Introduction to Big Data

APACHE SPARK

Le Ngoc Thanh – Nguyen Ngoc Thao

{lnthanh, nnthao}@fit.hcmus.edu.vn

Outline

- An introduction to Apache Spark
- Spark ecosystem
- How to run Spark on a cluster

The need of faster analytics

- Data is increasing in different aspects.

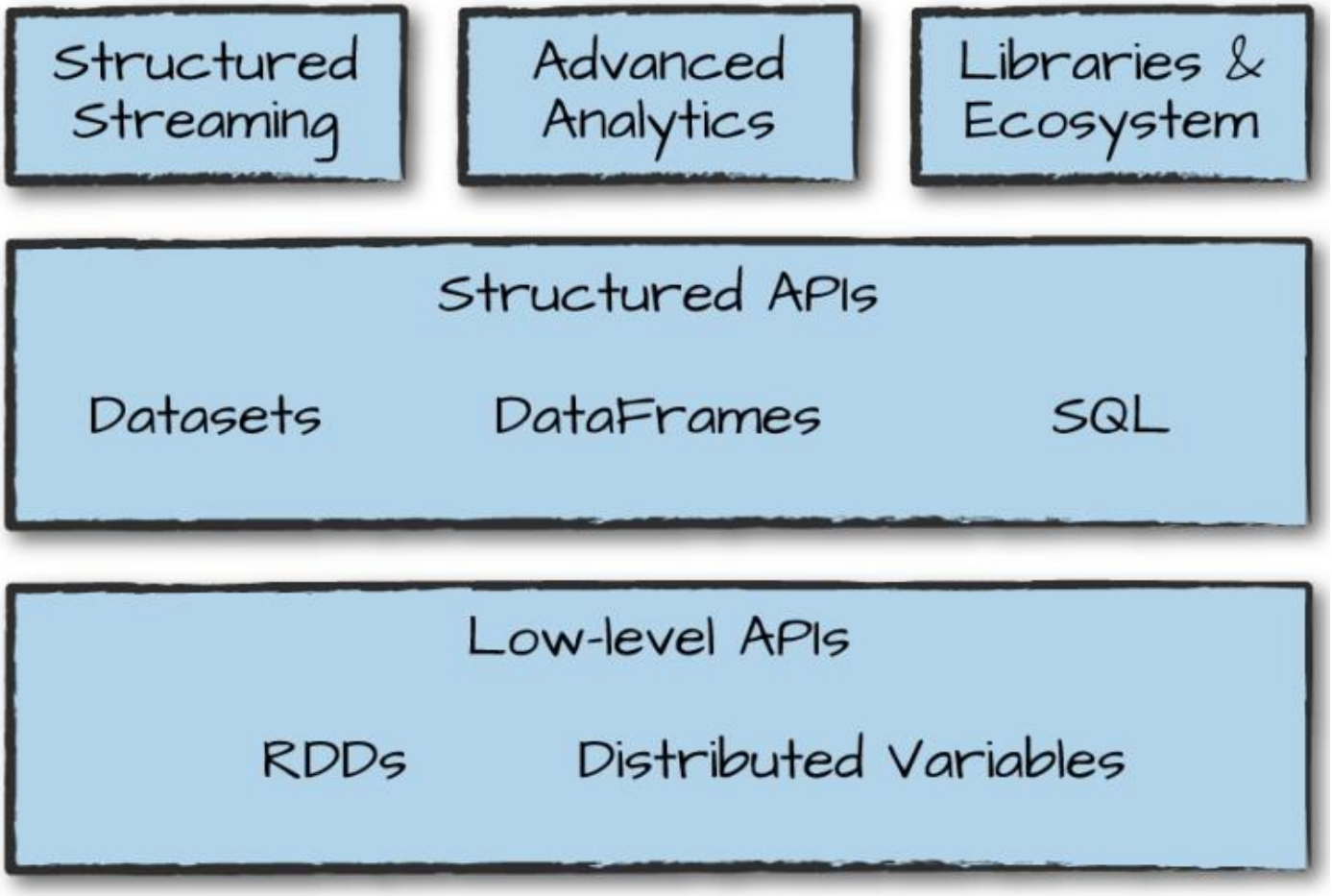
- The need of faster analytics results is increasingly important.

MapReduce

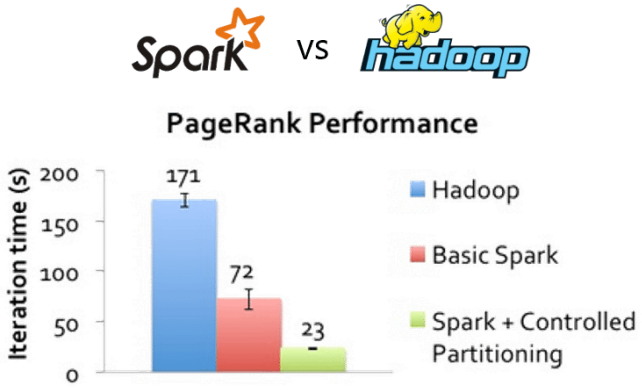
Spark

What is Apache Spark?

- A unified computing engine and a set of libraries for parallel data processing on computer clusters



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

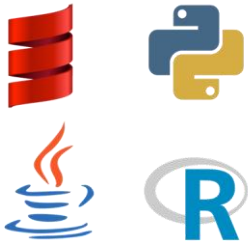
- In-memory computation: real-time, low latency
- Several computational models supported
- Massive scalability in clusters with thousands of nodes

Spark hadoop

Hadoop Integration

- Run on top of an existing Hadoop cluster with highly smooth compatibility

Features of Apache Spark



Polyglot

- High-level APIs in Java, Scala, Python and R
- Shells for Scala and Python

Lazy evaluation

- Delay the evaluation till it is necessary → key factor to speed

Machine learning

- A powerful and unified engine for big data processing
- High performance and easy to use

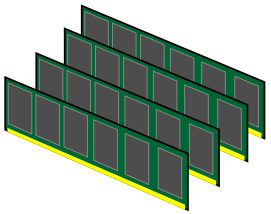
Multiple Format

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

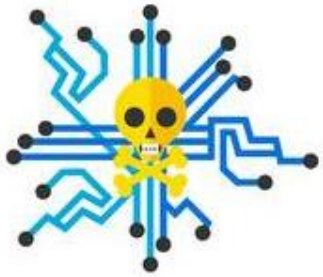
Who use Spark and Why?

Parallel distributed processing

Scalability



In-memory computing



Fault tolerance on commodity hardware

High level APIs

save

. . . .

Who use Spark and Why?

Data Scientist

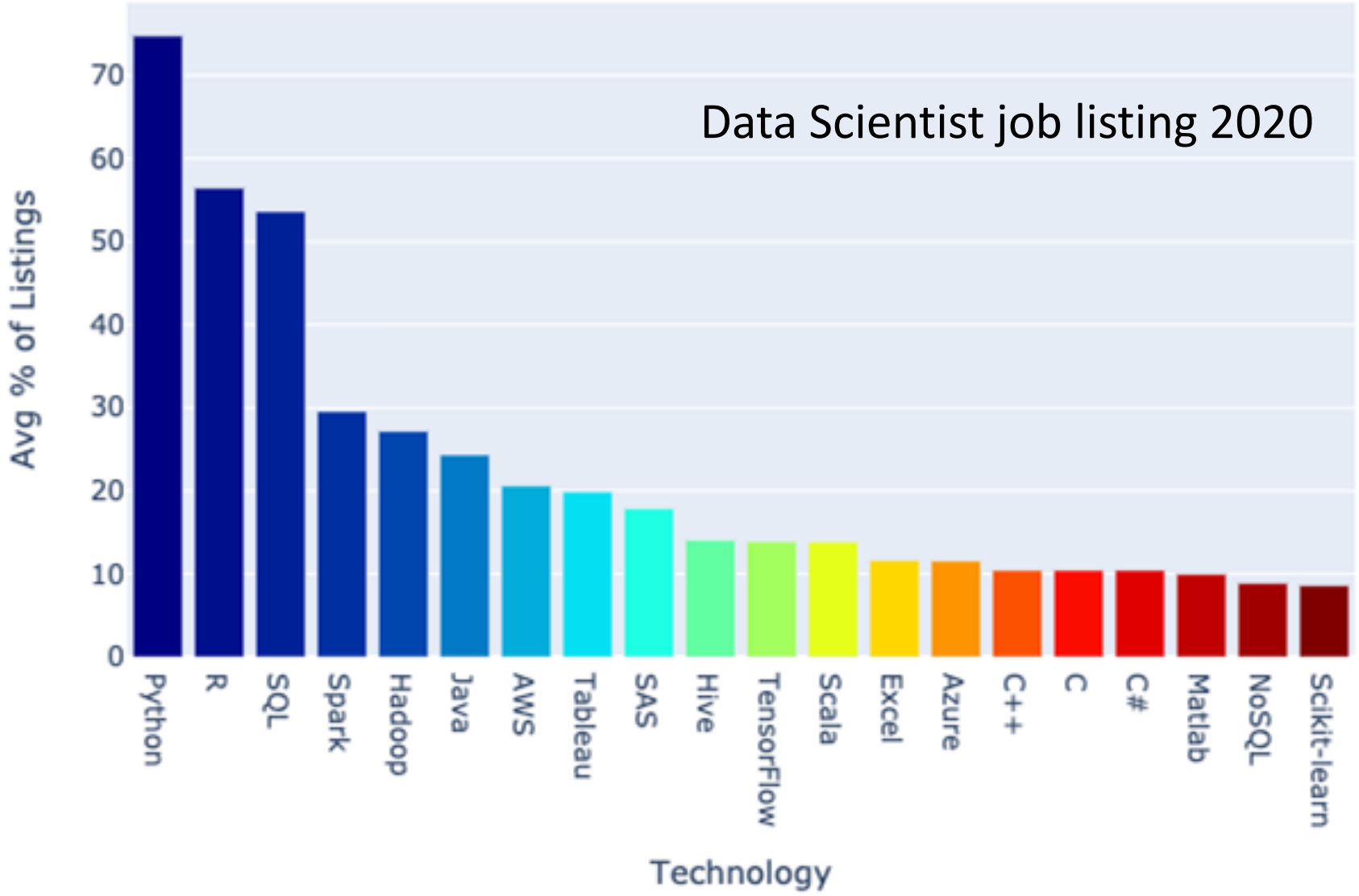
- Analyze and model the data to obtain insight
- Transforming the data into a useable format
- Statistics, machine learning, SQL

