Xử lý dữ liệu lớn với Apache Spark

- Giới thiệu Apache Spark và các modules.
- Spark Core và lập trình thông qua RDD APIs.
- Cài đặt Spark và chạy chương trình trên Cluster

Giảng Viên: Nguyễn Chí Thanh

Nguyễn **Chí Thanh**
Big Data Engineer/ Data Architect
Blog: https://karcuta.medium.com

ABOUT **ME**

- Trên 5 năm kinh nghiệm trong lĩnh vực Big Data Engineering.
- Tham gia xây dựng và triển khai hệ thống vBI, Viettel Data Lake cho Viettel Telecom.
- Sở hữu chứng chỉ Quốc tế về Hadoop, Spark do Cloudera, Databricks cấp (CCA 175, CRT020).
- Thiết kế phát triển các hệ thống trên nền tảng Hadoop Ecosytem: Hdfs, Spark, Kafka, Hive...

Databricks Training <kate.sullivan@databricks.com>
tới tôi

Tiếng Anh > Tiếng Việt Dịch thư

Congratulations on becoming a Databricks Certified Associate Developer for Apache Spark 2.4!

Your score was 92.6%
A passing score is 70.0%

Your scores will be reflected in our LMS at https://academy.databricks.com in about a week. In addition to that, a certificate will be made available for download via the LMS. We currently do not have a badging service but we are working to set up one up as we speak - more information will be shared with everyone once that service is ready.

Sincerely,
Databricks Training

Score	Pass

Score comments

Your exam is graded on each of the output requirements. The below information is a scoring log for each scenario. These are intended to help you study and further you skills.

Problem 1	Pass
Problem 2	Pass
Problem 3	Pass
Problem 4	Pass
Problem 5	Pass
Problem 6	Pass
Problem 7	Records contain incorrect data
Problem 8	Pass
Problem 9	Pass

Congratulations! You passed the CCA Spark and Hadoop Developer (CCA175).

cloudera®

Cloudera hereby recognizes that

Thanh Nguyen Chi

has successfully completed all requirements
and is hereby certified as a

CCA Spark and Hadoop Developer

License: 100-023-161
Issued: April 16, 2020
Expiration: April 16, 2022

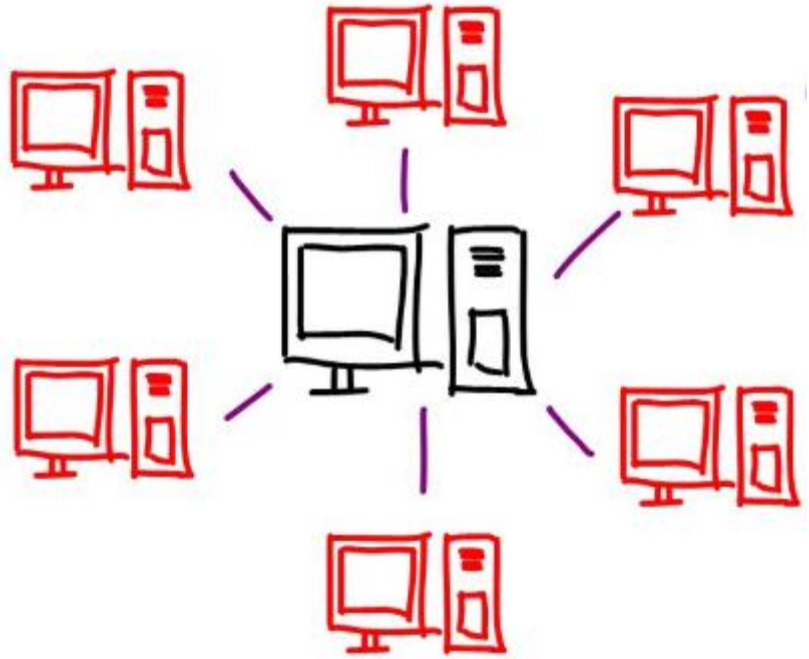
Tom Reilly
Chief Executive Officer

CCA cloudera
SPARK & HADOOP DEVELOPER

1.
Apache Spark Overview

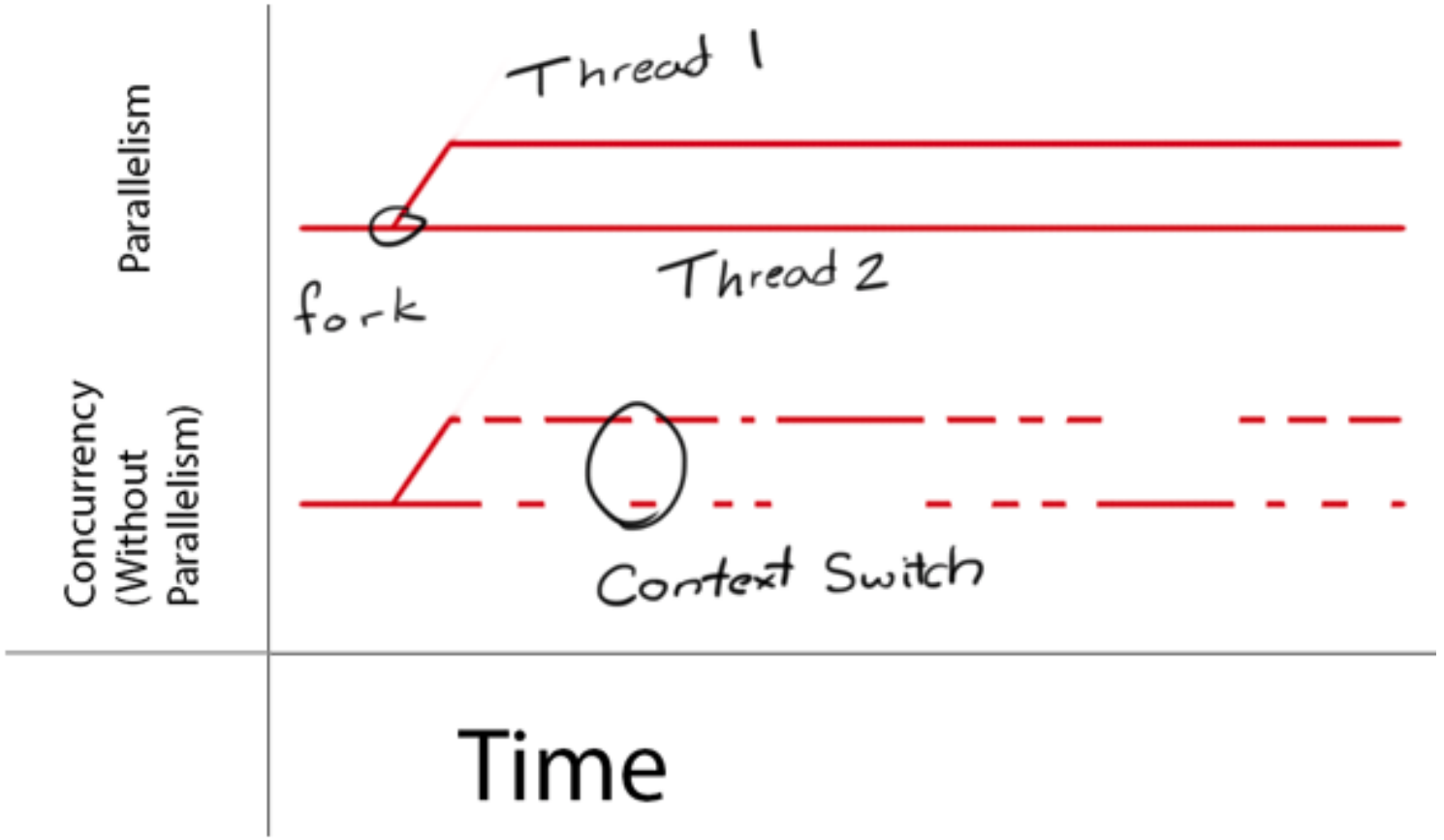
Hệ thống phân tán

- Bao gồm các máy tính độc lập không phụ thuộc lẫn nhau.
- Có thể là các máy tính có kiến trúc khác nhau.
- Được kết nối với nhau bằng mạng máy tính.
- Các phần mềm trên các máy này có khả năng phối hợp với nhau, chia sẻ tài nguyên hoặc thực hiện một nhiệm vụ chung.
- Hệ phân tán cung cấp dịch vụ một cách thông nhất, người sử dụng không cần quan tâm tới chi tiết của hệ thống.



