

GENERAL KNOWLEDGE OF  
MINING OF MASSIVE DATASETS



MINING  
OF  
MASSIVE  
DATASETS

DANG HOANG DAI NGHIA

TON DUC THANG  
UNIVERSITY

# Mục Lục

<b>I.</b>	<b>Khái niệm Machine Learning:</b> .....	4
1.	Giới thiệu:.....	4
2.	Ứng dụng của Machine Learning:.....	4
2.1.	Tự động phân loại.....	4
2.2	Ứng dụng trong các mạng xã hội.....	4
2.3.	Nhận diện hình ảnh.....	4
<b>II.</b>	<b>Khái niệm Pyspark:</b> .....	5
<b>III.</b>	<b>Thành phần của Spark:</b> .....	5
<b>IV.</b>	<b>Những đặc điểm nổi bật:</b> <sup>[1]</sup> .....	5
<b>V.</b>	<b>Pyspark trong các ngành công nghiệp:</b> <sup>[5]</sup> .....	6
1.	Ngành truyền thông: .....	6
2.	Ngành tài chính: .....	6
3.	Các ngành chăm sóc sức khỏe:.....	6
4.	Nhà bán lẻ và thương mại điện tử:.....	6
5.	Du lịch: .....	7
<b>VI.</b>	<b>Tìm hiểu:</b> .....	7
1.	Spark properties: <sup>[1]</sup> .....	7
2.	Dynamically Loading Spark Properties: <sup>[1]</sup> .....	8
3.	Viewing Spark Properties: <sup>[1]</sup> .....	9
4.	Available Properties: <sup>[1]</sup> .....	9
5.	Environment Variables: <sup>[1]</sup> .....	9
6.	Custom Haloop/Hive Configuration: <sup>[1]</sup> .....	10
7.	Spark RDD: <sup>[2]</sup> .....	11
7.1.	Thực thi trên Spark RDD: <sup>[3]</sup> .....	11
7.2.	Các loại RDD: <sup>[3]</sup> .....	12
7.3.	Các Transformation và Action với RDD: <sup>[2]</sup> .....	13
7.4.	Một số kỹ thuật với RDD: <sup>[3]</sup> .....	14
8.	Spark DataFrame: <sup>[4]</sup> .....	15
8.1.	Làm thế nào để tạo DataFrame ? .....	15
8.2.	Lượt đồ của DataFrame .....	15
8.3.	Tên cột và đếm hàng (Hàng và cột):.....	16
8.4.	Mô tả một cột đặc biệt:.....	17
8.5.	Chọn nhiều cột: .....	17
8.6.	Chọn nhiều cột khác biệt.....	18

<b>8.7.</b>	<b>Lọc dữ liệu .....</b>	<b>18</b>
<b>8.8.</b>	<b>Lọc dữ liệu (nhiều tham số).....</b>	<b>19</b>
<b>8.9.</b>	<b>Sắp xếp dữ liệu (OrderBy) .....</b>	<b>20</b>
<b>VII.</b>	<b>Tài liệu tham khảo:.....</b>	<b>21</b>

# Tìm hiểu về Pyspark trong Machine Learning

## I. Khái niệm Machine Learning:

### 1. Giới thiệu:

Machine Learning (Học Máy hoặc Máy Học) là một lĩnh vực nhỏ của Khoa Học Máy Tính, nó có khả năng tự học hỏi dựa trên dữ liệu đưa vào mà không cần phải được lập trình cụ thể. Trong những năm gần đây machine learning thu hút rất nhiều người quan tâm, từ các công ty lớn như Google, Facebook, Microsoft đến các công ty khởi nghiệp đều đầu tư vào machine learning. Hàng loạt các ứng dụng sử dụng machine learning ra đời trên mọi lĩnh vực của cuộc sống, tiêu biểu như xe tự vận hành của Google và Tesla, hệ thống tự tag khuôn mặt trong ảnh của Facebook, trợ lý ảo Siri của Apple, hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý phim của Netflix, máy chơi cờ vây AlphaGo của Google DeepMind, ..., ngoài ra còn nhiều ứng dụng khác bao trùm từ khoa học máy tính đến những ngành ít liên quan hơn như vật lý, hóa học, y học, chính trị.

Machine learning có 3 mối quan hệ sau:[3]

- Machine learning và trí tuệ nhân tạo.
- Machine learning và Big data.
- Machine learning và dự đoán tương lai.

Sau đây mình sẽ đi sâu tìm hiểu các mối quan hệ này.

### 2. Ứng dụng của Machine Learning: <sup>[6]</sup>

#### 2.1. Tự động phân loại

Phân loại tin tức là một ứng dụng của machine learning. Một vấn đề thực tế là bây giờ khối lượng thông tin đã tăng lên rất nhiều trên web. Tuy nhiên, mỗi người có sở thích hoặc lựa chọn cá nhân của mình. Vì vậy, để chọn hoặc thu thập một phần thông tin phù hợp trở thành một thách thức đối với người dùng từ vô số nội dung trên trang web.<sup>[6]</sup>

Phân loại các danh mục một cách rõ ràng, dễ điều hướng giúp cho các khách hàng mục tiêu chắc chắn sẽ tăng khả năng truy cập các trang tin tức. Hơn nữa, độc giả hoặc người dùng có thể tìm kiếm tin tức cụ thể một cách hiệu quả và nhanh chóng. Có một số thuật toán machine learning có thể giải quyết vấn đề này như Support Vector Machines, Naive Bayes, K-nearest neighbor,...<sup>[6]</sup>

#### 2.2 Ứng dụng trong các mạng xã hội

Machine learning đang được sử dụng trong một loạt các ứng dụng ngày nay. Một trong những ví dụ nổi tiếng nhất là Facebook News Feed. Nguồn cấp tin tức sử dụng học máy để cá nhân hóa từng nguồn cấp dữ liệu thành viên.

Nếu một thành viên thường xuyên dừng lại để đọc hoặc thích một bài đăng của một người bạn cụ thể, News Feed sẽ bắt đầu hiển thị nhiều hơn về hoạt động của người bạn đó trước đó trong nguồn cấp dữ liệu. Đằng sau hệ thống ấy, phần mềm sử dụng phân tích thống kê và phân tích dự đoán để xác định các mẫu trong dữ liệu người dùng và sử dụng các mẫu đó để điền vào News Feed. Nếu thành viên không còn dừng lại để đọc, thích hoặc bình luận trên các bài đăng của bạn bè, dữ liệu mới đó sẽ được bao gồm trong tập dữ liệu và News Feed sẽ điều chỉnh tương ứng.<sup>[6]</sup>

Không chỉ riêng facebook, ta có thể bắt gặp những tính năng tương tự đó qua các mạng xã hội khác như google, instagram,...