Data Engineer

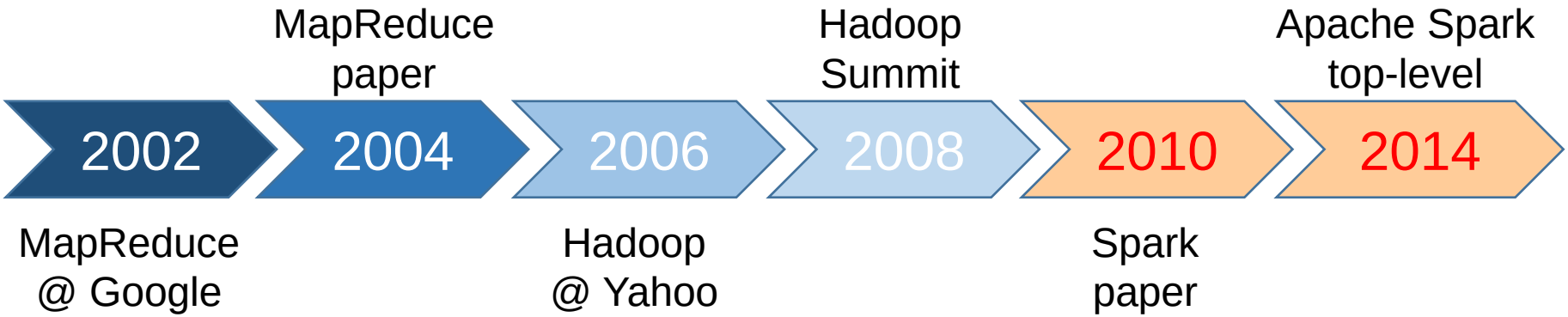
- Develop a data processing system or application
- Inspect and tune their applications
- Programming with the Spark's API

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

The popularity of Spark

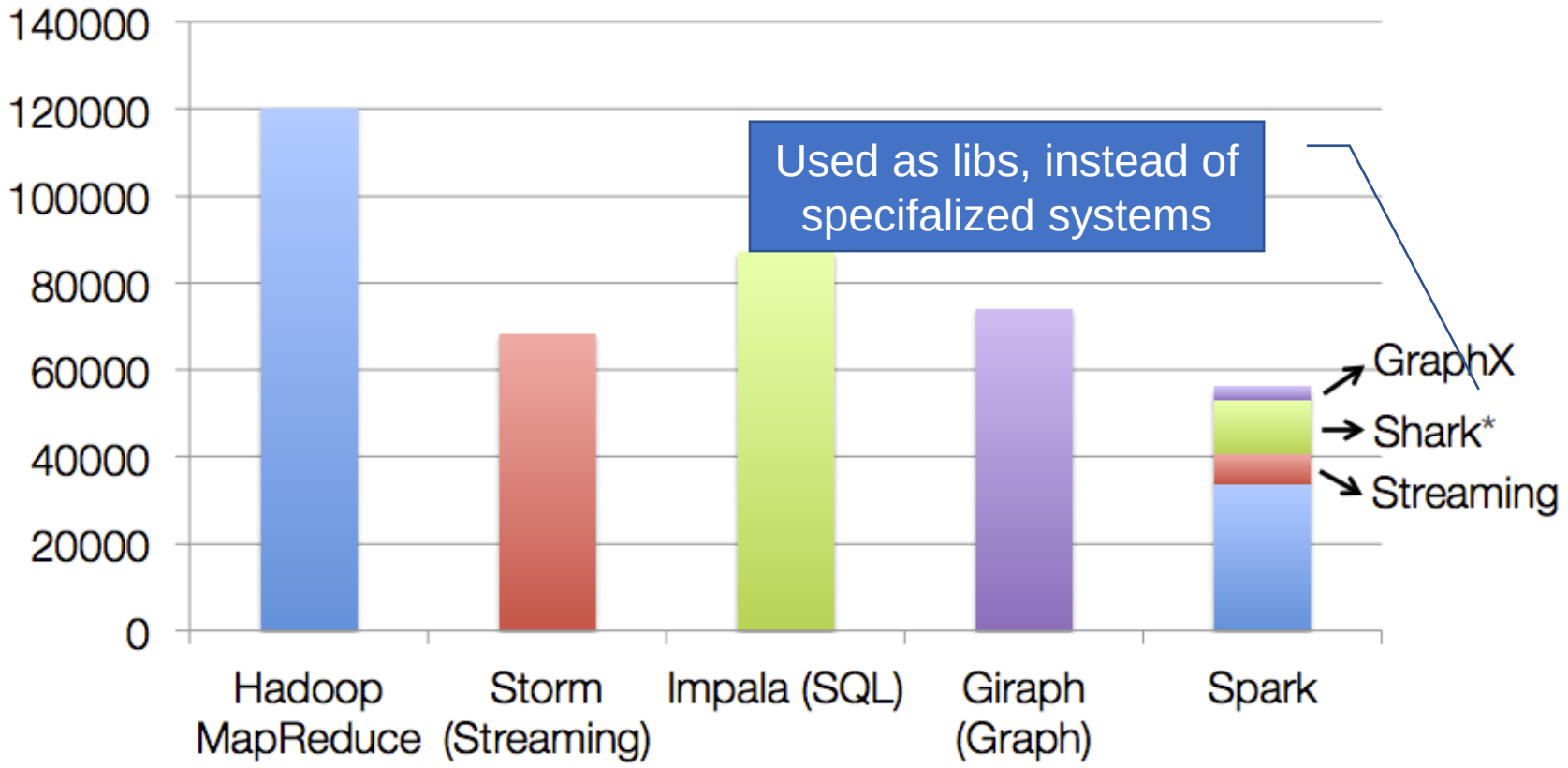


A brief history of Spark

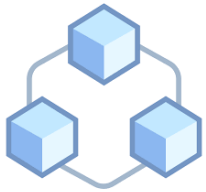


- MapReduce started a general batch processing paradigm
- Two limitations
  - Difficulty programming in MapReduce
  - Batch processing did not fit many use cases
- A lot of specialized systems (Storm, Impala, Gigraph, etc.)

Code sizes with various systems

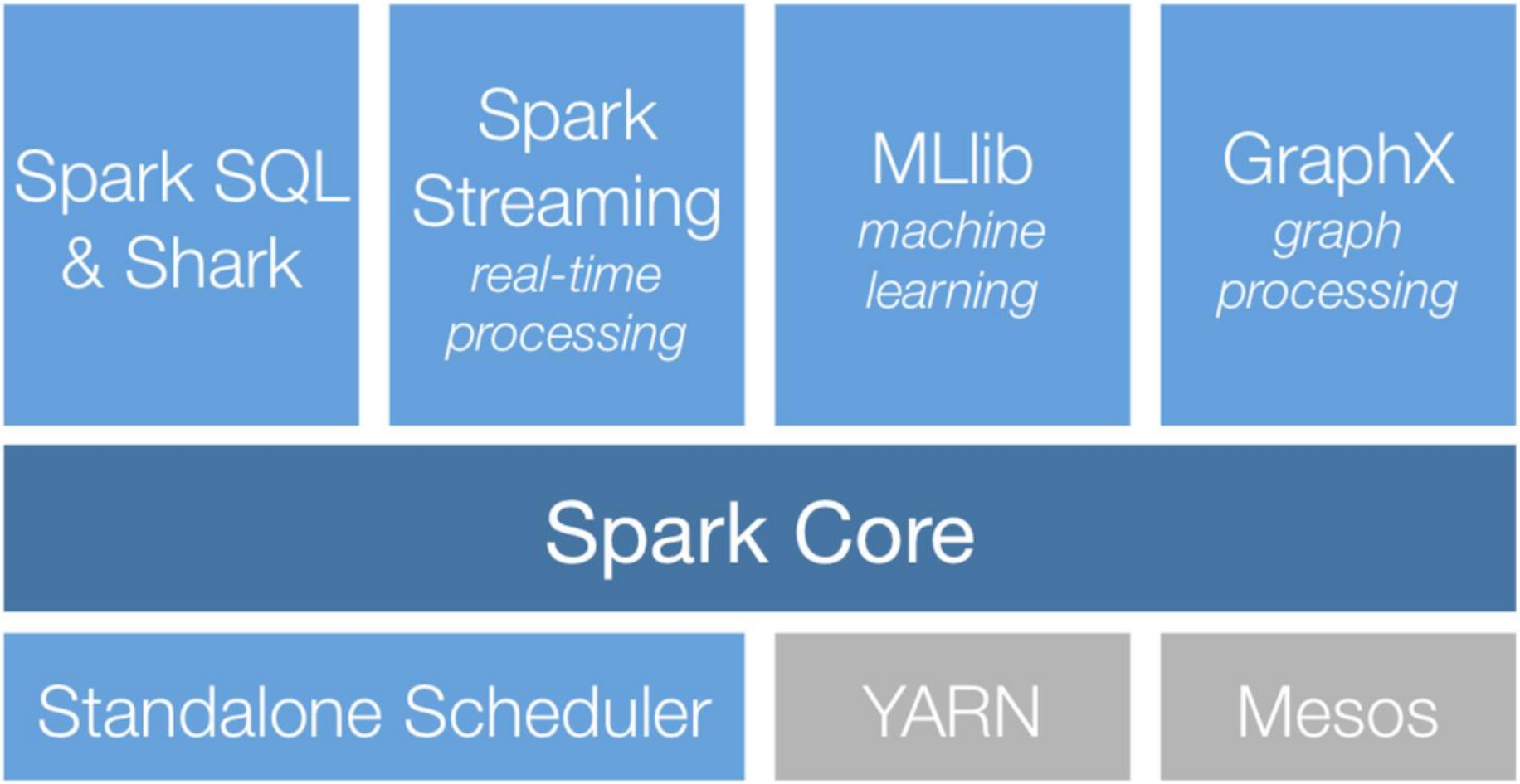


The State of Spark, and Where We're Going Next, Matei Zaharia, Spark Summit (2013)



