Understanding the Architecture of Apache Spark

JUNE 2024

ISSUED BY
Medha Prodduturi

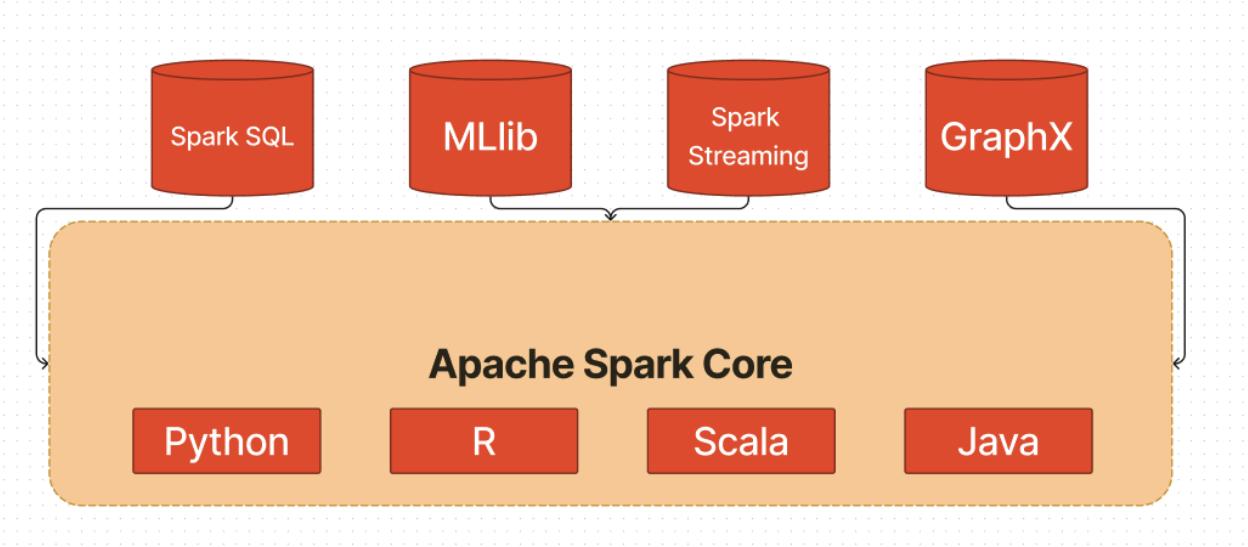
Introduction to Apache Spark

Apache Spark is an open-source, distributed computing system that allows for fast data processing of big data. It processes data in memory (RAM) instead of reading and writing from disk like Apache Hadoop, making it much more efficient and faster than traditional data processing frameworks.

In this article:

What It Offers

Spark Libraries and Languages



Spark Core

This is the foundational element that allows for basic functionality like task scheduling, memory management, fault recovery, and basic I/O operations. It supports multiple programming languages like Scala, Java, Python, and R, making it accessible to a wide range of developers.

Spark SQL

Allows users to query data via SQL, integrates with existing databases and supports both standard SQL and Hive Query Language.

MLlib

A library of machine learning algorithms that can be used to train ML models in classification, regression, clustering, and more.