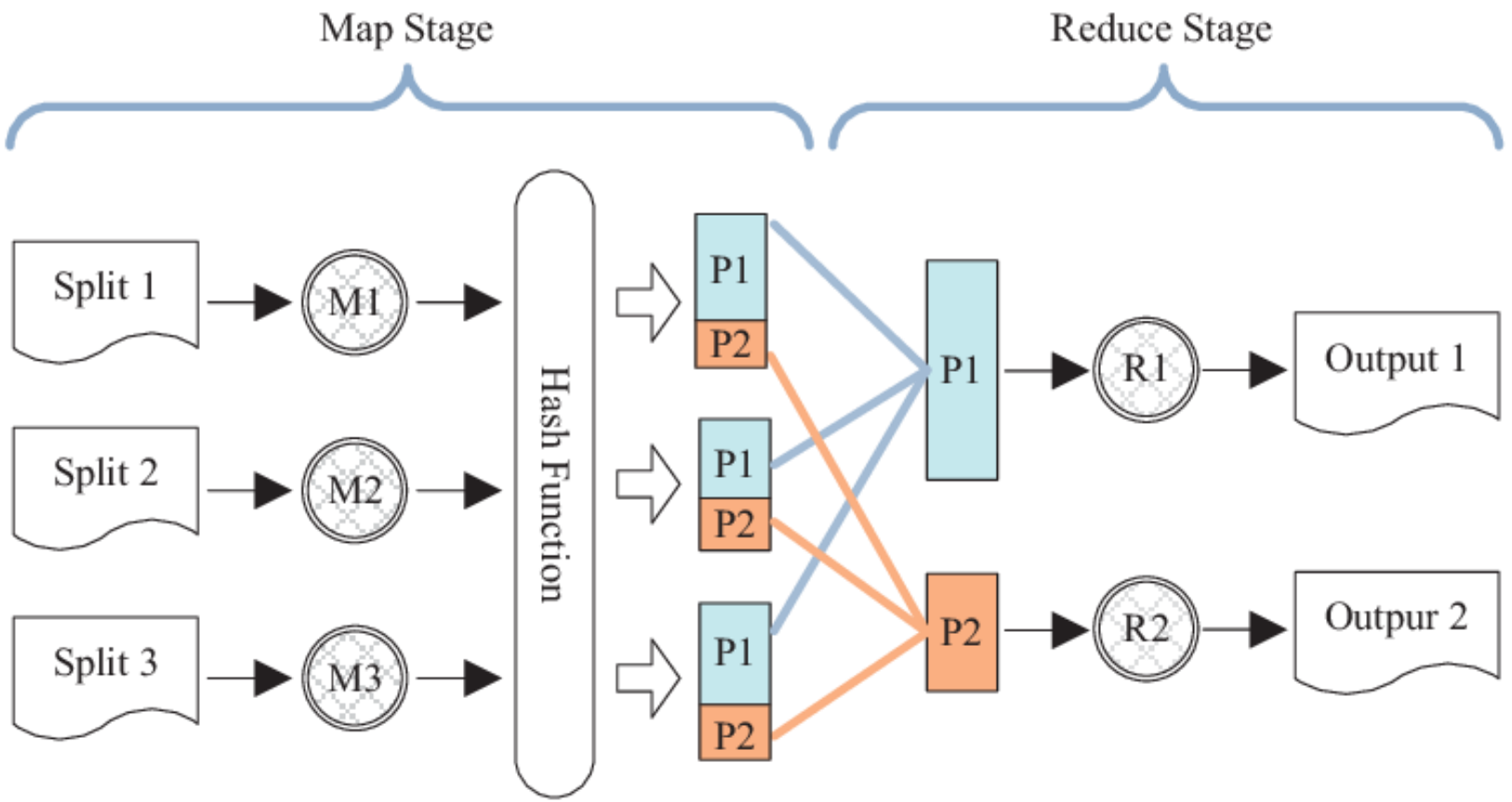
Concurrency & Parallelism

- Concurrency: Các tiến trình chia sẻ nhau CPU Time
- Parallelism: Thực sự song song khi có nhiều tiến trình xử lý công việc trong 1 thời điểm



MapReduce Programing Model

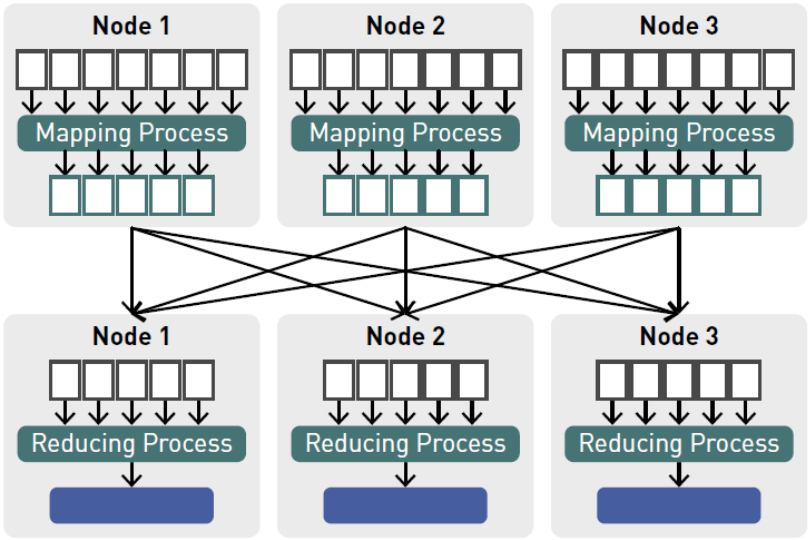
- Mô hình cho phép thực thi tính toán song song trong hệ thống phân tán.



Google Map-Reduce (1)

- Google sử dụng MR để index và exploding lượng lớn dữ liệu trên web trên cluster lớn.

- Concept của Google MR
  - **Distributed Data**: Dữ liệu được phân tán trên cụm
  - **Distributed Computation**: Tính toán phân tán.
  - **Tolerate faults**: Có khả năng tự động thực hiện lại tính toán khi có lỗi trên nodes bất kỳ

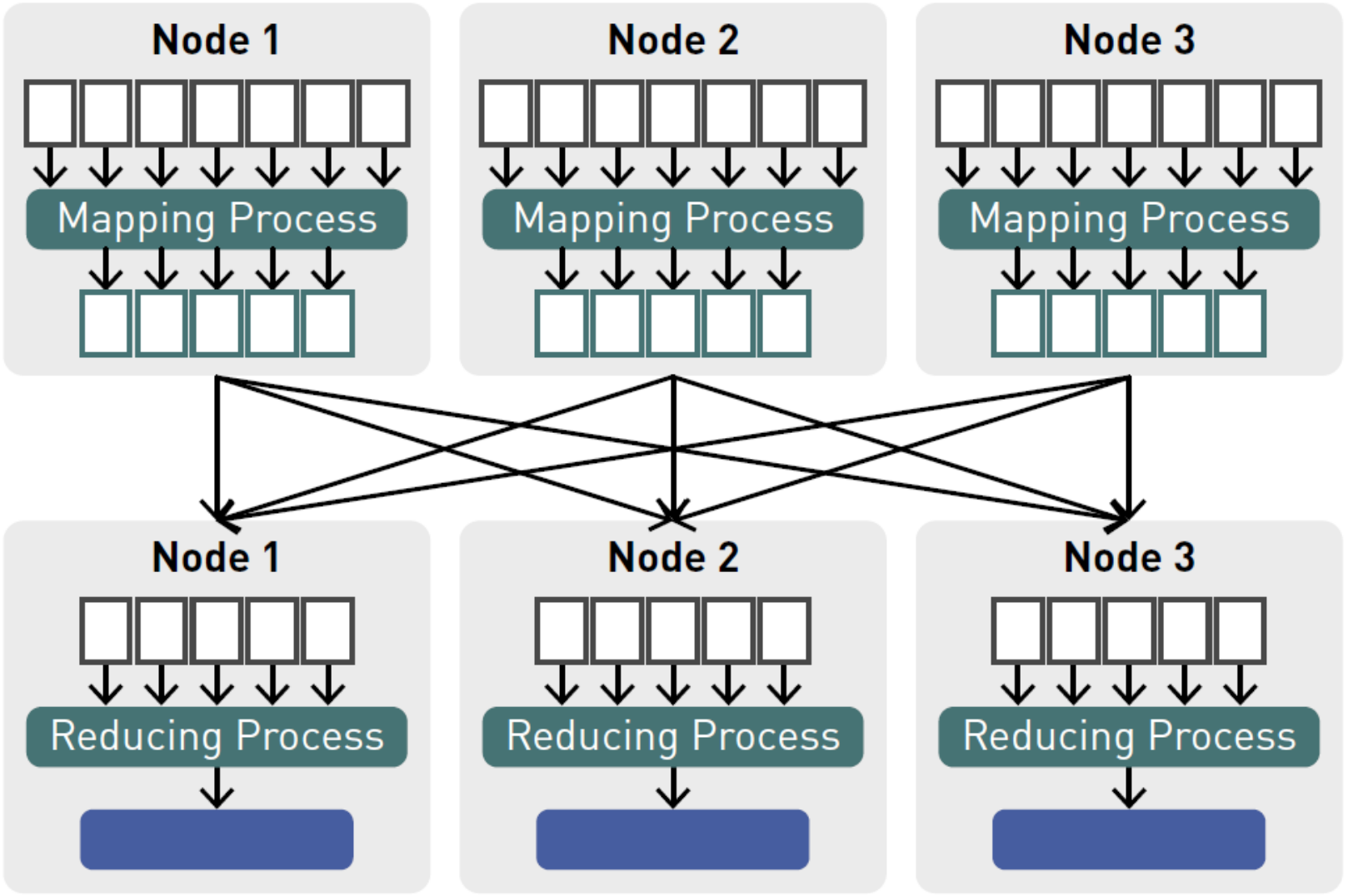


Map-Reduce (1)

