# **Homework 1: Tìm hiểu về spark**

Phần I: Tìm hiểu các phương thức trong class RDD

**1.1 Apache Spark:** Apache spark là một framework mã nguồn mở tính toán cụm (open source cluster computing framework). Apache spark cho phép bạn xây dựng những mô hình dự đoán nhanh chóng với khả năng thực hiện tính toán cùng lúc trên một nhóm các máy tính hay trên toàn bộ các tập dữ liệu mà không cần thiết phải trích xuất các mẫu tính toán thử nghiệm.

**1.2 Tìm hiểu về RDD:** 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 thành nhiều phần vùng logical. Có thể được tính toán trên các node khác nhau của một cụm máy chủ (cluster).

**1.3 Cách tạo ra RDD:**

* Tạo ra 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.

**1.4 Ý nghĩa của RDD:** Có một vấn đề tìm ẩn khi sử dụng dữ liệu hoặc chia sẻ dữ liệu trong các hệ thống máy tính phân tán hiện có (như MapReduce) và đó là chúng ta cần phải lưu trữ dữ liệu trong một storage phân tán ổn định trung gian như HDFS.

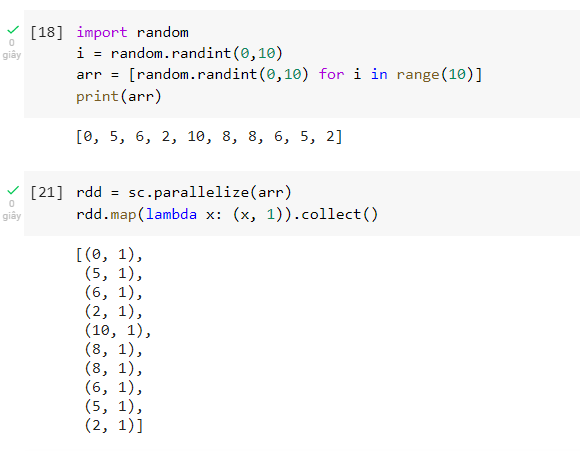
Để khắc phục được vấn đề của MapReduce, RDD hỗ trợ tính toán xử lý trong bộ nhớ. Điều này có nghĩ nó lưu trữ trạng thái của bộ nhớ dưới dạng một đối tượng trên các công việc và đối tượng có thể chia sẻ giữa các công việc đó. Việc xử lý dữ liệu trong bộ nhớ nhanh hơn 10 đến 100 lần so với network và disk.

Ngoài ra, vì RDD là bộ dữ liệu phân tán linh hoạt. Bằng cách thao tác chi tiết thô, các hoạt động được áp dụng trên tất cả các phần tử trong bộ dữ liệu, RDD được tạo ra qua một tập hợp các phép biến đổi, nó ghi các phép biến đổi đó chứ không phải dữ liệu thực tế. Trong trường hợp chúng ta mất một số phân vùng của RDD, chúng ta có thể phát lại chuyển đổi trên phân vùng đó theo dòng để đạt được cùng một tính toán, thay vì sao chép dữ liệu trên nhiều nút. Đặc điểm này là lợi ích của RDD vì nó tiết kiệm rất nhiều trong việc quản lý và mở rộng dữ liệu, từ đó đạt được những tính toán nhanh hơn.

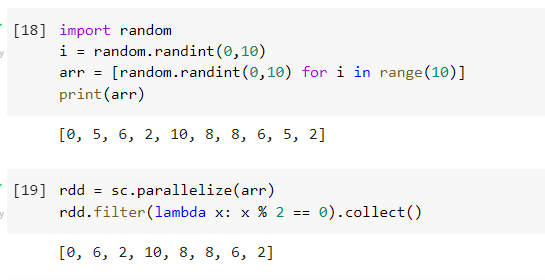
**1.5 Một số phương thức chính của RDD:**