#### 2.3. Nhận diện hình ảnh

Nhận dạng hình ảnh là một trong những ví dụ về machine learning và trí tuệ nhân tạo phổ biến nhất. Về cơ bản, nó là một cách tiếp cận để xác định và phát hiện các đặc trưng của một đối tượng trong hình ảnh kỹ thuật số. Hơn nữa, kỹ

thuật này có thể được sử dụng để phân tích sâu hơn, chẳng hạn như nhận dạng mẫu, nhận diện hình khuôn, nhận dạng khuôn mặt, nhận dạng ký tự quang học và nhiều hơn nữa,...<sup>[6]</sup>

## II. Khái niệm Pyspark:

Apache Spark cho phép xây dựng các mô hình dự đoán nhanh chóng với việc tính toán được thực hiện trên một nhóm các máy tính, có thể tính toán cùng lúc trên toàn bộ tập dữ liệu mà không cần phải trích xuất mẫu tính toán thử nghiệm. Tốc độ xử lý của Spark có được do việc tính toán được thực hiện cùng lúc trên nhiều máy khác nhau. Đồng thời việc tính toán được thực hiện ở bộ nhớ trong (in-memories) hay thực hiện hoàn toàn trên RAM.

Spark cho phép xử lý dữ liệu theo thời gian thực, vừa nhận dữ liệu từ các nguồn khác nhau đồng thời thực hiện ngay việc xử lý trên dữ liệu vừa nhận được ( Spark Streaming).

Spark không có hệ thống file của riêng mình, nó sử dụng hệ thống file khác như: HDFS, Cassandra, S3,... Spark hỗ trợ nhiều kiểu định dạng file khác nhau (text, csv, json...) đồng thời nó hoàn toàn không phụ thuộc vào bất cứ một hệ thống file nào.

## III. Thành phần của Spark:

Apache Spark gồm có 5 thành phần chính : Spark Core, Spark Streaming, Spark SQL, MLlib và GraphX, trong đó:

Thành phần trung của Spark là Spark Core: cung cấp những chức năng cơ bản nhất của Spark như lập lịch cho các tác vụ, quản lý bộ nhớ, fault recovery, tương tác với các hệ thống lưu trữ... Đặc biệt, Spark Core cung cấp API để định nghĩa RDD (Resilient Distributed DataSet) là tập hợp của các item được phân tán trên các node của cluster và có thể được xử lý song song. <sup>[2]</sup>

Spark có thể chạy trên nhiều loại Cluster Managers như Hadoop YARN, Apache Mesos hoặc trên chính cluster manager được cung cấp bởi Spark được gọi là Standalone Scheduler. • Spark SQL cho phép truy vấn dữ liệu cấu trúc qua các câu lệnh SQL. Spark SQL có thể thao tác với nhiều nguồn dữ liệu như Hive tables, Parquet, và JSON.

- Spark Streaming cung cấp API để dễ dàng xử lý dữ liệu stream,
- MLlib Cung cấp rất nhiều thuật toán của học máy như: classification, regression, clustering, collaborative filtering...
- GraphX là thư viện để xử lý đồ thị.

Trong các thư viện mà Spark cung cấp thì có 69% người dùng Spark SQL, 62% sử dụng DataFrames, Spark Streaming và MLlib + GraphX là 58%

## IV. Những đặc điểm nổi bật: <sup>[1]</sup>

Xử lý dữ liệu: Spark xử lý dữ liệu theo lô và thời gian thực

Tính tương thích: Có thể tích hợp với tất cả các nguồn dữ liệu và định dạng tệp được hỗ trợ bởi cụm Hadoop

Hỗ trợ ngôn ngữ: hỗ trợ Java, Scala, Python và R

Phân tích thời gian thực

Mục tiêu sử dụng:

- Xử lý dữ liệu nhanh và tương tác
- Xử lý đồ thị
- Công việc lặp đi lặp lại
- Xử lý thời gian thực
- joining Dataset
- Machine Learning
- Apache Spark là Framework thực thi dữ liệu dựa trên Hadoop HDFS. Apache Spark không thay thế cho Hadoop nhưng nó là một framework ứng dụng. Apache Spark tuy ra đời sau nhưng được nhiều người biết đến hơn Apache Hadoop vì khả năng xử lý hàng loạt và thời gian thực.

## V. Pyspark trong các ngành công nghiệp: [5]

Mọi ngành công nghiệp đều xoay quanh dữ liệu lớn và nơi có dữ liệu lớn, có phân tích liên quan. Vì vậy, hãy xem xét các ngành công nghiệp khác nhau nơi Apache Spark được sử dụng.

### 1. Ngành truyền thông:

**Truyền thông** là một trong những ngành công nghiệp lớn nhất phát triển theo hướng phát trực tuyến. **Netflix** sử dụng Apache Spark để xử lý luồng thời gian thực để cung cấp các đề xuất trực tuyến được cá nhân hóa cho khách hàng của mình. Nó xử lý **450 tỷ** sự kiện mỗi ngày chảy vào các ứng dụng phía máy chủ.

### 2. Ngành tài chính:

**Tài chính** là một lĩnh vực khác mà việc xử lý thời gian thực của Apache Spark đóng vai trò quan trọng. Các ngân hàng đang sử dụng Spark để truy cập và phân tích hồ sơ truyền thông xã hội để hiểu rõ hơn có thể giúp họ đưa ra quyết định kinh doanh đúng đắn để **đánh giá rủi ro tín dụng**, quảng cáo được nhắm mục tiêu và phân khúc khách hàng. **Khách hàng** cũng giảm thời gian sử dụng Spark. **Phát hiện gian lận** là một trong những lĩnh vực được sử dụng rộng rãi nhất trong học máy mà Spark có liên quan.

### 3. Các ngành chăm sóc sức khỏe:

Các nhà cung cấp dịch vụ **chăm sóc sức khỏe** đang sử dụng Apache Spark để phân tích hồ sơ bệnh nhân cùng với dữ liệu lâm sàng trong quá khứ để xác định bệnh nhân nào có khả năng phải đối mặt với các vấn đề sức khỏe sau khi được xuất viện. Apache Spark được sử dụng trong giải trình tự bộ gen để giảm thời gian cần thiết để xử lý dữ liệu bộ gen.