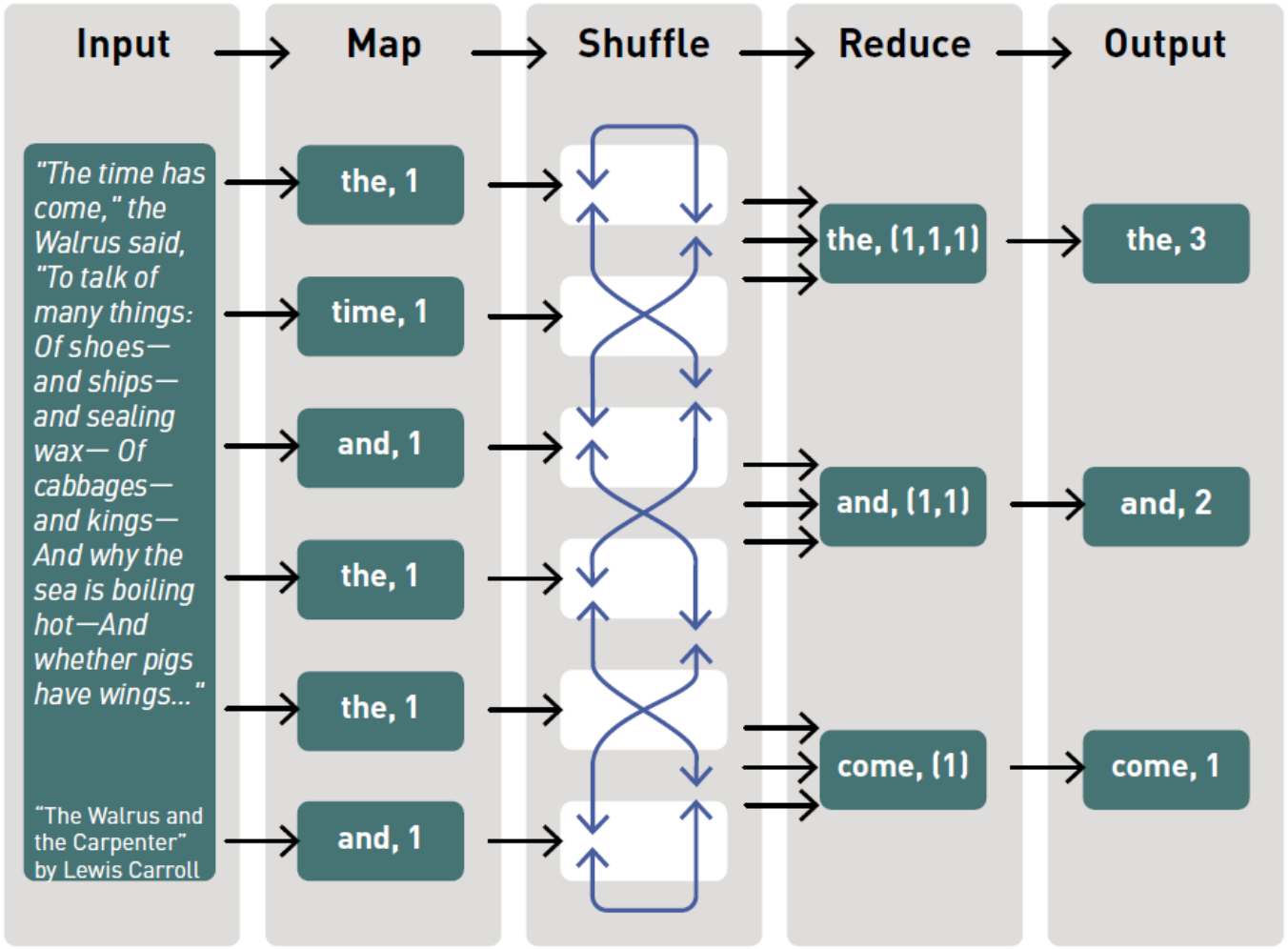
- **Được xây dựng dựa trên mô hình MapReduce của Google.**

- **Cơ chế hoạt động**
  - Input data được đọc từ HDFS
  - 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.

Map-Reduce (2)

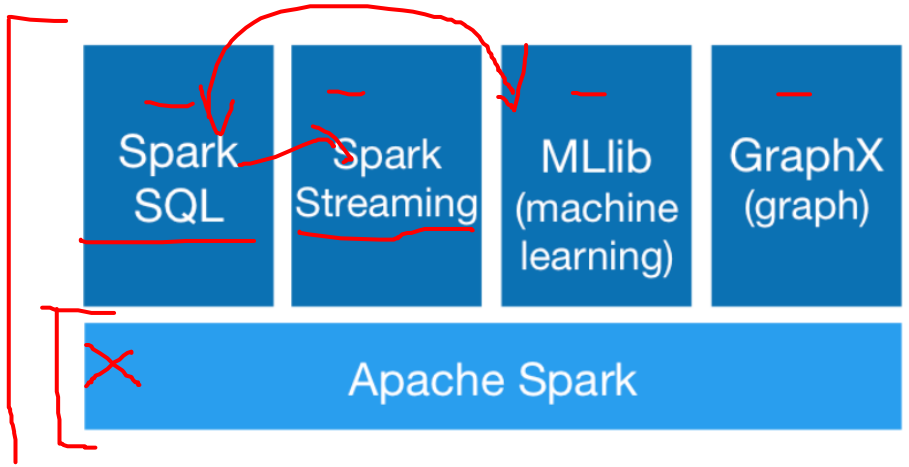

Node 1
Mapping Process
Node 2
Mapping Process
Node 3
Mapping Process
Node 1
Reducing Process
Node 2
Reducing Process
Node 3
Reducing Process

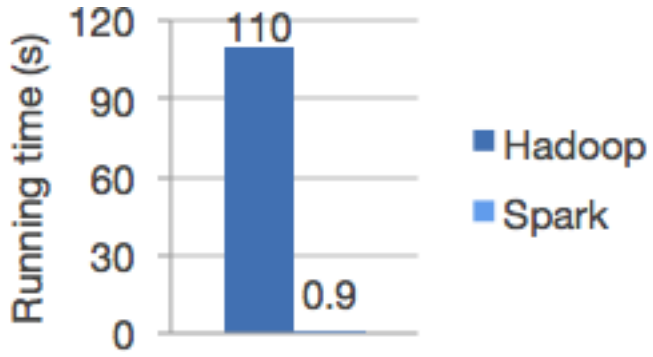
Map-Reduce – Word count



Apache Spark: An Engine for Large-Scale Data Processing (1)

- Engine xử lý dữ liệu in-memory, phân tán.
- Ban đầu là nghiên cứu của Berkeley AMPLab vào năm 2009.
  http://people.csail.mit.edu/matei/papers/2010/hotcloud_spark.pdf
  http://people.csail.mit.edu/matei/papers/2012/nsdi_spark.pdf
- Khắc phục nhược điểm của Hadoop MapReduce
- Hỗ trợ nhiều loại workload trong 1 engine:
  - Batch Processing (Spark Core, Spark SQL)
  - Machine Learning (MLlib)
  - Streaming (Spark Streaming)
  - Graph (GraphX)
  - Querying Structured Data (Spark SQL)





Apache Spark: An Engine for Large-Scale Data Processing (2)

- Là Opensource, viết bằng Scala và chạy trên JVM
- Dựa trên mô hình lập trình Map-Reduce
- Hỗ trợ nhiều loại ngôn ngữ lập trình:
  - **Scala (native)**
  - Python
  - Java
  - R
- Có thể chạy trên cluster hoặc chạy trên local.

Apache Spark: An Engine for Large-Scale Data Processing (3)

- **Hỗ trợ nhiều loại Resource Manager**
  - Hadoop YARN
  - Apache Mesos
  - Kubernetes
  - Spark Standalone

Apache Spark: An Engine for Large-Scale Data Processing (4)

