SPARK FUNDAMENTALS

Presented by: Le Ngoc Thanh

Outline

- Introduction to Apache Spark
  - Features of Spark
  - A brief history of Spark
- The Spark Ecosystem
- Working with Spark

@lnthanh

Real-time analytics use cases

fit@hcmus

FACEBOOK
Users like
4,166,667
posts

TWITTER
Users send
347,222
tweets

REDDIT
Users cast
18,327
votes

INSTAGRAM
Users like
1,736,111
posts

YOUTUBE
Users upload
300 hours
of new video

The amount of data generated every minute by social media leaders

Real-time analytics use cases

fit@hcmus

Banking

Government

Healthcare

Telecommunications

Stock market

The need of faster analytics

fit@hcmus

- D

- The need to have faster analytics results is thus become increasingly

MapReduce

Spark

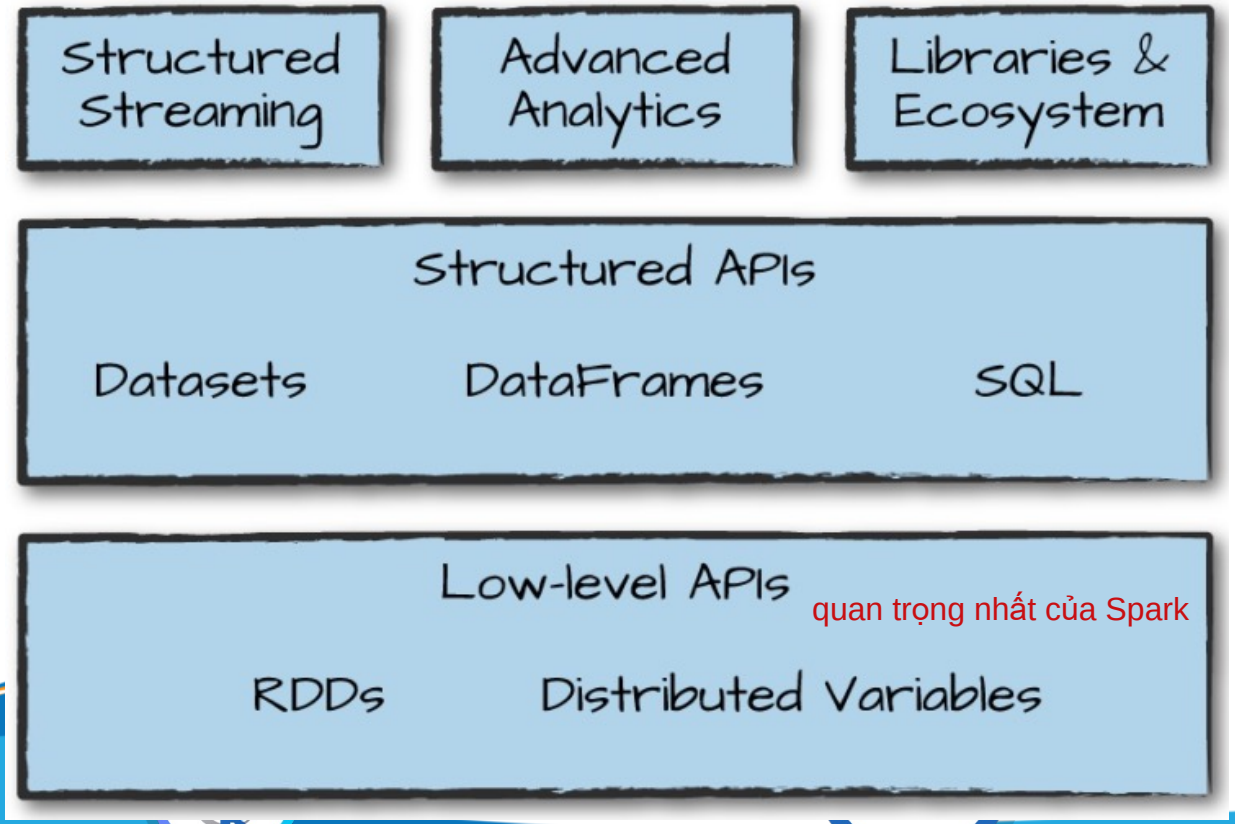
Introduction to Apache Spark

An open-source cluster computing framework for real-time processing

What is Apache Spark?

fit@hcmus

- **Apache Spark** is a unified computing engine and a set of libraries for parallel data processing on computer clusters.



What is Apache Spark?

fit@hcmus

Speed

In-memory computations

Faster than MapReduce for applications on disk

Generality

Cover a wide range of workloads in one system

Batch applications (e.g. MapReduce)

Iterative algorithms, interactive queries and streaming

Easy of use

APIs for Java, Python and Scala

Run on Hadoop cluster or as a standalone

Libraries for SQL, Machine learning, streaming and graph processing

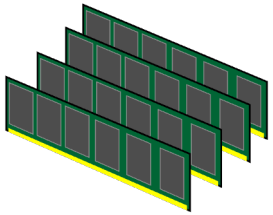
@lnthanh

Who use Spark and Why?