### 4. Nhà bán lẻ và thương mại điện tử:

**Bán lẻ và thương mại điện tử** là một ngành mà người ta không thể tưởng tượng nó đang chạy mà không sử dụng phân tích và quảng cáo được nhắm mục tiêu. Một trong những nền tảng thương mại điện tử lớn nhất hiện nay, Alibaba một số công việc Spark lớn nhất trên thế giới để phân tích petabyte dữ liệu. Alibaba

thực hiện trích xuất tính năng trong dữ liệu hình ảnh. **eBay** sử dụng Apache Spark để cung cấp các ưu đãi được nhắm mục tiêu, nâng cao trải nghiệm của khách hàng và tối ưu hóa hiệu suất tổng thể.

## 5. Du lịch:

Các ngành công nghiệp **du lịch** cũng sử dụng Apache Spark. **TripAdvisor**, một trang web du lịch hàng đầu giúp người dùng lên kế hoạch cho một chuyến đi hoàn hảo, đang sử dụng Apache Spark để tăng tốc các đề xuất tùy chỉnh được cá nhân hóa. TripAdvisor sử dụng Apache Spark để cung cấp lời khuyên cho hàng triệu khách du lịch bằng cách so sánh hàng trăm trang web để tìm giá khách sạn tốt nhất cho khách hàng của mình.

## VI. Tìm hiểu:

### 1. Spark properties: <sup>[1]</sup>

Spark properties kiểm soát hầu hết tham số ứng dụng và có thể được cài đặt bằng cách sử dụng *SparkConf* và nó sẽ được chuyển tới *SparkContext* của bạn. *SparkConf* cho phép bạn cấu hình một số thuộc tính phổ biến, cũng như các cặp **key-value** tùy ý thông qua phương thức *set()*. VD: ta có thể khởi tạo một ứng dụng với hai luồng như sau:

```
val conf = new SparkConf()

    .setMaster("local[2]")

    .setAppName("CountingSheep")

val sc = new SparkContext(conf)
```

Chú ý ta có thể có nhiều hơn 1 thread trong một chế độ local, và trong trường hợp giống như Spark Streaming, chúng ta chắc chắn có thể yêu cầu nhiều hơn 1 thread để ngăn chặn bất kỳ loại vấn đề “chết đói” (*starvation*) nào.

Các thuộc tính chỉ định một số khoảng thời gian nên được chỉ định cấu hình với một đơn vị thời gian. Định dạng sau được chấp nhận:

```
25ms (milliseconds)
5s (seconds)
10m or 10min (minutes)
3h (hours)
5d (days)
1y (years)
```

Thuộc tính chỉ định kích thước byte phải được cấu hình với đơn vị kích thước. Định dạng sau được chấp nhận:

1b (bytes)

1k or 1kb (kibibytes = 1024 bytes)

1m or 1mb (mebibytes = 1024 kibibytes)

1g or 1gb (gibibytes = 1024 mebibytes)

1t or 1tb (tebibytes = 1024 gibibytes)

1p or 1pb (pebibytes = 1024 tebibytes)

## 2. Dynamically Loading Spark Properties: <sup>[1]</sup>

Trong một số trường hợp, bạn có thể muốn tránh hard-coding các cấu hình nhất định trong *SparkConf*. Ví dụ: nếu bạn muốn chạy cùng một ứng dụng với các bản khác nhau hoặc số lượng bộ nhớ khác nhau. Spark cho phép bạn làm điều đó chỉ cần tạo một còng trống:

```
val sc = new SparkContext(new SparkConf())
```

Sau đó, bạn có thể cung cấp các giá trị cấu hình trong thời gian chạy:

```
./bin/spark-submit --name "My app" --master local[4] --conf spark.eventLog.enabled=false  
--conf "spark.executor.extraJavaOptions=-XX:+PrintGCDetails -XX:+PrintGCTimeStamps" myApp.jar
```

The Spark shell và `spark-submit` là công cụ hỗ trợ hai cách để tải cấu hình. Đầu tiên là các tùy chọn dòng lệnh như là `--master` đã được nêu bên trên. `spark-submit` có thể hỗ trợ bất kỳ Spark property nào sử dụng flag `--conf/-c`, nhưng sử dụng cờ đặc biệt cho các thuộc tính đóng một vai trò trong việc khởi chạy ứng dụng Spark. `./bin/spark-submit --help` sẽ hiển thị toàn bộ danh sách các tùy chọn này.

`./bin/spark-submit --help` cũng sẽ đọc các tùy chọn cấu hình từ `conf/spark-defaults.conf`, trong đó mỗi dòng bao gồm một khóa và một giá trị được phân tách bằng khoảng trắng. Ví dụ:

```
spark.master      spark://5.6.7.8:7077  
spark.executor.memory  4g  
spark.eventLog.enabled true  
spark.serializer  org.apache.spark.serializer.KryoSerializer
```

Mọi giá trị được chỉ định dưới dạng flag hoặc trong file thuộc tính sẽ được chuyển đến ứng dụng và được hợp nhất với những giá trị được chỉ định thông qua *SparkConf*. Các thuộc tính được đặt trực tiếp trên *SparkConf* được ưu tiên cao nhất, sau đó flag chuyển đến `spark-submit` hoặc `spark-shell`, sau đó là các tùy chọn trong file `spark-defaults.conf`. Một số khóa cấu hình đã được đổi tên kể từ các phiên bản Spark trước đó, trong trường hợp này, các tên khóa cũ hơn vẫn được chấp nhận, nhưng được ưu tiên thấp hơn bất kỳ trường hợp nào của khóa mới hơn.

Spark properties chủ yếu được chia thành 2 loại: một là liên quan đến triển khai, như là “`spark.driver.memory`”, “`spark.executor.instances`”, loại này không bị ảnh hưởng khi thiết lập lập trình thông



qua **SparkConf** trong thời gian chạy, hoặc hành vi phụ thuộc vào trình quản lý cụm và chế độ triển khai bạn chọn, vì vậy bạn nên cài đặt thông qua tệp cấu hình hoặc tùy chọn dòng lệnh `spark-submit`; một loại khác chủ yếu liên quan đến kiểm soát thời gian chạy Spark, như “`spark.task.maxFailures`”, loại thuộc tính này có thể được đặt theo một trong hai cách.