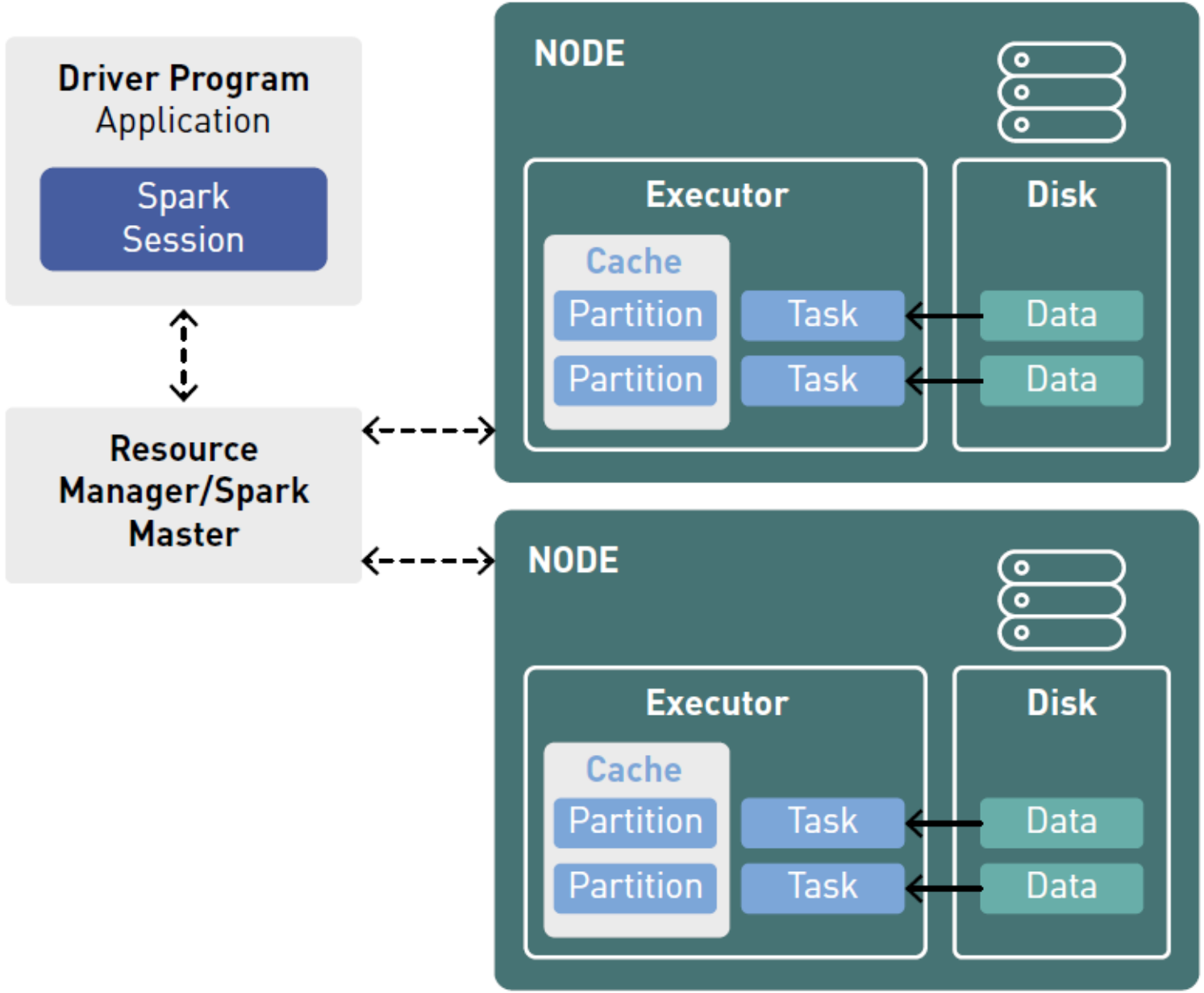
- **Tính năng, ưu điểm của Spark**

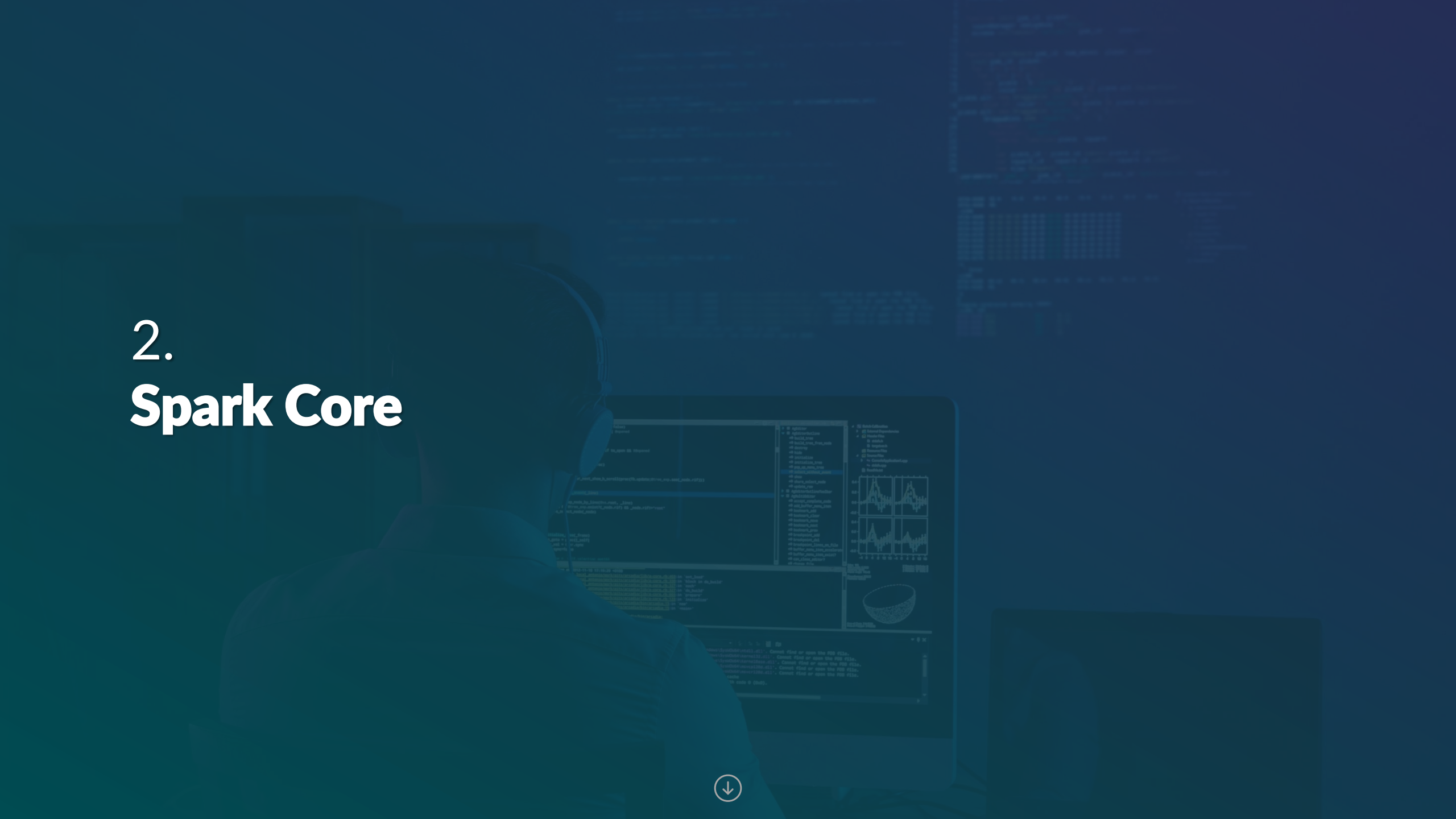
  - **Advanced Analytics**: Spark không chỉ hỗ trợ "Map" và "Reduce ", nó còn hỗ trợ Spark truy vấn SQL, Streaming data, Machine learning (ML) và các thuật toán xử lý đồ thị đóng vai trò như một bộ công cụ phân tích dữ liệu cực kì mạnh mẽ.

  - **Speed**: Spark giúp chạy một ứng dụng với tốc độ rất nhanh. So với Hadoop cluster, Spark Application nến chạy trên bộ nhớ nhanh hơn tới 100 lần và nhanh hơn 10 lần khi chạy trên đĩa. Điều này có được nhờ giảm số lượng các hoạt động đọc / ghi vào ổ đĩa.

Spark Application

- Bao gồm:
  - Driver Program: Chứa Spark session. Vai trò khởi chạy ứng dụng, điều phối code/ logic xử lý phân tán.
  - Executor: chịu trách nhiệm thực hiện các tính toán các logic nhận từ Driver.
  - **Resource Manager**: Quản lý và điều phối tài nguyên cho executor

