

Buổi 1:

Xử lý dữ liệu lớn với Apache Spark

- Các mô hình xử lý dữ liệu lớn.
- Một số công nghệ trong lĩnh vực xử lý dữ liệu.
- Giới thiệu Apache Spark và các modules.
- Spark Core và lập trình thông qua RDD APIs.

Nguyễn Chí Thanh – Ban Quản trị Dữ liệu





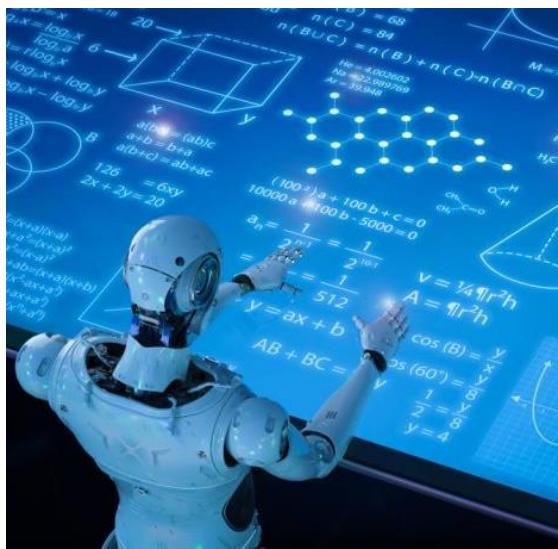
Nguyễn Chí Thành
Chuyên viên kiến trúc dữ liệu
Ban Quản trị Dữ liệu
Blog: <https://karcuta.medium.com>

ABOUT ME

- Trên 3 năm kinh nghiệm trong lĩnh vực Big Data Engineering.
- Tham gia xây dựng và triển khai hệ thống vBI, Viettel Data Lake cho VTT.
- 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 Ecosystem: Hdfs, Spark, Kafka, Hive...

1. Các mô hình xử lý dữ liệu





Big Data Internet of Things
Machine Learning
Chuyển đổi số
Artificial Intelligence
Cách mạng công nghiệp 4.0