### 3. Viewing Spark Properties: <sup>[1]</sup>

UI web ứng dụng tại <http://<driver>:4040> liệt kê các thuộc tính Spark trong tab “Environment”. Đây là một nơi hữu ích để đảm bảo rằng các thuộc tính của bạn đã được đặt chính xác. Lưu ý rằng chỉ các giá trị được chỉ định rõ ràng thông qua **spark-defaults.conf**, **SparkConf** hoặc dòng lệnh mới xuất hiện. Đối với tất cả các thuộc tính cấu hình khác, bạn có thể giả sử giá trị mặc định được sử dụng.

### 4. Available Properties: <sup>[1]</sup>

Hầu hết các thuộc tính kiểm soát cài đặt nội bộ đều có giá trị mặc định hợp lý. Một số tùy chọn phổ biến nhất để đặt là: Application Properties, Runtime Environment, Shuffle Behavior,...

### 5. Environment Variables: <sup>[1]</sup>

Một số cài đặt Spark nhất định có thể được định cấu hình thông qua các biến môi trường, được đọc từ `conf/spark-env.sh` trong thư mục nơi mà Spark được cài đặt. Ở chế độ Standalone và Mesos, file này có thể cung cấp thông tin cụ thể như tên máy chủ. Nó cũng có nguồn gốc khi chạy các ứng dụng Spark cục bộ hoặc các tập lệnh gửi.

Lưu ý rằng `conf/spark-env.sh` không tồn tại theo mặc định khi Spark được cài đặt. Tuy nhiên, bạn có thể sao chép `conf/spark-env.sh.template` để tạo nó. Đảm bảo rằng bạn thực thi bản sao.

Các biến sau có thể được đặt trong `spark-env.sh`:

Environment Variable	Meaning
JAVA_HOME	Vị trí nơi Java được cài đặt (Nếu nó không có trên PATH mặc định của bạn).
PYSPARK_PYTHON	Thực thi nhị phân Python để sử dụng cho PySpark trong cả trình điều khiển và người sử dụng (mặc định là <code>python2.7</code> nếu có, nếu không là <code>python</code> ). Thuộc tính <code>spark.pyspark.python</code> được ưu tiên nếu nó được đặt.
PYSPARK_DRIVER_PYTHON	Thực thi nhị phân Python để chỉ sử dụng cho PySpark trong trình điều khiển (mặc định là <code>PYSPARK_PYTHON</code> ). Thuộc tính <code>spark.pyspark.driver.python</code> được ưu tiên nếu nó được đặt.

SPARKR_DRIVER_R	Thực thi nhiệm vụ R để sử dụng cho trình bao SparkR (mặc định là R). Thuộc tính <code>spark.r.shell.command</code> được ưu tiên nếu nó được đặt
SPARK_LOCAL_IP	Địa chỉ IP của máy để liên kết.
SPARK_PUBLIC_DNS	Tên máy chủ chương trình Spark của bạn sẽ quảng cáo đến các máy khác.

Lưu ý: Khi chạy Spark trên YARN ở chế độ `cluster`, các biến môi trường cần được đặt bằng thuộc tính `spark.yarn.appMasterEnv. [EnvironmentVariableName]` trong tệp `conf / spark-defaults.conf` của bạn

## 6. Custom Haloop/Hive Configuration: <sup>[1]</sup>

Nếu ứng dụng Spark của bạn đang tương tác với Hadoop, Hive hoặc cả hai, có thể có các tệp cấu hình Hadoop / Hive trong đường dẫn nổi của Spark.

Nhiều ứng dụng đang chạy có thể yêu cầu cấu hình từ phía Client Haloop/Hive khác. Bạn có thể sao chép và sửa đổi `hdfs-site.xml`, `core-site.xml`, `fiber-site.xml`, `hive-site.xml` trong Spark's classpath cho mỗi ứng dụng. Trong Spark Cluster chạy trên YARN, các tệp cấu hình này được đặt trên cluster-wide, và ứng dụng không thể thay đổi một cách an toàn.

Sự lựa chọn tốt hơn là sử dụng các thuộc tính của `spark.hadoop` ở dạng `spark.hadoop`. Và sử dụng các thuộc tính của `spark hive` ở dạng `spark.hive`. Ví dụ: thêm cấu hình “`spark.hadoop.abc.def = xyz`” biểu thị việc thêm thuộc tính `hadoop` “`abc.def = xyz`” và thêm cấu hình “`spark.hive.abc = xyz`” biểu thị việc thêm thuộc tính `hive` “`hive.abc = xyz`” và thêm cấu hình “`spark.hive.abc = xyz`” đại diện cho việc thêm thuộc tính `hive` “`hive.abc = xyz`”. Chúng có thể được coi giống như các Spark properties bình thường có thể được đặt trong `$SPARK_HOME/conf/spark-defaults.conf`

Trong một số trường hợp, bạn có thể muốn tránh hard-coding các cấu hình nhất định trong *SparkConf*. Ví dụ, Spark cho phép bạn chỉ cần tạo một `conf` trống và thiết lập các thuộc tính `spark / spark.hadoop / spark.hive`.

```
val conf = new SparkConf().set("spark.hadoop.abc.def", "xyz")
val sc = new SparkContext(conf)
```

Ngoài ra, bạn có thể sửa đổi hoặc thêm cấu hình trong thời gian chạy:

```
./bin/spark-submit \
--name "My app" \
--master local[4] \
--conf spark.eventLog.enabled=false \
--conf "spark.executor.extraJavaOptions=-XX:+PrintGCDetails -XX:+PrintGCTimeStamps" \
--conf spark.hadoop.abc.def=xyz \
```

```
--conf spark.hive.abc=xyz
```