Spark ecosystem modules

Spark unified stack

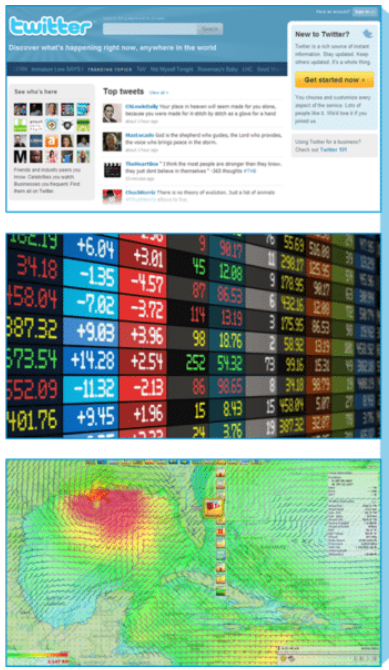


Spark Core

- 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, and interacting with storage systems
- Spark focuses on performing computations over the data, no matter where it resid

Spark Streaming

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



Spark Streaming

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.

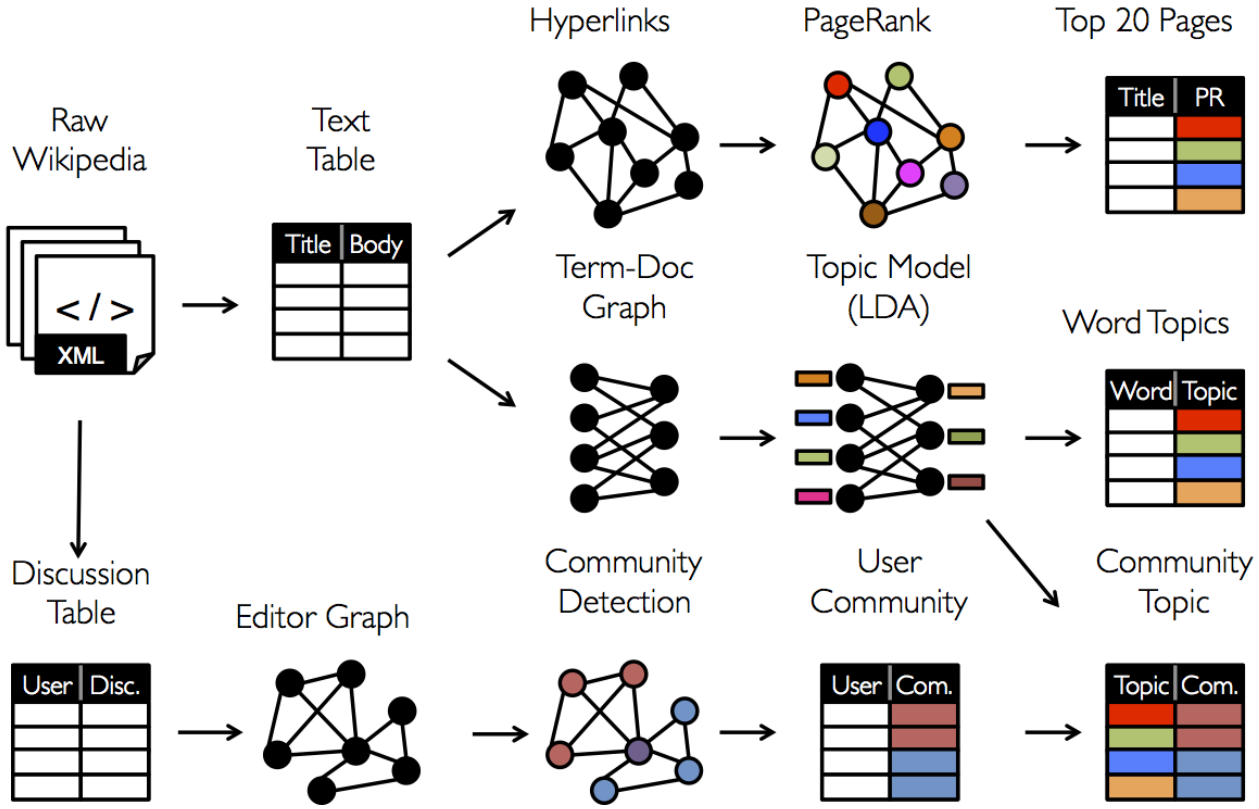
Spark SQL

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

A Spark SQL process uses all the four libraries in sequence.

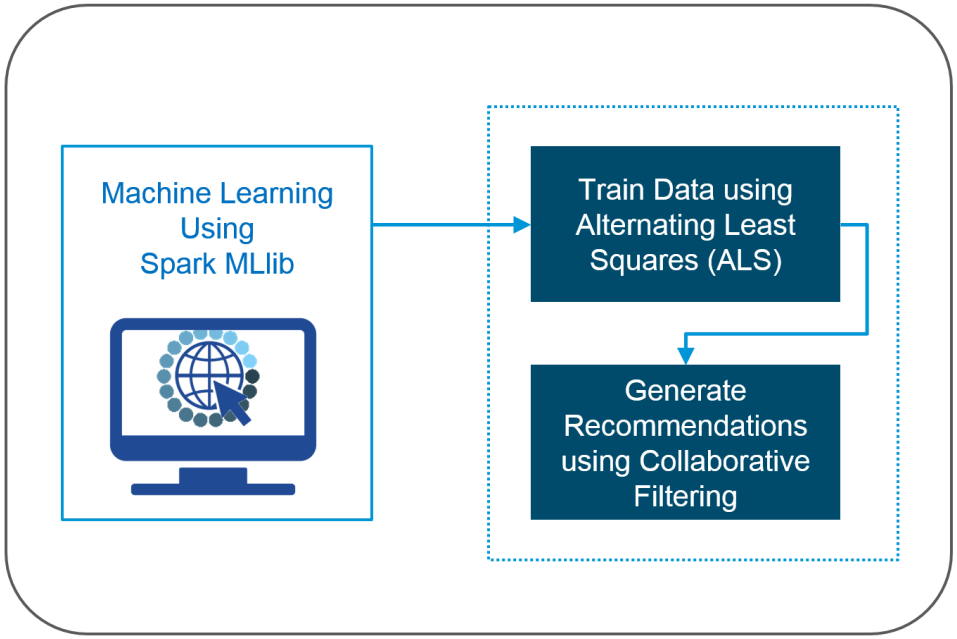
Spark GraphX

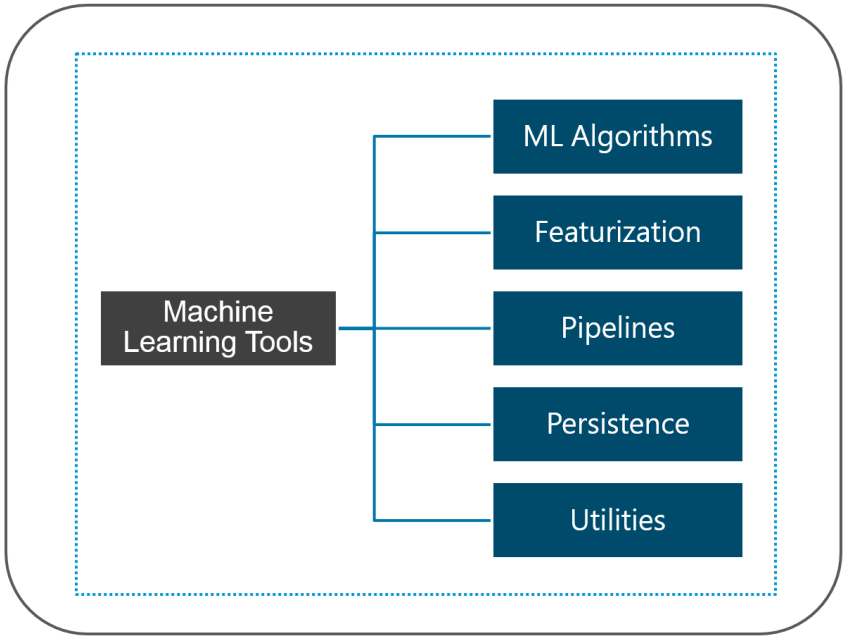
- Include a growing collection of graph algorithms and builders to simplify graph analytics tasks
- Support an optimized variant of the Pregel API