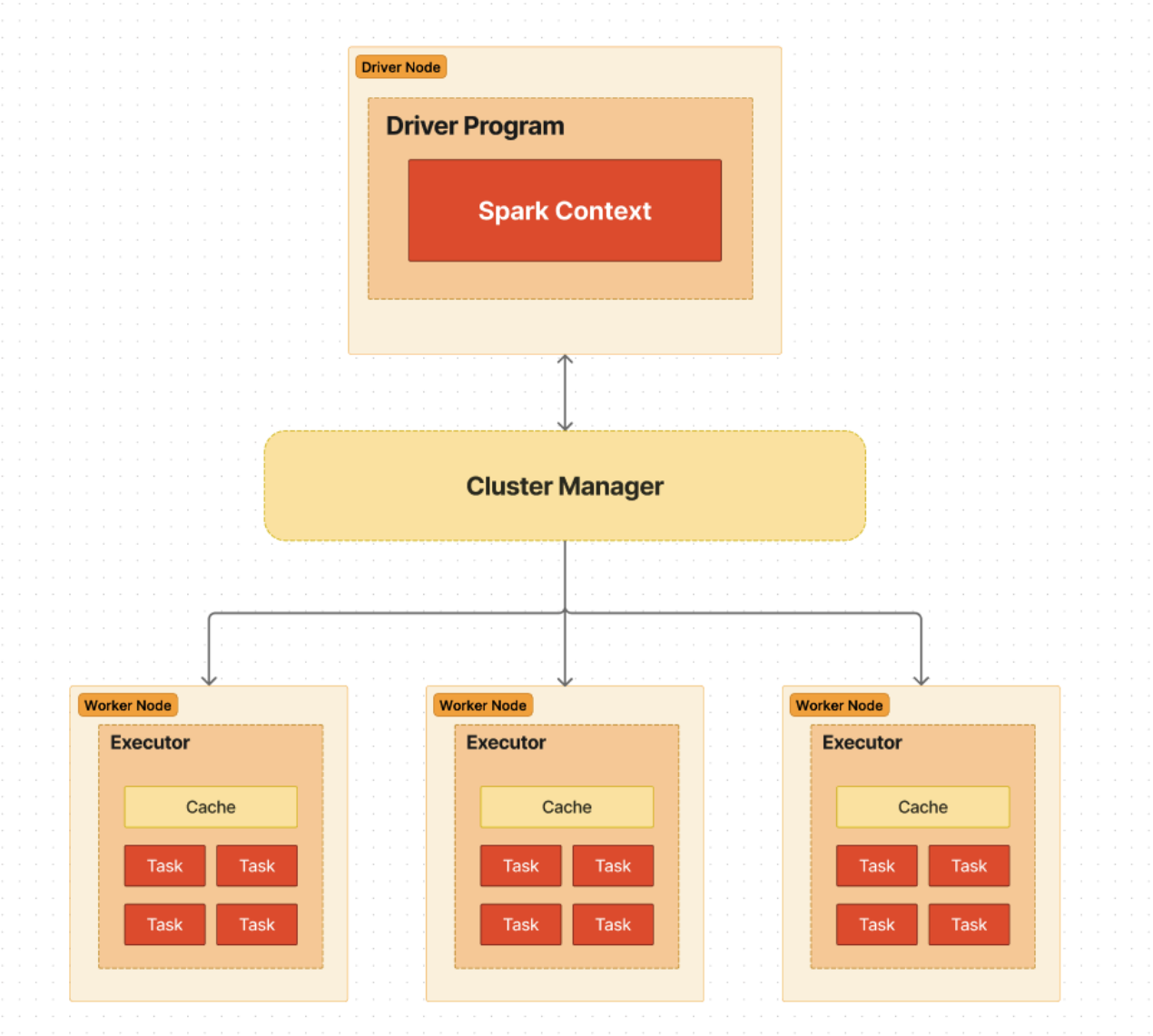
Spark Streaming

Enables processing of real-time data streams.

GraphX

A component for graph processing and analytics which allows users to build and transform a graph data structure at scale interactively.

Spark Architecture



Spark's Different Components

Driver Node
Essentially, this is the Boss. It keeps track of all spark information and analyzes the work that needs to be done. It is responsible for scheduling the jobs on the cluster and communicating with the cluster manager.

Driver Program
It is responsible for scheduling the jobs on the cluster and communicating with the cluster manager.

Spark Context
Sets up the connection to the Spark cluster. It is created in a driver node and is used to create RDD, accumulator, or broadcasting variables.

Worker Node
Machine in a Spark cluster that hosts executors and manages the resources allocated to them.

Executor
Runs the tasks assigned by the driver program.