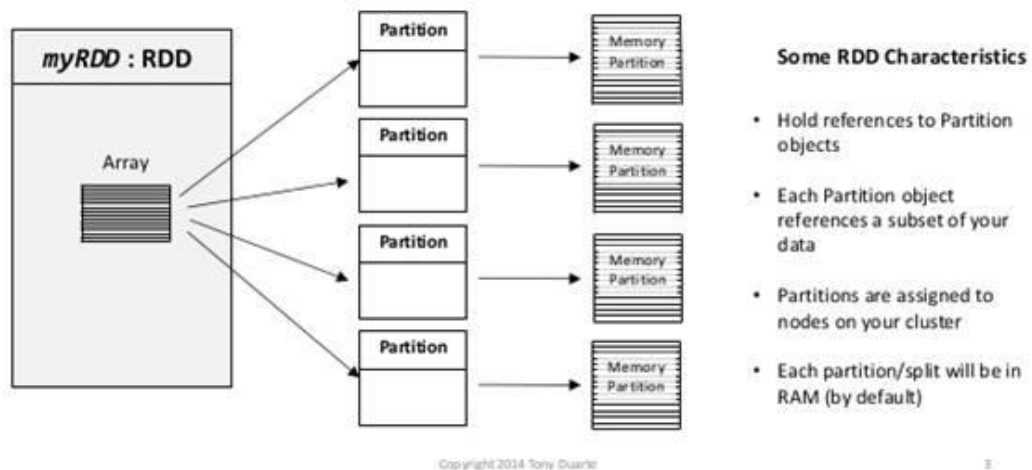
```
myApp.jar
```

## 7. Spark RDD: [2]

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.

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.

### What is an RDD?



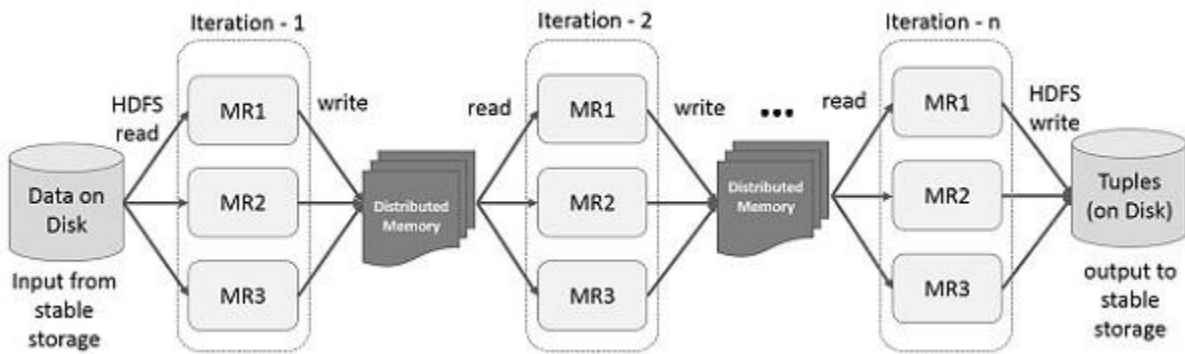
Trong 1 chương trình Spark, RDD là đại diện cho tập dữ liệu phân tán.

VD: Ta có 1 object `colors = Array {red, blue, black, white, yellow}`. Trong chương trình thông thường, phần dữ liệu (red, blue, black, white, yellow) nằm trên 1 máy tính duy nhất và thông qua biến `colors`, bạn có thể truy cập đến dữ liệu. Tuy nhiên trong hệ phân tán, nếu phần dữ liệu red, blue nằm trên máy tính A còn black, white, yellow lại nằm trên máy tính B thì sao? RDD sẽ giúp bạn truy cập 2 phần dữ liệu rời rạc này như 1 đối tượng thông thường. Đó là lý do tại sao mình nói RDD là đại diện cho tập dữ liệu phân tán.

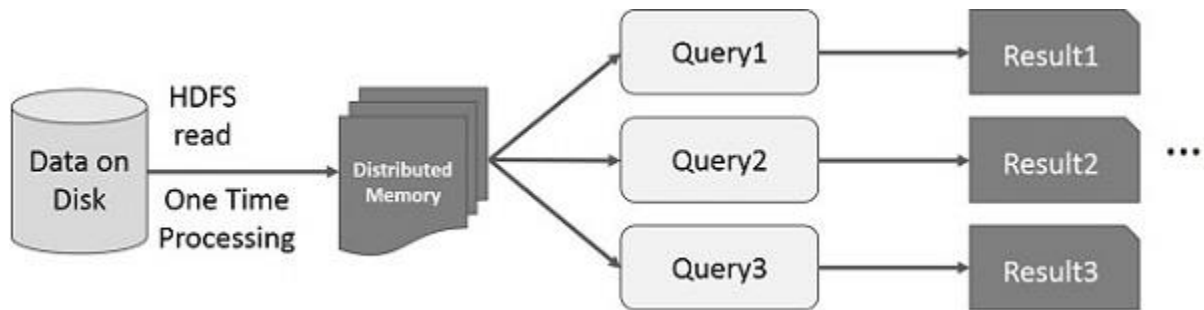
### 7.1. Thực thi trên Spark RDD: [3]

Spark RDD lưu trữ trạng thái của bộ nhớ dưới dạng một đối tượng trên các công việc và đối tượng có thể chia sẻ giữa các công việc đó. Việc xử lý dữ liệu trong bộ nhớ nhanh hơn 10 đến 100 lần so với network và disk.

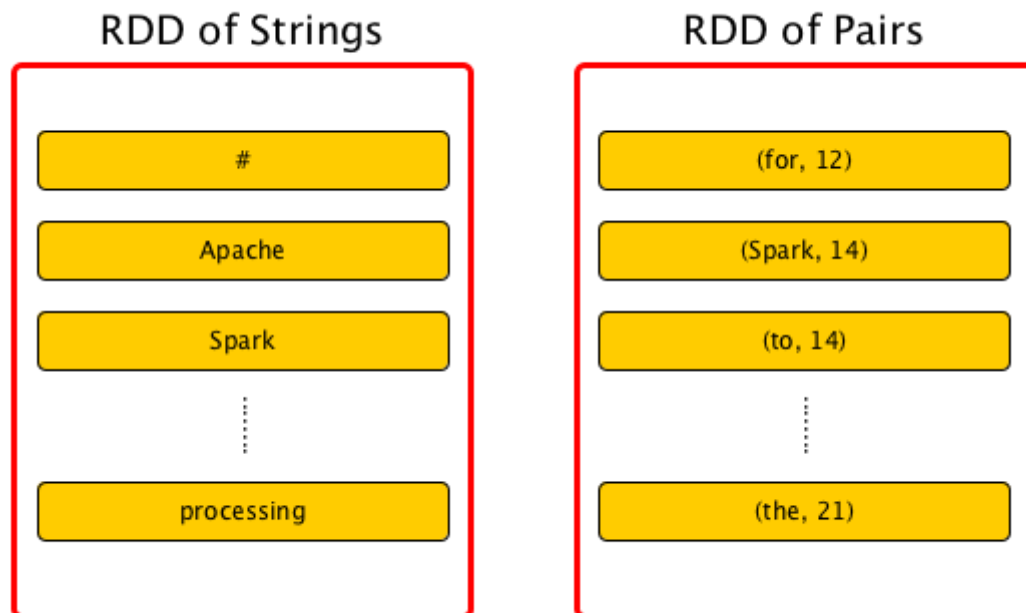
- Iterative Operation trên Spark RDD:



- Interactive Operations trên Spark RDD:



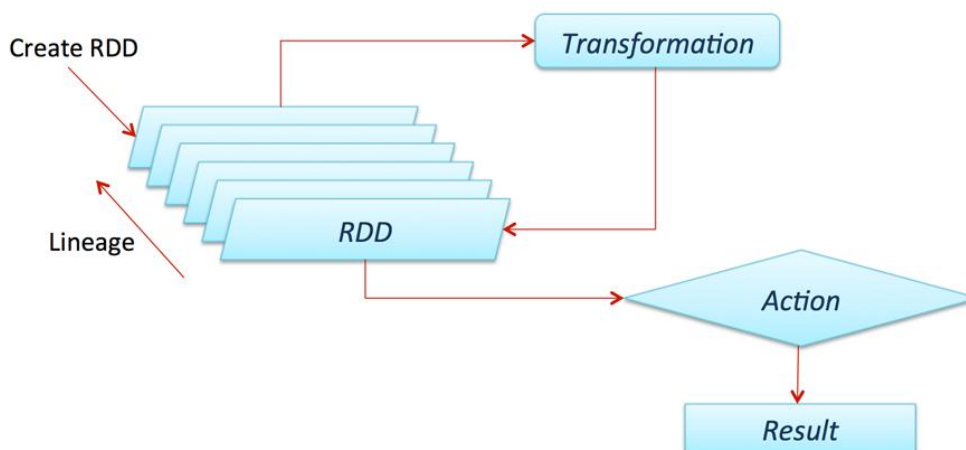
## 7.2. Các loại RDD: [3]



- Các RDD biểu diễn một tập hợp cố định, đã được phân vùng các record để có thể xử lý song song.
- Các record trong RDD có thể là đối tượng Java, Scala hay Python tùy lập trình viên chọn. Không giống như DataFrame, mỗi record của DataFrame phải là một dòng có cấu trúc chứa các field đã được định nghĩa sẵn.
- RDD đã từng là API chính được sử dụng trong series Spark 1.x và vẫn có thể sử dụng trong version 2.X nhưng không còn được dùng thường xuyên nữa.

- RDD API có thể được sử dụng trong Python, Scala hay Java:
  - Scala và Java: Performance tương đương trên hầu hết mọi phần. (Chi phí lớn nhất là khi xử lý các raw object)
  - Python: Mất một lượng performance, chủ yếu là cho việc serialization giữa tiến trình Python và JVM

### 7.3. Các Transformation và Action với RDD: <sup>[2]</sup>



#### 7.3.1. Transformations <sup>[2]</sup>

Phép biến đổi từ RDD này sang RDD khác là 1 transformation, như việc biến đổi tập web log ban đầu sang tập web log chỉ chứa log gọi qua app là 1 transformation.

Một số transformations:

Transformations	Meaning
<code>map(func)</code>	RDD mới được tạo thành bằng cách áp dụng func lên tất cả các bản ghi trên RDD ban đầu.
<code>filter(func: Boolean)</code>	RDD mới được tạo thành bằng cách áp dụng func lên tất cả các bản ghi trên RDD ban đầu và chỉ lấy những bản ghi mà func trả về true
<code>distinct([numPartitions])</code>	loại bỏ trùng lặp trong RDD
<code>flatMap(func)</code>	cung cấp một hàm đơn giản hơn hàm map. Yêu cầu output của map phải là một structure có thể lặp và mở rộng được.

<code>sortByKey([ascending], [numPartitions])</code>	mô tả một hàm để trích xuất dữ liệu từ các object của RDD và thực hiện sort được từ đó.
--	---

### 7.3.2. Actions <sup>[2]</sup>

Sau tất cả các phép biến đổi, khi muốn tương tác với kết quả cuối cùng (VD xem kết quả, collect kết quả, ghi kết quả...) ta gọi 1 action.

Có thể kể đến 1 số Action như:

Actions	Meaning
<code>take(n)</code>	lấy n bản ghi từ RDD về driver
<code>Collect</code>	lấy tất cả RDD về driver
<code>saveAsTextFile("path")</code>	ghi dữ liệu RDD ra file
<code>Count</code>	đếm số bản ghi của RDD

### 7.4. Một số kỹ thuật với RDD: <sup>[3]</sup>

- Lưu trữ file:

- Thực hiện ghi vào các file plain-text
- Có thể sử dụng các codec nén từ thư viện của Hadoop
- Lưu trữ vào các database bên ngoài yêu cầu ta phải lặp qua tất cả partition của RDD – Công việc được thực hiện ngầm trong các high-level API

- `sequenceFile` là một flat file chứa các cặp key-value, thường được sử dụng làm định dạng input/output của MapReduce. Spark có thể ghi các `sequenceFile` bằng cách ghi lại các cặp key-value

- Đồng thời, Spark cũng hỗ trợ ghi nhiều định dạng file khác nhau, cho phép define các class, định dạng output, config và compression scheme của Hadoop.