**Transformation:**

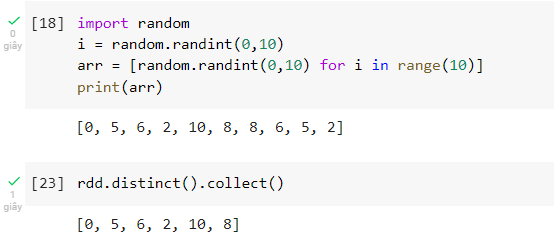
* map(func): Trả về một tập dữ liệu phân tán mới được hình thành bằng cách chuyển từng phần tử của nguồn thông qua một hàm chức năng

****

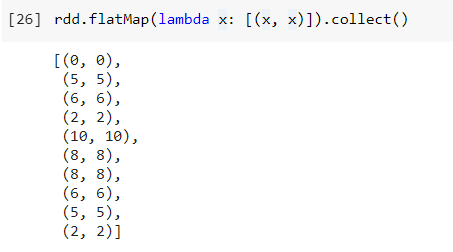
* filter(func): Trả về tập dữ liệu mới được hình thành bằng cách chọn các phần tử của nguồn mà func trả về true.

****

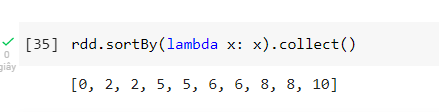
* distinct([*numPartitions*]): Loại bỏ trùng lặp trong RDD.

****

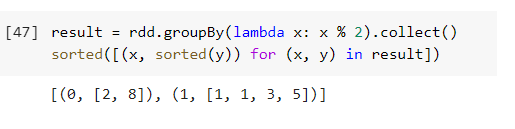
* flatMap(function): Cung cấp một hàm đơn giản hơn hàm map. Yêu cầu output của map phải là một structure có thể lặp và mở rộng được.

****

* sortBy([*ascending*], [*numPartitions*]): mô tả một hàm để trích xuất dữ liệu từ các object của RDD và thực hiện sort được từ đó.

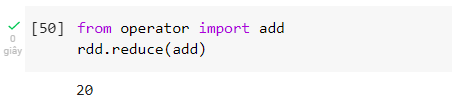


* groupBy([*ascending*], [*numPartitions*]): trả về RDD các mục được nhóm

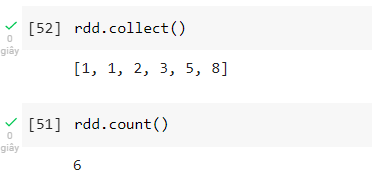


**Actions:**

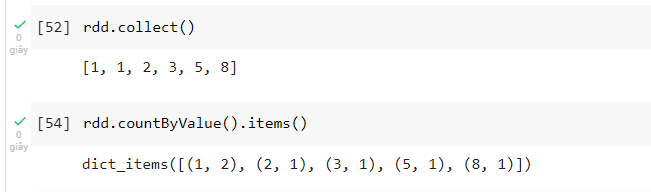
* reduce(func): thực hiện hàm reduce trên RDD để thu về 1 giá trị duy nhất.



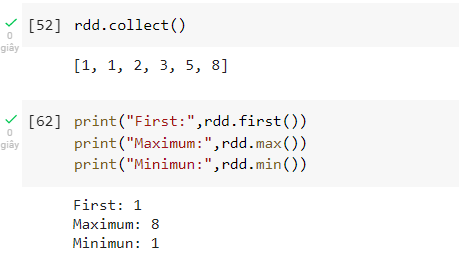
* count(): đếm số dòng trong RDD.



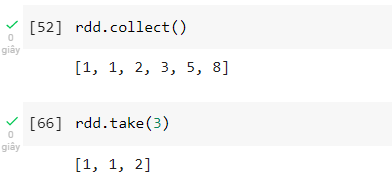
* countByValue(): đếm số lượng của từng giá trị duy nhất trong RDD dưới dạng dictionary.



* first(), max(), min(): lấy giá trị đầu tiên, maximum, minimun của RDD.



* take(n) và các method tương tự: lấy n giá trị đầu tiên của RDD.



* Ngoài ra còn có các method về phân phối như: mean(), variance(), stdev(),…