fit@hcmus

Parallel distributed processing

Scalability



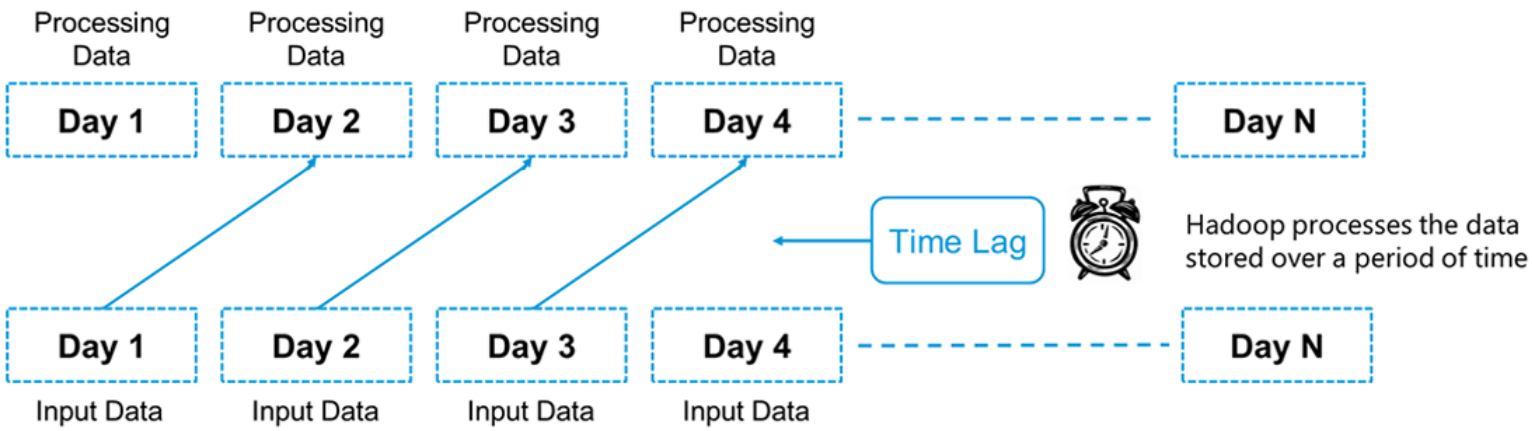
In-memory computing

Fault tolerance on commodity hardware

High level APIs

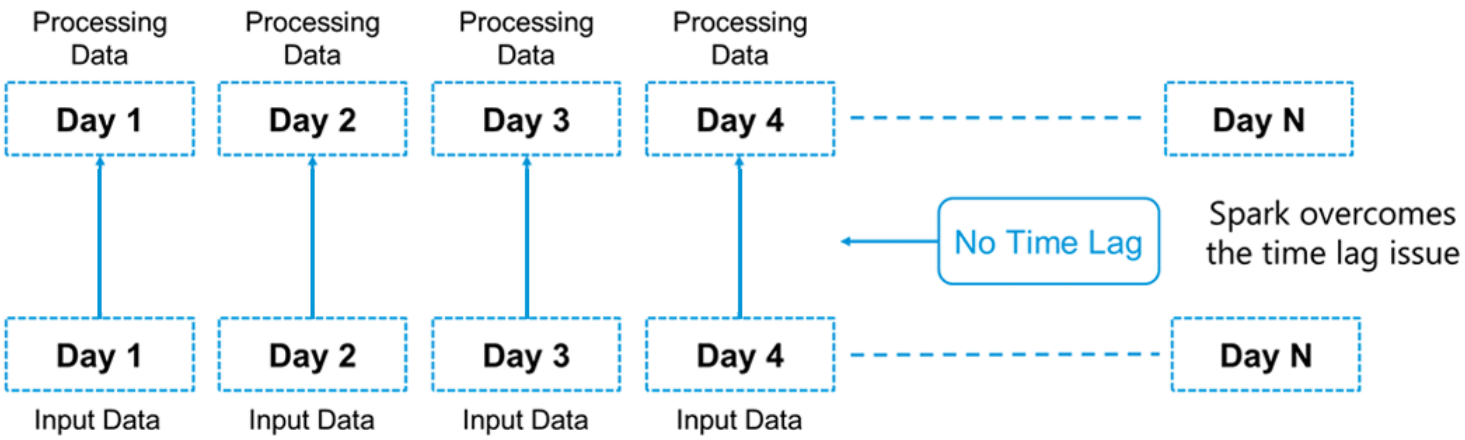
save

Processing Data Using MapReduce



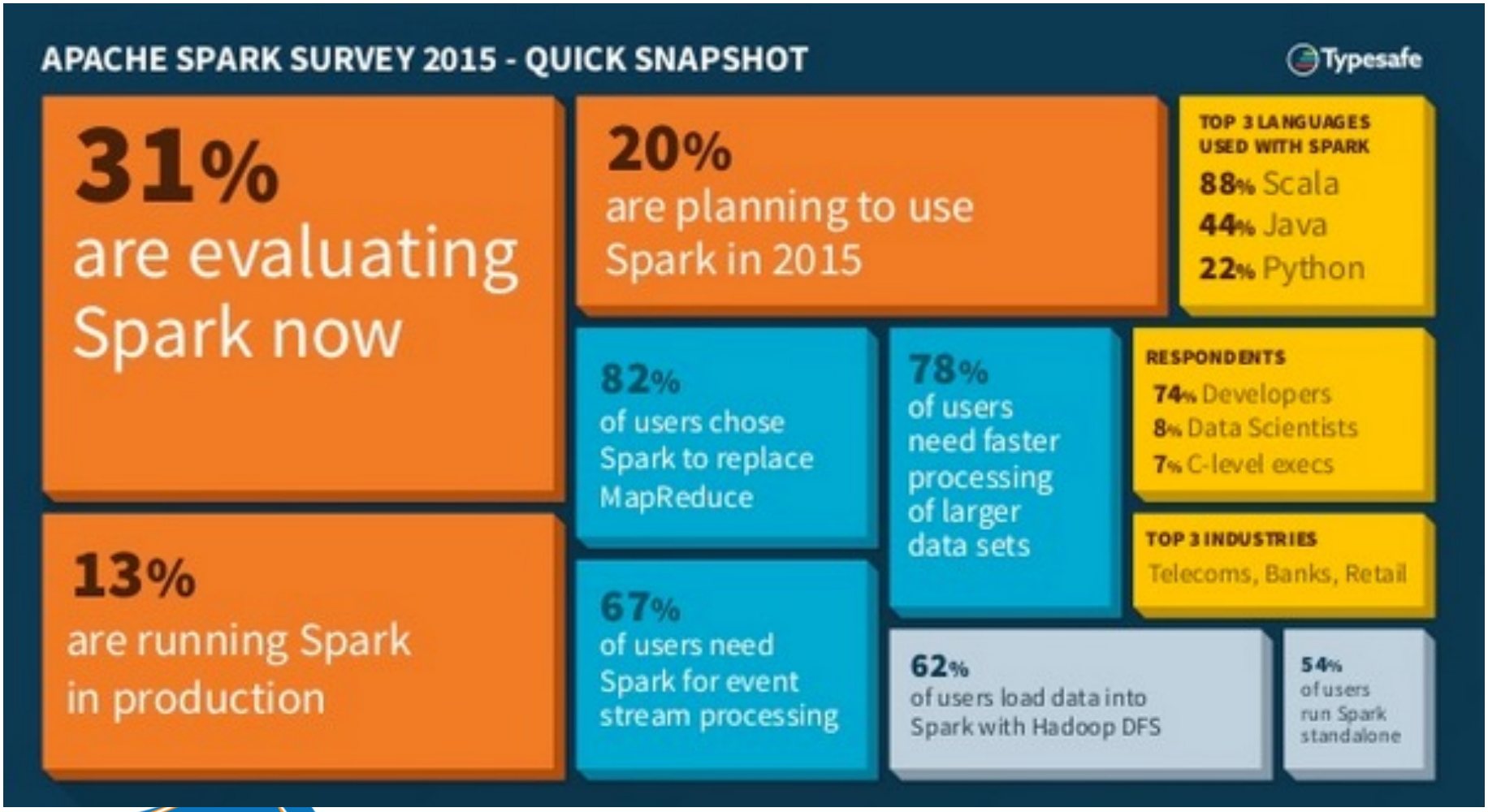
VS

Real Time Processing in Spark



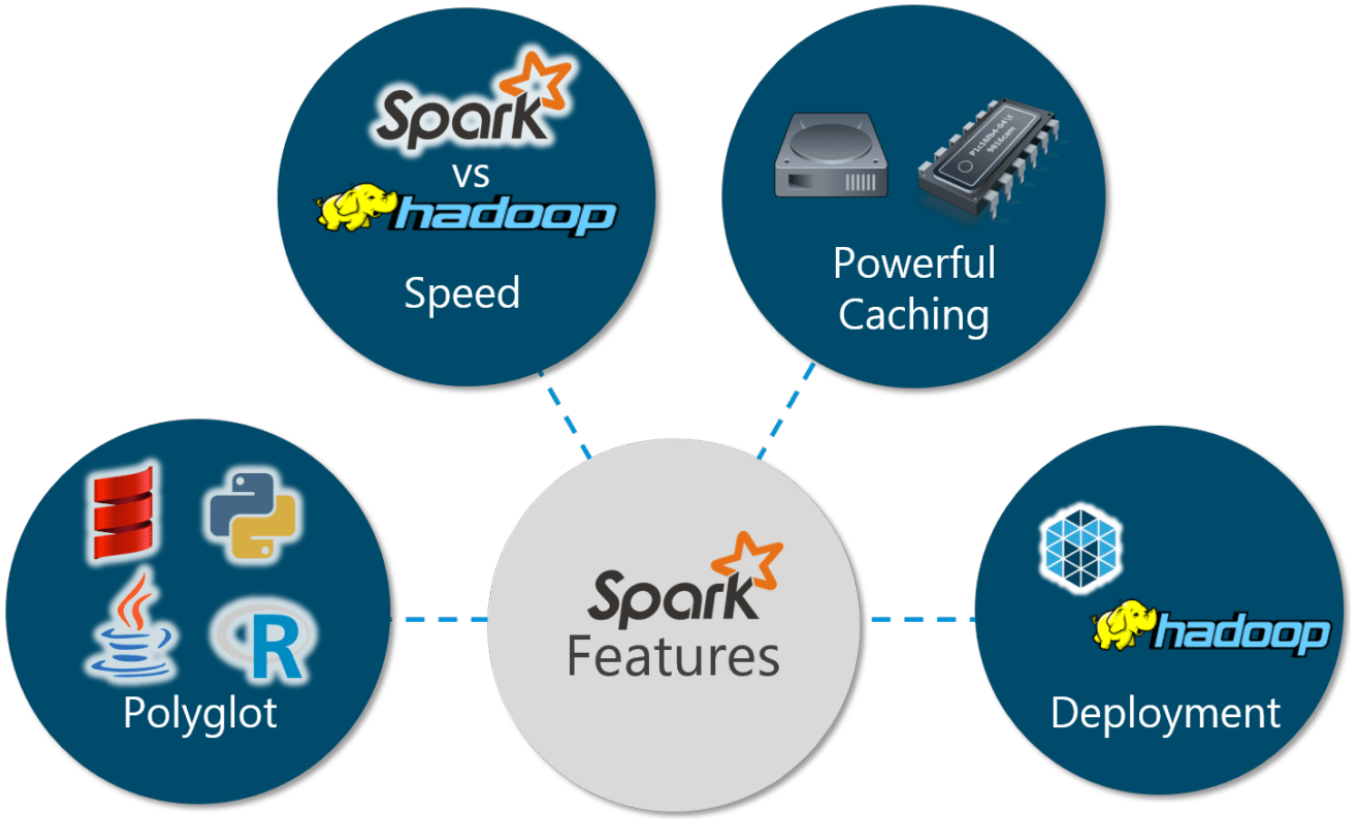
The popularity of Spark

fit@hcmus

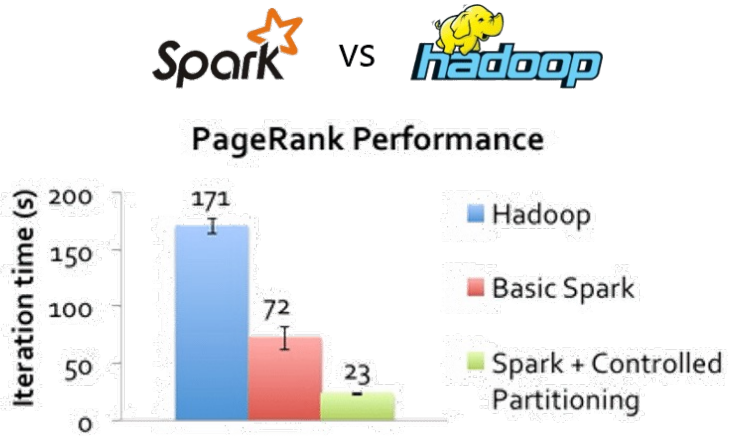


Source: 2015 Typesafe survey.