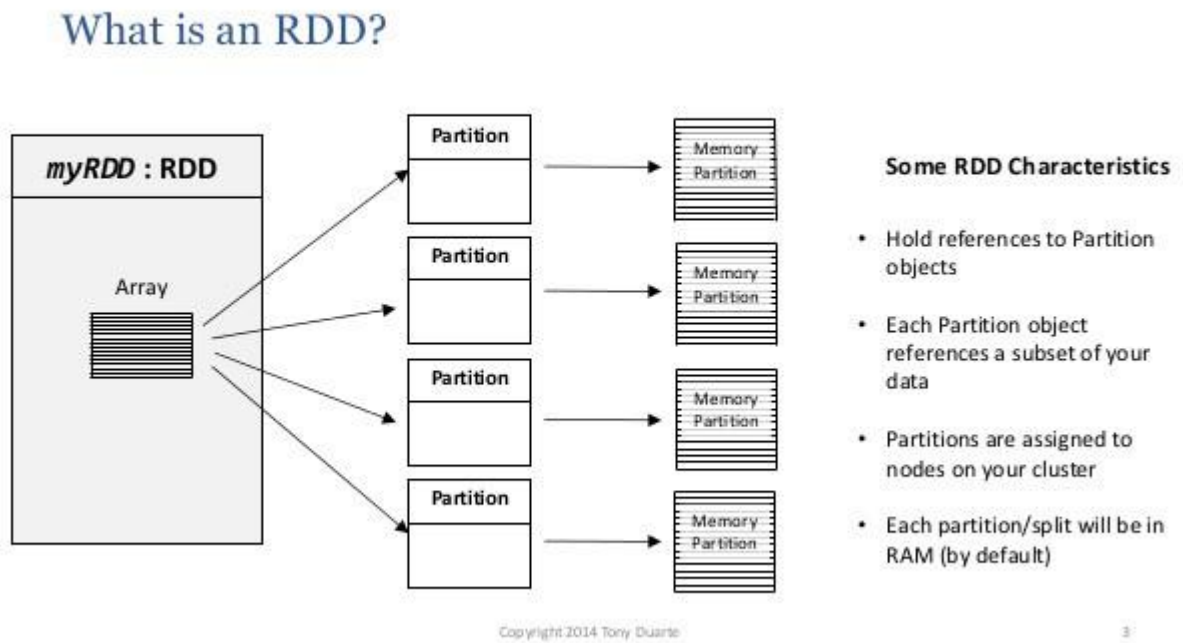




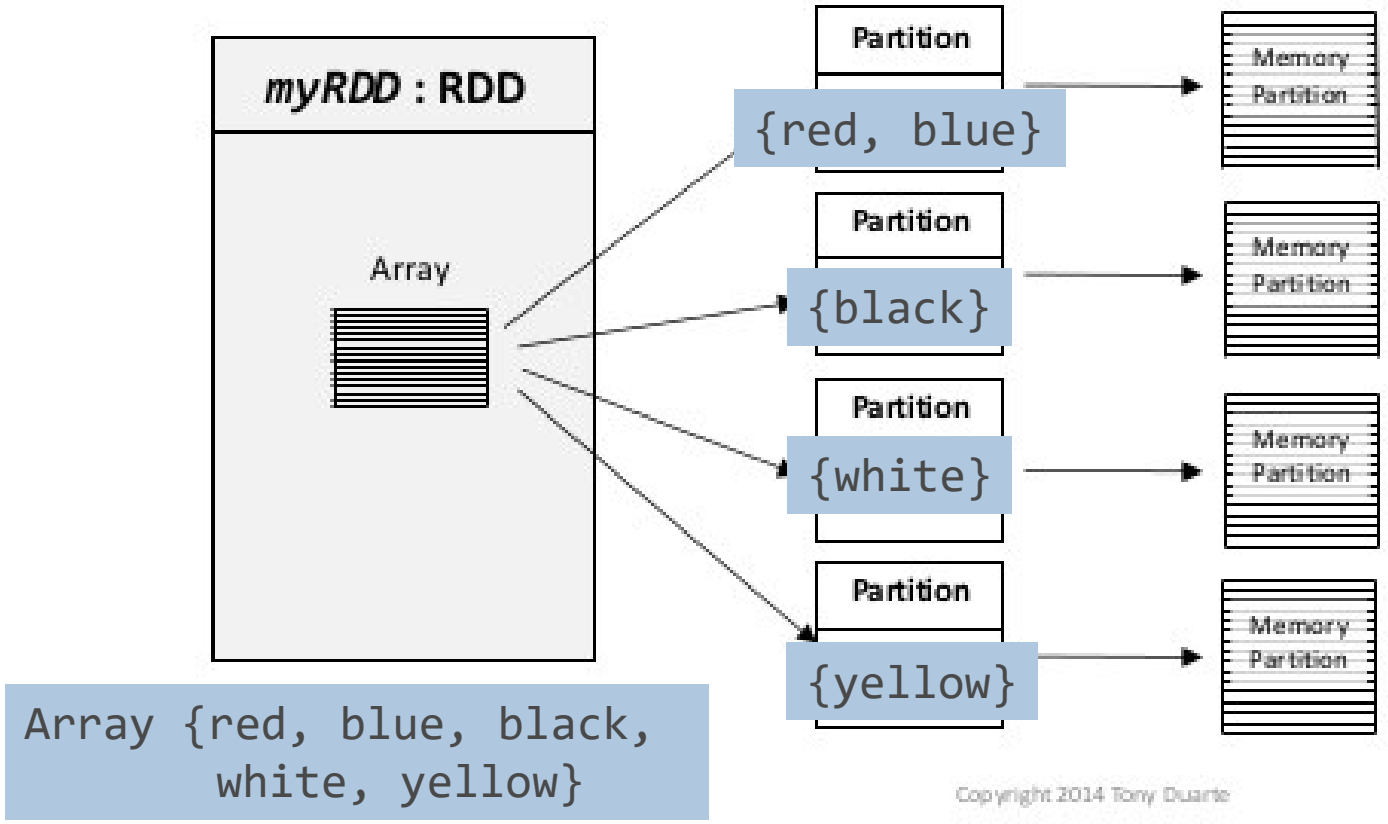
2.
Spark Core

Resilient Distributed Datasets (RDDs) (1)

- RDDs là thành phần core của Spark
- Tập dữ liệu có thể chứa bất kì object nào, phân tán
- RDD =
  - **Resilient**: Dữ liệu có thể tạo lại nếu bị mất.
  - **Distributed**: Dữ liệu được phân tán trên cụm
  - **Dataset**: Tập dữ liệu được tạo từ file hoặc thông qua coding.



Resilient Distributed Datasets (RDDs) (2)



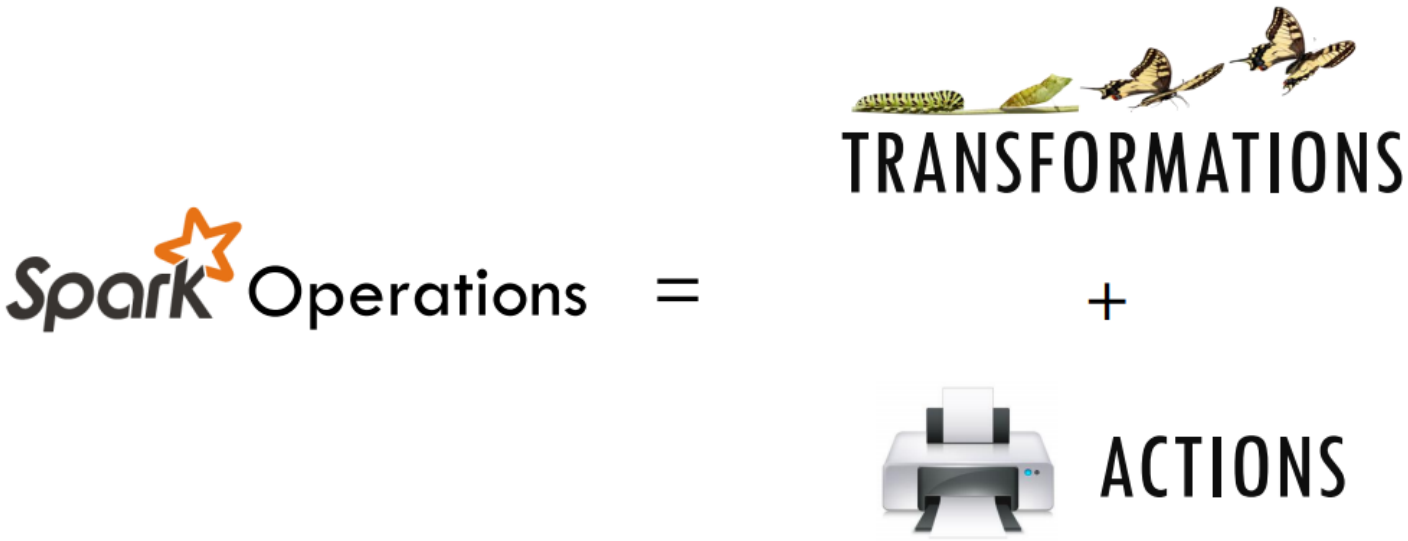
Some RDD Characteristics

- Hold references to Partition objects
- Each Partition object references a subset of your data
- Partitions are assigned to nodes on your cluster
- Each partition/split will be in RAM (by default)

Spark Operations

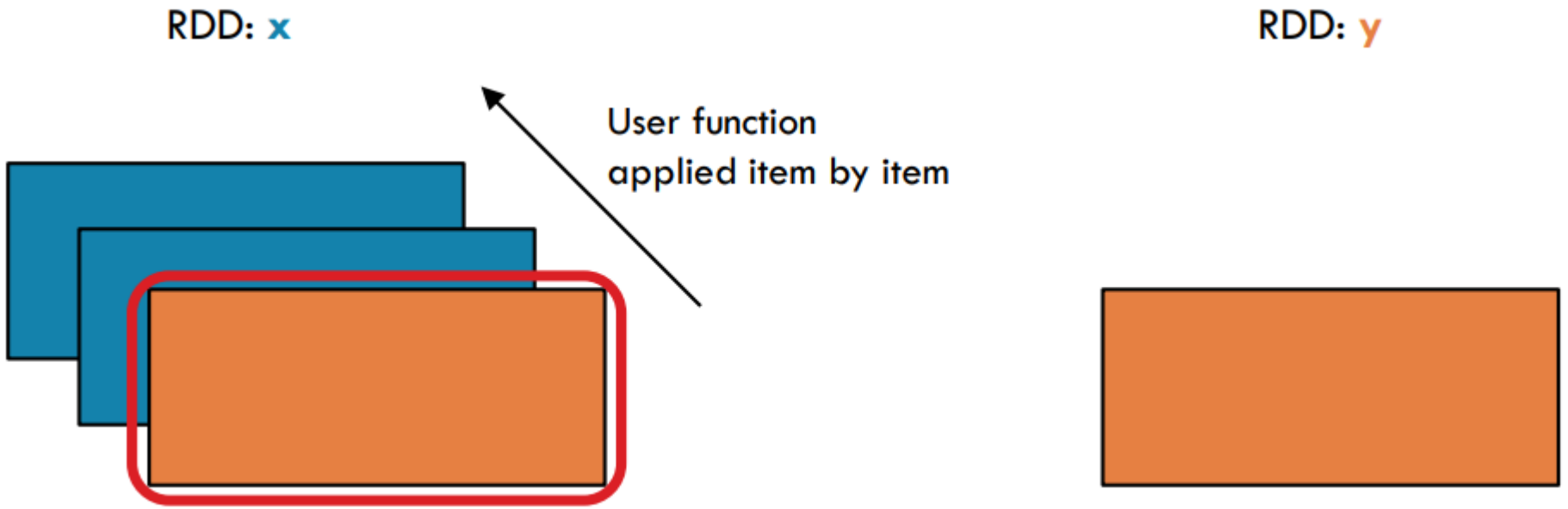
- Có 2 loại Operations
  - **Actions:** Trả về dữ liệu cho driver hoặc ghi dữ liệu
  - **Transformations**: Biến đổi RDD thành 1 RDD khác