"Nếu muốn làm AI,
ML chúng ta phải có
Big Data

Dữ liệu trên Internet



2.2 tỷ
người dùng

mỗi 60s trên Facebook



520k comments



293k status



136k photos



4,5 tỷ nội dung được chia sẻ mỗi ngày



167 triệu paying subs

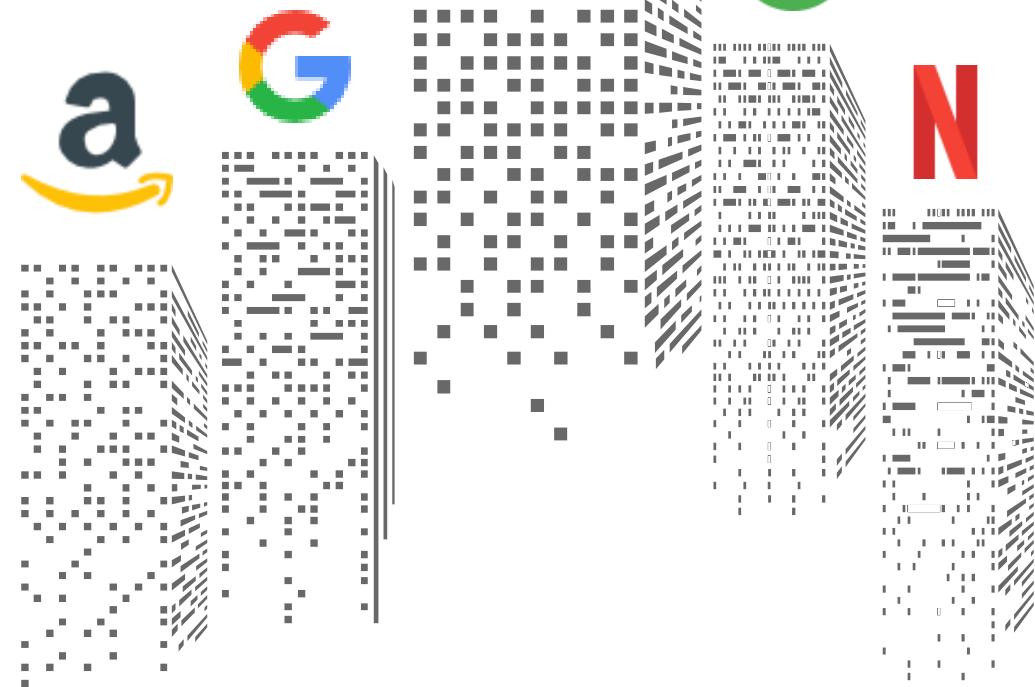
190 quốc gia



250+ triệu giờ videos

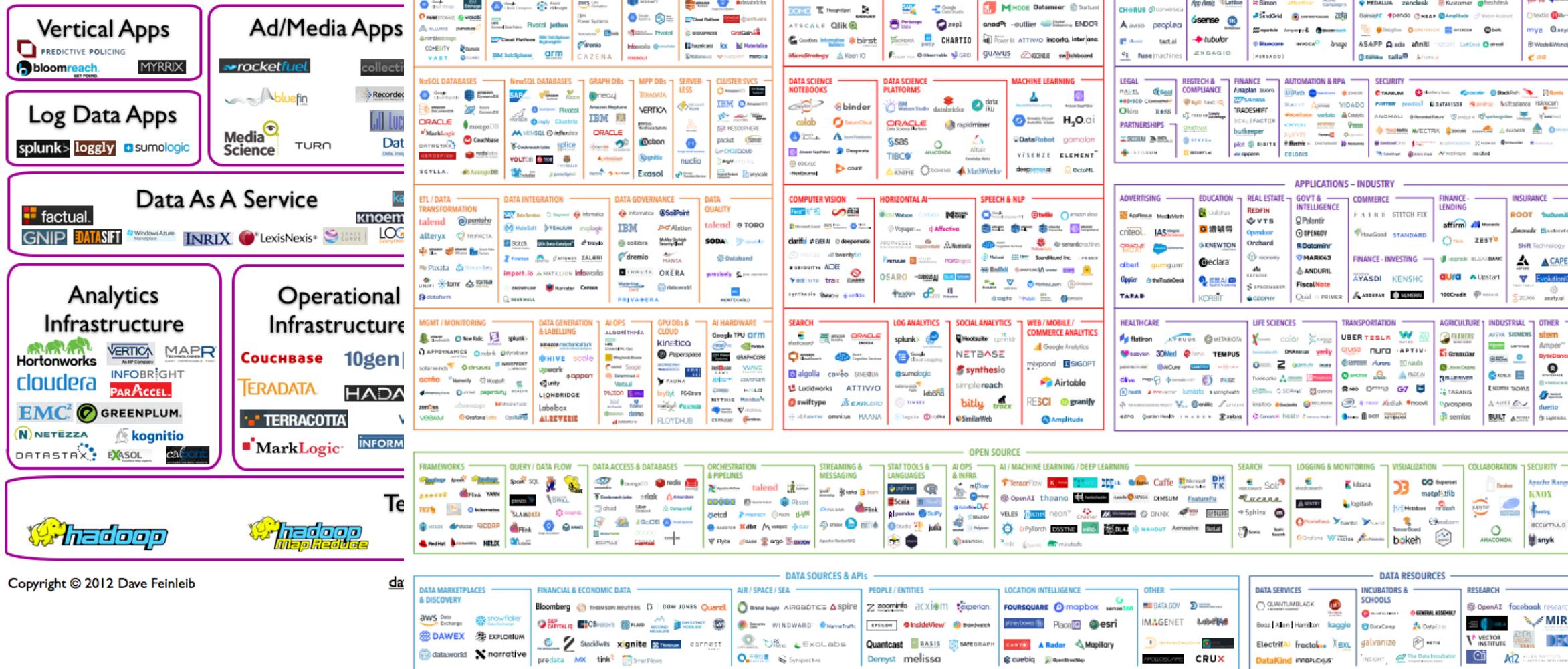


đưa ra recommend real time, khi user "nhấn" play



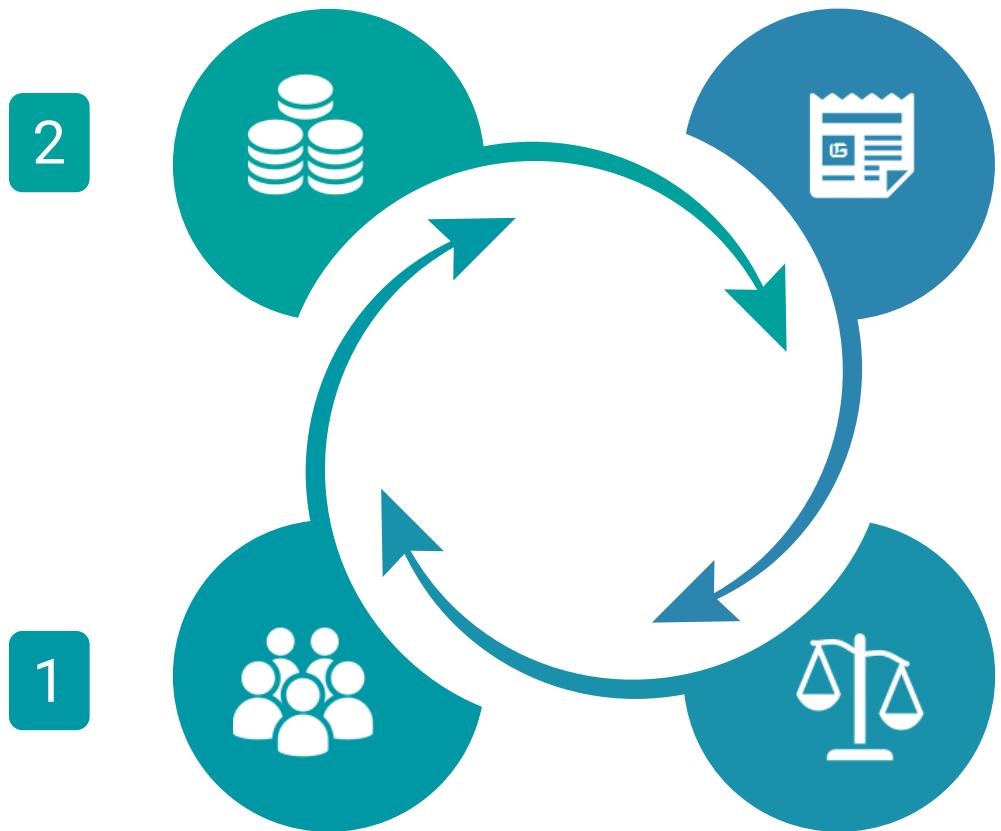
Sự phát triển của công nghệ Big Data

Big Data



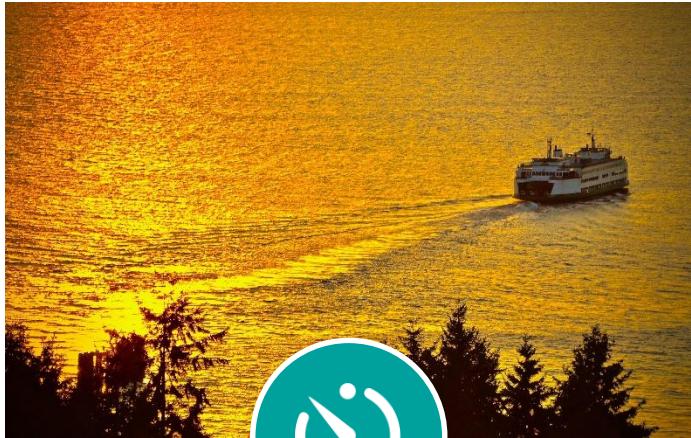
Vòng tuần hoàn dữ liệu

- CƠ SỞ DỮ LIỆU**
Các bản ghi logs của khách hàng được lưu trữ, phân loại trong các hệ thống CSDL của doanh nghiệp
- DỮ LIỆU NGƯỜI DÙNG**
Người dùng sử dụng các sản phẩm, dịch vụ của doanh nghiệp, tạo ra các dữ liệu



- XÂY DỰNG DASHBOARD/MÔ HÌNH DỰ ĐOÁN, DỰ BÁO**
Thực hiện phân tích, tổng hợp số liệu để tạo ra các báo cáo kinh doanh, các mô hình dự đoán, dự báo hành vi, xu hướng khách hàng
- RA QUYẾT ĐỊNH KINH DOANH**
Số liệu giúp người chỉ huy có được cái nhìn đúng nhất để đưa ra quyết định

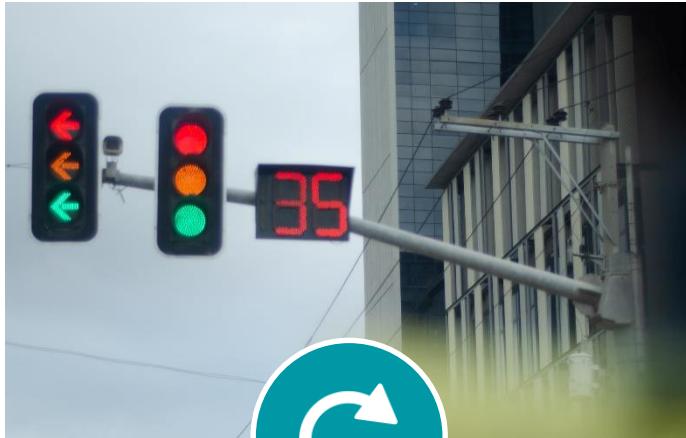
Các mô hình xử lý dữ liệu



BATCH PROCESSING

Xử lý một lượng lớn dữ liệu theo lô, có thể dựa trên chu kỳ thời gian nhất định.

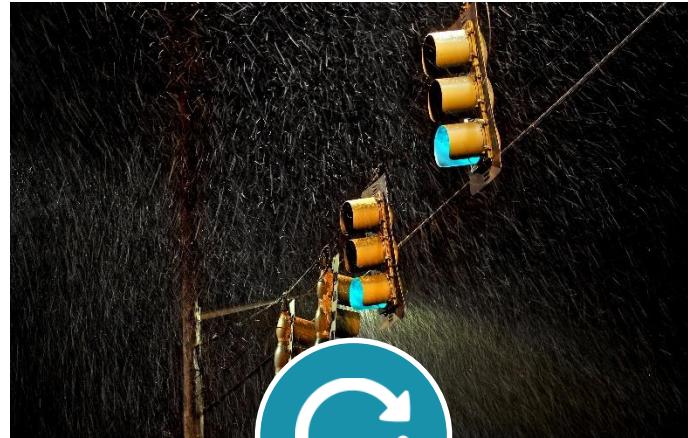
01



NEAR REAL-TIME PROCESSING

Xử lý dữ liệu theo lô, dựa trên các khoảng thời gian định sẵn nhỏ, khoảng vài phút.

02



REAL-TIME PROCESSING

Xử lý dữ liệu dữ liệu trên các khoảng thời gian cực nhỏ hoặc xử lý từng sự kiện theo thời gian thực.

03

Batch Processing



Xử lý theo lô

Hệ thống ghi nhận các giao dịch của khách hàng, nhưng sẽ gom nhóm tới ngưỡng nhất định mới xử lý.



Dữ liệu theo chu kỳ

Mô hình batch processing rất hiệu quả với các dữ liệu có chu kỳ như tài chính, tích lũy điểm thưởng



Lượng lớn dữ liệu

Có thể lên đến hàng triệu/ tỉ bản ghi mỗi lần xử lý.



Chi phí vận hành thấp

Không yêu cầu hệ thống vận hành liên tục, phù hợp với doanh nghiệp không có đội ngũ IT chuyên trách



Real time/ near-real time Processing

Độ trễ thấp

Dữ liệu được xử lý gần như tức thời hoặc độ trễ rất nhỏ, vài giây cho tới vài phút.



Vận hành phức tạp

Thông thường các hệ thống real time/ near realtime có yêu cầu cao hơn về phần cứng, phần mềm và đội ngũ nhân sự giám sát, vận hành và phát triển.



Xu hướng trong tương lai

Trước đây chỉ có 1 số hệ thống đặc thù yêu cầu như ATM, OCS thì nay sẽ có thêm nhiều hệ thống đòi hỏi xử lý độ trễ thấp như SmartCity, stream music/ videos...





THẢO LUẬN

- Các bài toán xử lý dữ liệu tại đơn vị.
- Cách thiết kế và tiếp cận

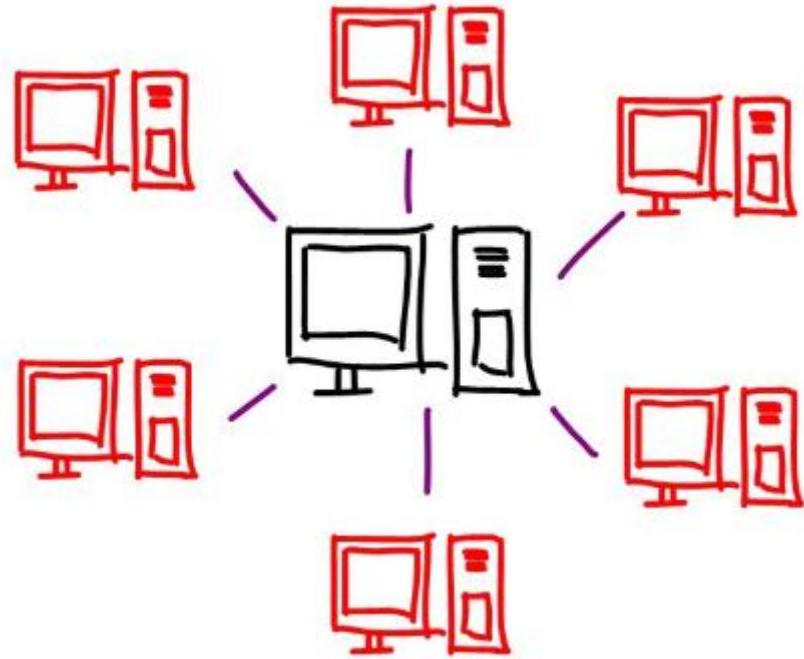
2.