Features of Apache Spark



Features of Apache Spark



Speed

- Up to 100× faster than MapReduce for large-scale data processing
- Parallelize distributed data processing with minimal network traffic

Real-time computation

- In-memory computation: real-time, low latency
- Designed for massive scalability
- Production clusters with thousands of nodes
- Several computational models supported

Features of Apache Spark

fit@hcmus

Polyglot

- High-level APIs in Java, Scala, Python and R
- Shells for Scala (./bin/spark-shell) and Python (./bin/pyspark)

Multiple Format

- Parquet, JSON, Hive and Cassandra
- Text files, CSV, RDBMS tables

Hadoop Integration

- Smooth combability with Hadoop
- Run on top of an existing Hadoop cluster

@lnthanh

Features of Apache Spark

fit@hcmus

+ Spark sẽ đợi đến khi user cần or xuất ra màn hình thì mới làm, trước đó thì nó delay và ko làm
+ Vì nó delay như thế nên nó có đc toàn bộ những gì user mún làm và tối ưu hóa nó mà ko lm thay đổi kq (tránh làm dư)

Lazy evaluation

- Delay the evaluation till it is really necessary
- One of the key factors to speed

Machine Learning

- Handy for big data processing
- A powerful, unified engine that is both fast and easy to use

Who use Spark and Why?

fit@hcmus

- Analyze and model the data to obtain insight using ad-hoc analysis
- Transforming the data into a useable format
- Statistics, machine learning, SQL

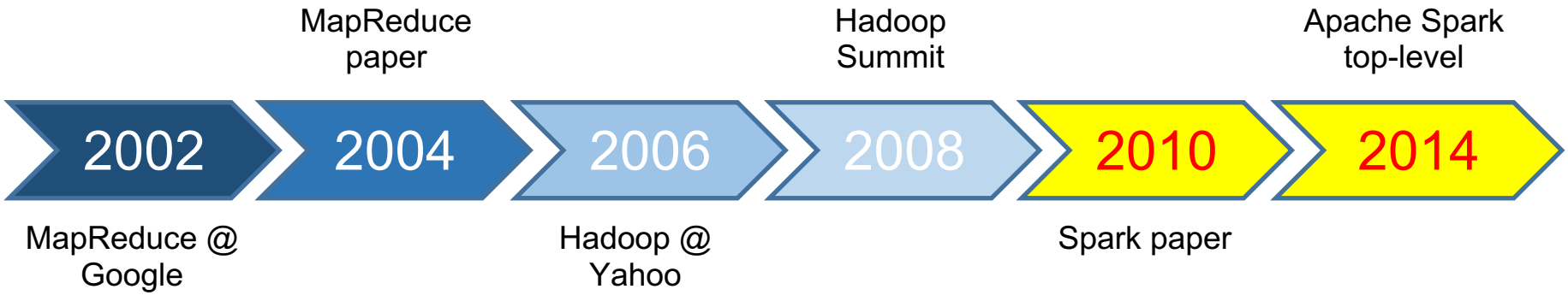
- Develop a data processing system or application
- Inspect and tune their applications
- Programming with the Spark's API

- Easy of use
- Wide variety of functionality
- Mature and reliable

@lnthanh

A brief history of Spark

fit@hcmus

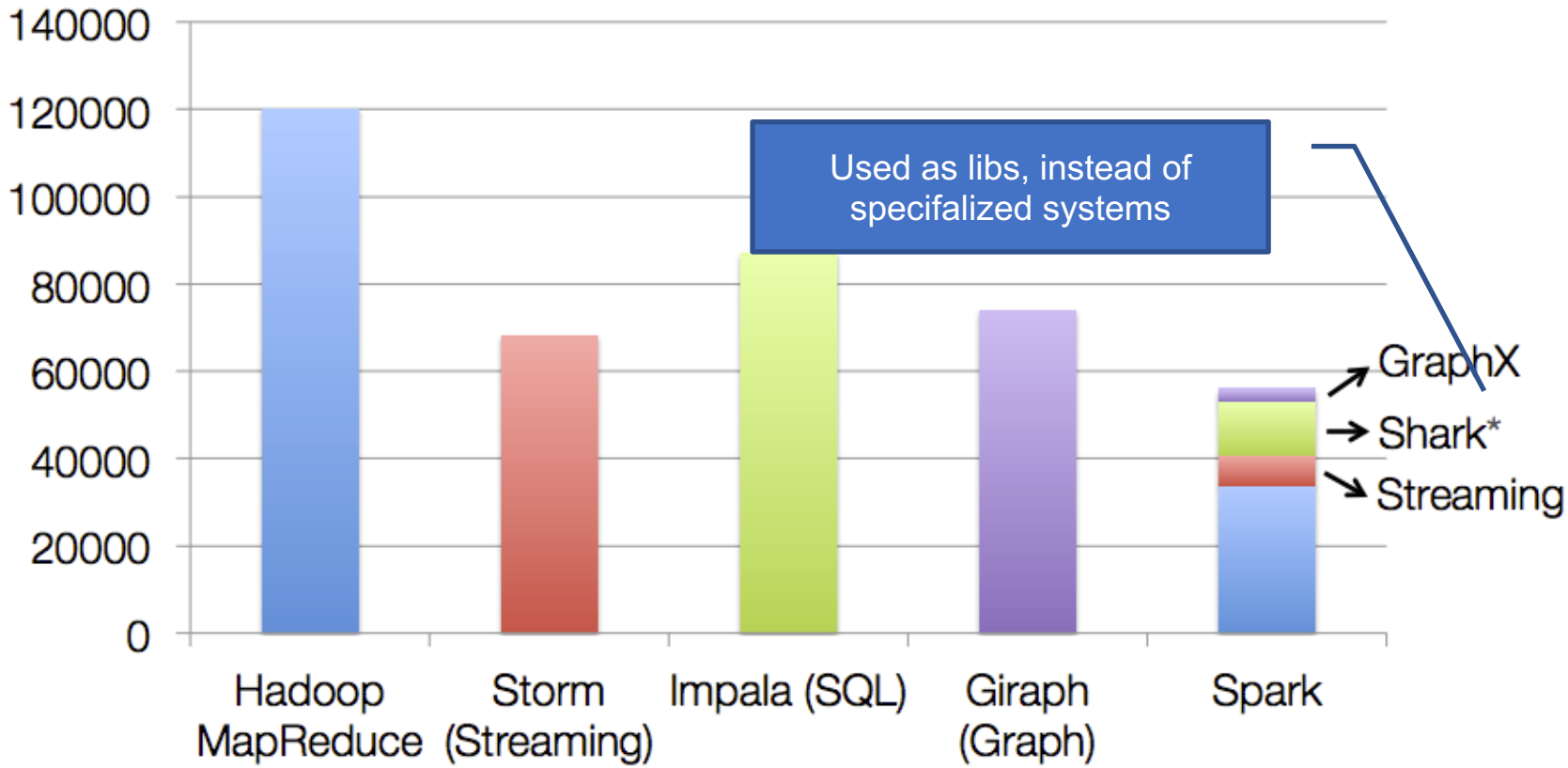


- MapReduce started a general batch processing paradigm
- Two limitations
  - Difficulty programming in MapReduce
  - Batch processing did not fit many use cases
- Spawned a lot of specialized systems (Storm, Impala, Gigraph, etc.)

@lnthanh

Comparison of code sizes

fit@hcmus

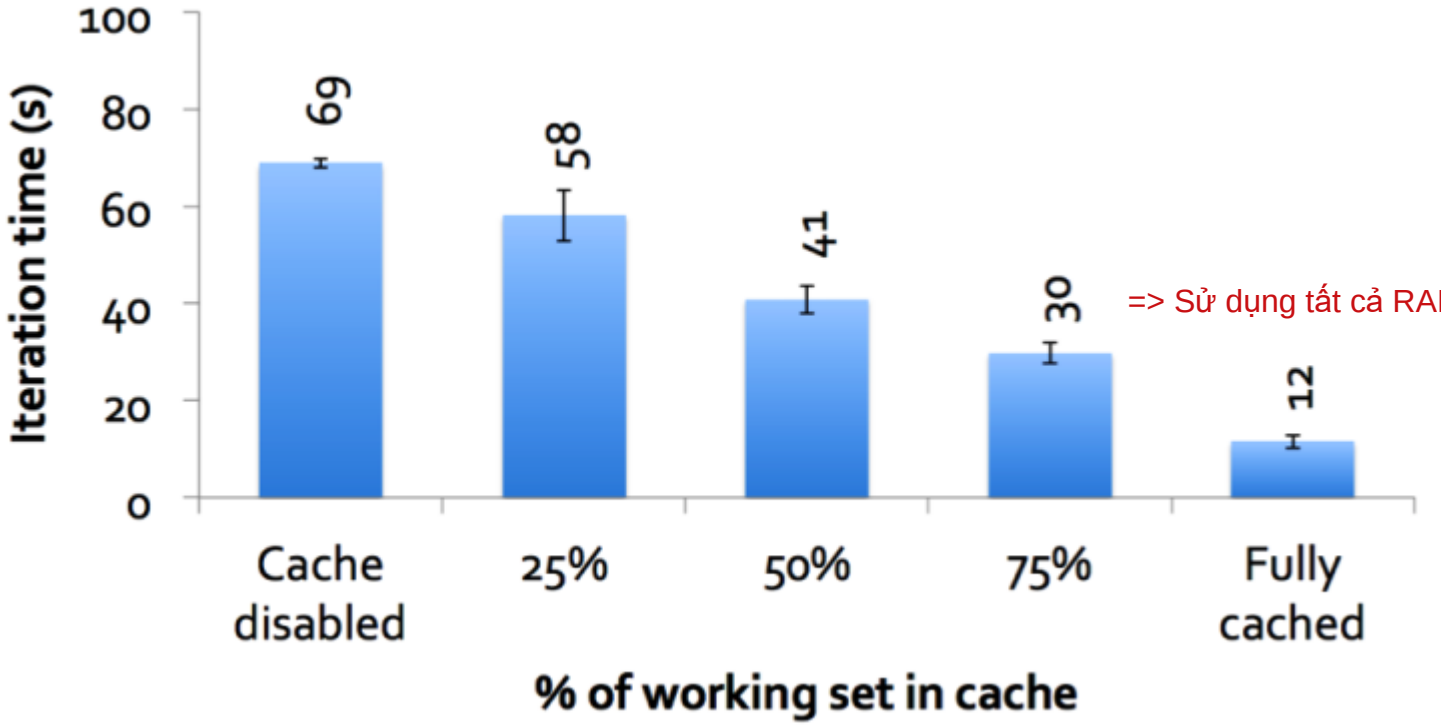


non-test, non-example source lines

* also calls into Hive

The State of Spark, and Where We're Going Next, Matei Zaharia, Spark Summit (2013)
https://www.youtube.com/watch?v=nU6vO2EJAb4

Spark behaviors with less RAM



=> Sử dụng tất cả RAM trong máy tính

The Ecosystem of Spark

Spark Core, Spark SQL, MLlib, etc.