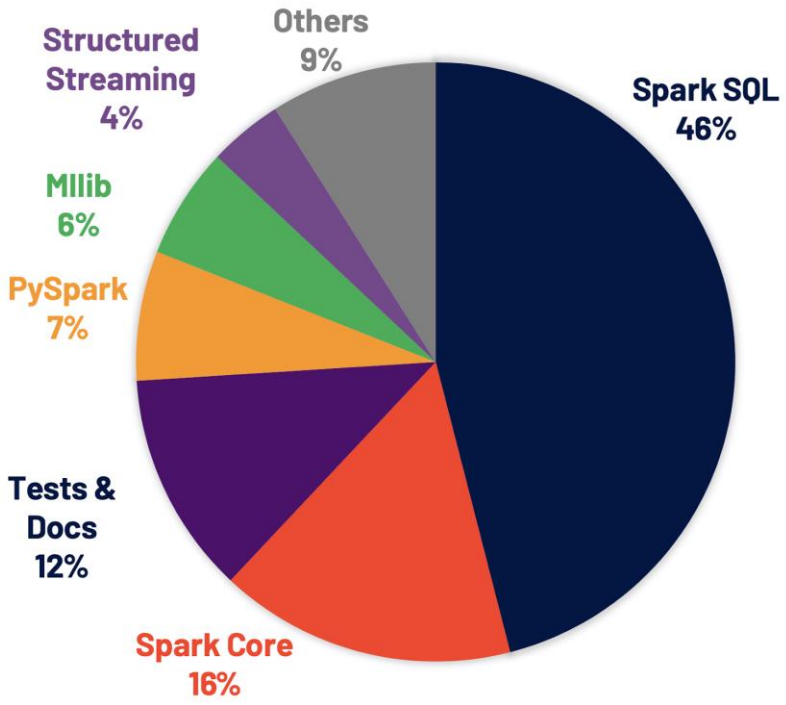
Spark MLlib

- A **M**achine **L**earning **lib**rary that supports various machine learning tasks in Spark

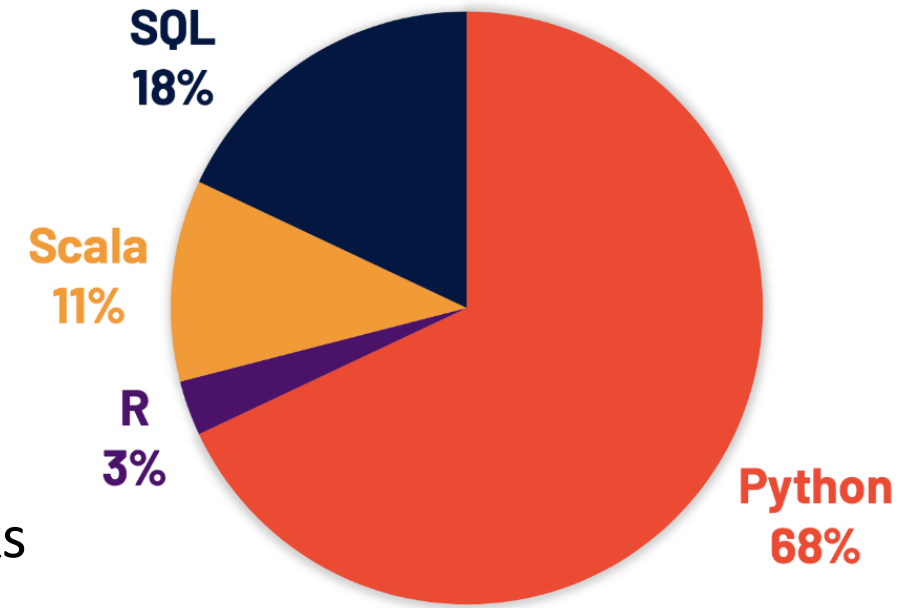




Spark usage statistics



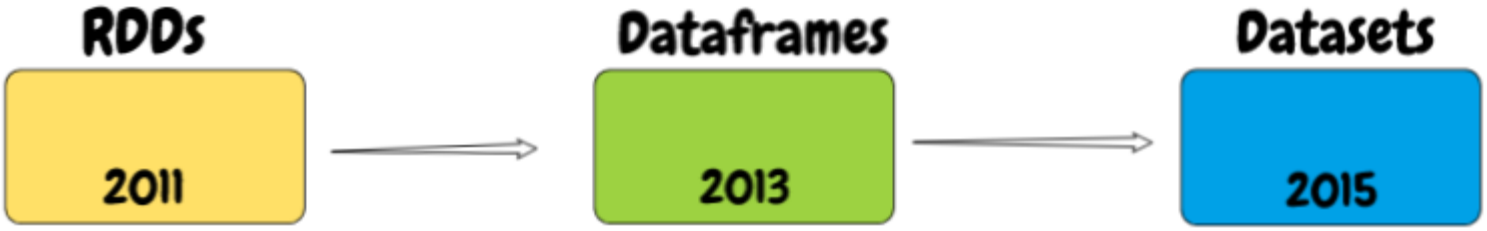
The popularity of modules in Spark

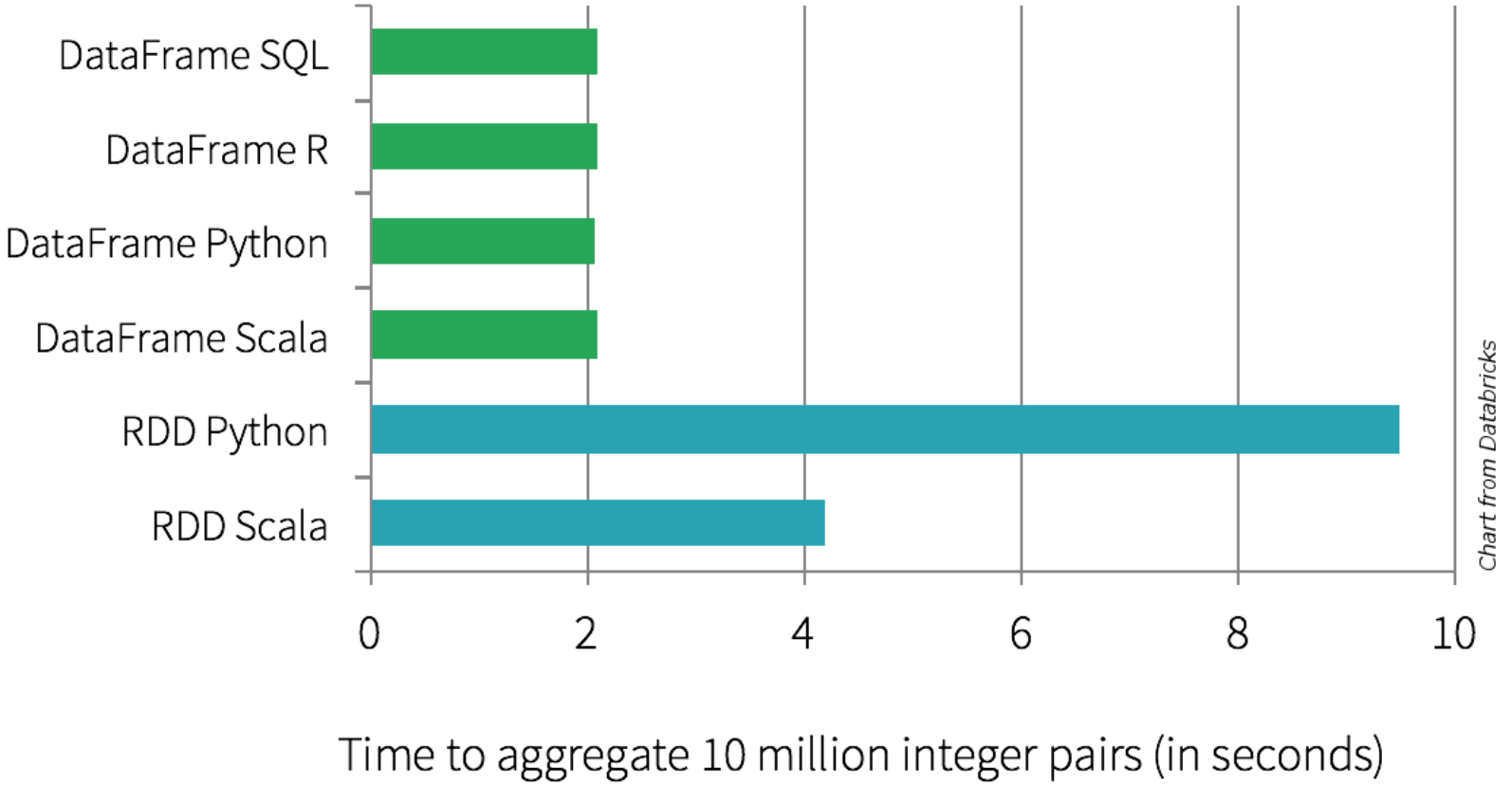


Language used in notebooks

Image credit: Databricks

Spark APIs


RDDs
2011
Dataframes
2013
Datasets
2015


DataFrame SQL
DataFrame R
DataFrame Python
DataFrame Scala
RDD Python
RDD Scala
0
2
4
6
8
10
Chart from Databricks
Time to aggregate 10 million integer pairs (in seconds)