Apache Spark Overview



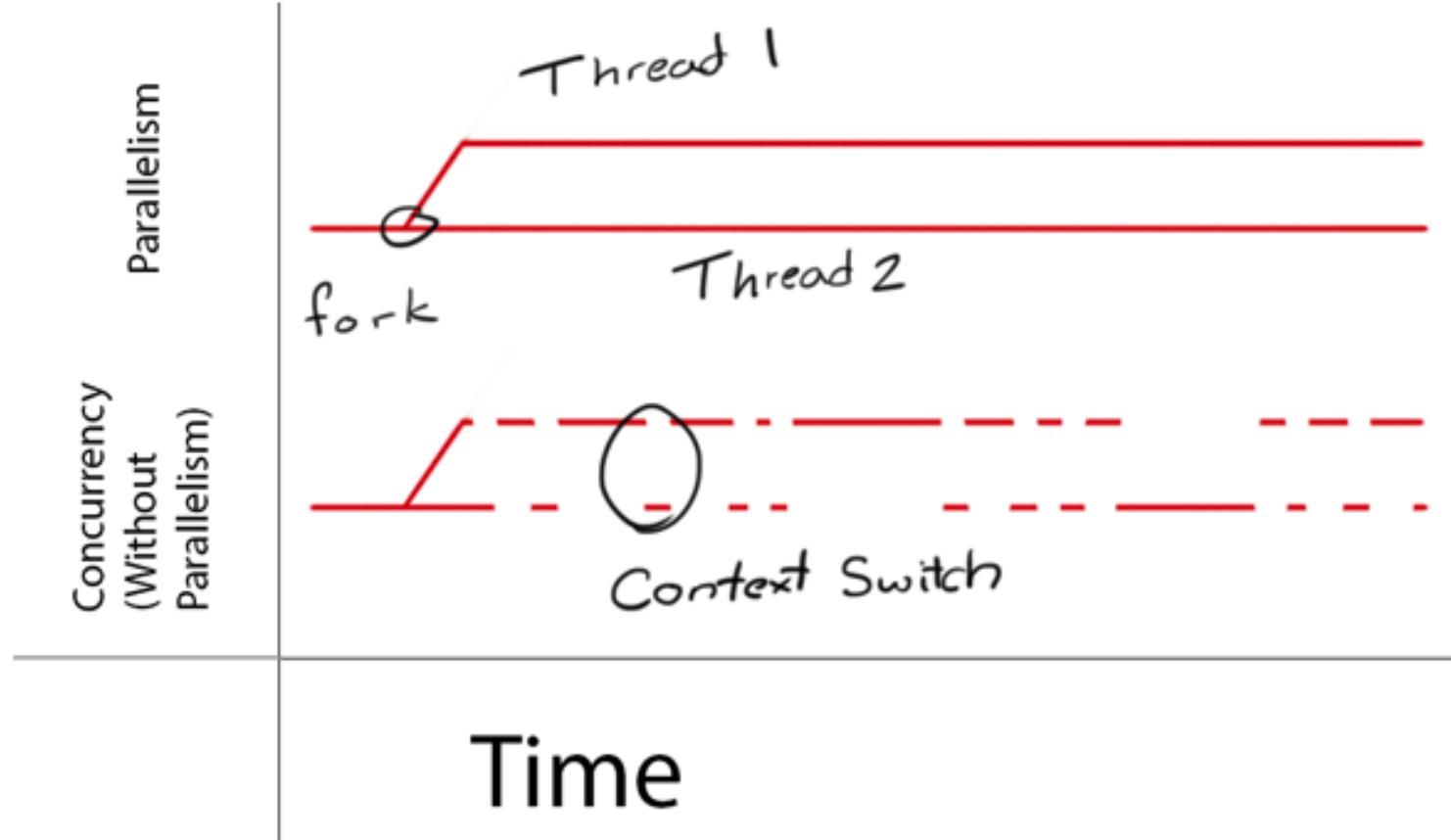
Hệ thống phân tán

- 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.



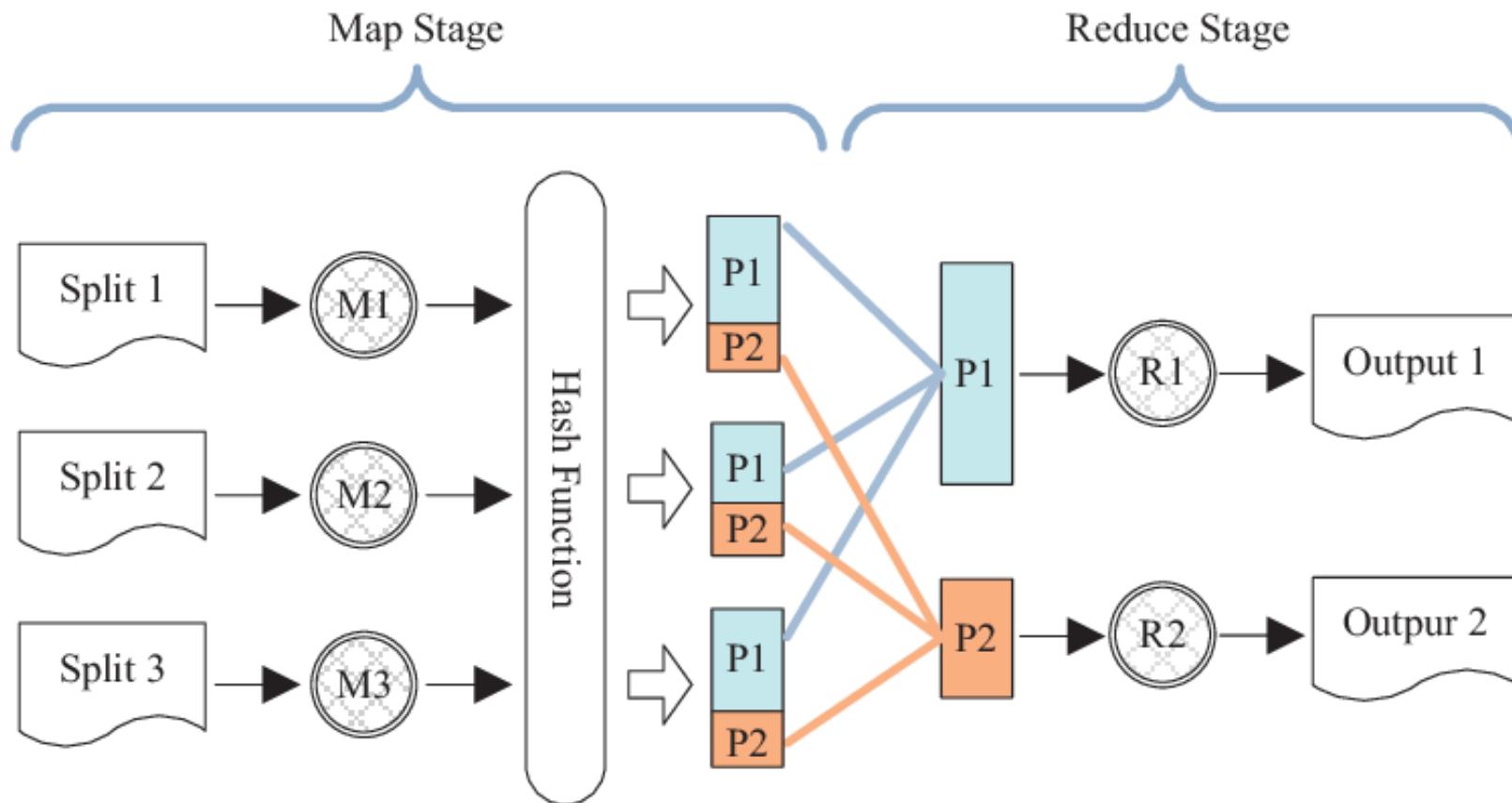
Concurrency & Parallelism

- 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



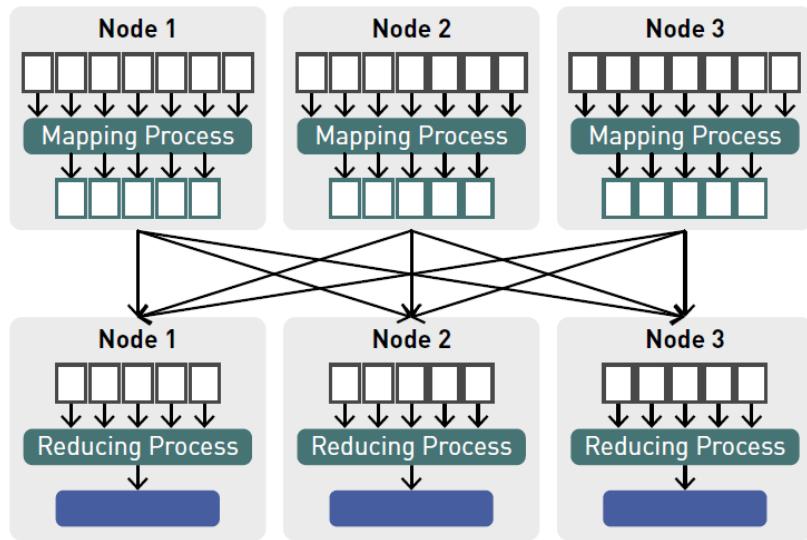
MapReduce Programming Model

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



Google Map-Reduce (1)

