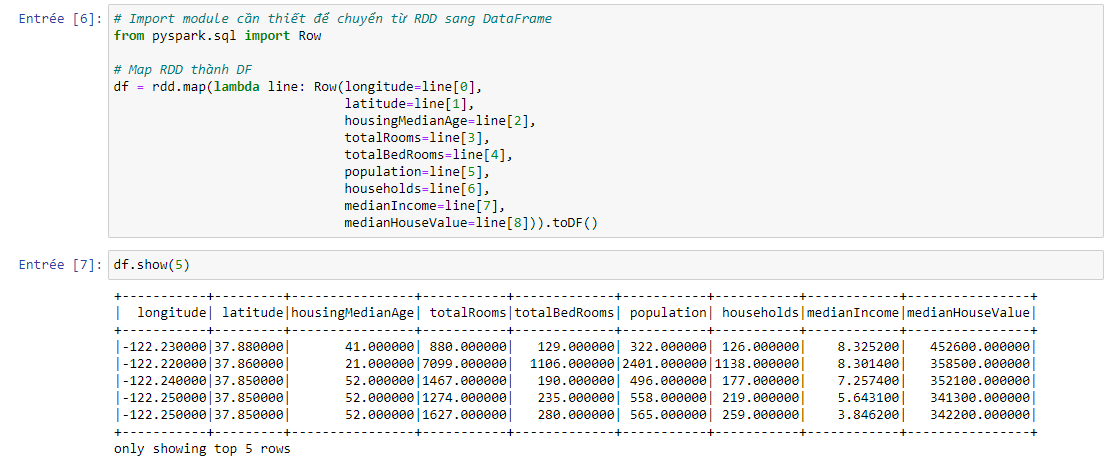
- Đầu tiên import SparkContext và SparkSession:

* SparkContext để cung cấp phương thức tạo và tính toán trên RDD
* SparkSession là giao diện cao cấp hơn của SparkContext, chức năng như SparkContext và còn hỗ trợ cho cả DataFrame.

A screenshot of a computer

Description automatically generated with medium confidence

- Tiếp đến là tách các dòng bằng dấu phẩy, sau đó để chuyển từ RDD sang DataFrame



- Để tính toán hoặc thực hiện các phép toán số học, chúng ta cần phải chuyển đổi kiểu dữ liệu từ chuỗi sang kiểu số.

A screenshot of a computer code

Description automatically generated with low confidence

## 3.3 Thăm dò và khai thác thông tin:

- Ta sẽ có những thông tin cơ bản như số bản ghi, giá trị trung bình, giá trị lớn nhất và nhỏ nhất của mỗi cột

A screenshot of a computer code

Description automatically generated with low confidence

- Do DataFrame của Spark không thể sử dụng Matplotlib nên ta chuyển sang DataFrame của Pandas.

A screenshot of a phone

Description automatically generated with low confidence

- Trực quan dữ liệu bằng biểu đồ để xem phân bố dữ liệu

A screenshot of a graph

Description automatically generated with low confidence

- Nhận thấy các cột như HousingMedianAge, medianHouseValue, medianIncome tăng đột ngột ở giá trị cuối, có thể đây là giới hạn khi người làm dữ liệu đã cho vào, nên ta xóa các giá trị đó đi để mô hình học máy chính xác hơn

A screenshot of a computer program

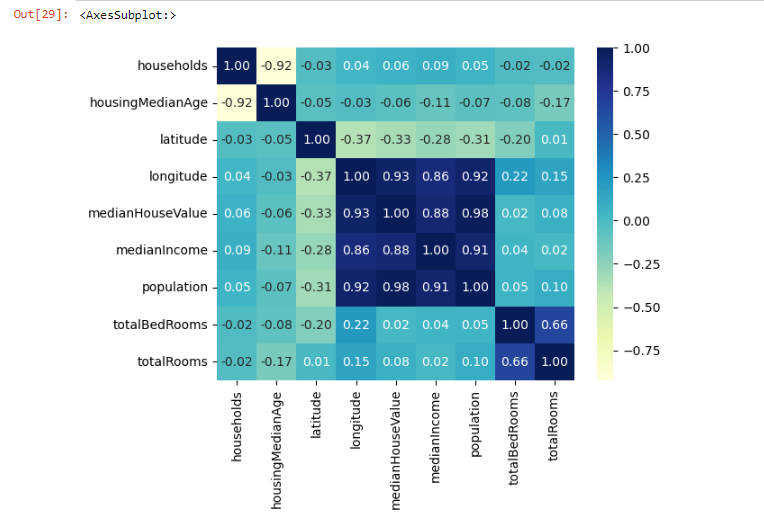
Description automatically generated with low confidence

- Sau khi xử lý:

A picture containing diagram, plot

Description automatically generated

- Tiếp theo ta sử dụng headmap để xem xét tự tương quan giữa các thuộc tính trong dữ liệu



- Sau đó để hiểu dữ liệu hơn chúng ta sẽ sử dụng kỹ thuật them một số đặc điểm. Thêm các đặc trưng mới có thể giúp mô hình học máy hiểu dữ liệu một cách tốt hơn, do đó cải thiện khả năng dự đoán của nó.

A screenshot of a computer code

Description automatically generated with low confidence

- Tiếp đến ta sẽ sử dụng phương pháp chia tỷ lệ dữ liệu (Scaling) để đưa các biến độc lập (independent variables) về cùng một khoảng giá trị. Điều này giúp cho các biến có thể được so sánh và đánh giá trọng số của chúng trong mô hình học máy một cách chính xác hơn.

A screenshot of a computer program

Description automatically generated with low confidence

# CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 4.1 Kết luận:

- Tổng kết lại, trong đề tài này, chúng ta đã tìm hiểu về cách xử lý dữ liệu với Apache Spark. Đầu tiên, chúng ta đã nắm được các khái niệm cơ bản của Apache Spark, bao gồm Spark RDD, Spark DataFrame và Spark SQL. Sau đó, chúng ta đã tìm hiểu về cách sử dụng Apache Spark để đọc dữ liệu từ file, chuyển đổi dữ liệu, lọc và lấy mẫu dữ liệu.

- Sau khi đã tải và xử lý dữ liệu, chúng ta đã sử dụng các thư viện khác nhau của Python như Pandas, NumPy và Matplotlib để phân tích và trực quan hóa dữ liệu. Chúng ta đã thực hiện các thao tác tiền xử lý như điền giá trị thiếu, xử lý giá trị ngoại lai, đánh giá mô hình và điều chỉnh độ giãn dữ liệu.

**4.2 Hướng phát triển:**

- Về hướng phát triển, với các kinh nghiệm đã học được từ đề tài này, chúng ta có thể tiếp tục nghiên cứu các kỹ thuật xử lý dữ liệu khác như kỹ thuật gom cụm, mạng nơ-ron nhân tạo, học sâu, học máy tăng cường và phân tích đồ thị. Chúng ta cũng có thể thử nghiệm với các bộ dữ liệu khác và thực hiện các thao tác xử lý dữ liệu khác để cải thiện độ chính xác của mô hình. Cuối cùng, chúng ta có thể triển khai ứng dụng của Apache Spark trên các hệ thống phân tán để xử lý các tập dữ liệu lớn.