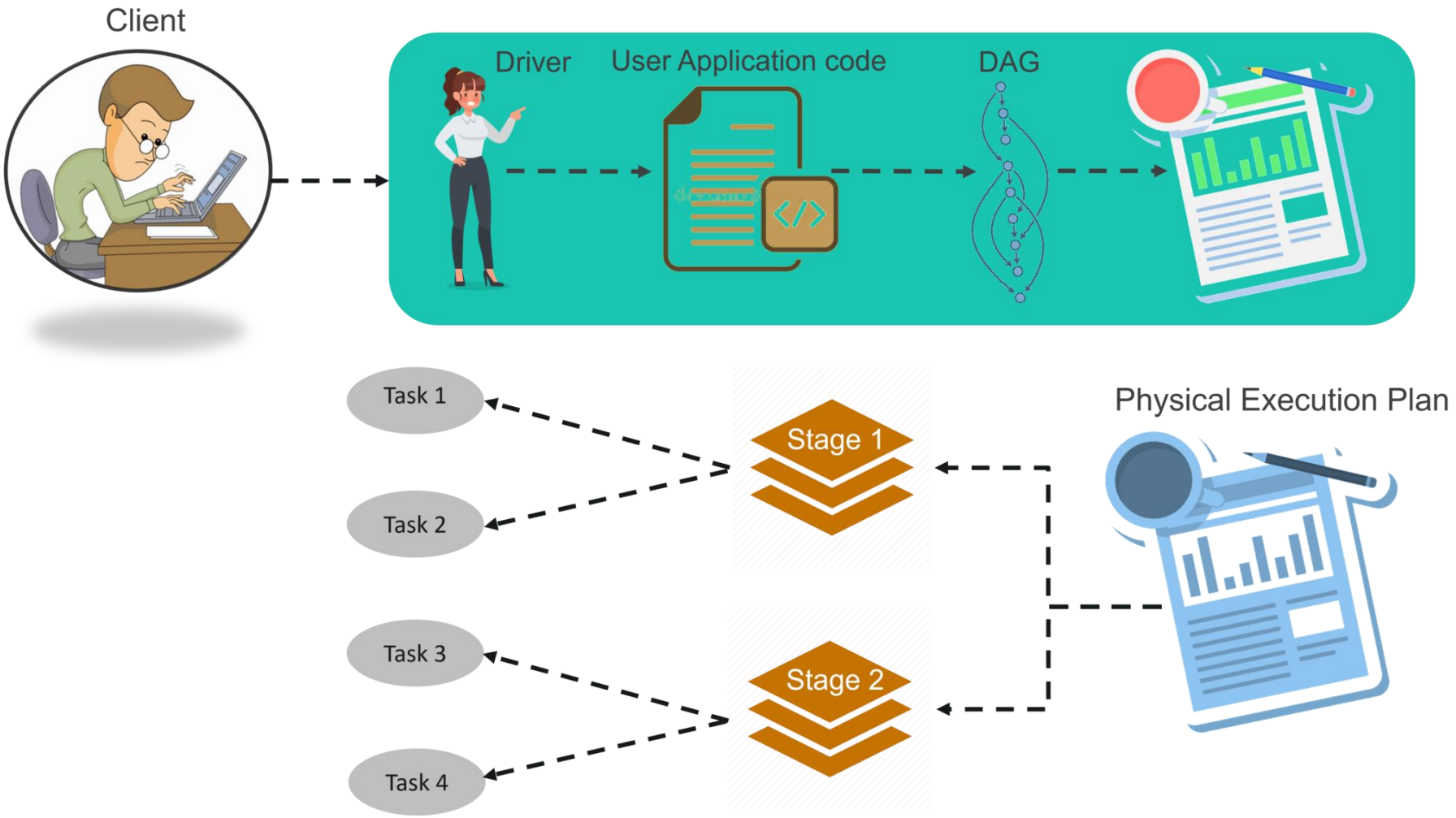
Apache Spark 3.0

2x performance improvement on TPC-DS over Spark 2.4, enabled by adaptive query execution, dynamic partition pruning and other optimizations
ANSI SQL compliance
Important improvements in pandas APIs, including Python type hints and pandas UDFs
Better Python error handling, simplifying PySpark exceptions
New UI for structured streaming
Up to 40x speedups for calling R user-defined functions
Over 3,400 Jira tickets resolved
No major code changes are required to adopt this version.

Source

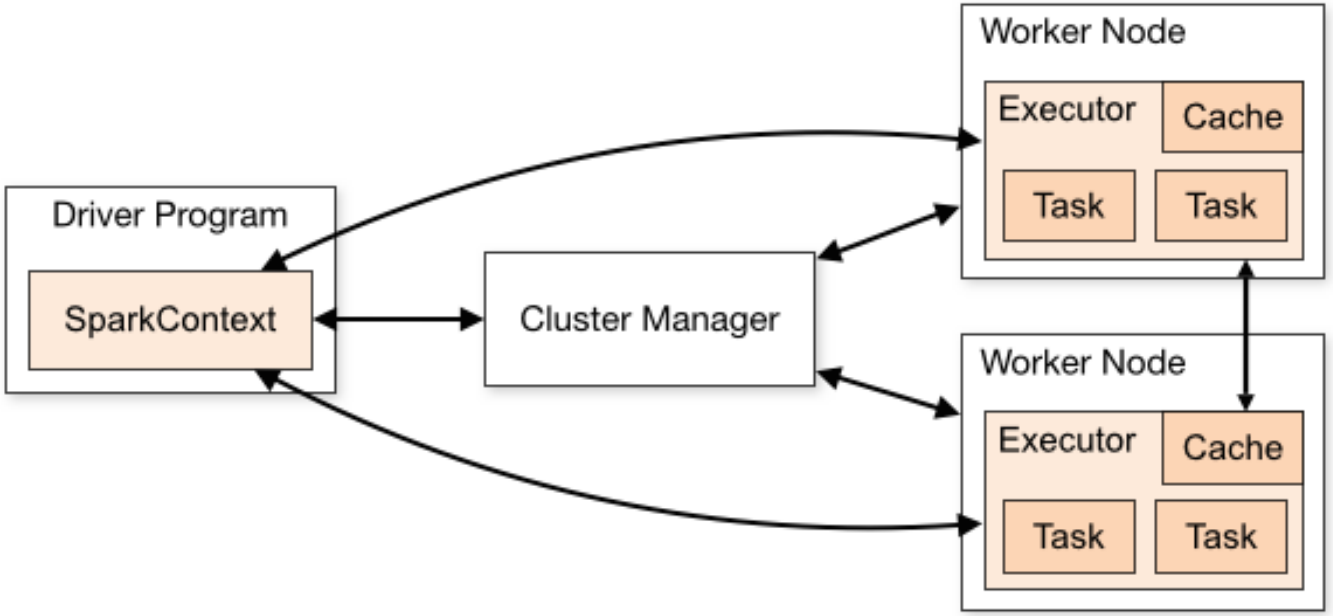
How Spark runs on a cluster

Spark architecture



Spark execution model

- A Spark application has a single driver process and a set of executor processes distributed across the hosts in a cluster.



Spark execution model

- A job is a parallel computation of multiple tasks that gets spawned in response to a Spark action.
  - The driver converts the application into one or more Spark jobs, each of which corresponds to a DAG.
- Each job gets divided into a set of stages that depend on each other.
- Each stage is comprised of tasks (unit of execution), which are then federated across each executor.
  - Each task maps to a single core and works on a single partition of data.
  - E.g., an executor with 16 cores can have 16 or more tasks working on 16 or more partitions in parallel

Spark execution model

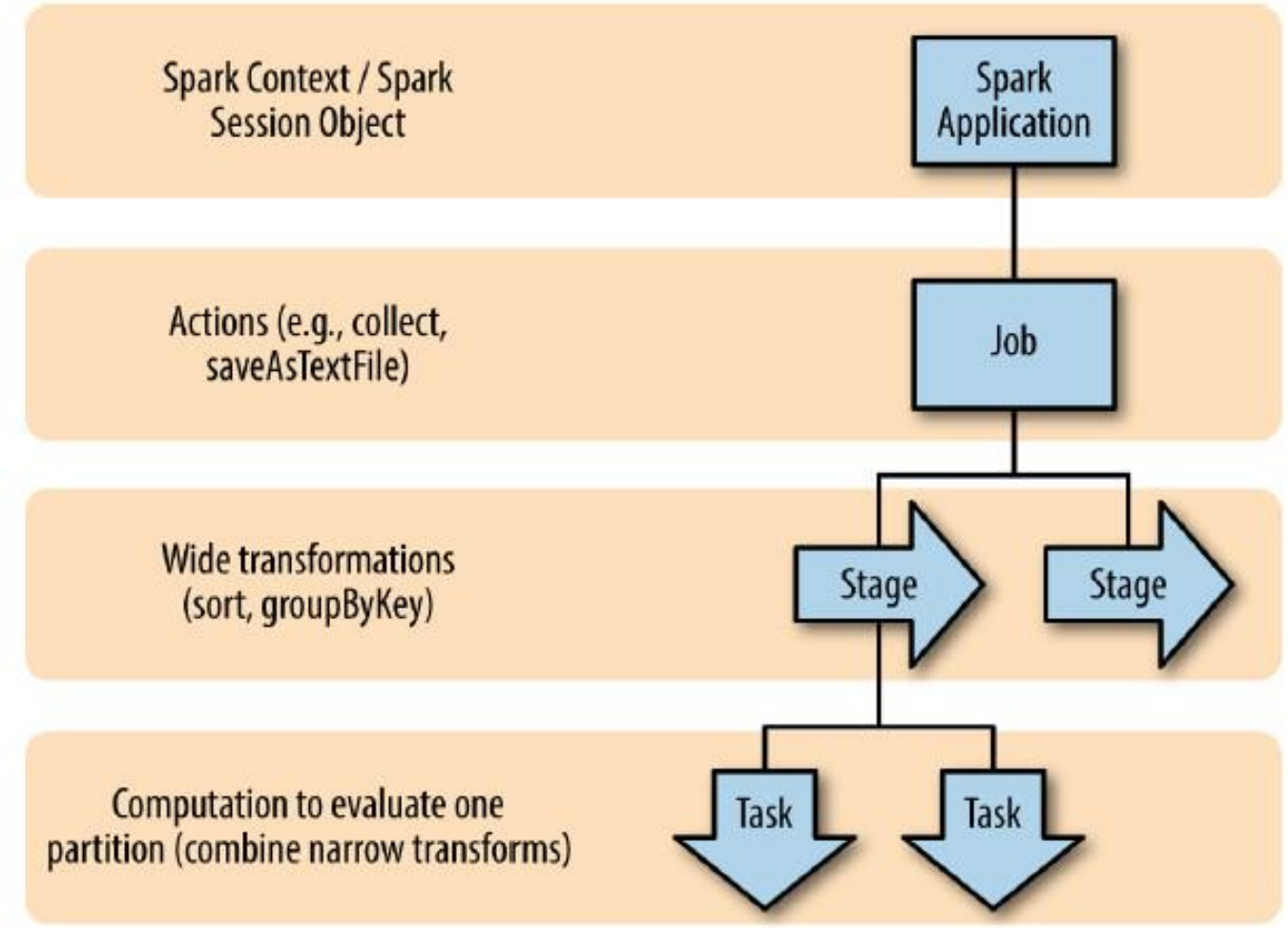
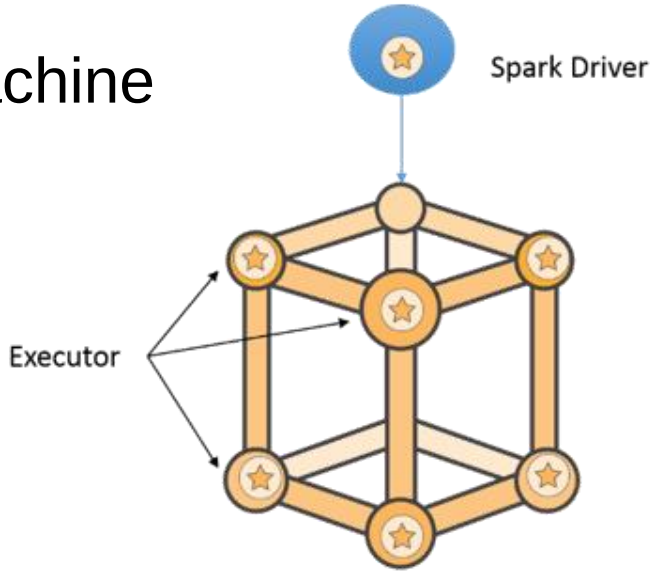


