What is Apache Spark?

Apache Spark is an open-source, distributed processing system used for big data workloads. It utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size

## How does Apache Spark work?

Hadoop MapReduce is a programming model for processing big data sets with a parallel, distributed algorithm.

However, a challenge to MapReduce is the sequential multi-step process it takes to run a job.

Because each step requires a disk read, and write, MapReduce jobs are slower due to the latency of disk I/O.

Spark was created to address the limitations to MapReduce, by doing processing in-memory, reducing the number of steps in a job, and by reusing data across multiple parallel operations.

 only one-step is needed where data is read into memory, operations performed, and the results written back—resulting in a much faster execution

Spark also reuses data by using an in-memory cache to greatly speed up machine learning algorithms that repeatedly call a function on the same dataset.

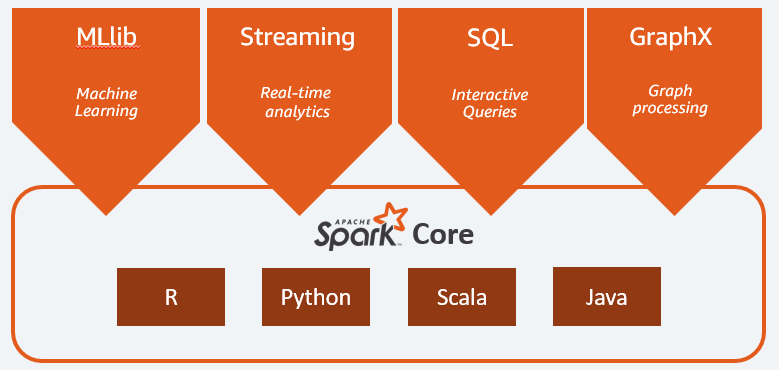
Data re-use is accomplished through the creation of DataFrames, an abstraction over Resilient Distributed Dataset (RDD), which is a collection of objects that is cached in memory, and reused in multiple Spark operations.

## What are the benefits of Apache Spark?

#### **Fast**

#### **Developer friendly**

#### **Multiple workloads**



## What are Apache Spark Workloads?

The Spark framework includes:

* Spark Core as the foundation for the platform
* Spark SQL for interactive queries
* Spark Streaming for real-time analytics
* Spark MLlib for machine learning
* Spark GraphX for graph processing

### **Spark Core**

Spark Core is the foundation of the platform. It is responsible for memory management, fault recovery, scheduling, distributing & monitoring jobs, and interacting with storage systems.

### **MLlib**

#### Machine Learning

Spark includes MLlib, a library of algorithms to do machine learning on data at scale. Machine Learning models can be trained by data scientists with R or Python on any Hadoop data source, saved using MLlib, and imported into a Java or Scala-based pipeline.

### **Spark Streaming**

#### Real-time

Spark Streaming is a real-time solution that leverages Spark Core’s fast scheduling capability to do streaming analytics. It ingests data in mini-batches, and enables analytics on that data with the same application code written for batch analytics.

### **Spark SQL**

#### Interactive Queries

Spark SQL is a distributed query engine that provides low-latency, interactive queries up to 100x faster than MapReduce. It includes a cost-based optimizer, columnar storage, and code generation for fast queries, while scaling to thousands of nodes.

### **GraphX**

#### Graph Processing

Spark GraphX is a distributed graph processing framework built on top of Spark. GraphX provides ETL, exploratory analysis, and iterative graph computation to enable users to interactively build, and transform a graph data structure at scale

## What are the use cases of Apache Spark?

### **Financial Services**

Spark is used in banking to predict customer churn, and recommend new financial products. In investment banking, Spark is used to analyze stock prices to predict future trends.

### **Healthcare**

Spark is used to build comprehensive patient care, by making data available to front-line health workers for every patient interaction. Spark can also be used to predict/recommend patient treatment.

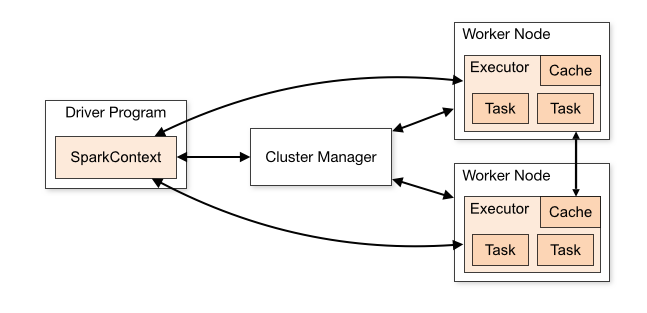
### **Manufacturing**

Spark is used to eliminate downtime of internet-connected equipment, by recommending when to do preventive maintenance.

### **Retail**

Spark is used to attract, and keep customers through personalized services and offers.

# Spark Architecture: A Deep Dive



The Apache Spark framework uses a **master-slave architecture** that consists of a driver, which runs as a master node, and many executors that run across as worker nodes in the cluster

## The Spark driver

The driver is the program or process responsible for coordinating the execution of the Spark application.

## The Spark executors

Executors are worker processes responsible for executing tasks in Spark applications. They are launched on worker nodes and communicate with the driver program and cluster manager.

## The cluster manager

The cluster manager is responsible for allocating resources and managing the cluster on which the Spark application runs.

## sparkContext

SparkContext is the entry point for any Spark functionality. It represents the connection to a Spark cluster and can be used to create RDDs (Resilient Distributed Datasets), accumulators, and broadcast variables. SparkContext also coordinates the execution of tasks.

# Working Of Spark Architecture

When the Driver Program in the Apache Spark architecture executes, it calls the real program of an application and creates a SparkContext.

The Spark Driver includes several other components, including a DAG Scheduler, Task Scheduler, Backend Scheduler, and Block Manager, all of which are responsible for translating user-written code into jobs that are actually executed on the cluster.

The Cluster Manager manages the execution of various jobs in the cluster. Spark Driver works in conjunction with the Cluster Manager to control the execution of various other jobs.

 The cluster Manager does the task of allocating resources for the job. Once the job has been broken down into smaller jobs, which are then distributed to worker nodes, SparkDriver will control the execution.

Many worker nodes can be used to process an RDD(Resilient Distributed Dataset) created in SparkContext, and the results can also be cached

The SparkContext receives task information from the Cluster Manager and enqueues it on worker nodes.

 The executor is in charge of carrying out these duties

# Two Main Abstractions of Apache Spark

**Resilient Distributed Dataset (RDD)**:

1. Immutable (Read only)
2. Fundamental collection of elements or items that can be operated on many devices at the same time (spark parallel processing)
3. Each dataset in RDD can be divided into logical portions which are then executed on different nodes of a cluster

**Directed Acyclic Graph (DAG)**:

DAG is the scheduling layer of the Apache Spark architecture that implements stage-oriented scheduling

Apache Spark can create DAGs that contain many stages.

# Execution Modes

## Cluster mode

Cluster mode is probably the most common way of running Spark Applications

 In cluster mode, a user submits a pre-compiled JAR, Python script, or R script to a cluster manager.

The cluster manager then launches the driver process on a worker node inside the cluster, in addition to the executor processes.

This means that the cluster manager is responsible for maintaining all Spark Application–related processes.

## Client mode

Client mode is nearly the same as cluster mode except that the Spark driver remains on the client machine that submitted the application.

This means that the client machine is responsible for maintaining the Spark driver process, and the cluster manager maintains the executor processes.

## Local mode

Local mode is a significant departure from the previous two modes: it runs the entire Spark Application on a single machine.

It achieves parallelism through threads on that single machine. This is a common way to learn Spark, test your applications, or experiment iteratively with local development.