Apache Spark: Concepts and Architecture

# Introduction

Apache Spark is a fast, in-memory, distributed computing system used for big data processing, analytics, machine learning, and graph processing.

# Core Concepts

|  |  |
| --- | --- |
| Component | Description |
| Spark Core | Base engine for scheduling, memory management, fault recovery, etc. |
| RDD | Resilient Distributed Dataset – low-level data structure for distributed data |
| DataFrame | Tabular, schema-based API (like SQL table) – built on top of RDD |
| Dataset | Type-safe, object-oriented API (mainly in Scala & Java) |
| Spark SQL | Module for querying structured data with SQL |
| Spark MLlib | Scalable machine learning library |
| Spark Streaming | Real-time stream processing engine |
| GraphX | For graph processing and computation |

# Apache Spark Architecture

The architecture of Apache Spark includes a Driver, Cluster Manager, SparkContext, Executors, and Workers. The Driver converts the user program into jobs and stages and coordinates the execution.

## Architecture Diagram

+------------------------+  
 | Spark Application |  
 +------------------------+  
 |  
 +--------------+--------------+  
 | |  
+------------+ +----------------+  
| Driver | | Cluster Mgr |  
+------------+ +----------------+  
 |  
+--------------+  
| SparkContext |  
+--------------+  
 |  
+--------------+  
| Task Scheduler| --> divides jobs into tasks  
+--------------+  
 |  
+-------------+  
| Executors | <-- on multiple Worker Nodes  
+-------------+  
| Tasks | (Run the actual computations)  
+-------------+

## Component Descriptions

- Driver Program: Main program that runs the SparkContext and coordinates execution.

- Cluster Manager: Allocates resources; can be Standalone, YARN, Mesos, or Kubernetes.

- Worker Nodes & Executors: Executors run tasks and store data.

- SparkContext: Entry point to Spark; connects to cluster manager.

- Tasks and Stages: Jobs are broken into stages and tasks that run in parallel.

# Execution Flow

User code is translated into a DAG (Directed Acyclic Graph). The driver submits the job to the cluster manager. The cluster manager assigns tasks to executors on worker nodes. The results are collected and returned to the driver.

# Spark vs. Hadoop MapReduce

|  |  |  |
| --- | --- | --- |
| Feature | Spark | Hadoop MapReduce |
| Processing | In-memory | Disk-based |
| Speed | Much faster (10–100x) | Slower |
| APIs | Rich APIs (RDD, DF, SQL, MLlib) | Java APIs only |
| Real-time | Supports streaming | Not suitable |
| Ease of Use | Python, Scala, SQL support | Java-only, verbose |

# Use Cases

|  |  |
| --- | --- |
| Use Case | Description |
| ETL Processing | Clean and transform large data |
| Stream Processing | Analyze real-time logs or events |
| Machine Learning | Scalable ML model training |
| Graph Processing | Analyze social or recommendation graphs |
| Interactive SQL Queries | BI dashboards on top of DataFrames |

# Quick Summary

|  |  |
| --- | --- |
| Term | Meaning |
| RDD | Low-level, fault-tolerant distributed data structure |
| DataFrame | High-level table-like API for structured data |
| DAG | Directed Acyclic Graph – execution plan for tasks |
| Executor | Process running tasks and caching data |
| Cluster Manager | Controls resource allocation (YARN, Kubernetes, etc.) |
| SparkContext | Interface between Driver and Cluster |