Spark unified stack

fit@hcmus

Spark SQL & Shark

Spark Streaming
real-time processing

MLlib
machine learning

GraphX
graph processing

Spark Core

Tài nguyên do Spark quản lý

Standalone Scheduler

YARN

Mesos

=> Sử dụng cơ chế điều phối tài nguyên của Yarn lq đến HDFS =

Spark Core

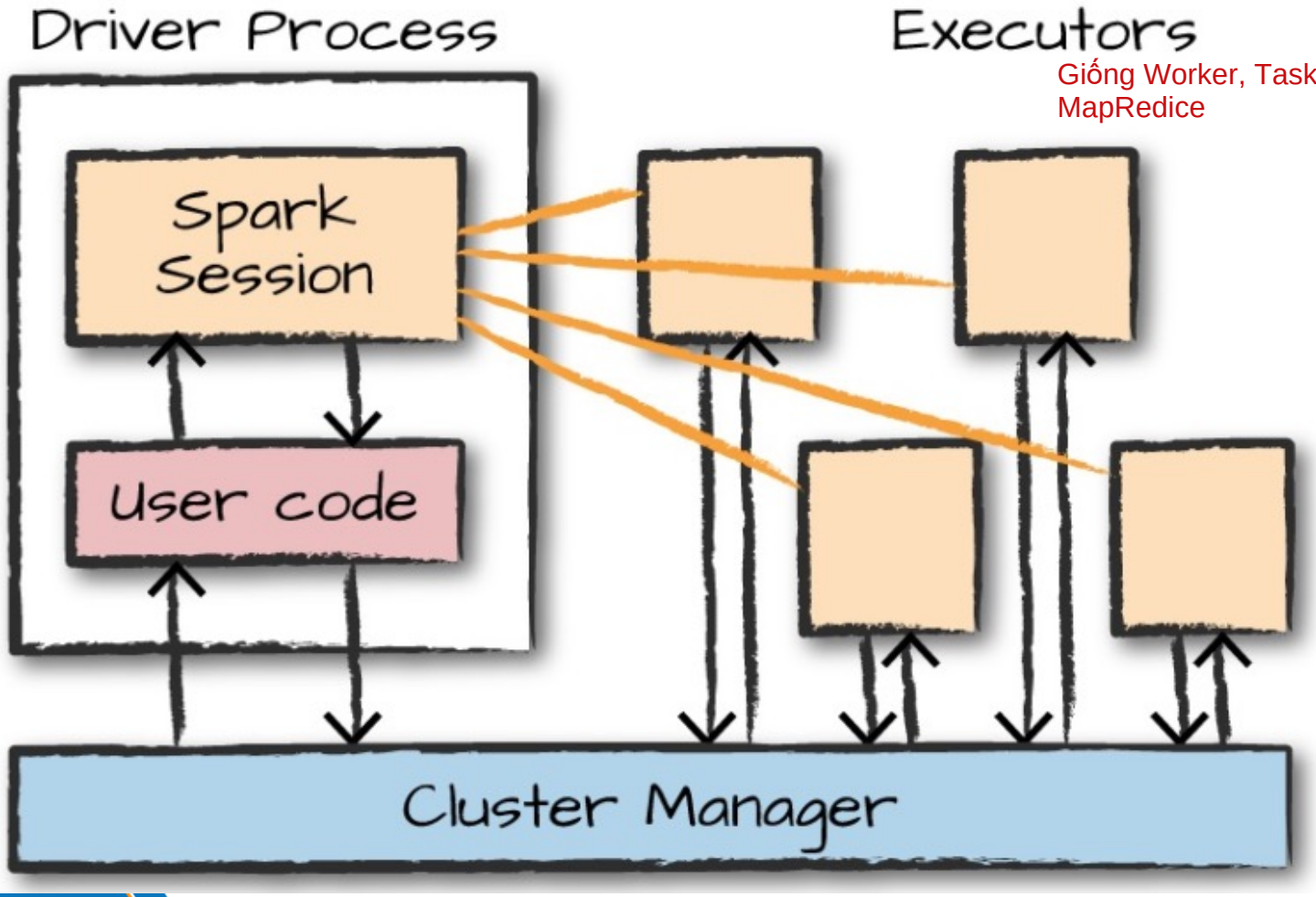
- **Distributed execution engine:** the base engine for large-scale parallel and distributed data processing
- Java, Scala, and Python APIs offered
- Additional libraries built atop allow diverse workloads
- Memory management and fault recovery
- Scheduling, distributing and monitoring jobs on a cluster
- Interacting with storage systems

Deployment environments

fit@hcmus

- Spark applications can run locally (on a single JVM) or using cluster resources.
  - Specify the deploy mode using *-deploy-mode* option of *spark-submit* or *spark.submit.deployMode* Spark property
- **Local mode** (pseudo cluster)**:** The driver and executors run (as threads) on an individual computer instead of a cluster.
- **Cluster mode:** Distrbuted mode with a cluster manager
  - Spark's own built-in Standalone cluster manager
  - Hadoop YARN
  - Apache Mesos
  - Kubernetes, Amazon EC2

Deployment environments



Giống Worker, TaskTracker trog MapRedice

The architecture of a Spark Application

Supported data sources

- Spark focuses on performing computations over the data, no matter where it resides.

Nên dùng trog DL phân tán

Parquet

HIVE

MySQL

PostgreSQL

and more ...

{ JSON }

Dùng trog làm web

amazon web services S3

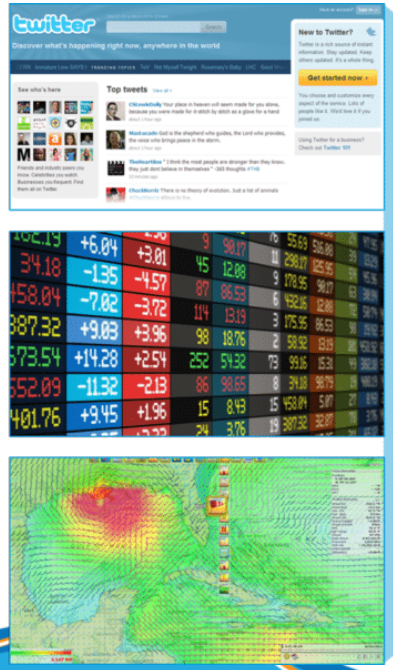
openstack

HDFS

Formats and Sources supported by DataFrames

Spark Streaming

- Enable high-throughput and fault-tolerant stream processing of live data streams
- **DStream** – fundamental stream unit: a series of RDDs



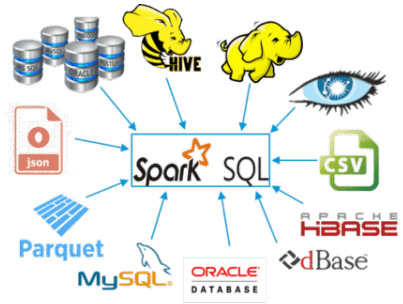
Spark Streaming is used to stream real-time data from various sources like Twitter, Stock Market and Geographical Systems and perform powerful analytics to help businesses.

Spark SQL

- Integrate relational processing with functional programming
- Support data query either via SQL or Hive Query Language

Spark SQL

Ko lưu trữ giống MySQL chỉ làm việc giống

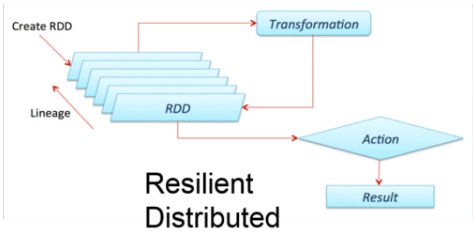


Data Source API

DataFrame API

Named Columns

Interpreter & Optimizer



Resilient Distributed Dataset

Spark SQL Service

Spark SQL RESULTS

Figure: *The flow diagram represents a Spark SQL process using all the four libraries in sequence*

@lnthanh

Spark SQL libraries

fit@hcmus

- Data Source API (Application Programming Interface)
  - An universal API for loading and storing structured data
  - Built-in supports for Hive, Avro, JSON, JDBC, Parquet, etc.
  - Third-party integration through Spark packages
- DataFrame API
  - Process huge data (KBs or PBs) on single/multi-node clusters
  - Support different data formats and storage systems
    - Avro, CSV, Elastic Search and Cassandra, etc.
    - HDFS, HIVE Tables, MySQL, etc.
  - Easily integrated with all Big Data tools/frameworks via Spark Core
  - Provides API for Python, Java, Scala, and R Programming