Image was from excellent book "High Performance Spark" by Holden Karau & Rachel Warren

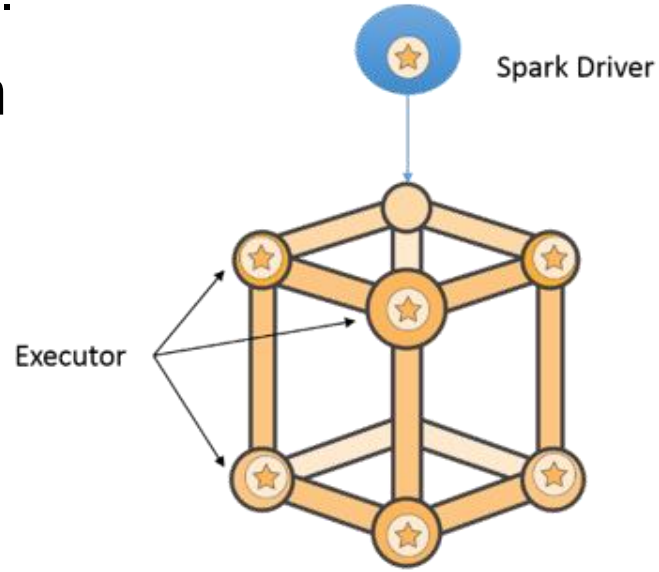
Spark execution model: Driver

- Maintain all information during the lifetime of an application running on the cluster
  - Keep track all the state and tasks of the executors
  - Interface with the cluster manager in order to get physical resources and launch executors.
- Appear as a process on a physical machine



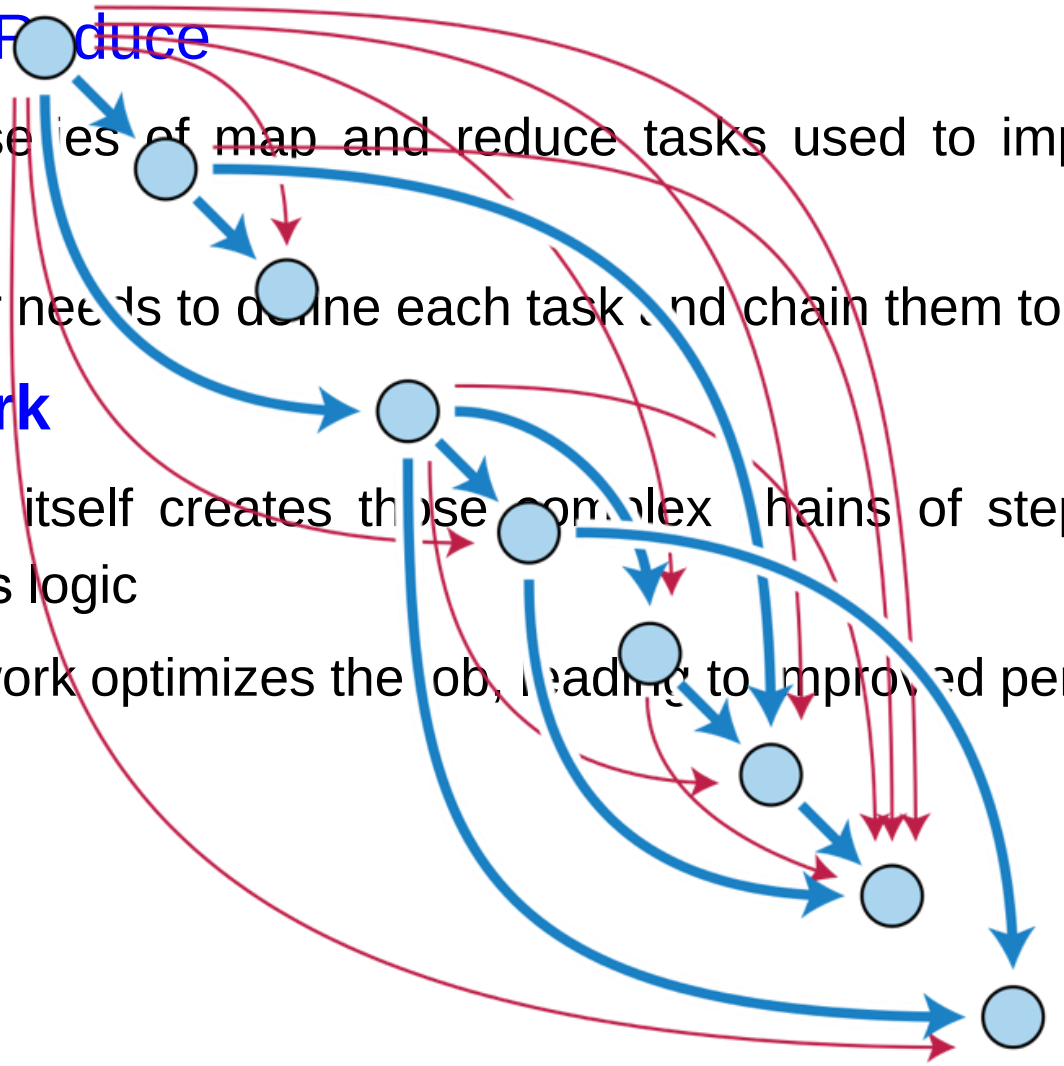
Spark execution model: Executors

- Carry out the work that the driver assigns to
  - Execute code assigned to it by the driver
  - Report the state of the computation back to the driver node
- Use multiple threads to execute a number of tasks
  - Application code needs to be thread-safe.
- An executor does NOT run tasks from multiple applications



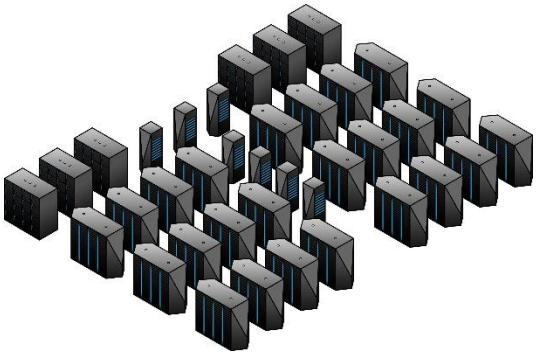
DAG for a Spark application

- Hadoop 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.
- **Apache Spark**
  - The engine itself creates those complex chains of steps from the application's logic
  - The framework optimizes the job, leading to improved performance

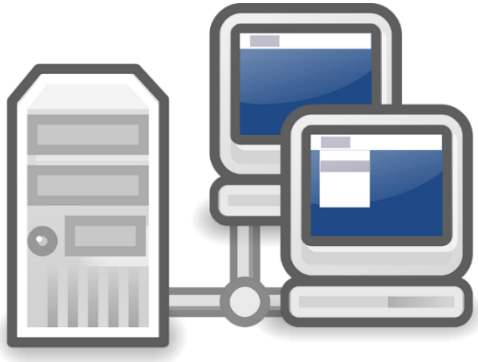


Execution modes

- An execution mode determines where the resources are physically located when running an application.
- Spark currently supports three following modes that can be specified using
  - -deploy-mode option of spark-submit, or
  - spark.submit.deployMode Spark property