Task
A single unit of work sent to an executor by the driver.

Cluster Manager
Allocates and deallocates the resources on the nodes of a cluster. Some famous cluster managers are YARN, Mesos, and Spark standalone.

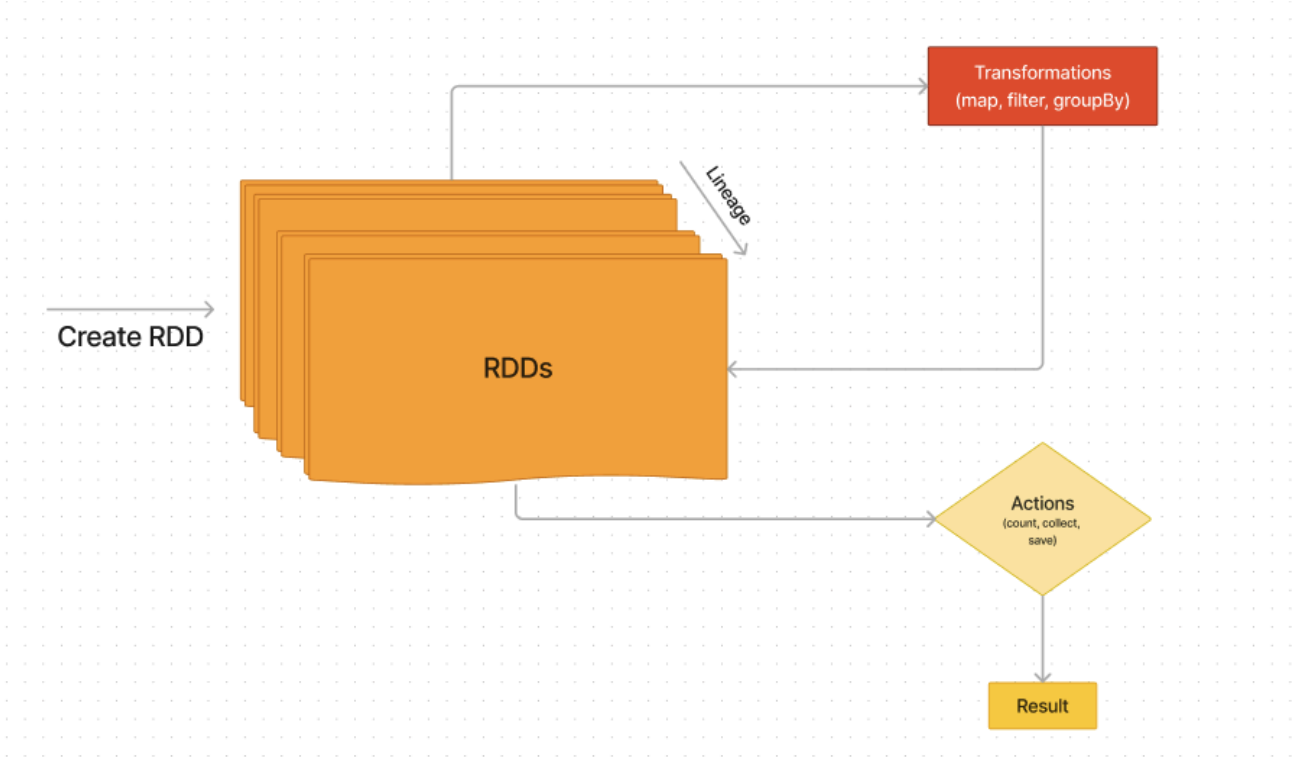
Job Scheduler
Responsible for dividing the job into smaller tasks and scheduling these tasks across the available executors.

Resilient Distributed Dataset (RDD)

The fundamental data structure of spark which is a fault-tolerant, immutable collection of elements which can operate in parallel. In RDD, each dataset is divided into logical partitions which are computed on different nodes of the cluster.