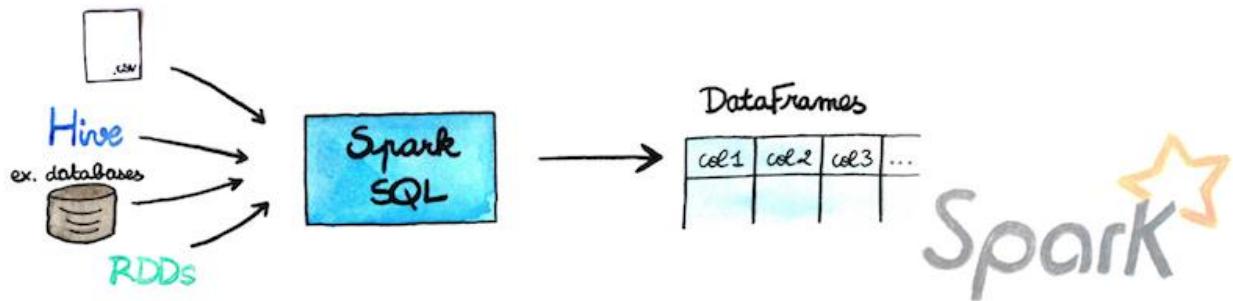
- Caching: Tăng tốc xử lý bằng cache

- Caching với RDD, Dataset hay DataFrame có nguyên lý như nhau.
- Chúng ta có thể lựa chọn cache hay persist một RDD, và mặc định, chỉ xử lý dữ liệu trong bộ nhớ

- Checkpointing: Lưu trữ lại các bước xử lý để phục hồi

- Checkpointing lưu RDD vào đĩa cứng để các tiến trình khác để thể sử dụng lại RDD point này làm partition trung gian thay vì tính toán lại RDD từ các nguồn dữ liệu gốc
- Checkpointing cũng tương tự như cache, chỉ khác nhau là lưu trữ vào đĩa cứng và không dùng được trong API của DataFrame
- Cần sử dụng nhiều để tối ưu tính toán.

## 8. Spark DataFrame: <sup>[4]</sup>



Spark DataFrame là phiên bản Spark 1.3. Nó là một tập hợp phân phối dữ liệu được sắp xếp vào các cột được đặt tên. Khái niệm khôn ngoan, nó bằng với bảng trong cơ sở dữ liệu quan hệ hoặc khung dữ liệu trong [R](#) / Python.

### 8.1. Làm thế nào để tạo DataFrame ?

Hãy tải dữ liệu từ tệp CSV. Ở đây chúng ta sẽ sử dụng phương thức **spark.read.csv** để tải dữ liệu vào DataFrame , `fifa_df`. Phương thức thực tế là **spark.read.format [csv / json]** .

```
from pyspark.sql import *

fifa_df = spark.read.csv("sample_data/click_data_sample.csv", inferSchema = True, header = True)

fifa_df.show(5)
```

```
+-----+-----+-----+
|      click.at|user.id|campaign.id|
+-----+-----+-----+
|2015-04-27 20:40:40| 144012|Campaign077|
|2015-04-27 00:27:55|  24485|Campaign063|
|2015-04-27 00:28:13|  24485|Campaign063|
|2015-04-27 00:33:42|  24485|Campaign038|
|2015-04-27 01:00:04|  24485|Campaign063|
+-----+-----+-----+
only showing top 5 rows
```

### 8.2. Lướt đồ của DataFrame

Để xem sơ đồ, tức là cấu trúc của DataFrame, chúng ta sẽ sử dụng phương thức **printSchema** . Điều này sẽ cung cấp cho chúng tôi các cột khác nhau trong DataFrame của chúng tôi, cùng với loại dữ liệu và các điều kiện không thể thực hiện được cho cột cụ thể đó.

```

from pyspark.sql import *

fifa_df = spark.read.csv("sample_data/click_data_sample.csv", inferSchema = True, header = True)

#fifa_df.show(5)

fifa_df.printSchema()

```

```

root
 |-- click.at: string (nullable = true)
 |-- user.id: integer (nullable = true)
 |-- campaign.id: string (nullable = true)

```

### 8.3. Tên cột và đếm hàng (Hàng và cột):

Khi chúng tôi muốn xem tên và số lượng hàng và cột của một DataFrame cụ thể, chúng tôi sử dụng các phương pháp sau.

#### 8.3.1. Tên cột:

```

9. from pyspark.sql import *
10.
11. fifa_df = spark.read.csv("sample_data/click_data_sample.csv", inferSchema = True, header = True)
12.
13. #fifa_df.show(5)
14.
15. #fifa_df.printSchema()
16.
17. fifa_df.columns #Column Names

```

```
['click.at', 'user.id', 'campaign.id']
```

#### 8.3.2. Đếm hàng:

```

9. from pyspark.sql import *
10.
11. fifa_df = spark.read.csv("sample_data/click_data_sample.csv", inferSchema = True, header = True)
12.
13. #fifa_df.show(5)
14.
15. #fifa_df.printSchema()
16.
17. #fifa_df.columns #Column Names
18.
19. fifa_df.count() #Row Count

```

```
327430
```



### 8.3.3. Đếm cột:

```
9. from pyspark.sql import *
10.
11. fifa_df = spark.read.csv("sample_data/click_data_sample.csv", inferSchema = True, header = True)
12. #fifa_df.show(5)
13. #fifa_df.printSchema()
14. #fifa_df.columns #Column Names
15. #fifa_df.count() #Row Count
16. len(fifa_df.columns) #Column Count
17.
```

3

### 8.4. Mô tả một cột đặc biệt:

Nếu chúng ta muốn xem tóm tắt về bất kỳ cột cụ thể nào của DataFrame, chúng tôi sử dụng describe phương thức này. Phương pháp này cung cấp cho chúng tôi bản tóm tắt thống kê của cột đã cho, nếu không được chỉ định, nó cung cấp tóm tắt thống kê của DataFrame.

```
from pyspark.sql import *

fifa_df = spark.read.csv("sample_data/click_data_sample.csv", inferSchema = True, header = True)

fifa_df.describe("`campaign.id`").show(2)
fifa_df.describe("`user.id`").show(2)
```

```
+-----+-----+
|summary|campaign.id|
+-----+-----+
| count|      327430|
| mean|         null|
+-----+-----+
only showing top 2 rows

+-----+-----+
|summary|          user.id|
+-----+-----+
| count|          327430|
| mean|83915.91885288458|
+-----+-----+
only showing top 2 rows
```

### 8.5. Chọn nhiều cột:

Nếu chúng tôi muốn chọn các cột cụ thể từ DataFrame, chúng tôi sử dụng select phương thức.

```
from pyspark.sql import *

fifa_df = spark.read.csv("sample_data/click_data_sample.csv", inferSchema = True, header = True)
```

```
fifa_df.describe("`campaign.id`,`user.id`).show(5)
```

```
+-----+-----+-----+
|summary|campaign.id|      user.id|
+-----+-----+-----+
|  count|      327430|      327430|
|   mean|         null|83915.91885288458|
| stddev|         null|48937.088016689886|
|    min|Campaign001|           4|
|    max|Campaign133|      169570|
+-----+-----+-----+
```