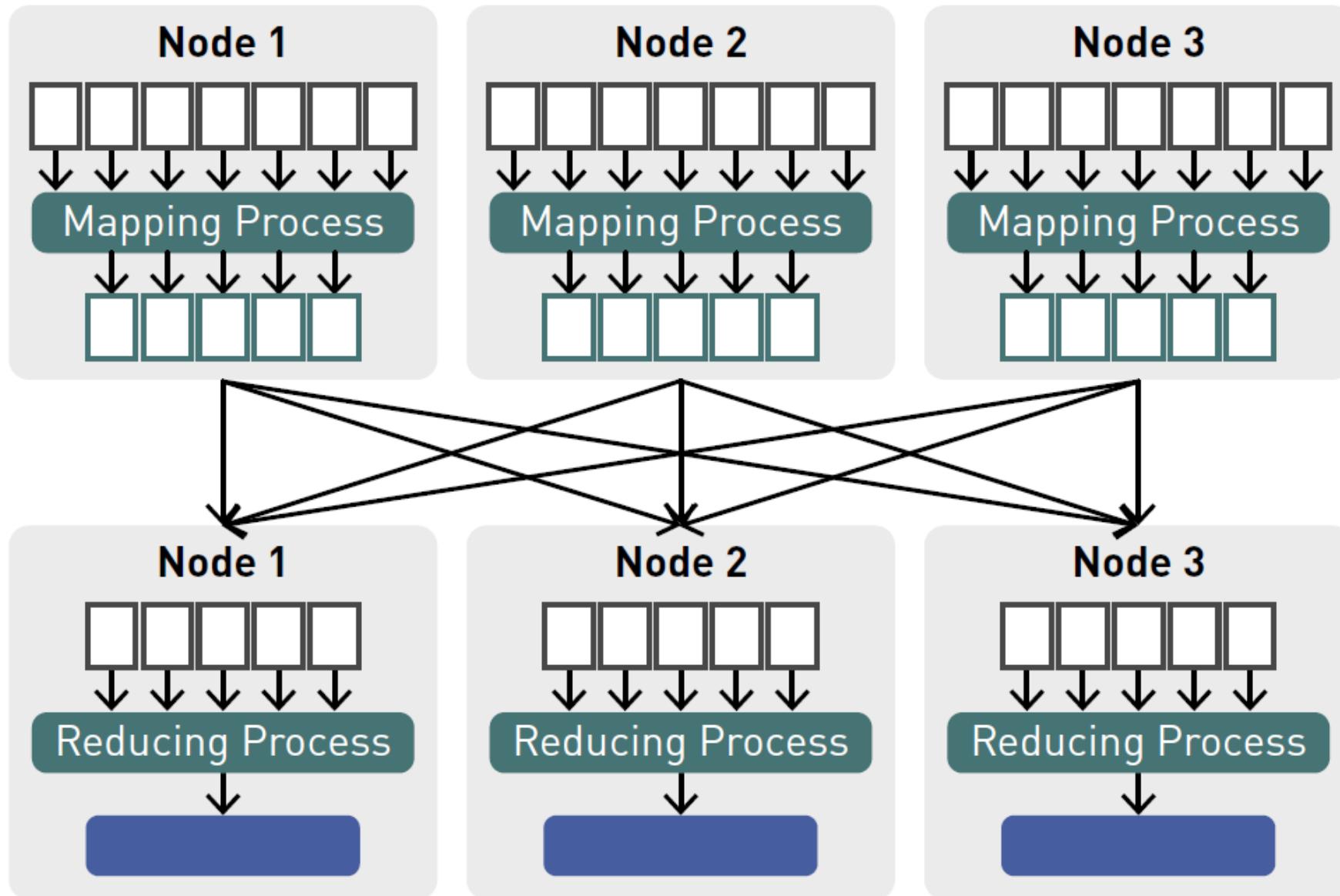
- Google sử dụng MR để index và exploding lượng lớn dữ liệu trên web trên cluster lớn.
- Concept của Google MR
 - **Distributed Data:** Dữ liệu được phân tán trên cụm
 - **Distributed Computation:** Tính toán phân tán.
 - **Tolerate faults:** Có khả năng tự động thực hiện lại tính toán khi có lỗi trên nodes bất kỳ



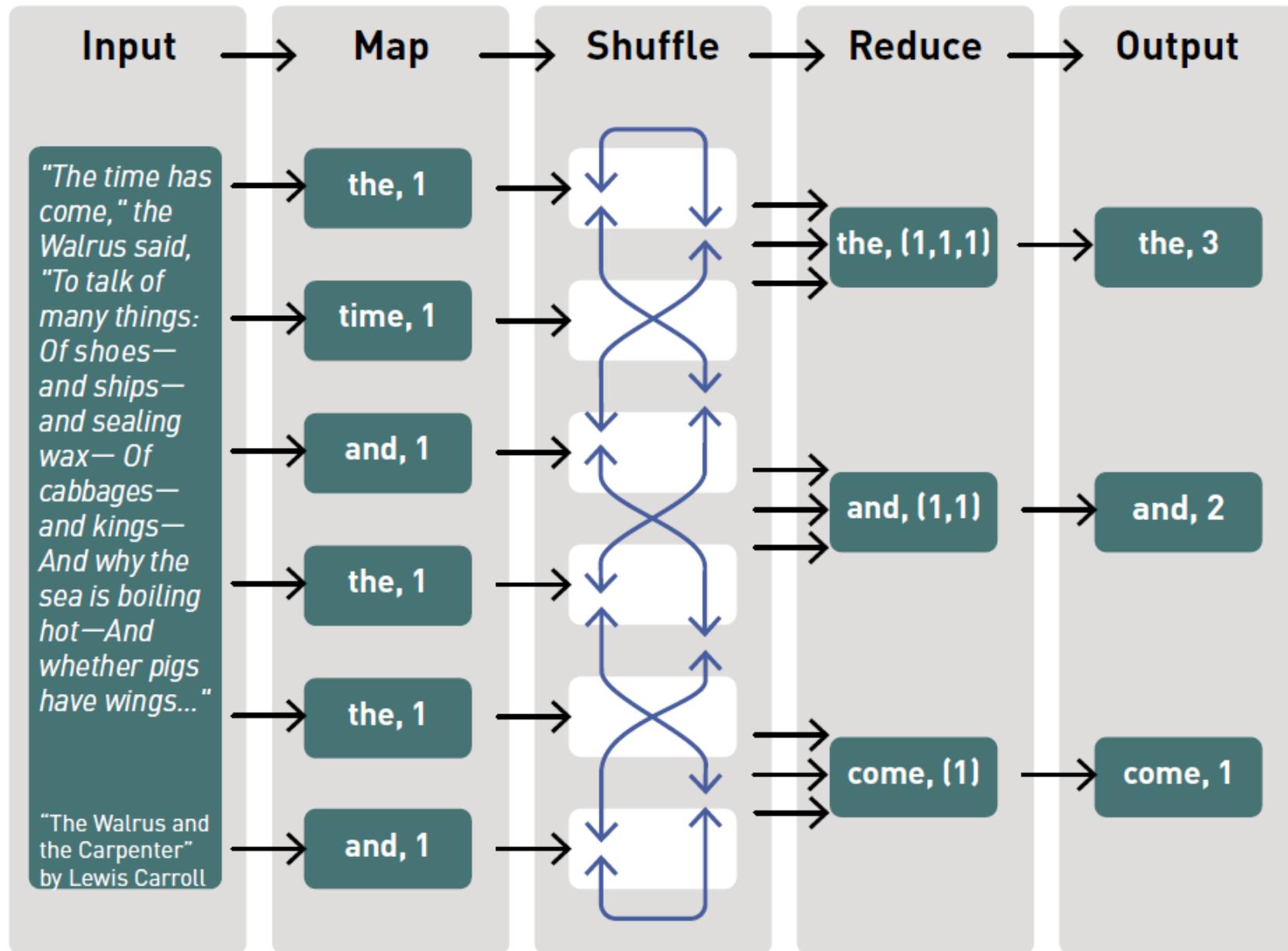
Hadoop MapReduce

- 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)