- Action là sự kiện trigger để bắt đầu thực thi các Transformations



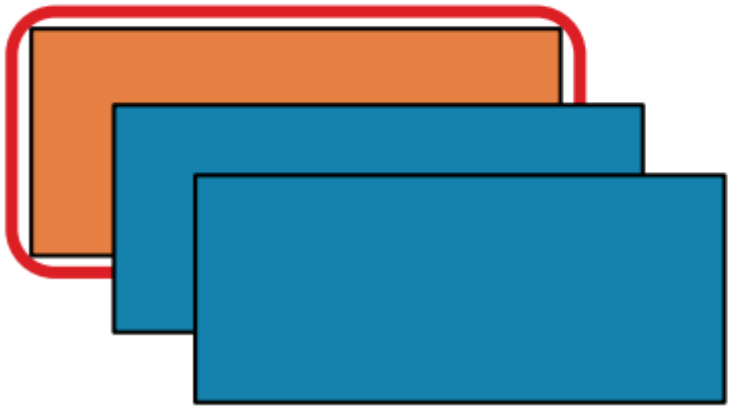
Map Transformation (1)

- `map(function)`
- Tạo ra RDD mới bằng cách apply *function* vào tất cả bản ghi trong RDD ban đầu

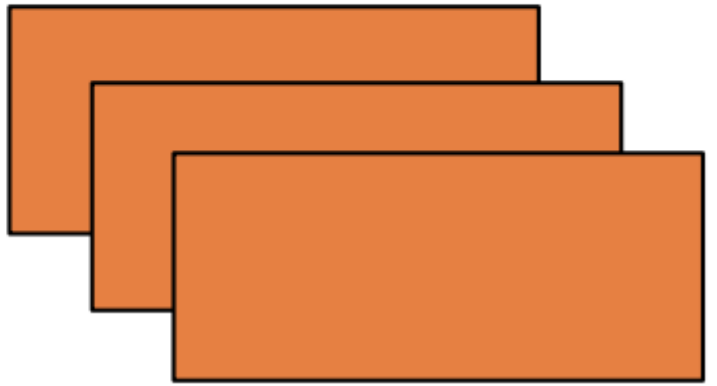


Map Transformation (2)

RDD: x



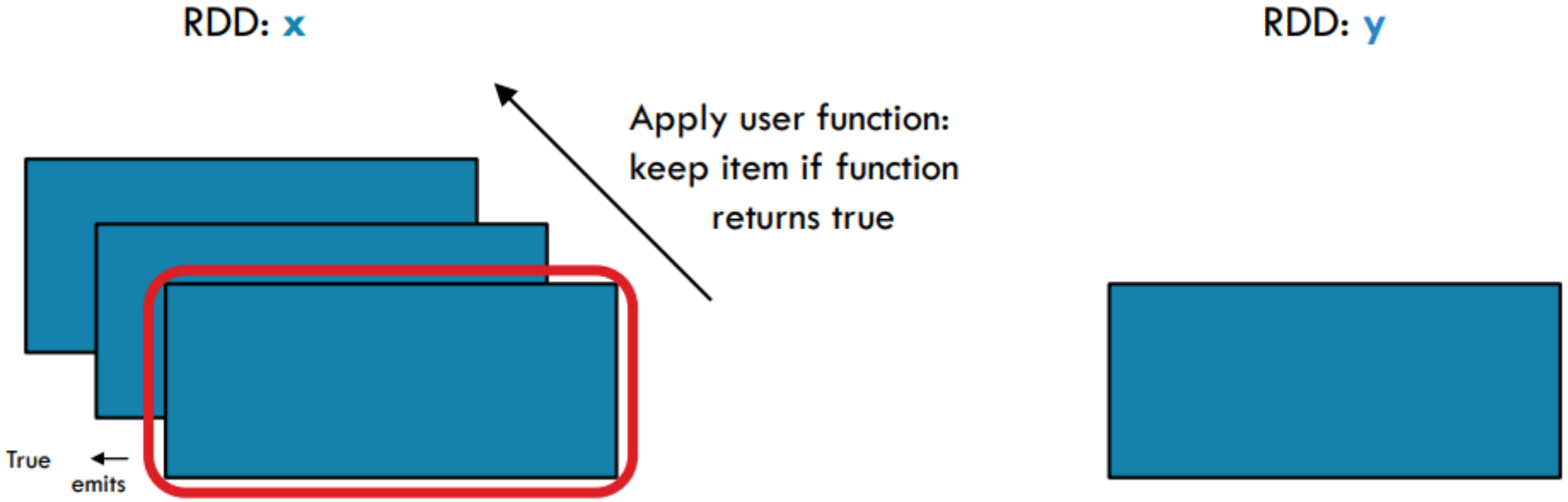
RDD: y



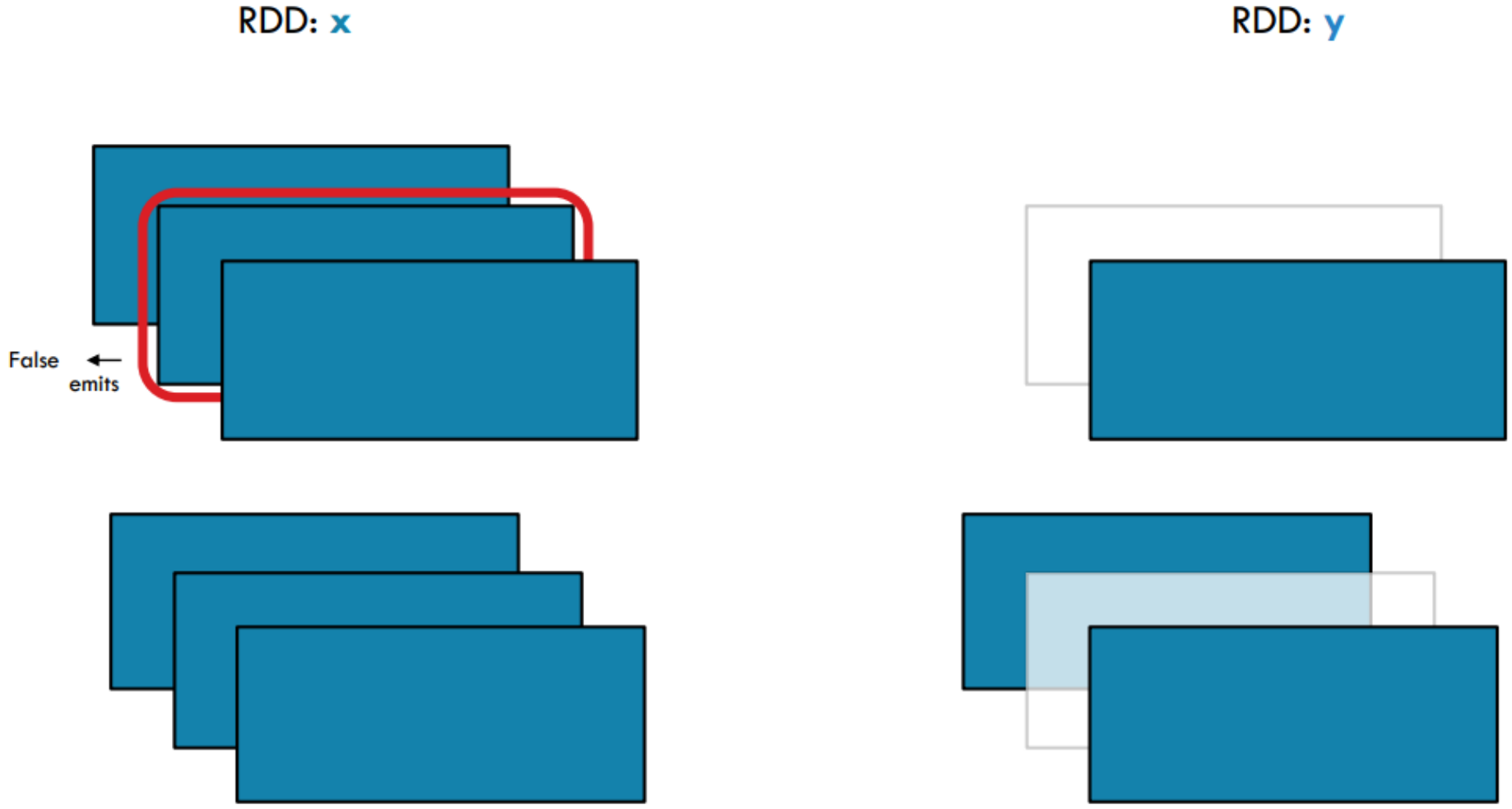
Filter Transformation (1)

- `filter(`***`function`***`)`

- Tạo ra RDD mới bằng cách apply ***`function`*** vào tất cả bản ghi trong RDD ban đầu và giữ lại các bản ghi mà ***`function`*** trả về *true*



Filter Transformation (2)

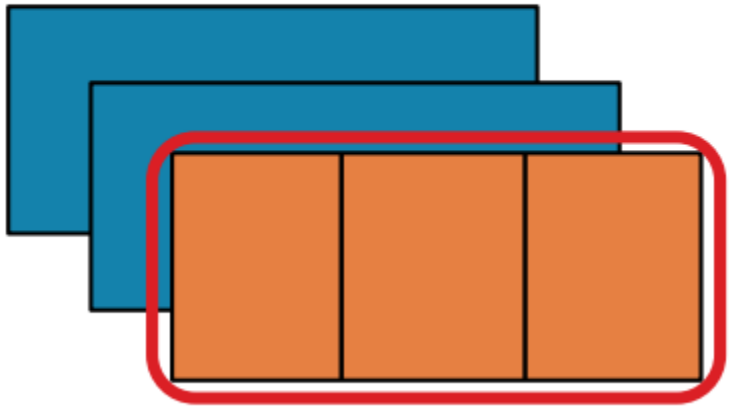


Flatmap Transformation (2)

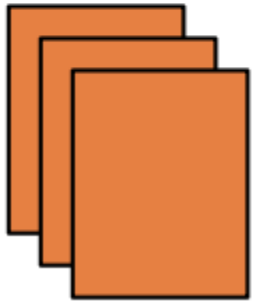
- `flatMap(`***`function`***`)`

- Tạo ra RDD mới bằng cách apply ***`function`*** vào tất cả bản ghi trong RDD ban đầu và flat kết quả.