Spark SQL libraries

fit@hcmus

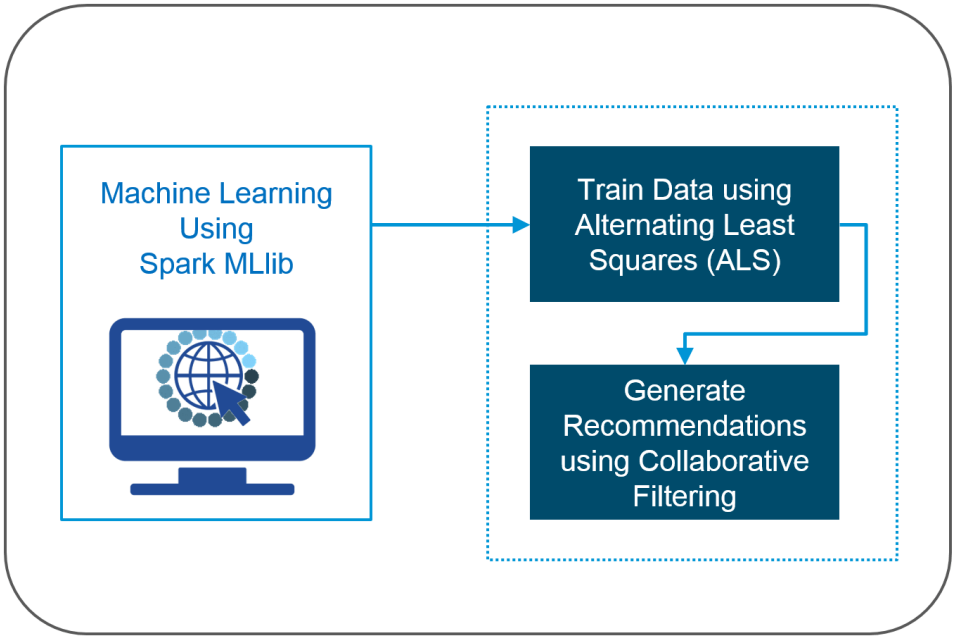
- SQL Interpreter And Optimizer
  - The newest and most technically evolved component of SparkSQL
  - Perform analysis/evaluation, optimization, planning, and run time code spawning
  - Support cost based optimization (run time and resource) and rule based optimization (e.g., Catalyst)
    - Making queries run much faster than their RDD counterparts
- SQL Service
  - An entry point for working along structured data in Spark
  - Allow the creation of DataFrame objects as well as the execution of SQL queries.

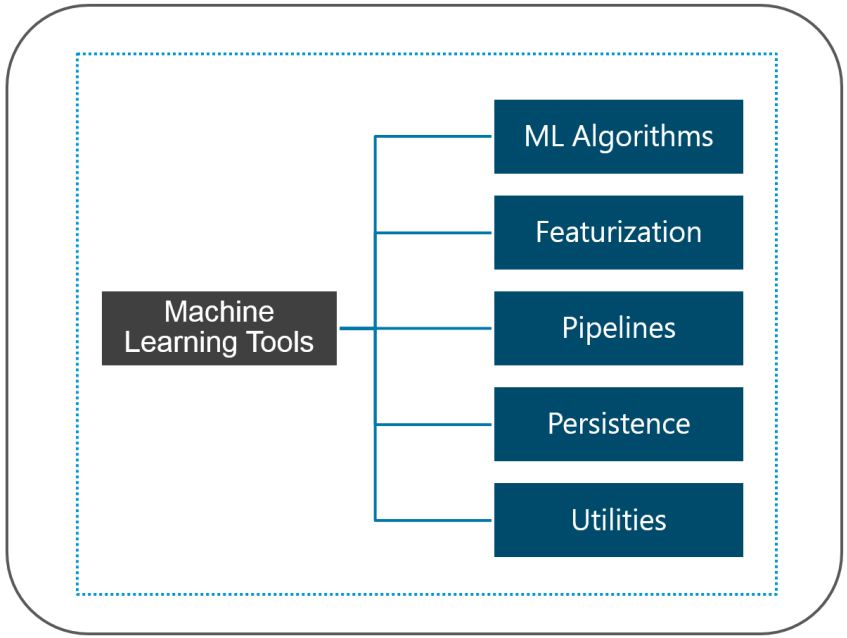
GraphX

- Spark API for graphs and graph-parallel computation
- Extend the Spark RDD abstraction to a Resilient Distributed Property Graph
  - A directed multigraph with properties attached to each vertex and edge
- Support a set of fundamental operators (e.g., subgraph, joinVertices, and mapReduceTriplets) and an optimized variant of the Pregel API
- Include a growing collection of graph algorithms and builders to simplify graph analytics tasks

Spark MLlib

- **M**achine **L**earning **Lib**rary: perform various machine learning tasks in Spark.



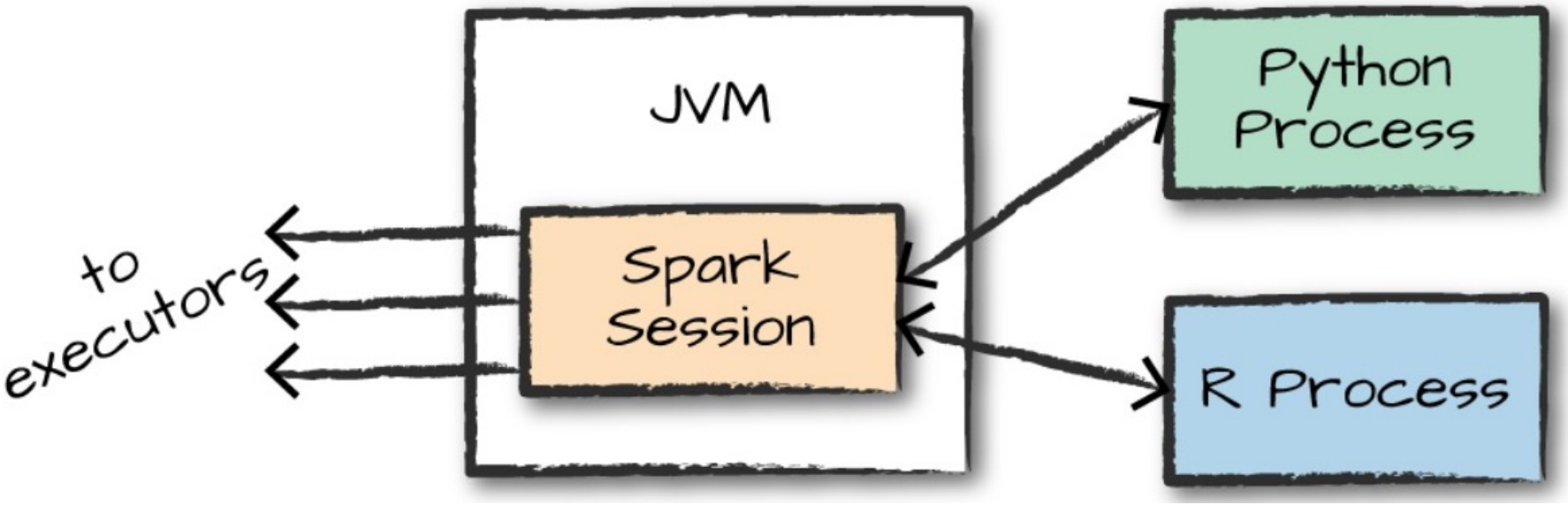


Spark's Language APIs

fit@hcmus

- Spark presents some core "concepts" in every language, which are then translated into Spark code.
- Scala: default language
- Java
- Python
  - Python supports nearly all constructs that Scala supports.
- SQL
  - A subset of the ANSI SQL 2003 standard supported
- R
  - Two commonly used R libraries: one as a part of Spark core (SparkR) and another as an R community-driven package (sparklyr)

Spark's Language APIs



The relationship between the SparkSession and Spark's Language API

- Do not have to write explicit JVM instructions
- Write Python and R code and Spark translates into code that can run on the executor JVMs.

