BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Introduction to Apache Spark

Pravin Y Pawar

What?

Apache Spark is

- a fast and general-purpose cluster computing system
- Extending popular MapReduce model to support more types of computations
- introduced by Apache Software Foundation for speeding up the Hadoop computational computing software process
- **in-memory cluster computing** that increases the processing speed of an application
- System which can cover large number of workloads like batch, streaming etc.
- Designed to be highly accessible, offering simple APIs in Python, Java, Scala and SQL and rich built-in libraries
- Easy to integrate with other big data tools

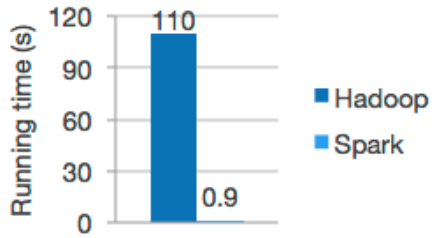
Apache Spark™ is a unified analytics engine for large-scale data processing.

Execution Speed

Speed

Run workloads 100x faster.

Apache Spark achieves high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.



Logistic regression in Hadoop and Spark

Source : https://spark.apache.org/

Ease of Use

Ease of Use

Write applications quickly in Java, Scala, Python, R, and SQL.

Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala, Python, R, and SQL shells.

```
df = spark.read.json("logs.json")
df.where("age > 21")
  .select("name.first").show()
```

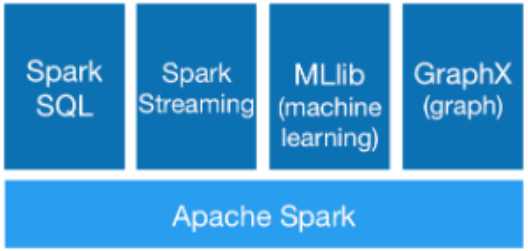
Spark's Python DataFrame API
Read JSON files with automatic schema inference

Source : https://spark.apache.org/

Generality

Combine SQL, streaming, and complex analytics.

Spark powers a stack of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming. You can combine these libraries seamlessly in the same application.



Source : https://spark.apache.org/

Runs Everywhere

Spark runs on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud. It can access diverse data sources.

You can run Spark using its standalone cluster mode, on EC2, on Hadoop YARN, on Mesos, or on Kubernetes. Access data in HDFS, Alluxio, Apache Cassandra, Apache HBase, Apache Hive, and hundreds of other data sources.