RDD: x

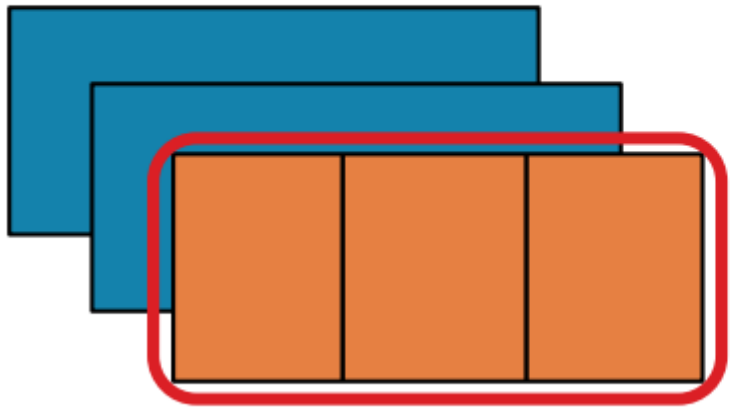


RDD: y

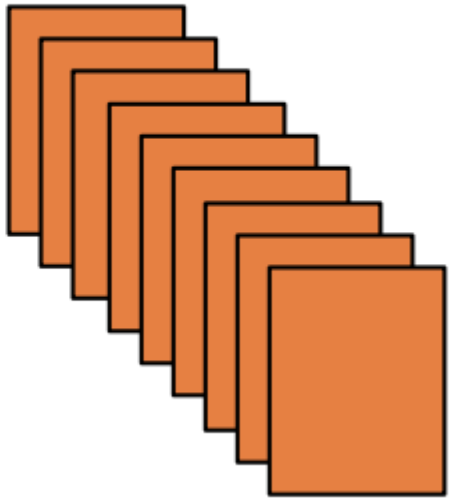


Flatmap Transformation (2)

RDD: x

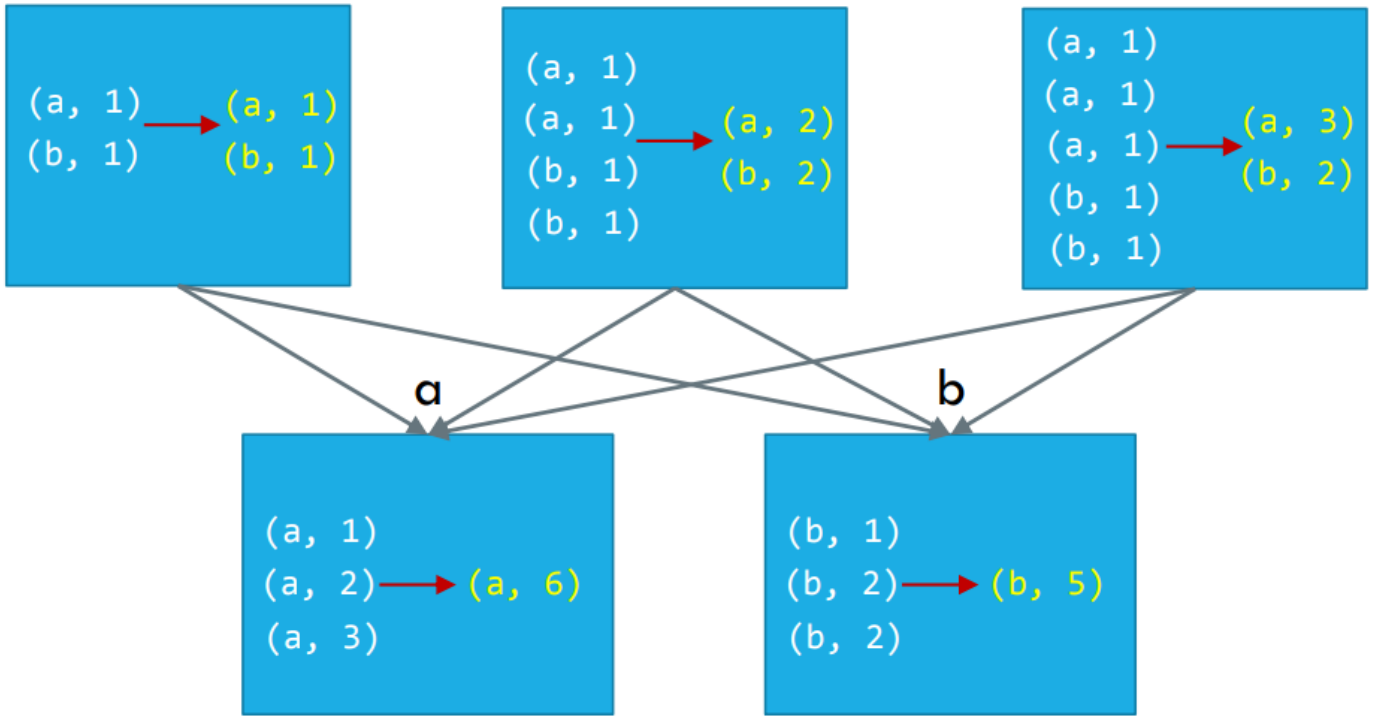


RDD: y



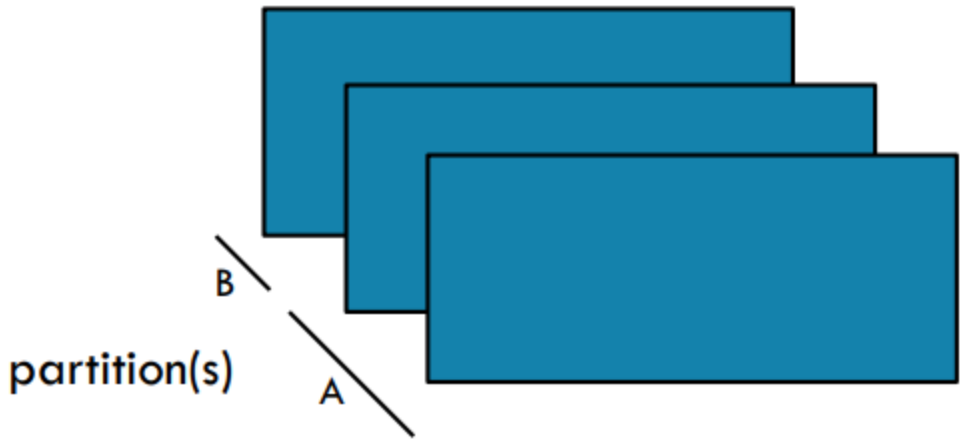
ReduceByKey Transformation (2)

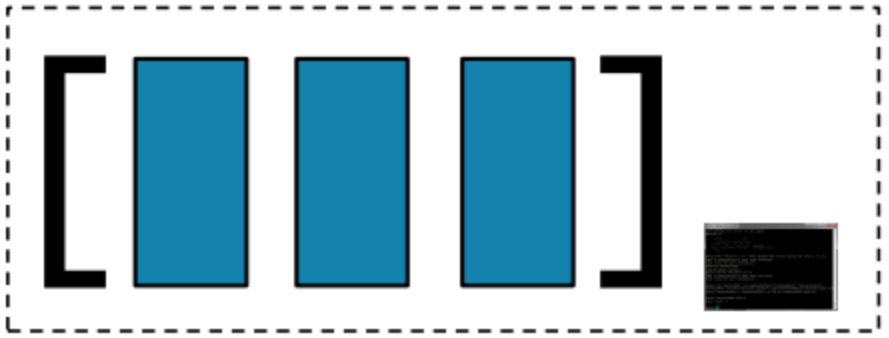
- Áp dụng với `PairRDD (RDD[Tuple2])`
- Nhóm các bản ghi có cùng key vào vào thực hiện tính toán



Collect Action

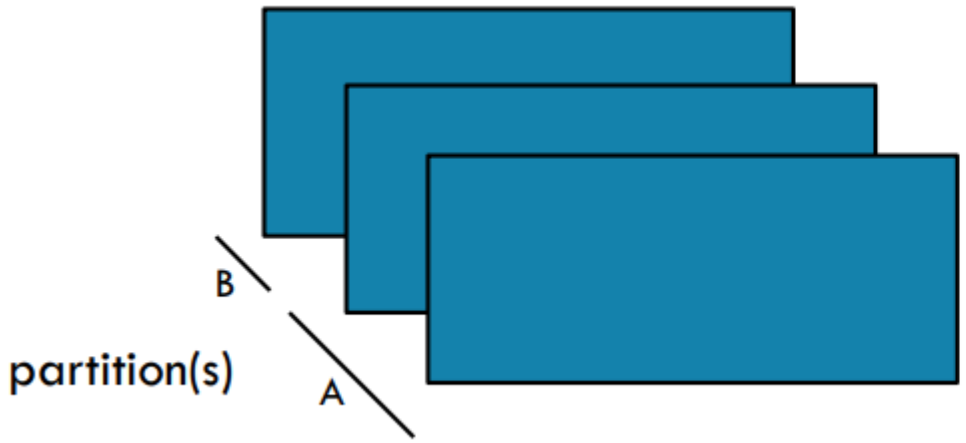
- Trả về toàn bộ dữ liệu từ RDD về Driver ở dạng List