## 8.6. Chọn nhiều cột khác biệt

```
9. from pyspark.sql import *
10.
11. fifa_df = spark.read.csv("sample_data/click_data_sample.csv", inferSchema = True, header = True)
12.
13. fifa_df.describe("`campaign.id`,`user.id`).distinct().show(5)
```

```
+-----+-----+-----+
|summary|campaign.id|      user.id|
+-----+-----+-----+
|    max|Campaign133|      169570|
| stddev|         null|48937.088016689886|
|  count|      327430|      327430|
|    min|Campaign001|           4|
|   mean|         null|83915.91885288458|
+-----+-----+-----+
```

## 8.7. Lọc dữ liệu

Để lọc dữ liệu, theo điều kiện được chỉ định, chúng tôi sử dụng `filter` lệnh. Ở đây, chúng tôi đang lọc DataFrame của chúng tôi dựa trên điều kiện ID đối sánh phải bằng 1096 và sau đó chúng tôi đang tính toán có bao nhiêu bản ghi / hàng trong đầu ra được lọc.

```
from pyspark.sql import *

fifa_df = spark.read.csv("sample_data/click_data_sample.csv", inferSchema = True, header = True)

fifa_df.filter(fifa_df.user=='24485').show()
```

```

+-----+-----+-----+
|      click.at| user|   campaign|
+-----+-----+-----+
| 4/27/2015 0:27|24485|Campaign063|
| 4/27/2015 0:28|24485|Campaign063|
| 4/27/2015 0:33|24485|Campaign038|
| 4/27/2015 1:00|24485|Campaign063|
| 4/29/2015 0:55|24485|Campaign074|
|4/29/2015 14:17|24485|Campaign011|
|4/29/2015 20:59|24485|Campaign011|
| 4/30/2015 0:49|24485|Campaign011|
| 4/30/2015 1:19|24485|Campaign011|
| 5/1/2015 0:19|24485|Campaign011|
| 5/1/2015 22:24|24485|Campaign011|
| 5/3/2015 9:08|24485|Campaign011|
| 5/3/2015 14:53|24485|Campaign011|
| 5/3/2015 15:09|24485|Campaign011|
+-----+-----+-----+

```

## 8.8. Lọc dữ liệu (nhiều tham số)

Chúng tôi có thể lọc dữ liệu của mình dựa trên nhiều điều kiện (AND hoặc OR)

```

from pyspark.sql import *

fifa_df = spark.read.csv("sample_data/click_data_sample.csv", inferSchema = True, header = True)

fifa_df.filter((fifa_df.user=='144012') | (fifa_df.campaign=='Campaign007')).show(5)

```

```

+-----+-----+-----+
|      click.at| user|   campaign|
+-----+-----+-----+
|4/27/2015 20:40|144012|Campaign077|
|4/27/2015 23:31|164138|Campaign007|
| 4/27/2015 1:26|147766|Campaign007|
|4/27/2015 23:26| 7650|Campaign007|
|4/27/2015 23:55| 61412|Campaign007|
+-----+-----+-----+
only showing top 5 rows

```

## 8.9. Sắp xếp dữ liệu (OrderBy)

Để sắp xếp dữ liệu chúng tôi sử dụng `OrderBy` phương pháp. Theo mặc định, nó sắp xếp theo thứ tự tăng dần, nhưng chúng ta cũng có thể thay đổi nó theo thứ tự giảm dần.

```
from pyspark.sql import *

fifa_df = spark.read.csv("sample_data/click_data_sample.csv", inferSchema = True, header = True)

fifa_df.orderBy(fifa_df.user).show()
```

```
+-----+-----+-----+
|      click.at|user|   campaign|
+-----+-----+-----+
|4/28/2015 12:58|  4| Campaign074|
| 4/29/2015  2:39|  4| Campaign063|
|4/30/2015 17:52|  4| Campaign038|
|4/27/2015 20:48|  4| Campaign108|
|4/28/2015 17:57|  4| Campaign074|
|4/30/2015 13:53|  4| Campaign063|
|4/29/2015 18:30|  4| Campaign063|
|4/28/2015 22:24|  9| Campaign016|
|4/29/2015 14:13|  9| Campaign016|
|4/28/2015 22:27| 10| Campaign124|
+-----+-----+-----+
only showing top 10 rows
```

## VII. Tài liệu tham khảo:

[1]: <https://spark.apache.org/docs/latest/configuration.html>

[2]: <https://www.facebook.com/notes/c%E1%BB%99ng-%C4%91%E1%BB%93ng-big-data-vi%E1%BB%87t-nam/apache-spark-fundamentals-ph%E1%BA%A7n-2-spark-core-v%C3%A0-rdd/514714606074061/>

[3]: [https://laptrinh.vn/books/apache-spark/page/apache-spark-rdd#:~:text=Resilient%20Distributed%20Datasets%20\(RDD\)%20l%C3%A0,t%C3%A1n%20c%E1%BB%A7a%20m%E1%BB%99t%20%C4%91%E1%BB%91i%20t%C6%B0%E1%BB%A3ng.&text=RDDs%20c%C3%B3%20th%E1%BB%83%20ch%E1%BB%A9a%20b%E1%BA%A5t,do%20ng%C6%B0%E1%BB%9Di%20d%C3%B9ng%20%C4%91%E1%BB%8Bnh%20ngh%C4%A9a.](https://laptrinh.vn/books/apache-spark/page/apache-spark-rdd#:~:text=Resilient%20Distributed%20Datasets%20(RDD)%20l%C3%A0,t%C3%A1n%20c%E1%BB%A7a%20m%E1%BB%99t%20%C4%91%E1%BB%91i%20t%C6%B0%E1%BB%A3ng.&text=RDDs%20c%C3%B3%20th%E1%BB%83%20ch%E1%BB%A9a%20b%E1%BA%A5t,do%20ng%C6%B0%E1%BB%9Di%20d%C3%B9ng%20%C4%91%E1%BB%8Bnh%20ngh%C4%A9a.)

[4]: <https://helpex.vn/article/huong-dan-pyspark-dataframe-gioi-thieu-ve-dataframes-5c6b21e6ae03f628d053c29e>

[5]: <https://helpex.vn/article/huong-dan-pyspark-tim-hieu-apache-spark-bang-python-5c6b201dae03f628d053c256>

[6]: <https://longvan.net/machine-learning-la-gi-ung-dung-cua-machine-learning.html>