Source : https://spark.apache.org/

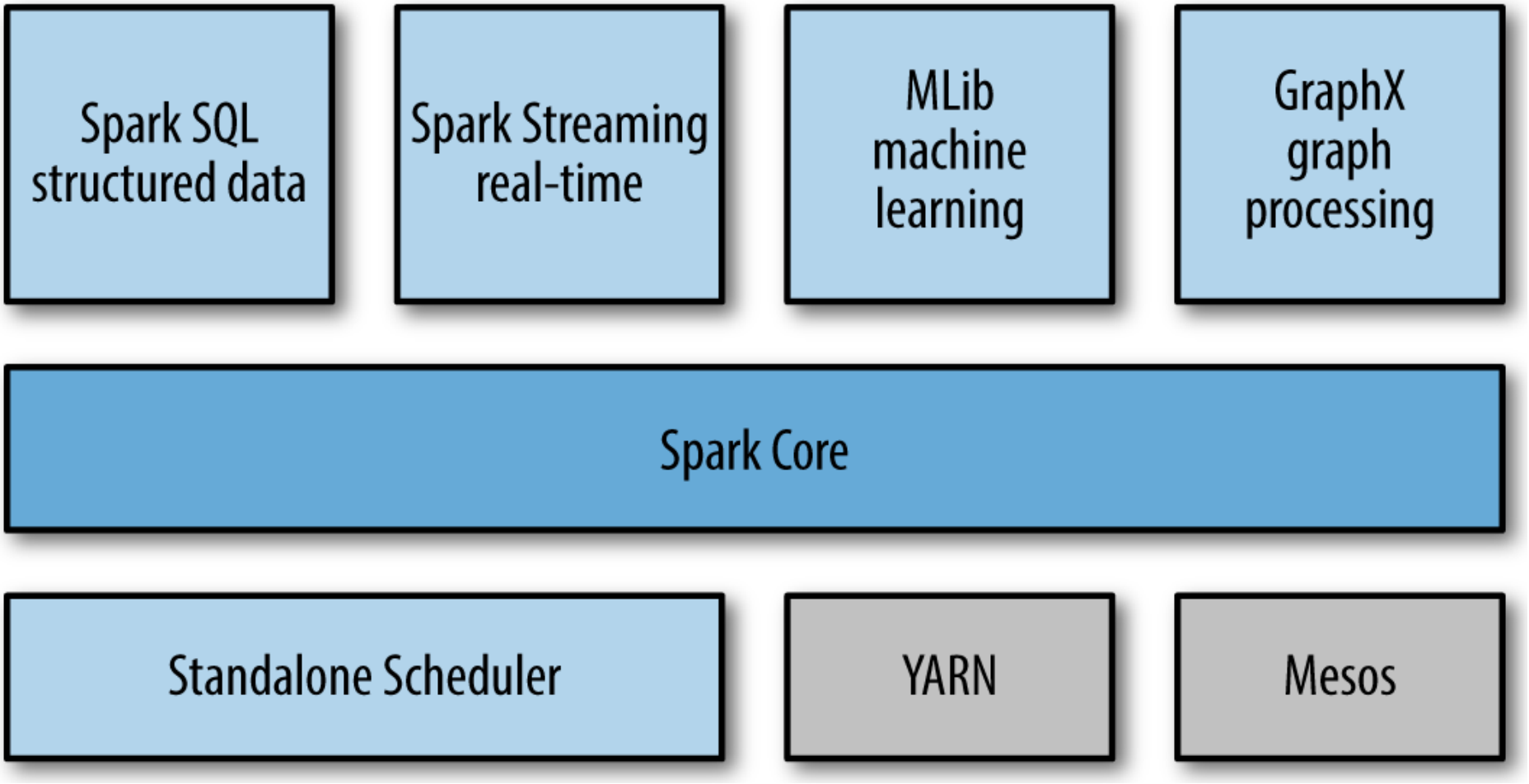
Cluster Manager Types

- The system currently supports several cluster managers:

- Standalone
  - ✓ a simple cluster manager included with Spark that makes it easy to set up a cluster.
- Apache Mesos
  - ✓ a general cluster manager that can also run Hadoop MapReduce and service applications.
- Hadoop YARN
  - ✓ the resource manager in Hadoop 2.
- Kubernetes
  - ✓ an open-source system for automating deployment, scaling, and management of containerized applications.

Source : https://spark.apache.org/

Unified Stack



Ref : https://www.oreilly.com/library/view/learning-spark/9781449359034/ch01.html

Spark Core

- Contains the basic functionality of Spark, including components for
  - ✓ task scheduling
  - ✓ memory management
  - ✓ fault recovery
  - ✓ interacting with storage systems etc.

- Home to the API that defines resilient distributed datasets (RDDs), which are Spark's main programming abstraction

- Provides many APIs for building and manipulating these collections

Spark SQL

- Spark's package for working with structured data
- Allows querying data via SQL as well as the Apache Hive variant of SQL—called the Hive Query Language (HQL)
- Allows developers to intermix SQL queries with the programmatic data manipulations supported by RDDs in Python, Java, and Scala, all within a single application
- Tight integration with the rich computing environment provided by Spark makes Spark SQL unlike any other open source data warehouse tool

Spark Streaming

- Spark component that enables processing of live streams of data
- Examples of data streams include web servers log files
- Provides an API for manipulating data streams that closely matches the Spark Core's RDD API
- Designed to provide the same degree of fault tolerance, throughput, and scalability as Spark Core.

- Built in library containing common machine learning (ML) functionality
- Provides multiple types of machine learning algorithms, including
  - ✓ Classification
  - ✓ Regression
  - ✓ Clustering
  - ✓ collaborative filtering
  - ✓ supporting functionality such as model evaluation and data import
- Provides some lower-level ML primitives, including a generic gradient descent optimization algorithm
- All of these methods are designed to scale out across a cluster.