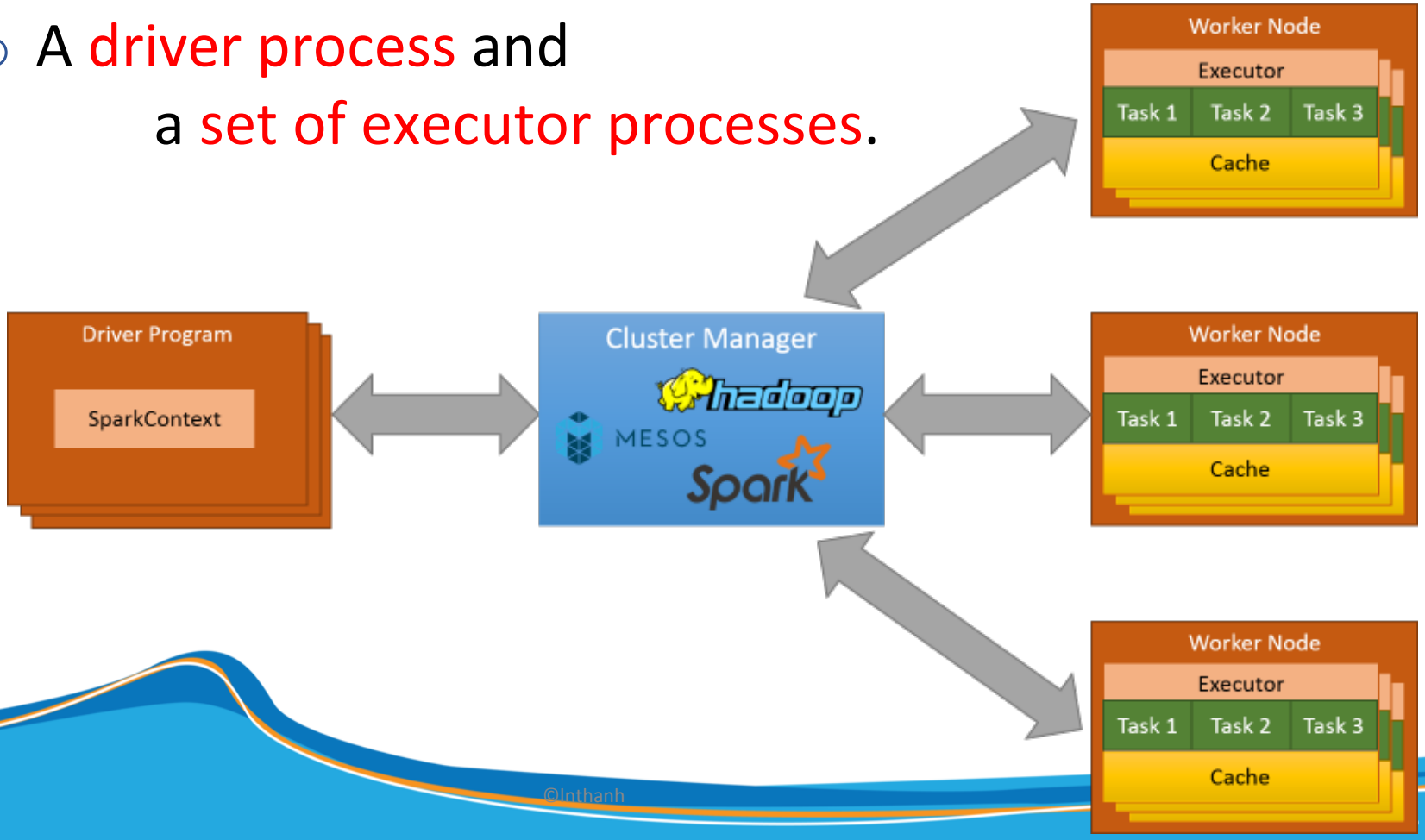
The Architecture of Spark

Where all the spark components and layers are loosely coupled…

The architecture of Spark

fit@hcmus

- A well-defined layered architecture where all components and layers are loosely coupled.
- A driver process and
  a set of executor processes.

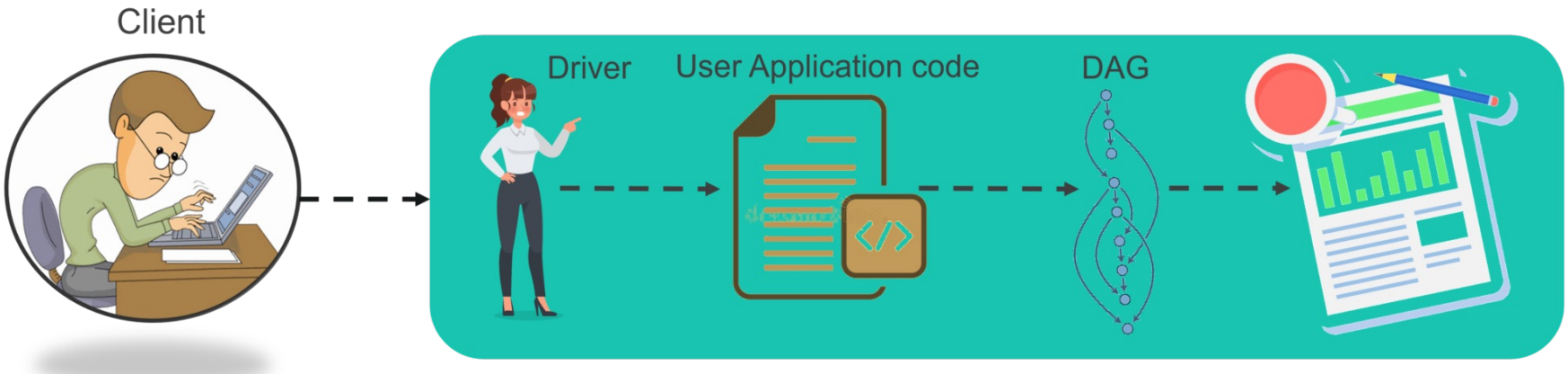


@lnthanh

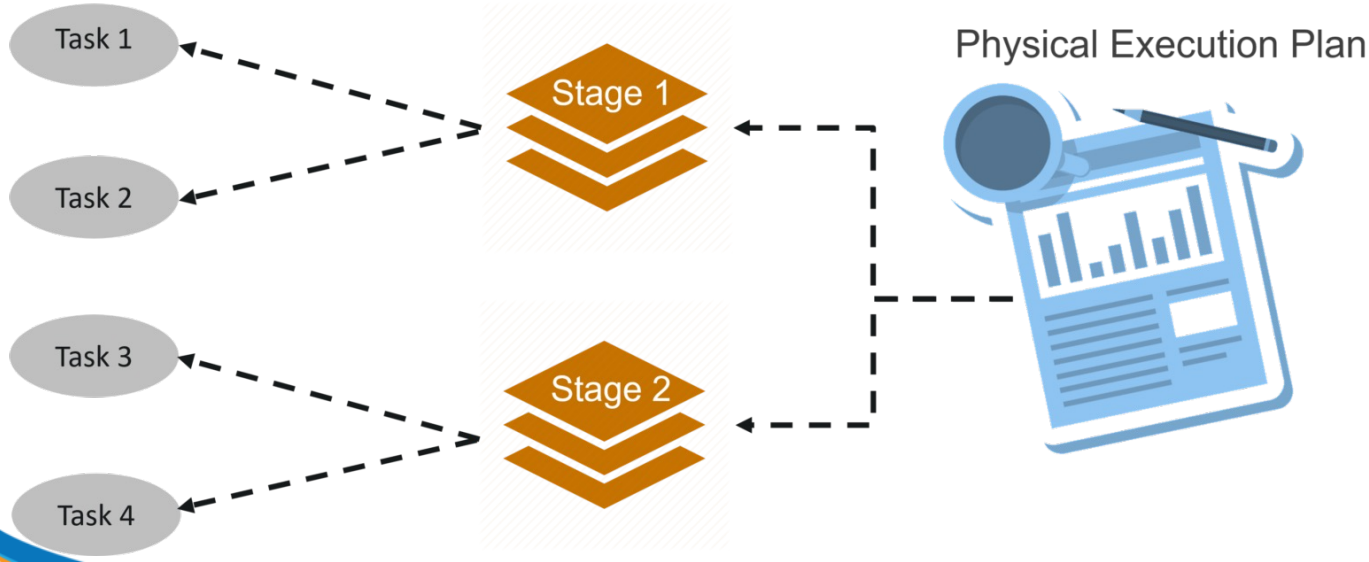
The architecture of Spark

fit@hcmus

DAG: Đồ thị ko có chu trình và có hướng



=> Excutor sẽ quản lý các task
=> Cluster Manager sẽ quản lý các Stage

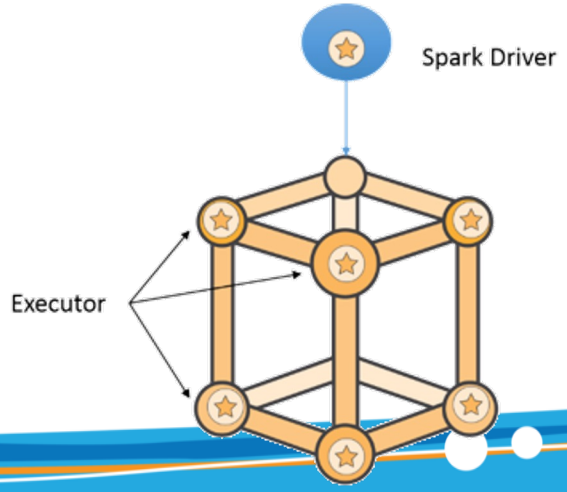


@lnthanh

Spark applications: Driver process

fit@hcmus

- Maintain all relevant information during the lifetime of the application.
- Run the main() function and sits on a node in the cluster
  - Maintain information about the Spark Application
  - Respond to a user's program or input
  - Analyze, distributing, and schedule work across the executors

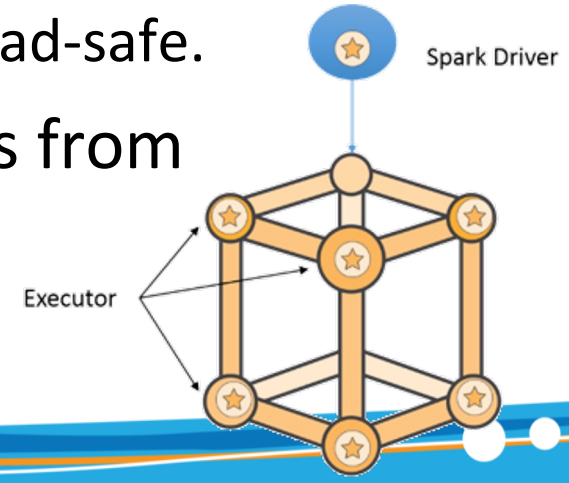


@lnthanh

Spark applications: Executors

fit@hcmus

- Actually carry out the work that the driver assigns them
- Each executor is responsible for only two things
  - Execute code assigned to it by the driver
  - Report the state of the computation back to the driver node
- Executor uses multiple threads to execute the tasks
  - Application code needs to be thread-safe.
- An executor does NOT run tasks from multiple applications