Cluster mode

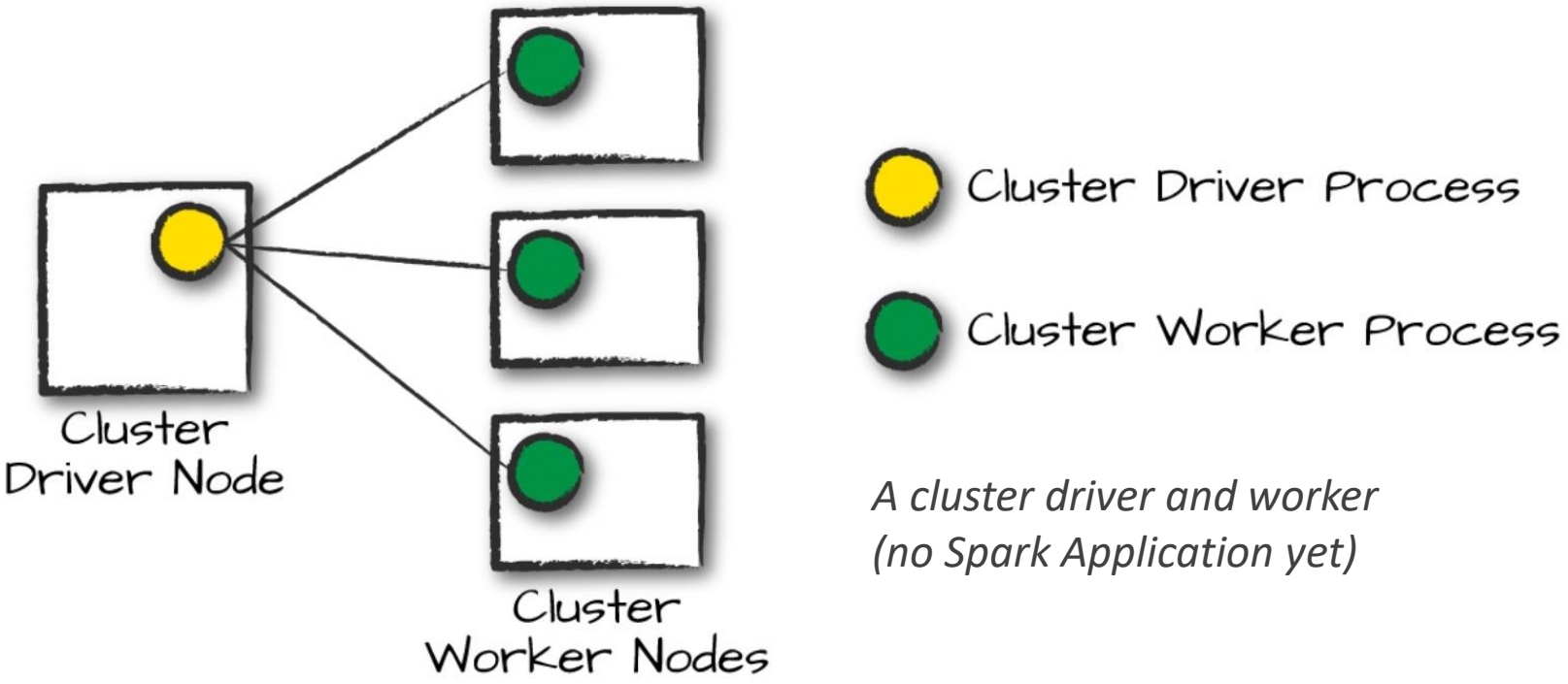


Client mode

Local mode

Cluster manager

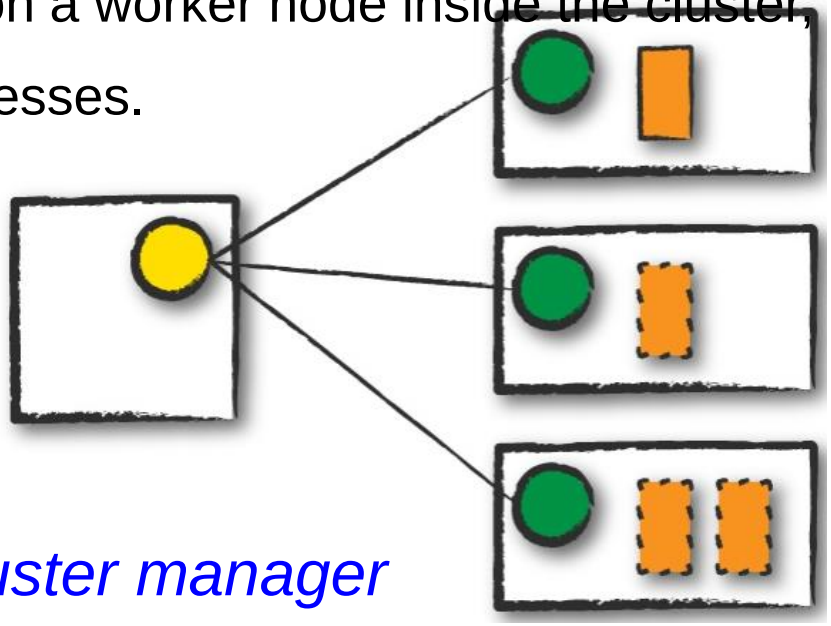
- A cluster manager has its own “driver” (sometimes called master) and “worker” abstractions
  - These are tied to physical machines rather than processes.
- Maintain a cluster of machines that run Spark applications



A cluster driver and worker (no Spark Application yet)

Execution modes: Cluster mode

- The most common way of running Spark applications
- The cluster manager governs all Spark application processes
  - A user submits a pre-compiled JAR, Python script, or R script to it.
  - It launches the driver process on a worker node inside the cluster, in addition to the executor processes.



Hadoop YARN

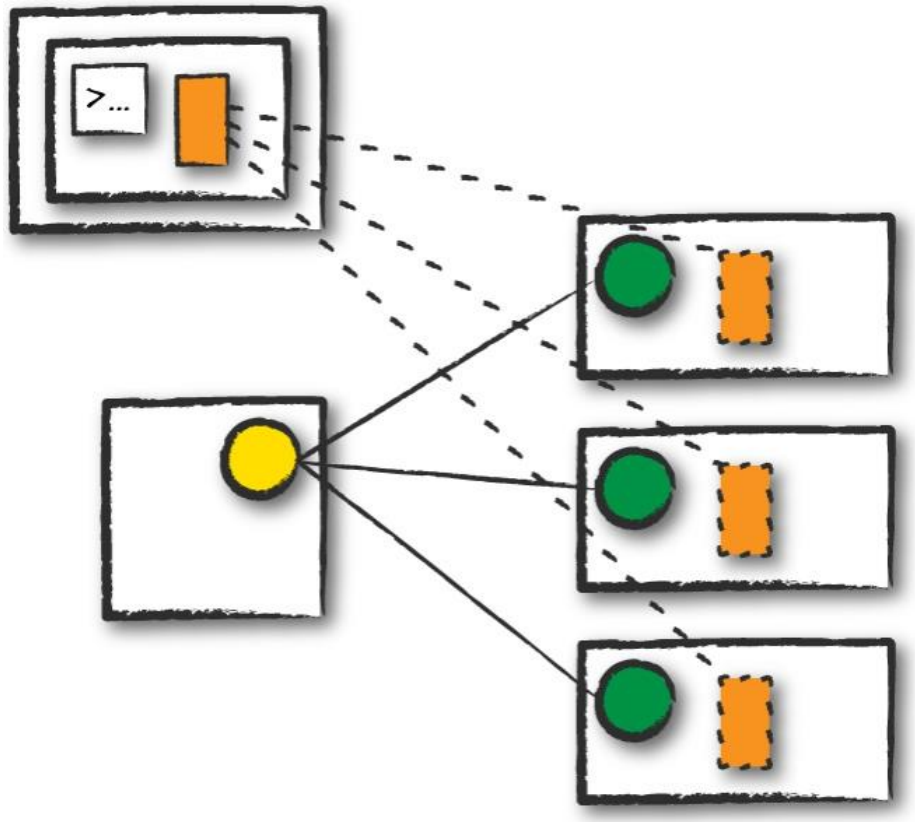
Apache Mesos

Kubernetes, Amazon EC2

Spark's built-in Standalone cluster manager

Execution modes: Client mode

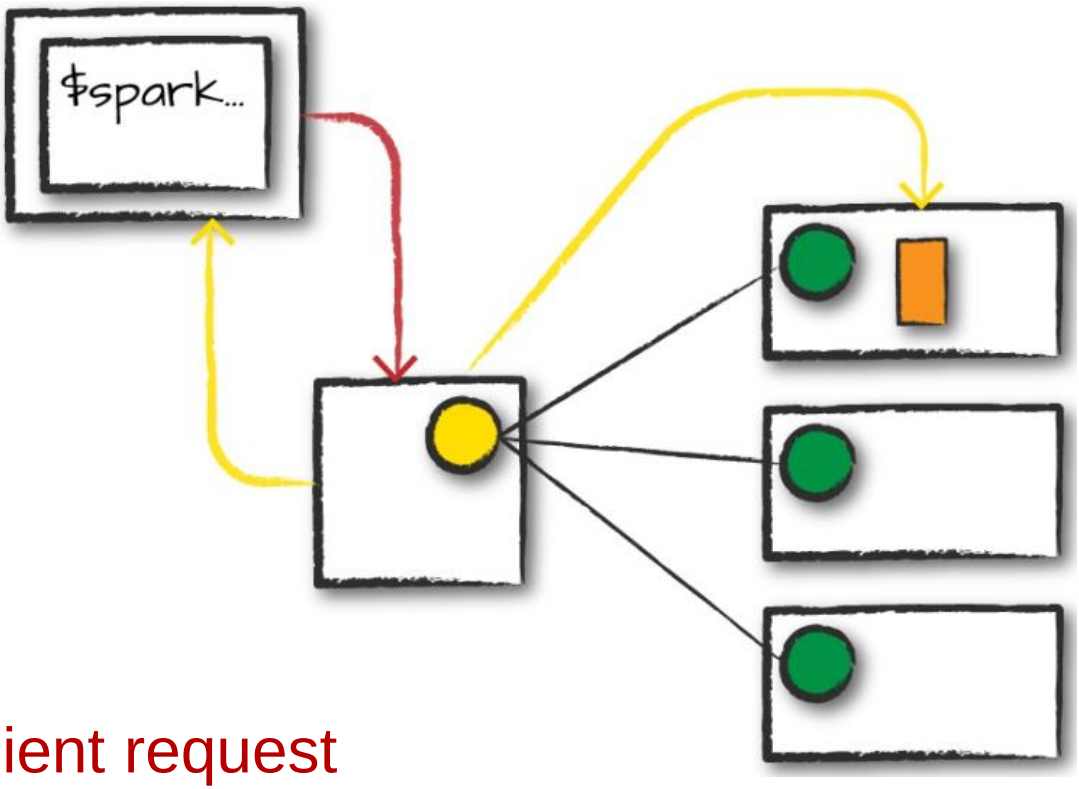
- The client machine maintains the Spark driver process, and the cluster manager manages the executor processes.



