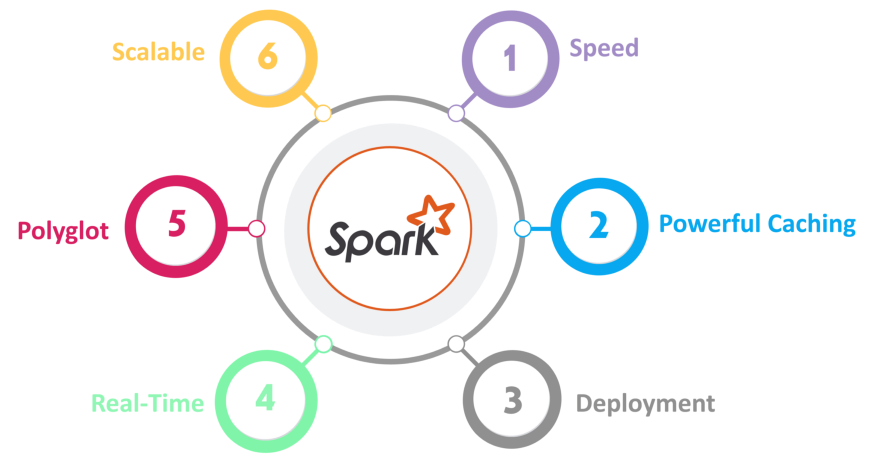
Features of Apache Spark:



1. **Speed —**Spark runs up to 100 times faster than Hadoop MapReduce for large-scale data processing. It is also able to achieve this speed through controlled partitioning.
2. **Powerful Caching —**Simple programming layer provides powerful caching and disk persistence capabilities.
3. **Deployment —**It can be deployed through **Mesos, Hadoop via YARN, or Spark’s own cluster manager.**
4. **Real-Time —**It offers Real-time computation & low latency because of **in-memory computation.**
5. **Polyglot —**Spark provides high-level APIs in Java, Scala, Python, and R. Spark code can be written in any of these four languages. It also provides a shell in Scala and Python.

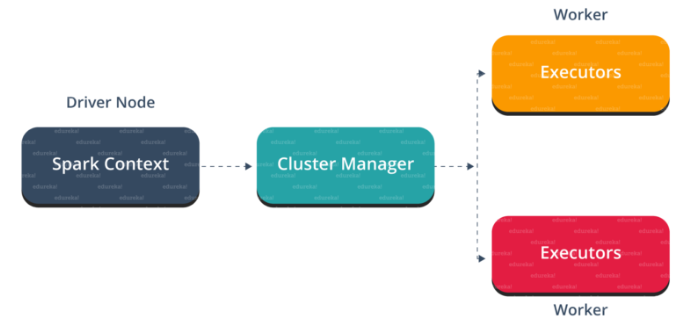
# **Major components of Apache Spark’s distributed architecture.**