RDD supports two types of operations:

1. Actions
2. Transformations



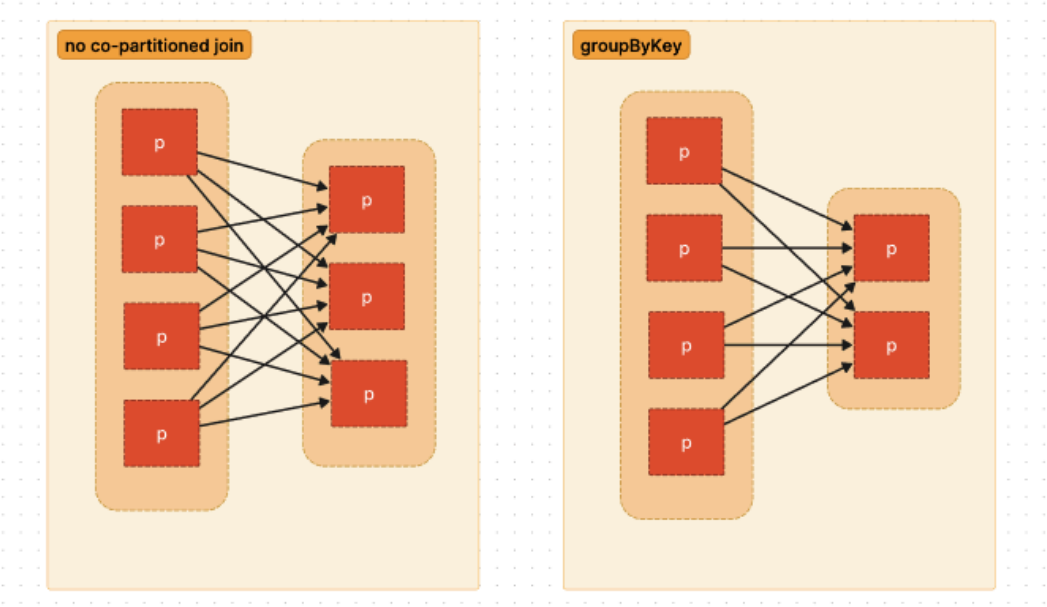
Actions

Operations that trigger the execution of the transformation to compute and return a result to the driver program.