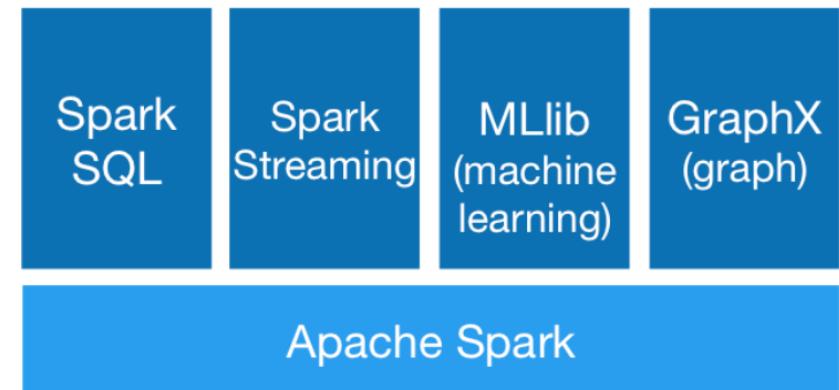


Reduce – Word Count



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)

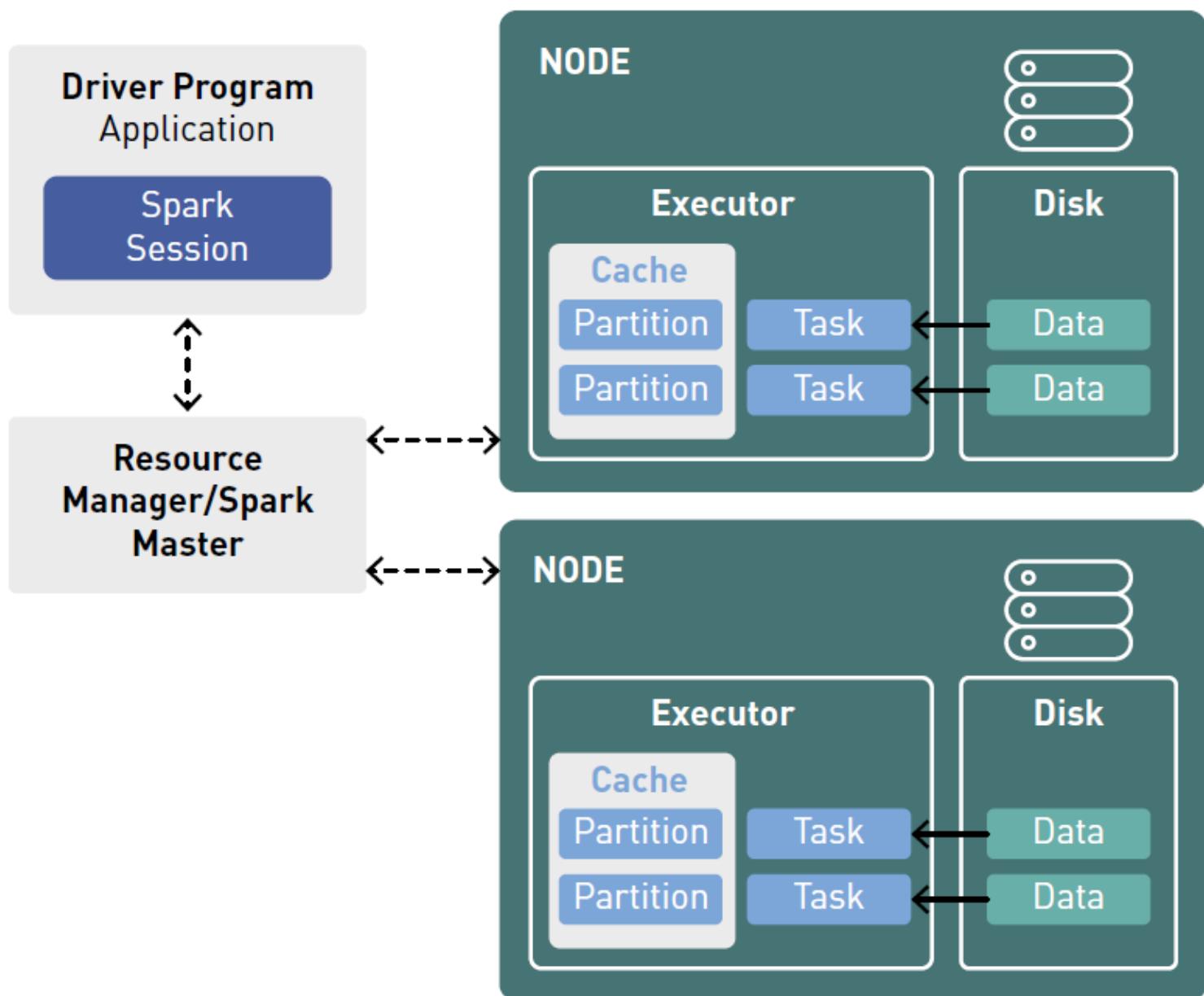
- Tính năng, ưu điểm của Spark
 - Advanced Analytics: Spark không chỉ hỗ trợ "Map" và "Reduce", nó còn hỗ trợ Spark truy vấn SQL, Streaming data, Machine learning (ML) và các thuật toán xử lý đồ thị đóng vai trò như một bộ công cụ phân tích dữ liệu cực kì mạnh mẽ.
 - Speed: Spark giúp chạy một ứng dụng với tốc độ rất nhanh. So với Hadoop cluster, Spark Application nén 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



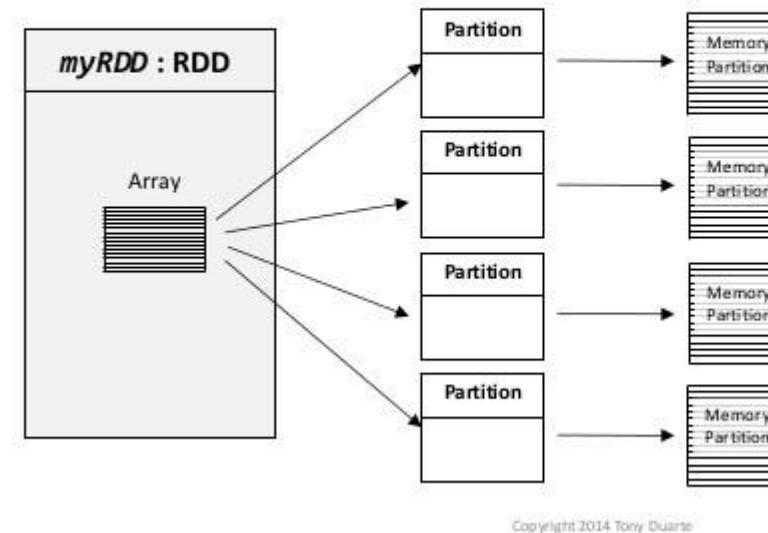
3. Spark Core



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.

