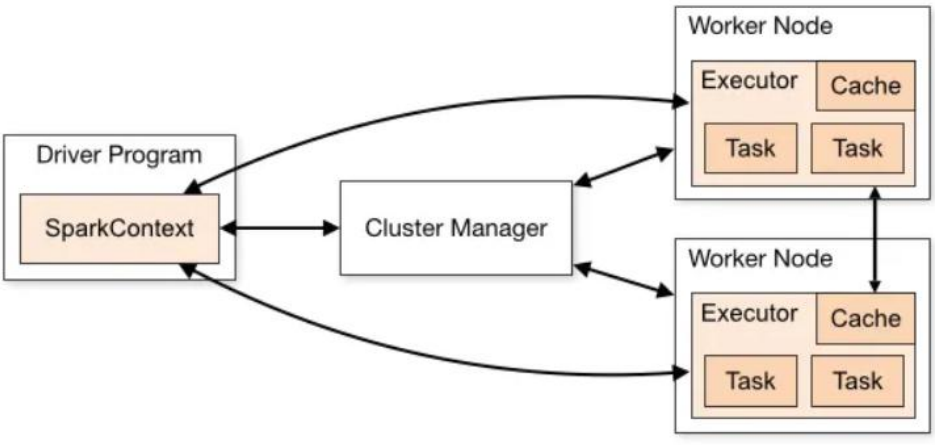
Apache Spark is an open-source distributed computing system designed for big data processing and analytics. Spark is known for its speed and efficiency.

we will explore the following topics:

- spark Architecture And Applications
- working of spark Architecture
- Abstractions of Apache Spark
- Cluster Manager Types
- Execution Modes

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. Apache Spark can be used for batch processing and real-time processing as well.




Before understanding the Spark architecture let's understand the Applications of Spark Architecture which exists in the above diagram:

### The Spark driver

The driver is the program or process responsible for coordinating the execution of the Spark application. It runs the main function and creates the SparkContext, which connects to the cluster manager.

### 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. Executors run tasks concurrently and store data in memory or disk for caching and intermediate storage.

### The cluster manager

The cluster manager is responsible for allocating resources and managing the cluster on which the Spark application runs. Spark supports various cluster managers like Apache Mesos, Hadoop YARN, and standalone cluster manager.

### 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.

### Task

A task is the smallest unit of work in Spark, representing a unit of computation that can be performed on a single partition of data. The driver program divides the Spark job into tasks and assigns them to the executor nodes for execution.

### 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. SparkContext contains all of the basic functions. 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. Spark Driver and

SparkContext collectively watch over the job execution within 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. The lifespan of executors is the same as that of the Spark Application. We can increase the number of workers if we want to improve the performance of the system. In this way, we can divide jobs into more coherent parts.

**Two Main Abstractions of Apache Spark**

Apache Spark has a well-defined layer architecture that is designed on two main abstractions:

Resilient Distributed Dataset (RDD): RDD is an immutable (read-only), fundamental collection of elements or items that can be operated on many devices at the same time (spark parallel processing). Each dataset in an 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. Compared to MapReduce which creates a graph in two stages, Map and Reduce, Apache Spark can create DAGs that contain many stages.

**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.

**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.

## Conclusion

We learned about the Apache Spark Architecture in order to understand how to build big data applications efficiently. They're accessible and consist of components, which is very beneficial for cluster computing and big data technology. Spark calculates the desired outcomes in an easy way and is popular for batch processing.