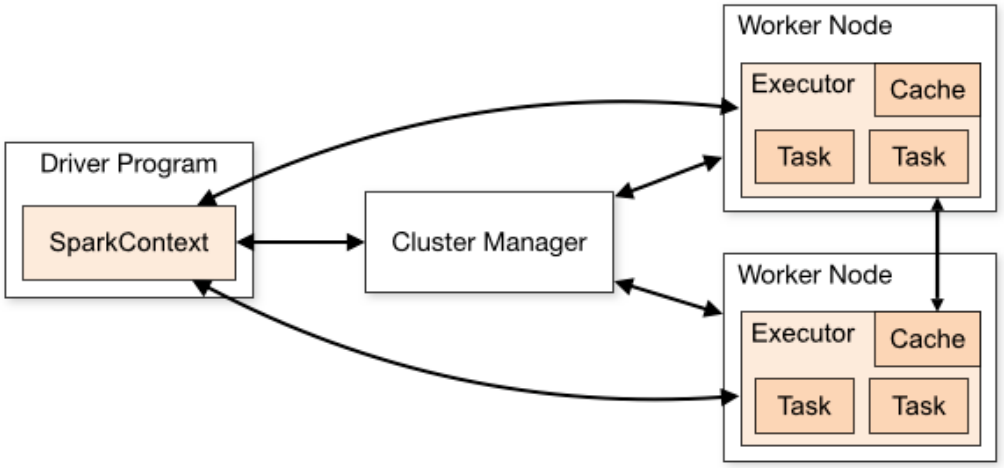
GraphX

- Library for
  - ✓ manipulating graphs (e.g., a social network's relations graph)
  - ✓ performing graph-parallel computations.
- Extends the Spark RDD API, allowing us to create a directed graph with arbitrary properties attached to each vertex and edge
- Provides
  - ✓ various operators for manipulating graphs (e.g., subgraph and mapVertices)
  - ✓ library of common graph algorithms (e.g., PageRank and triangle counting)

Cluster Managers

- Spark is designed to efficiently scale up from one to many thousands of compute nodes
- Can run over a variety of cluster managers, including
  - ✓ Hadoop YARN
  - ✓ Apache Mesos
  - ✓ Simple cluster manager included in Spark itself called the Standalone Scheduler
- If you are just installing Spark on an empty set of machines
  - ✓ the Standalone Scheduler provides an easy way to get started
- If you already have a Hadoop YARN or Mesos cluster,
  - ✓ Spark's support for these cluster managers allows your applications to also run on them

Components



Source : https://spark.apache.org/docs/2.4.4/cluster-overview.html

Cluster Mode Overview (2)

- Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in your main program (called the driver program).
- Specifically, to run on a cluster, the SparkContext can connect to several types of cluster managers (either Spark's own standalone cluster manager, Mesos or YARN), which allocate resources across applications.
- Once connected, Spark acquires executors on nodes in the cluster, which are processes that run computations and store data for your application.
- Next, it sends your application code (defined by JAR or Python files passed to SparkContext) to the executors.
- Finally, SparkContext sends tasks to the executors to run.

Source : https://spark.apache.org/docs/2.4.4/cluster-overview.html

Spark Cluster Concepts

Term	Meaning
Application	User program built on Spark. Consists of a *driver program* and *executors* on the cluster.
Application jar	A jar containing the user's Spark application. In some cases users will want to create an "uber jar" containing their application along with its dependencies. The user's jar should never include Hadoop or Spark libraries, however, these will be added at runtime.
Driver program	The process running the main() function of the application and creating the SparkContext
Cluster manager	An external service for acquiring resources on the cluster (e.g. standalone manager, Mesos, YARN)
Deploy mode	Distinguishes where the driver process runs. In "cluster" mode, the framework launches the driver inside of the cluster. In "client" mode, the submitter launches the driver outside of the cluster.
Worker node	Any node that can run application code in the cluster
Executor	A process launched for an application on a worker node, that runs tasks and keeps data in memory or disk storage across them. Each application has its own executors.
Task	A unit of work that will be sent to one executor
Job	A parallel computation consisting of multiple tasks that gets spawned in response to a Spark action (e.g. `save`, `collect`); you'll see this term used in the driver's logs.
Stage	Each job gets divided into smaller sets of tasks called *stages* that depend on each other (similar to the map and reduce stages in MapReduce); you'll see this term used in the driver's logs.

Source : https://spark.apache.org/docs/2.4.4/cluster-overview.html

Thank You!