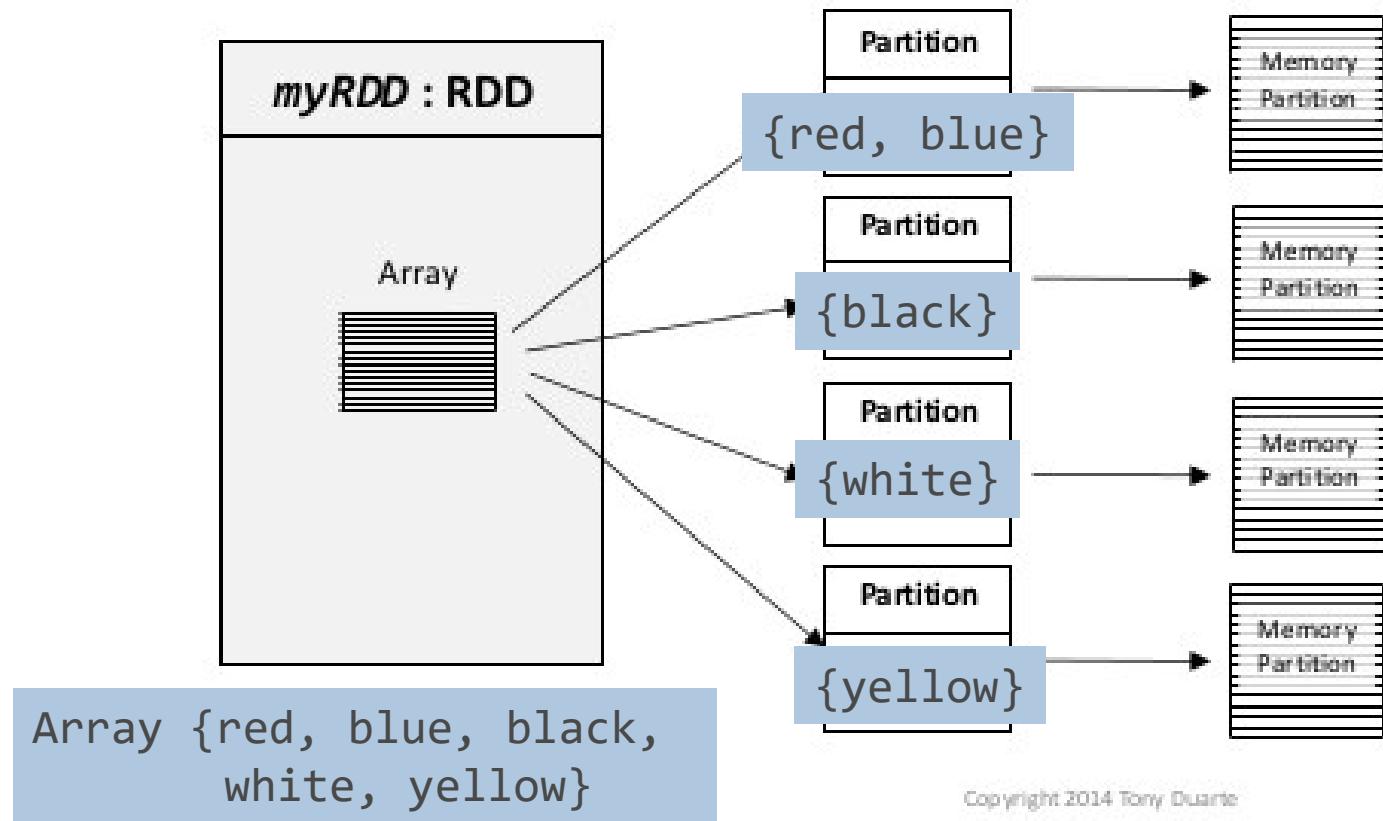
What is an RDD?



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)

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)

RDD Operations

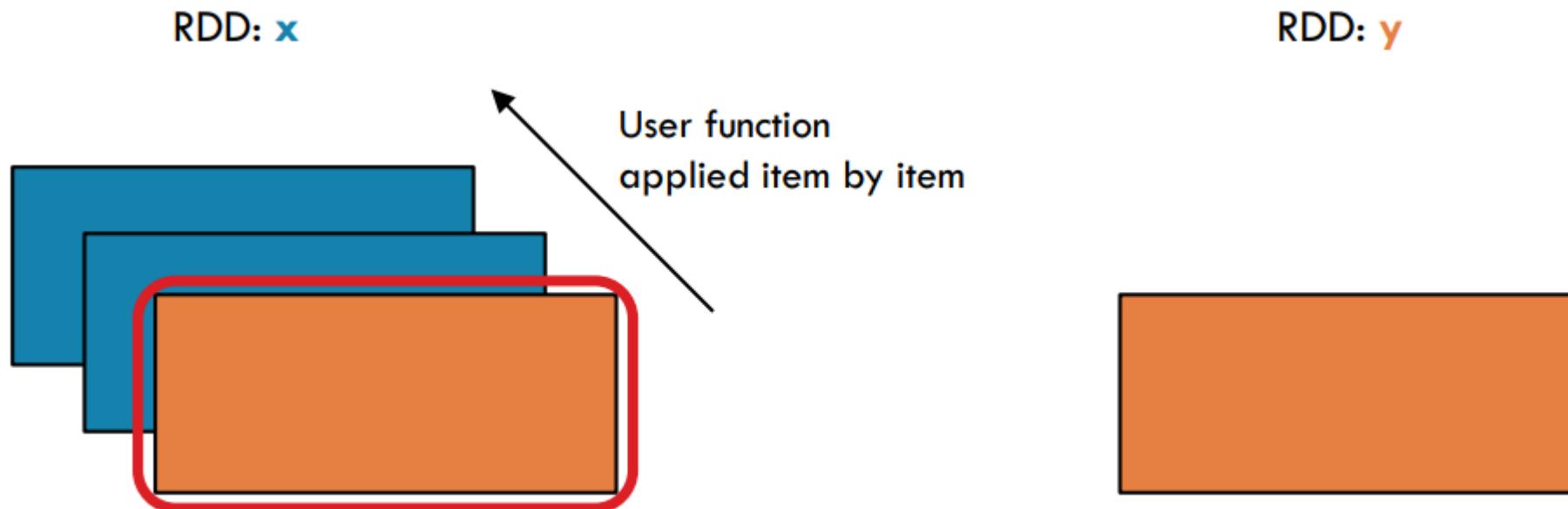
- Có 2 loại Operations
 - Actions: Trả về dữ liệu cho driver hoặc ghi dữ liệu
 - Transformations: Biến đổi RDD thành 1 RDD khác
- Action là sự kiện trigger để bắt đầu thực thi các Transformations

 Operations =



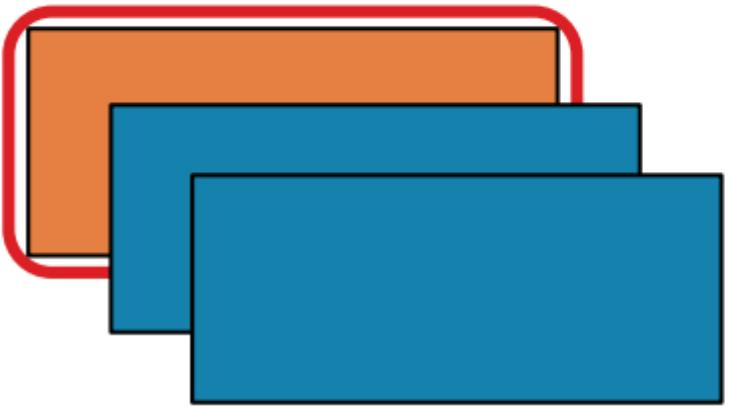
Map Transformation (1)

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

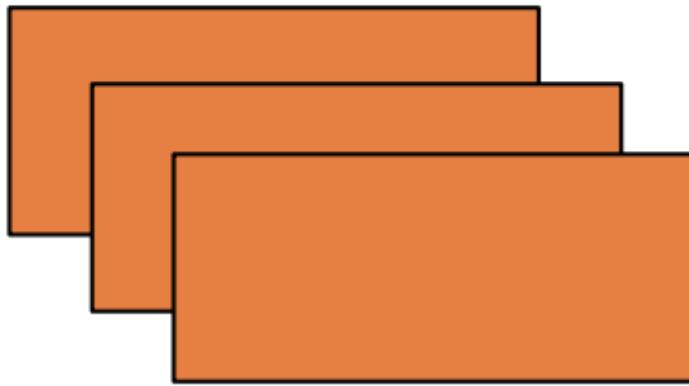


Map Transformation (2)