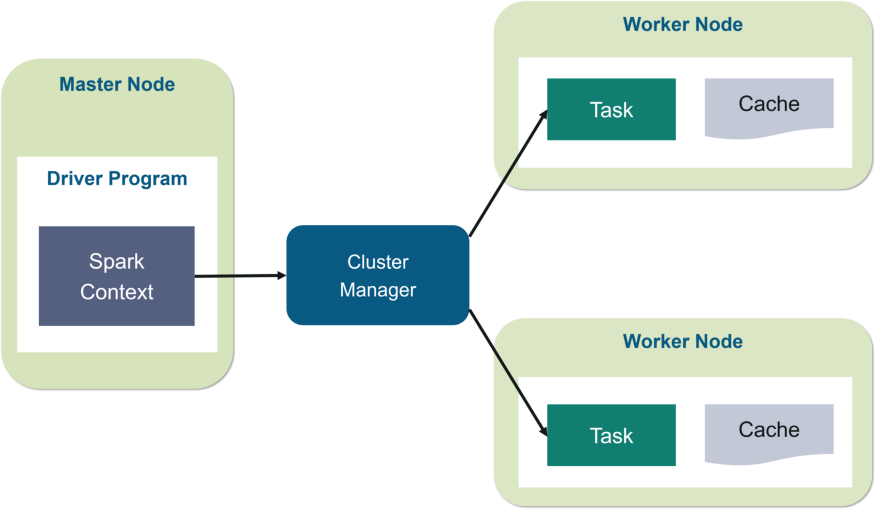
## Spark Architecture Overview

* **Resilient Distributed Dataset (RDD) ( Outdated )**: RDD is an immutable (read-only), fundamental collection of elements or items that can be operated on many devices at the same time (parallel processing). Each dataset in an RDD can be divided into logical portions, which are then executed on different nodes of a cluster. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster. RDDs can contain any type of Python, Java, or Scala objects, including user-defined classes.  
  There are two ways to create RDDs − **parallelizing** an existing collection in your driver program, or **referencing a dataset** in an external storage system, such as a shared file system, HDFS, HBase, or any data source offering a Hadoop Input Format.
* **Directed Acyclic Graph (DAG**): DAG is the scheduling layer of the Apache Spark architecture that implements stage-oriented scheduling. Compared to MapReduce that creates a graph in two stages, Map and Reduce, Apache Spark can create DAGs that contain many stages.

## When to use RDD

1. You want low-level transformation and actions and control on your dataset;
2. Your data is unstructured, such as media streams or streams of text;
3. You want to manipulate your data with functional programming constructs than domain specific expressions;
4. You don’t care about imposing a schema, such as columnar format while processing or accessing data attributes by name or column; and
5. You can forgo some optimization and performance benefits available with DataFrames and Datasets for structured and semi-structured data.





* **Driver Program** in the Apache Spark architecture calls the main program of an application and creates **SparkContext**. A **SparkContext**consists of all the basic functionalities. **Spark Driver** contains various other components such as **DAG Scheduler, Task Scheduler, Backend Scheduler, and Block Manager**, which are responsible for translating the 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. **Spark Driver** works with the**Cluster Manager** to manage various other jobs.**Cluster Manager** does the resource allocating work. And then, the job is split into multiple smaller tasks which are further distributed to worker nodes.
* Whenever an **RDD**is created in the **SparkContext**, it can be distributed across many worker nodes and can also be cached there.
* Worker nodes execute the tasks assigned by the **Cluster Manager** and return it back to the **Spark Context**.
* An **executor**is responsible for the **execution**of these tasks. The lifetime of executors is the same as that of the Spark Application. If we want to increase the performance of the system, we can increase the number of workers so that the jobs can be divided into more logical portions.

## Spark Eco-System

## 

1. **Spark Core**Spark Core is the base engine for large-scale parallel and distributed data processing. Further, additional libraries which are built on the top of the core allows diverse workloads for streaming, SQL, and machine learning. It is responsible for memory management and fault recovery, scheduling, distributing and monitoring jobs on a cluster & interacting with storage systems. Spark Core API is available in R, SQL, Python, Scala and Java.
2. **Spark Streaming**Spark Streaming is the component of Spark which is used to process real-time streaming data. Thus, it is a useful addition to the core Spark API. It enables high-throughput and fault-tolerant stream processing of live data streams.
3. **Spark SQL + DataFrames**Spark SQL is a new module in Spark which integrates relational processing with Spark’s functional programming API. It supports querying data either via SQL or via the Hive Query Language. For those of you familiar with RDBMS, Spark SQL will be an easy transition from your earlier tools where you can extend the boundaries of traditional relational data processing.
4. **GraphX**GraphX is the Spark API for graphs and graph-parallel computation. Thus, it extends the Spark RDD with a Resilient Distributed Property Graph. At a high-level, GraphX extends the Spark RDD abstraction by introducing the Resilient Distributed Property Graph (a directed multigraph with properties attached to each vertex and edge).
5. **MLlib (Machine Learning)**  
   MLlib stands for Machine Learning Library. Spark MLlib is used to perform machine learning in Apache Spark.
6. **SparkR**It is an R package that provides a distributed data frame implementation. It also supports operations like selection, filtering, aggregation but on large data-sets.