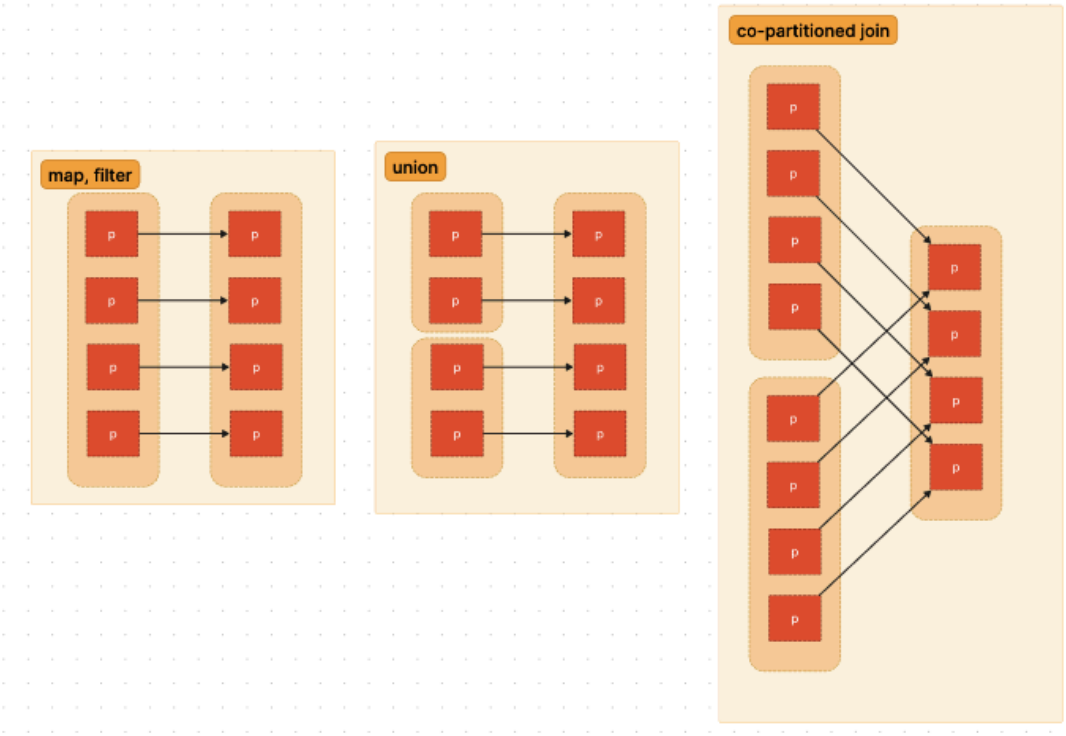
Transformations

Transformations take RDD as an input and return one or more RDD as output. However, the data in the existent RDD in Spark does not change as it is immutable. They are "lazy," meaning they do not immediately compute results; instead, they wait till an action is called to execute. There are two types of transformations:

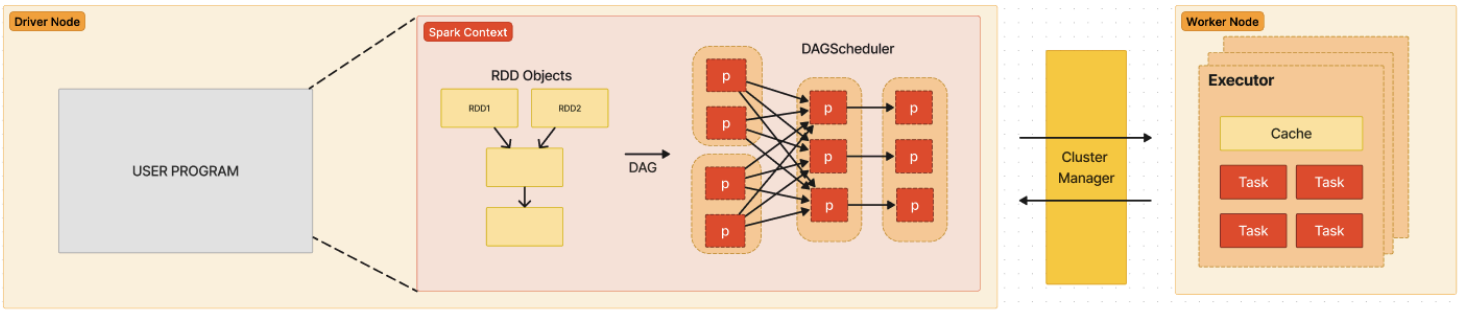
1. Wide Transformations - shuffling data across the network because they require data from multiple partitions to produce a single output partition.



2. Narrow Transformations - each input partition contributes to only one output partition.



Execution Workflow Recap



A Spark Application, often referred to as the Driver Node consists of the SparkContext and user code that interacts with it. This user code creates RDDs and applies a series of transformations to achieve the desired result. The transformations on RDDs are translated into a Directed Acyclic Graph (DAG) and submitted to the scheduler, which then executes the tasks on a set of worker nodes.

Conclusion

Understanding Apache Spark's components and architecture is key to using its powerful data processing capabilities effectively. A Spark application, driven by the SparkContext and user-defined transformations, translates data workflows into a Directed Acyclic Graph (DAG). The scheduler then executes these workflows across a distributed cluster. By differentiating between narrow and wide transformations, and knowing the roles of the driver program, executors, and worker nodes, you can optimize your Spark applications for better performance and scalability. Whether you're dealing with large-scale data analytics, real-time stream processing, or machine learning tasks, Spark's architecture and flexible programming model offer diverse solutions for today's big data challenges.