RDD: **x**

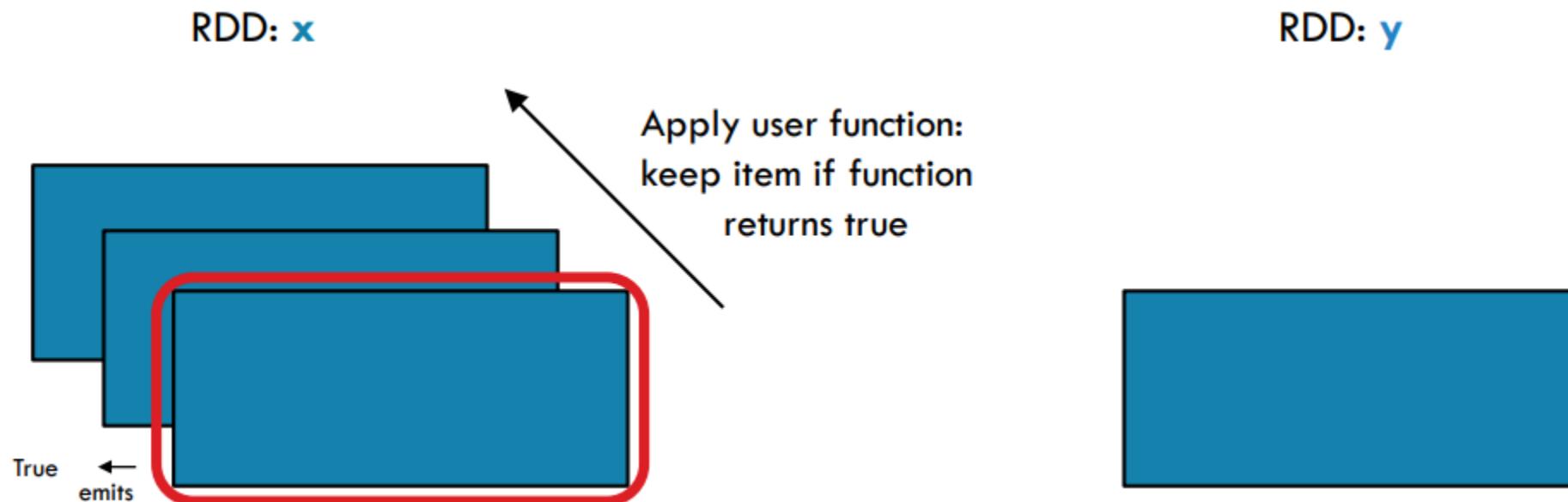


RDD: **y**

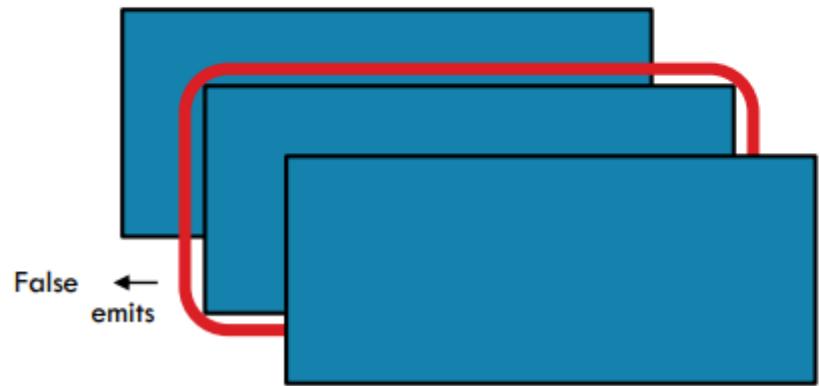
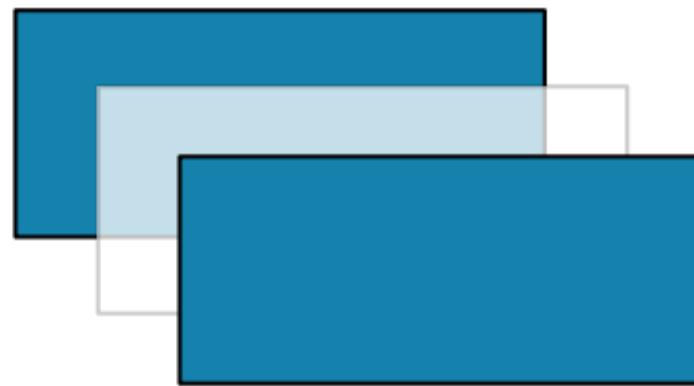
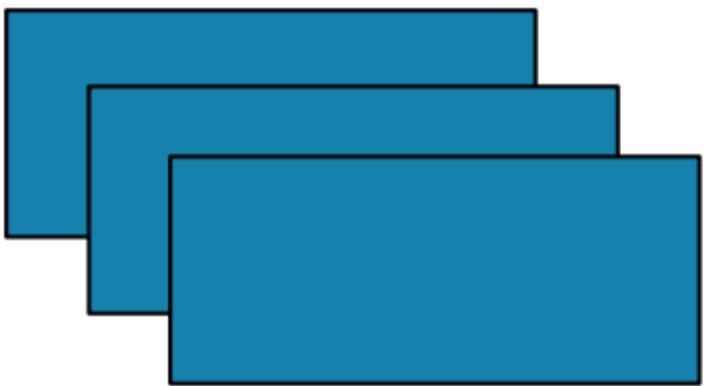
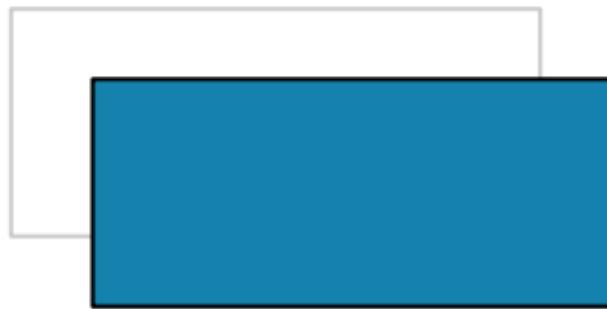


— Filter Transformation (1)

- `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*

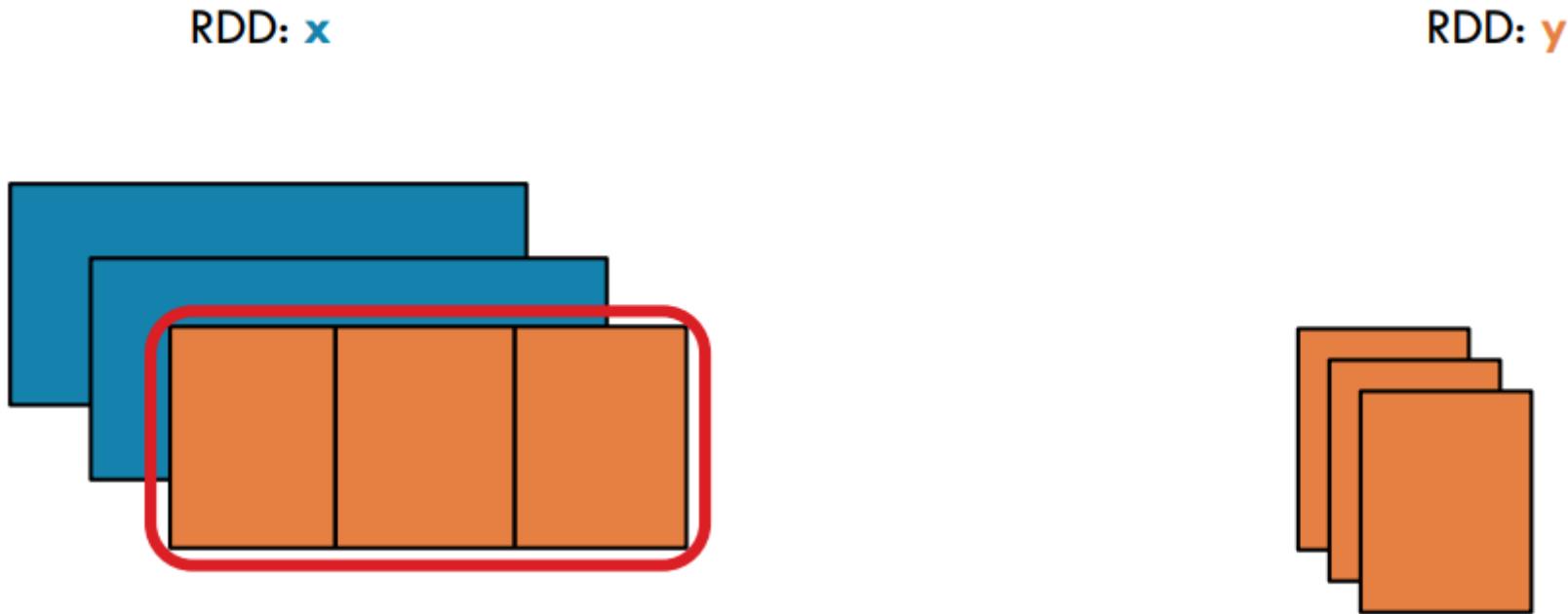


Filter Transformation (2)

RDD: **x**RDD: **y**

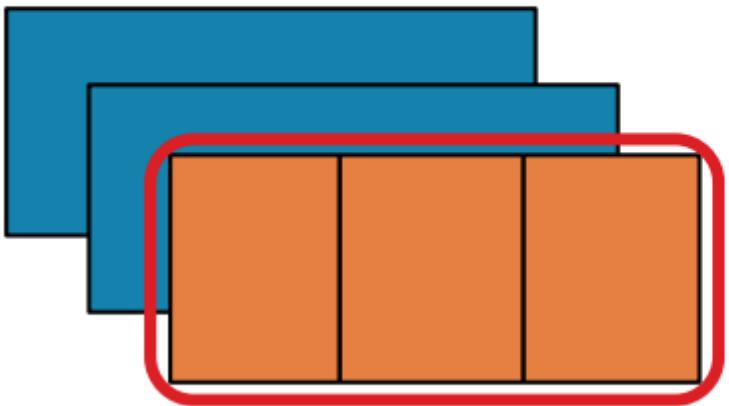
Flatmap Transformation (1)

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

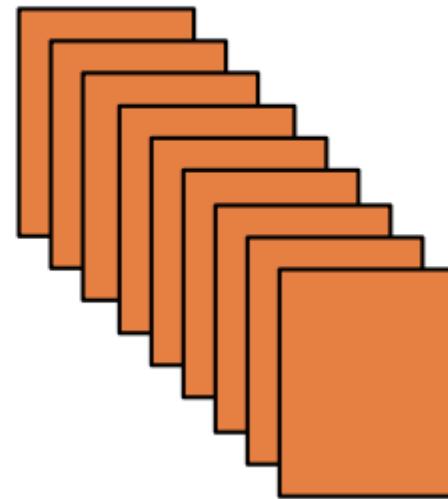


Flatmap Transformation (2)