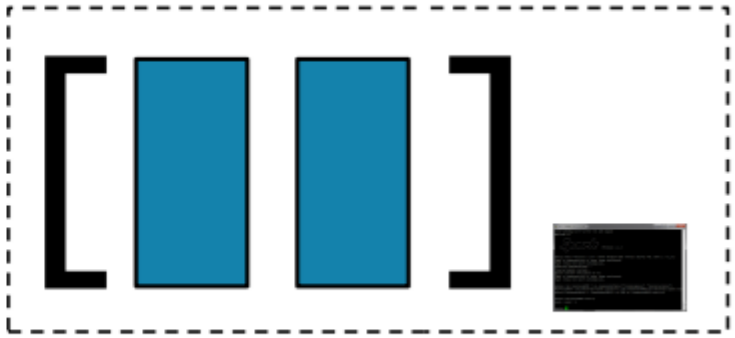




Take Action

- `take(`***n***`)`
- Trả về ***n*** bản ghi từ RDD về Driver ở dạng List





SaveAsTextFile Action

- `saveAsTextFile(`***`path`***`)`
- Ghi dữ liệu ra file

3.
Cài đặt Spark

Cài đặt Spark – local mode (1)

- cd /usr/local
- wget https://mirror.downloadvn.com/apache/spark/spark-3.1.1/spark-3.1.1-bin-hadoop3.2.tgz
- tar -xvf spark-3.1.1-bin-hadoop3.2.tgz
- mv spark-3.1.1-bin-hadoop3.2/ spark
- cd /usr/local/spark

- **Thêm SPARK_HOME vào biến môi trường.**
  - nano ~/.bashrc
  - Thêm: export SPARK_HOME=/usr/local/spark
  - source ~/.bashrc

Cài đặt Spark – local mode (2)

- $SPARK_HOME/bin/spark-shell
- SparkUI: http://master:4040

```
[hduser@master spark]$ $SPARK_HOME/bin/spark-shell
bash: /bin/spark-shell: No such file or directory
[hduser@master spark]$ source ~/.bashrc
[hduser@master spark]$ $SPARK_HOME/bin/spark-shell
21/04/12 13:17:25 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://master:4040
Spark context available as 'sc' (master = local[*], app id = local-1618247855752).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.1.1
      /_/

Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 1.8.0_41)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

Cài đặt Spark – cluster mode (1)

- cp conf/spark-env.sh.template conf/spark-env.sh
- Thêm HADOOP_CONF_DIR vào môi trường của SPARK.
  - File: $SPARK_HOME/conf/spark-env.sh
  - export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop/

```
# Options for native BLAS, like Intel MKL, OpenBLAS, and so on.
# You might get better performance to enable these options if using native BLAS (see SPARK-21305).
# - MKL_NUM_THREADS=1        Disable multi-threading of Intel MKL
# - OPENBLAS_NUM_THREADS=1   Disable multi-threading of OpenBLAS
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop/
```

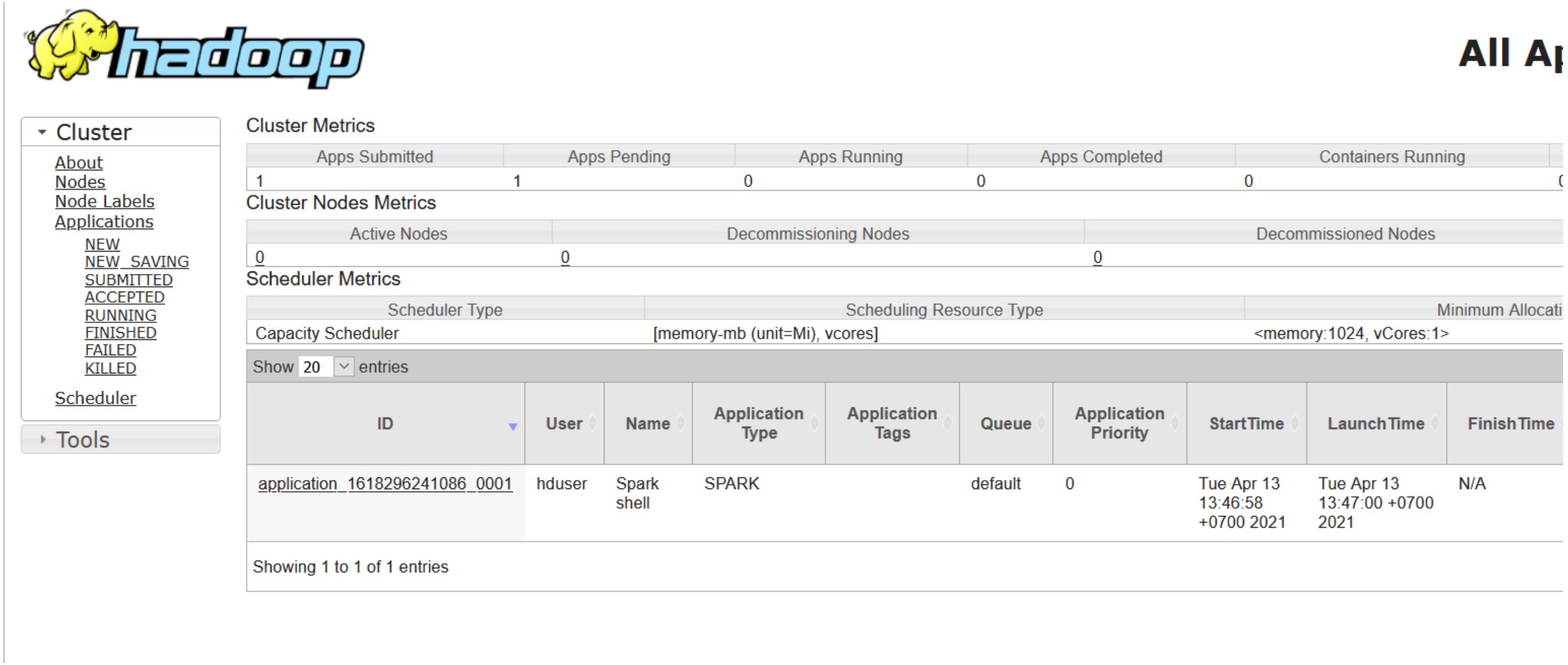
Cài đặt Spark – cluster mode (2)

- $SPARK_HOME/bin/spark-shell --master yarn
- RMUI: http://master:8088
- SparkUI: http://master:4040

```
[hduser@master spark]$ ./bin/spark-shell --master yarn
21/04/13 02:46:37 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
21/04/13 02:46:52 WARN Client: Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
Spark context Web UI available at http://master:4040
Spark context available as 'sc' (master = yarn, app id = application_1618296241086_0001).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.1.1
      /_/

Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 1.8.0_41)
Type in expressions to have them evaluated.
Type :help for more information.
```

Cài đặt Spark – `cluster mode` (3)



4.
Thực hành với Spark

Kiểm tra SparkSession

- spark

```
scala> spark
res0: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@6f77917c
```

- sc

```
scala> sc
res1: org.apache.spark.SparkContext = org.apache.spark.SparkContext@586be63c
```

Tạo RDD từ List Data

```
> val fruits = Array("Peach", "Orange", "Apple")
    > val rdd =  sc.parallelize(fruits)
    > val rdd =  spark.sparkContext.parallelize(fruits)

> rdd.getNumPartitions
> val rdd =  sc.parallelize(fruits, 2)
> rdd.getNumPartitions
```

Đọc ghi file

Ghi dữ liệu:

```
> rdd.saveAsTextFile("path")
```

Đọc dữ liệu:

```
> sc.textFile("path")
```

Đọc ghi file

- `hdfs dfs -mkdir -p /aiacad/spark/inp`
- `hdfs dfs -mkdir -p /aiacad/spark/out`

- `hdfs dfs -put $SPARK_HOME/LICENSE /aiacad/spark/inp`

```
> val fruits = Array("Peach", "Orange", "Apple")
> val rdd =  sc.parallelize(fruits)
> rdd.saveAsTextFile("/aiacad/spark/out/fruits")
```

Actions

```
> rdd.count()
> rdd.take(2)
> rdd.take(2).foreach(println)
> rdd.collect.foreach(println)
```

Transformations

```
> rdd.map(a=>a.length).collect
> rdd.map(a=>a.toUpperCase).take(3).foreach(println)

> val sentence = Array("Hello, my name is Thanh")
> val rdd = sc.parallelize(sentence)
> rdd.flatMap(a=>a.split(" ")).count
> rdd.flatMap(a=>a.split(" ")).take(10).foreach(println)
```

WordCount

```
> val rddFile = sc.textFile("/aiacad/spark/inp")
> rddFile.flatMap(a=>a.split(" ")) //Tách RDD theo dòng ban đầu thành RDD theo từng ký tự, cách nhau bởi dấu cách
>         .map(a=>(a, 1)) //Đưa dữ liệu từng ký tự đơn thành dạng tuple2, mỗi từ ban đầu coi như xuất hiện 1 lần, key là từ, value là số lần xuất hiện
>         .reduceByKey((a, b) => (a+b)) // cộng số lần xuất hiện nếu của cùng 1 từ
>         .collect
```