Execution modes: Local mode

- The entire Spark application is run on a single machine, using threads to achieve parallelism.
- Learn Spark, test applications, or experiment iteratively with local development
- Run production applications: not recommended

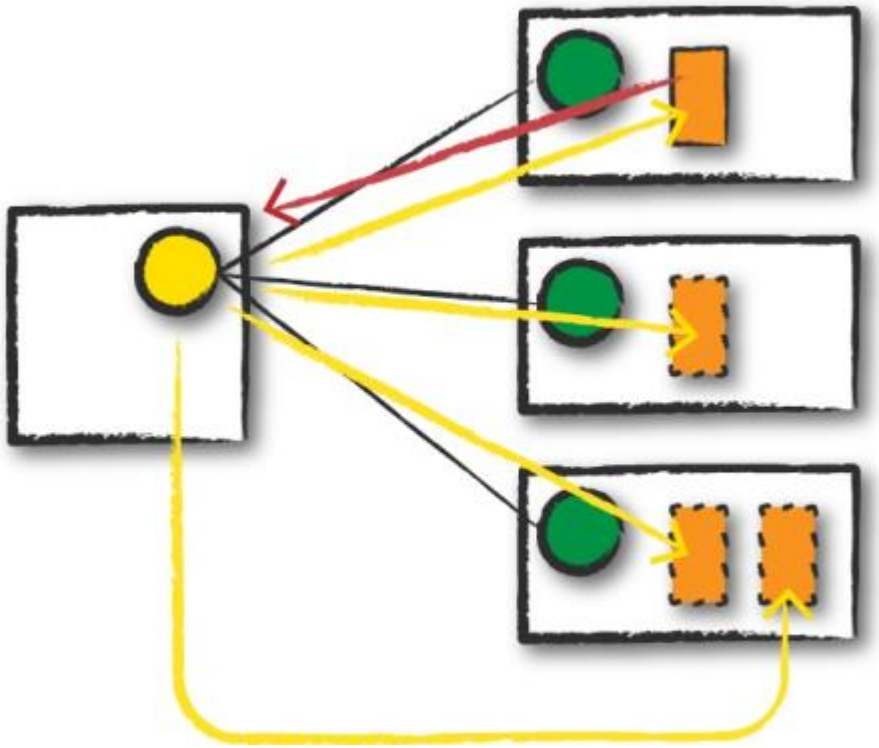
The life cycle of a Spark application




```
./bin/spark-submit \
  --class <main-class> \
  --master <master-url> \
  --deploy-mode cluster \
  --conf <key>=<value> \
  ... # other options
  <application-jar> \
  [application-arguments]
```

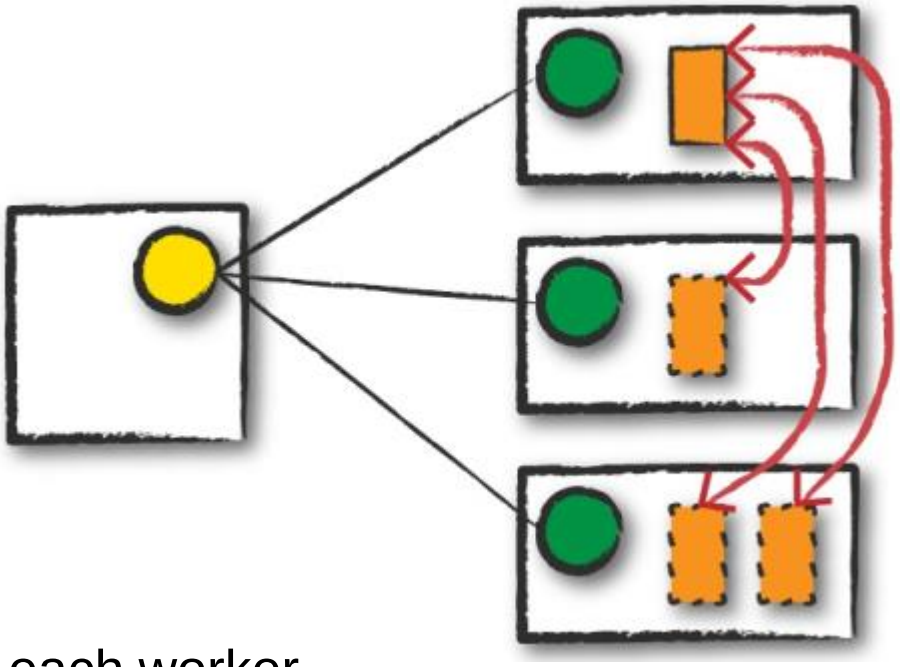
- Client request
  - Submit an actual application: a pre-compiled JAR or library
  - Explicitly ask for resources for the Spark driver process only

The life cycle of a Spark application



- Launch
  - A SparkSession asks the cluster manager to launch executor processes across the cluster.
  - The locations of these executors are sent to the driver process.

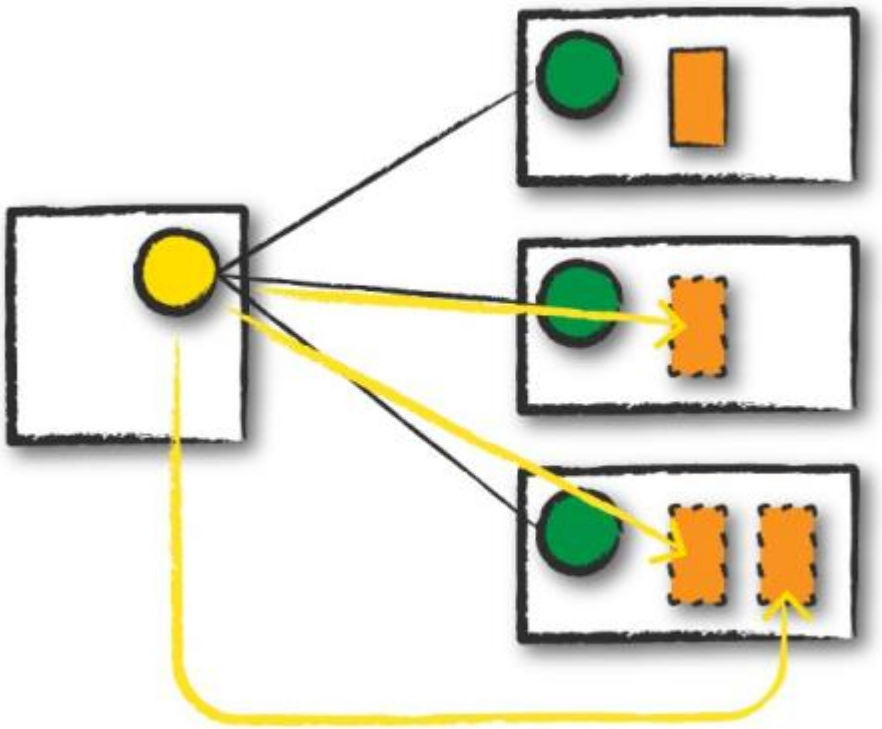
The life cycle of a Spark application



- Execution
  - The driver schedules tasks onto each worker
  - Each worker responds with the status of those tasks and success or failure.

The life cycle of a Spark application

- Completion
  - The driver process exits with either success or failure.
  - The cluster manager then shuts down the executors in that Spark cluster for the driver.



The life cycle of a Spark application

- Each application is made up of one or more Spark jobs.
- Spark jobs within an application are executed serially
  - Unless threading is used to launch multiple actions in parallel.

- **Spark 1.x:** both `SparkContext` and `SQLContext`
  - `SparkContext` focused on more fine-grained control of Spark's central abstractions.
  - `SQLContext` focused on the higher-level tools like Spark SQL.
- **Spark 2.x:** the centralized `SparkSession`
  - We never need to use `SQLContext` and rarely use `SparkContext`.
  - However, they are still accessible via the `SparkSession`.

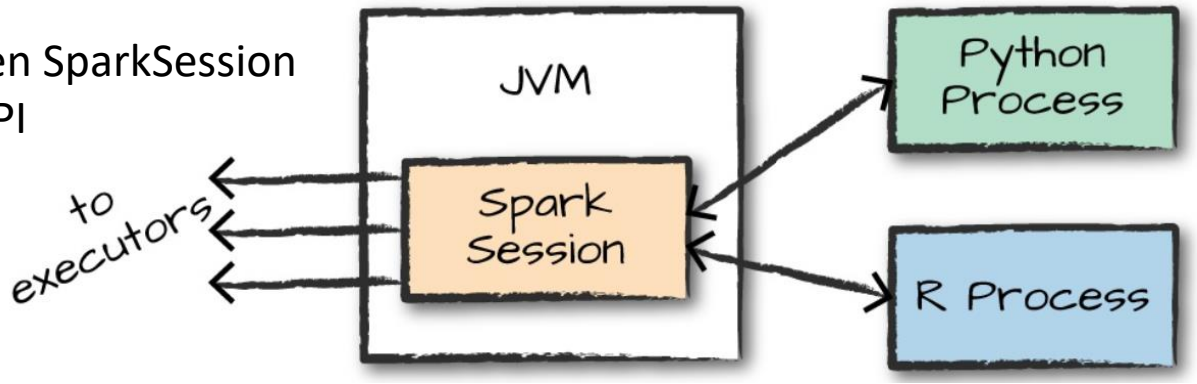
SparkSession

- The first step of any application is creating a `SparkSession`.

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local").appName("Word Count")\
            .config("spark.some.config.option", "some-value")\
            .getOrCreate()
```

- Spark translates user code into some internal representation that can run on the executor JVMs.

The relationship between SparkSession and Spark's Language API



SparkContext

- An object within `SparkSession` that connects to the cluster
  - Create RDDs, accumulators, and broadcast variables, run code on the cluster, etc.
- Not explicitly initialize a `SparkContext`, access it through `SparkSession` instead.
- Or, it should be created in the most general way as follows.

```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
```

References

- Spark Tutorial: A Beginner’s Guide to Apache Spark (link)
- Apache Spark Architecture – Spark Cluster Architecture Explained (link)
- Submitting User Applications with spark-submit (link)


...the end.