RDD: x

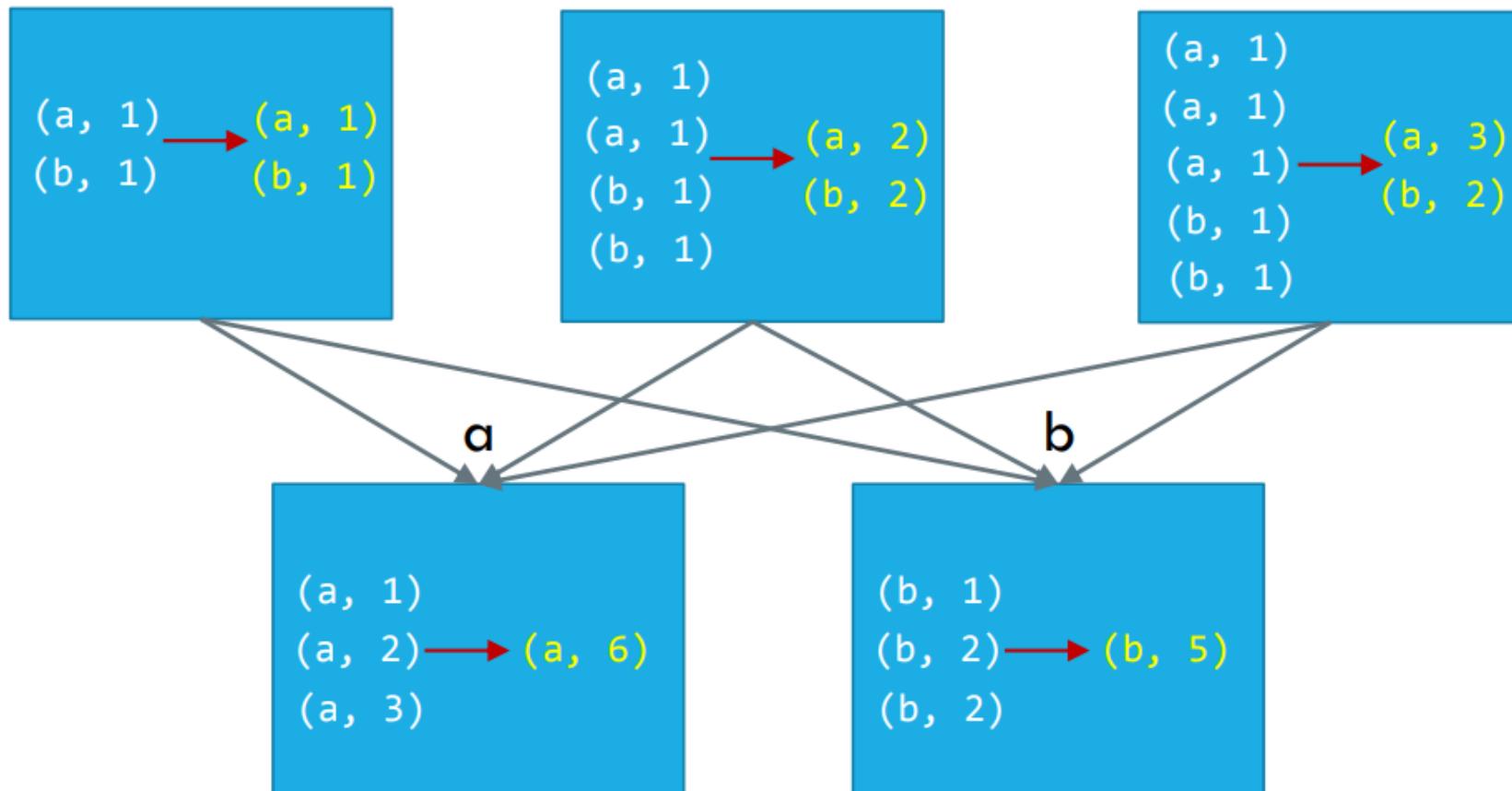


RDD: y



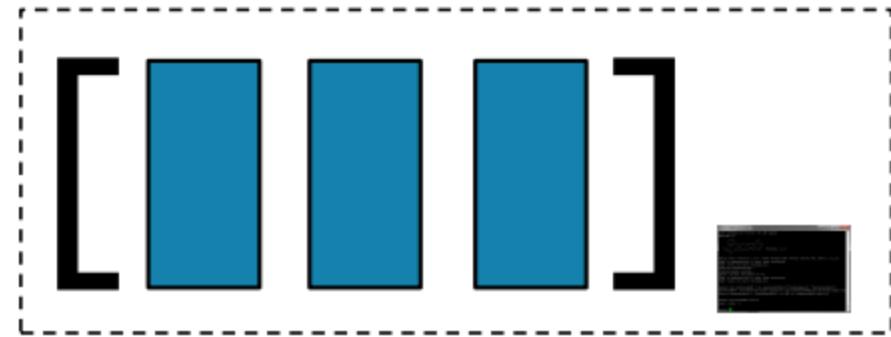
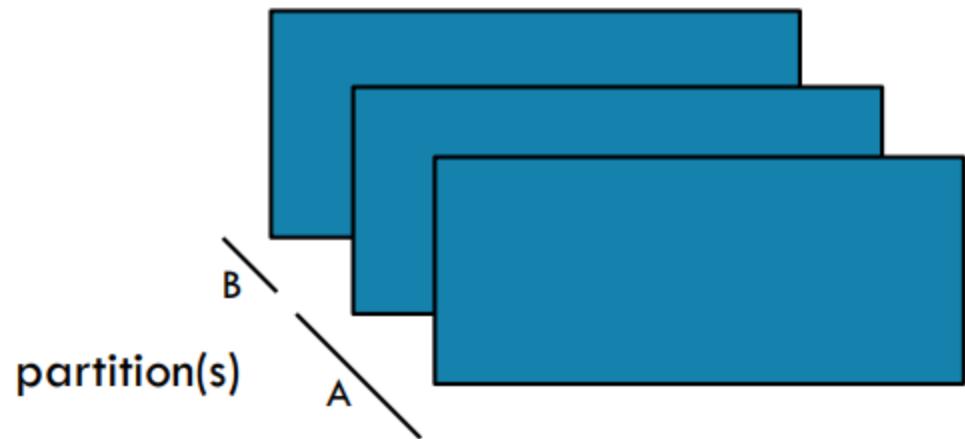
ReduceByKey Transformation (2)

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



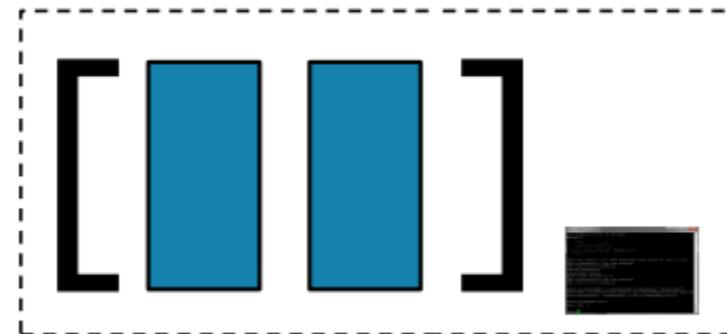
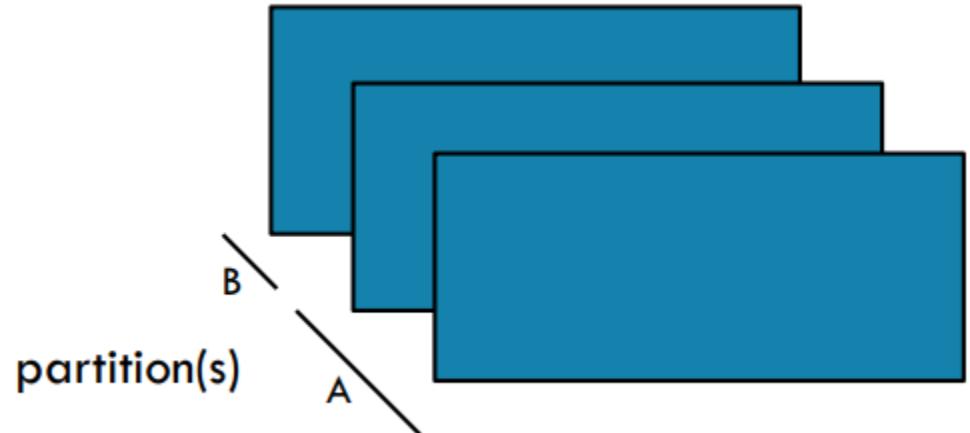
Collect Action

- 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



4. Thực hành