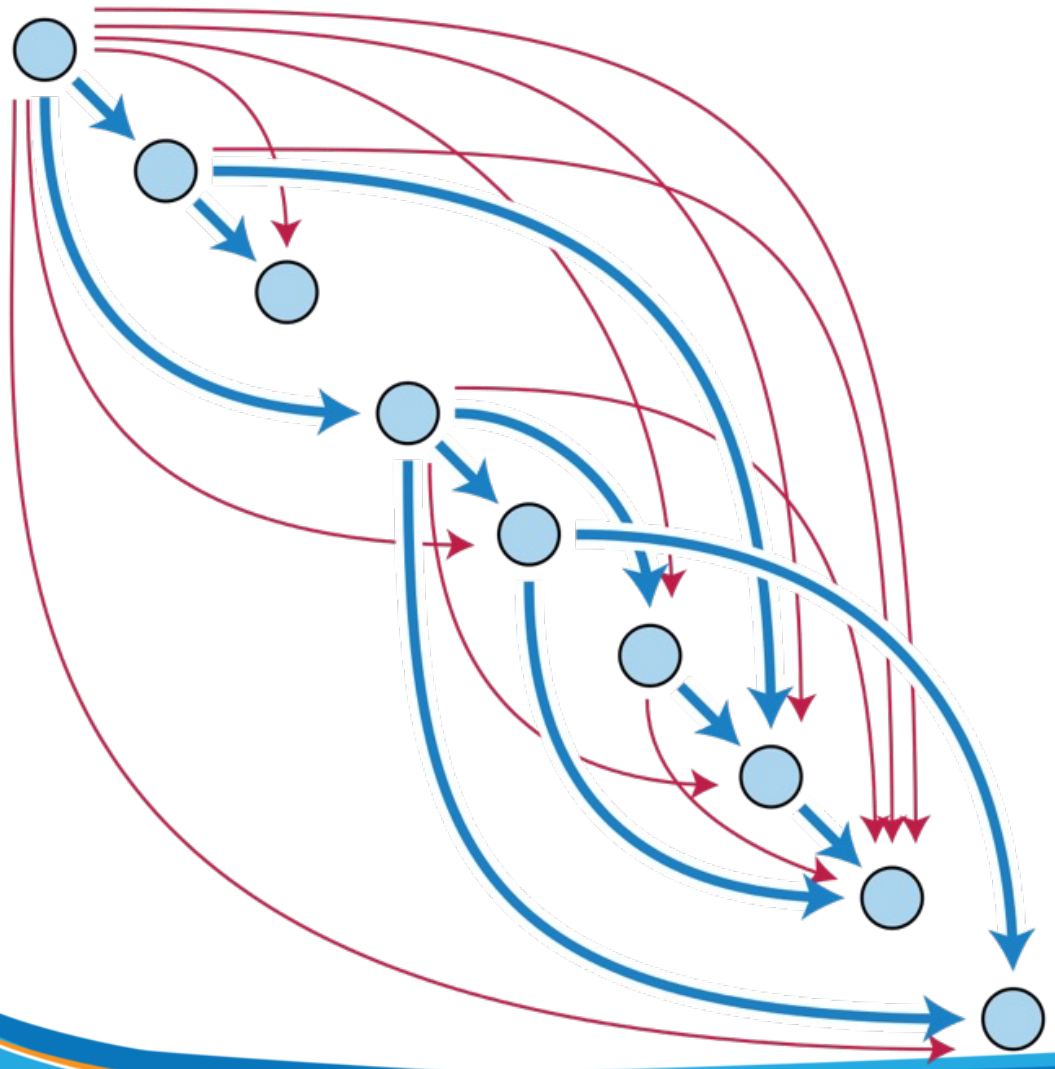
@lnthanh

Spark applications: DAG

fit@hcmus

- MapReduce
  - DAG is a series of map and reduce tasks used to implement the application
  - A developer needs to define each task and chain them together.
- **Spark**
  - The engine itself creates those complex chains of steps from the application's logic
  - Allow the framework to optimize the job as a whole, leading to improved performance.

Spark applications: DAG

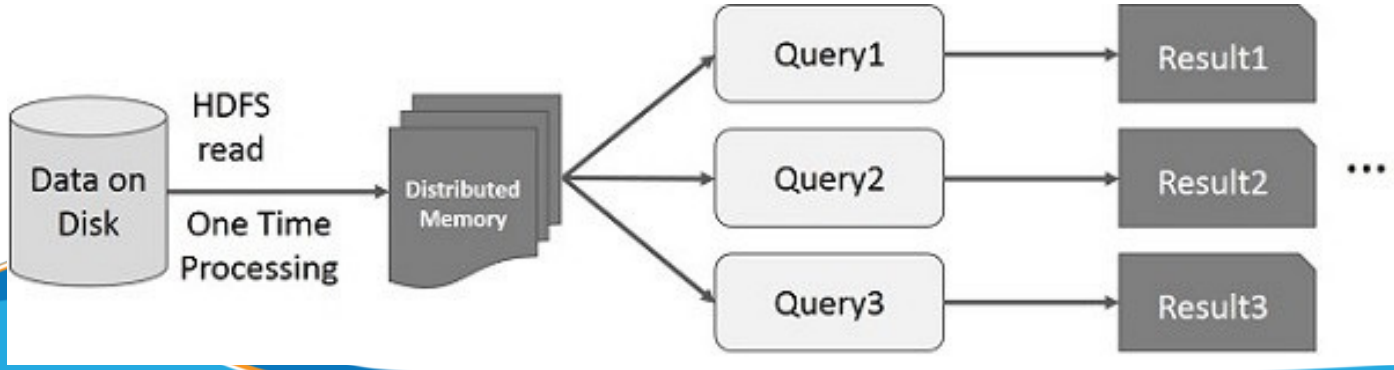


@lnthanh

Resilient Distributed Dataset (RDD)

=> Tất cả DL đổ lên Spark thì nó sẽ cấu trúc theo RDD

- **Resilient Distributed Dataset (RDD)** is the primary data abstraction in Apache Spark and the core of Spark.
- It represents an immutable, partitioned collection of elements that can be operated on in parallel.
- Lets programmers perform in-memory computations on large clusters in a fault-tolerant manner.



Resilient Distributed Dataset (RDD)

fit@hcmus

- Immutable (read-only) in nature and follows lazy transformations (the data inside RDD is not available or transformed until an action is executed that triggers the execution)
- **Resilient**
  - Fault tolerant and is capable of rebuilding data on failure
- **Distributed**
  - Distributed data among the multiple nodes in a cluster
- **Dataset**
  - Collection of partitioned data with values

Resilient Distributed Dataset (RDD)

fit@hcmus

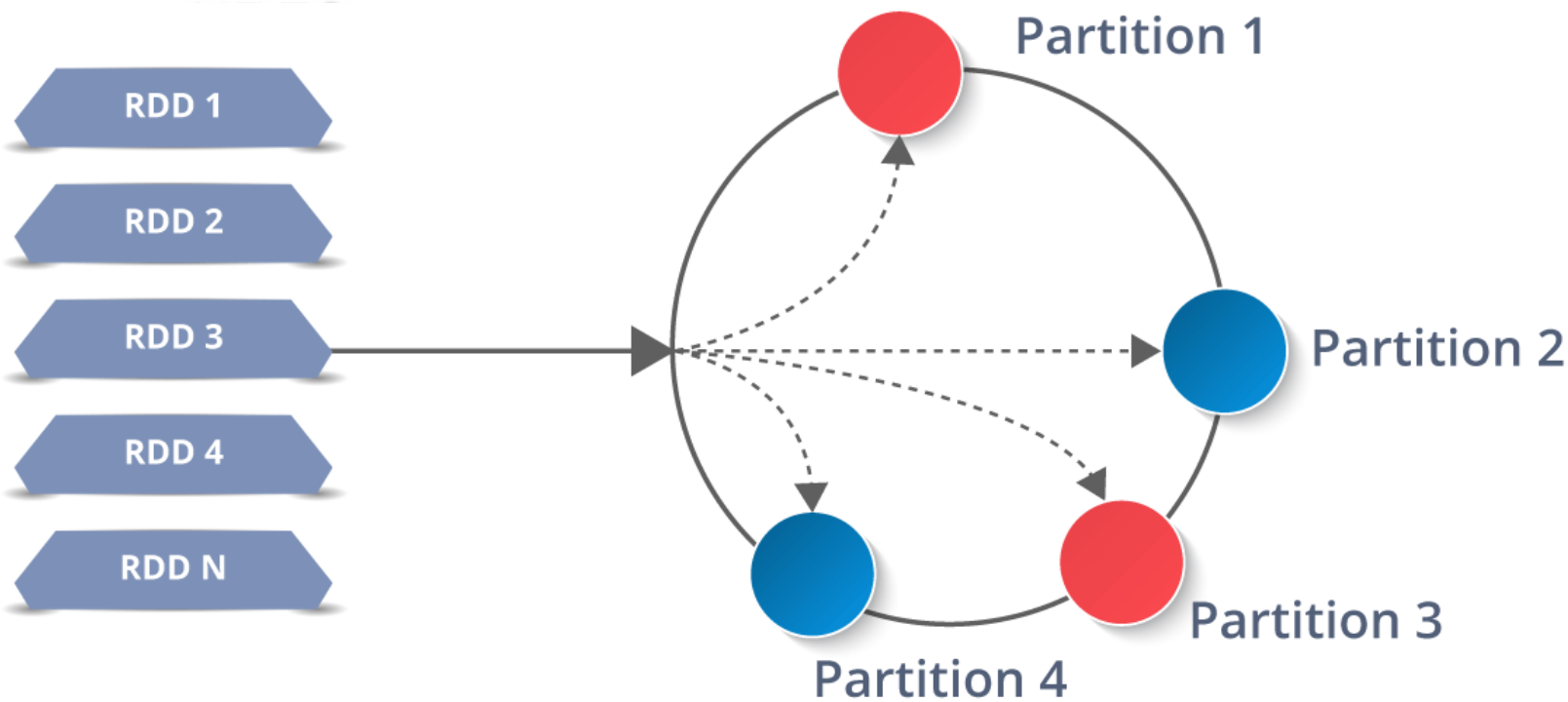
- Each RDD is characterized by the following main properties
  - A list of partitions
  - A function for computing each split
  - A list of dependencies on other RDDs => Nếu RDD nào ko phụ thuộc thì có thể chạy song song
- Also optionally,
  - A Partitioner for key-value RDDs
    - E.g., to say that the RDD is hash-partitioned
  - A list of preferred locations on which to compute each split
    - E.g., block locations for a HDFS file

@lnthanh

RDDs: Creating RDDs

fit@hcmus

- Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster.



Partitions are the units of parallelism. You can control the number of partitions of a RDD using repartition or coalesce transformations.

@lnthanh

RDDs: Creating RDDs

fit@hcmus

- Parallelize an existing collection in your driver program

```
# in Python
myCollection = "Spark The Definitive Guide : Big Data Processing Made Simple".split(" ")
words = spark.sparkContext.parallelize(myCollection, 2)  => Đây là 1 RDD
```

```
# in Python
spark.sparkContext.textFile("/some/path/withTextFiles")
Or
```

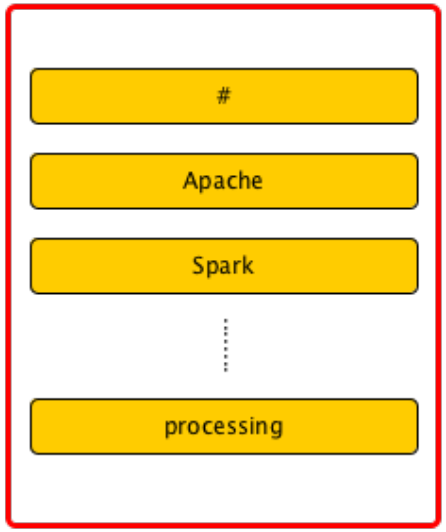
- Reference a dataset in an external storage system, such as a shared file system, HDFS, HBase, etc.

```
spark.sparkContext.wholeTextFiles("/some/path/withTextFiles")
```

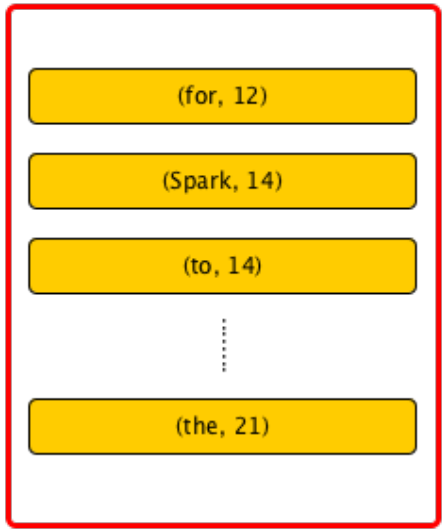
RDDs: Examples of RDDs

fit@hcmus

RDD of Strings

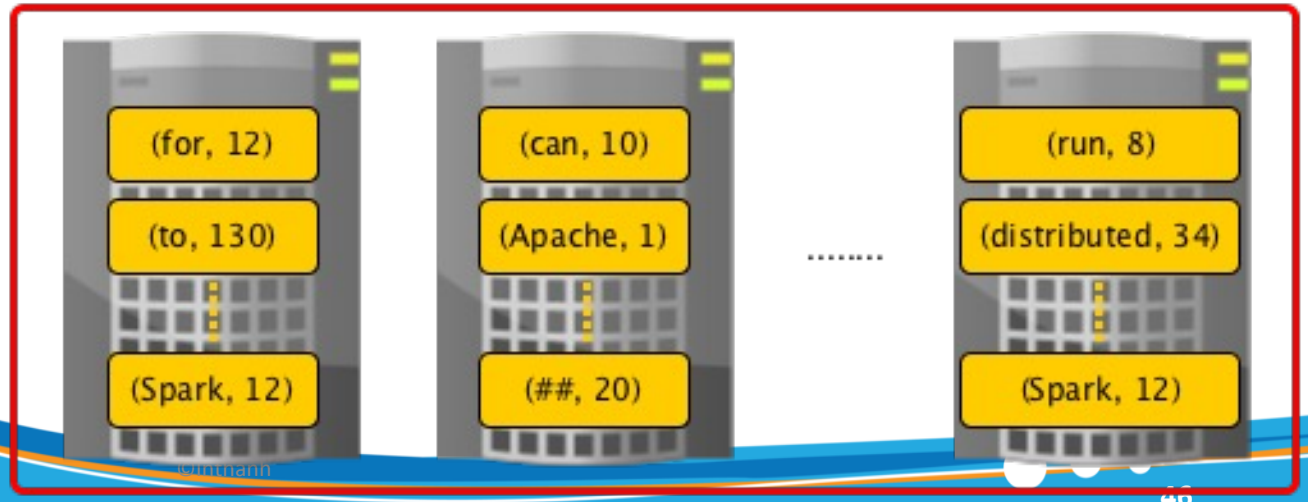


RDD of Pairs



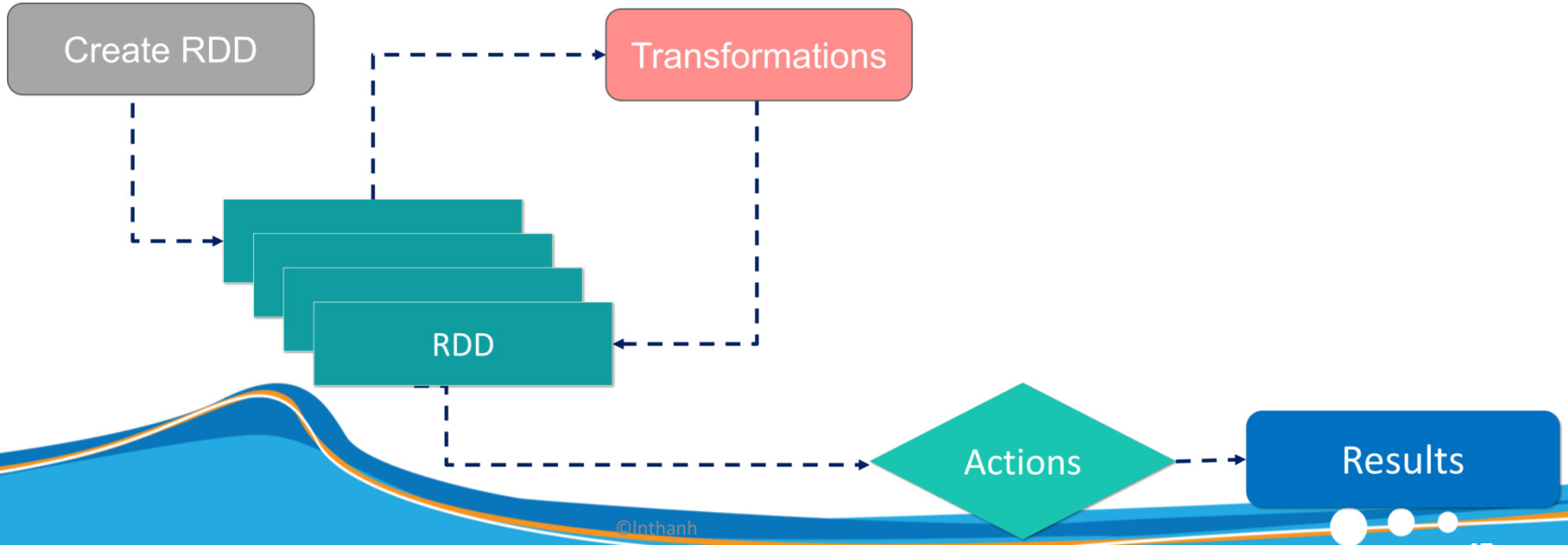
Đa số DL này đều nằm trên RAM (cache)

distributed and partitioned RDD



RDDs: Operations on RDDs

- Transformations are operations that are applied to create a new RDD
- Actions applied on an RDD to instruct Spark to apply computation and pass the result back to the driver.



@lnthanh

Operations on RDDs

=> Khi nào ta gọi lệnh Action thì các lệnh liên quan đến Transformation mới đc thực hiện

Transformations	Actions
map(func)	take(N)
flatMap(func)	count()
filter(func)	collect()
groupByKey()	reduce(func)
reduceByKey(func)	takeOrdered(N)
mapValues(func)	top(N)
...	...

References

- Spark Tutorial: A Beginner's Guide to Apache Spark

  https://www.edureka.co/blog/spark-tutorial/

- Apache Spark Architecture – Spark Cluster Architecture Explained

  https://www.edureka.co/blog/spark-architecture/

- Submitting User Applications with spark-submit

  https://aws.amazon.com/blogs/big-data/submitting-user-applications-with-spark-submit/

THE END