# **Foundation**

## **HDFS:**

Hadoop Distributed File System is a distributed file system designed to store and manage large volumes of data across multiple commodity hardware nodes in a distributed computing environment. It is part of the Apache Hadoop project and is one of the core components of the Hadoop ecosystem.

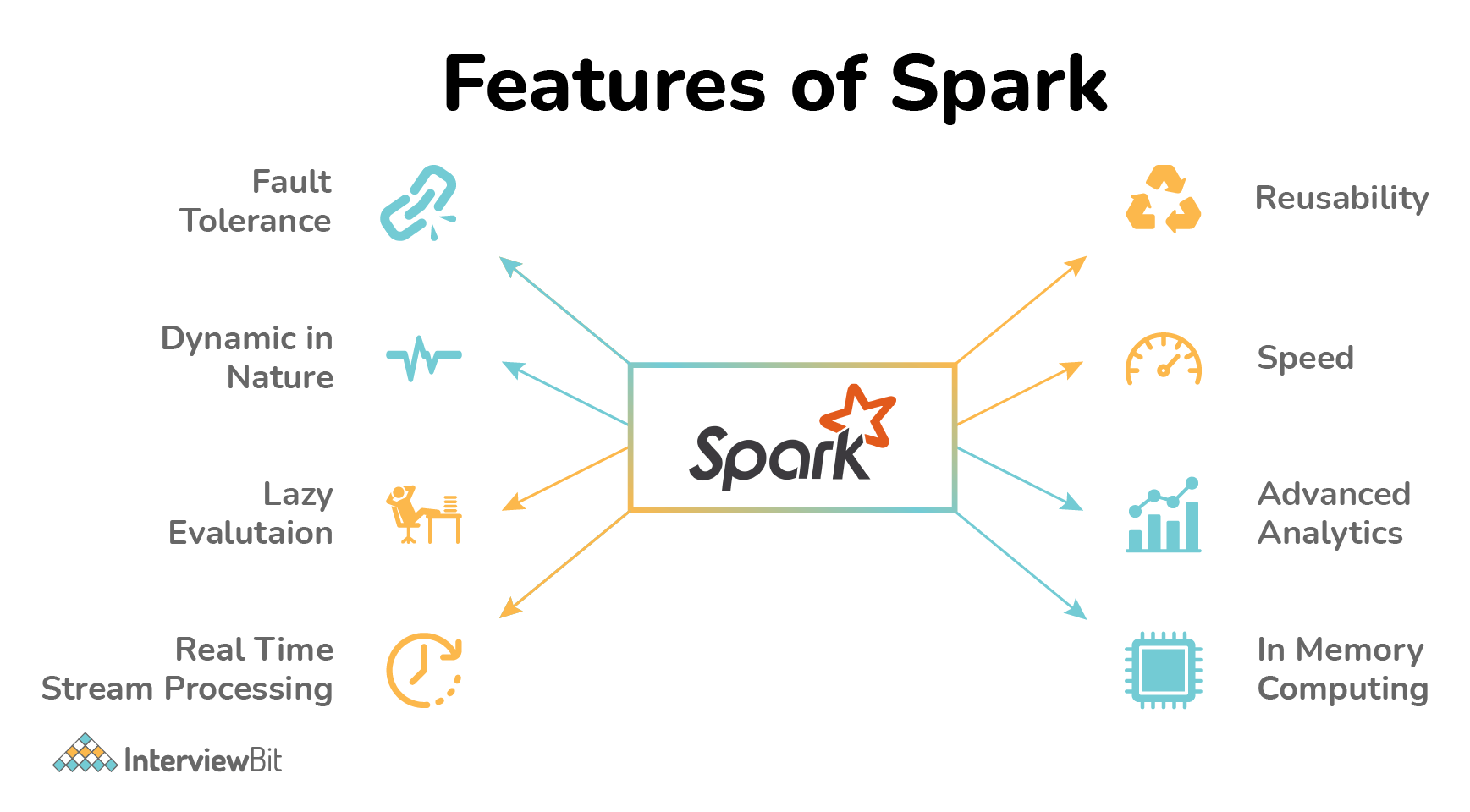
## **Map Reduce:**

MapReduce is a programming model and processing framework for parallel and distributed computation of large datasets across clusters of computers. It was popularized by Google and later adopted and implemented in the Apache Hadoop project as a core component of the Hadoop ecosystem.

# **Spark**

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application.

## **Features of Apache Spark:**



**Supports multiple languages −** Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.

**Advanced Analytics −** Spark not only supports ‘Map’ and ‘reduce’. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

**High Processing Speed:** Apache Spark helps in the achievement of a very high processing speed of data by reducing read-write operations to disk. The speed is almost 100x faster while performing in-memory computation and 10x faster while performing disk computation.

**Dynamic Nature:** Spark provides 80 high-level operators which help in the easy development of parallel applications.

**In-Memory Computation:** The in-memory computation feature of Spark due to its DAG execution engine increases the speed of data processing. This also supports data caching and reduces the time required to fetch data from the disk.

**Reusability:** Spark codes can be reused for batch-processing, data streaming, running ad-hoc queries, etc.

**Fault Tolerance:** Spark supports fault tolerance using RDD. Spark RDDs are the abstractions designed to handle failures of worker nodes which ensures zero data loss.

**Stream Processing:** Spark supports stream processing in real-time. The problem in the earlier MapReduce framework was that it could process only already existing data.

**Lazy Evaluation:** Spark transformations done using Spark RDDs are lazy. Meaning, they do not generate results right away, but they create new RDDs from existing RDD. This lazy evaluation increases the system efficiency.

**Hadoop Integration:** Spark also supports the Hadoop YARN cluster manager thereby making it flexible. Supports Spark GraphX for graph parallel execution, Spark SQL, libraries for Machine learning, etc.

**Cost Efficiency:** Apache Spark is considered a better cost-efficient solution when compared to Hadoop as Hadoop required large storage and data centers while data processing and replication.

**Active Developer’s Community**: Apache Spark has a large developers base involved in continuous development. It is considered to be the most important project undertaken by the Apache community.

## **Components of Spark or Spark EcoSystem:**

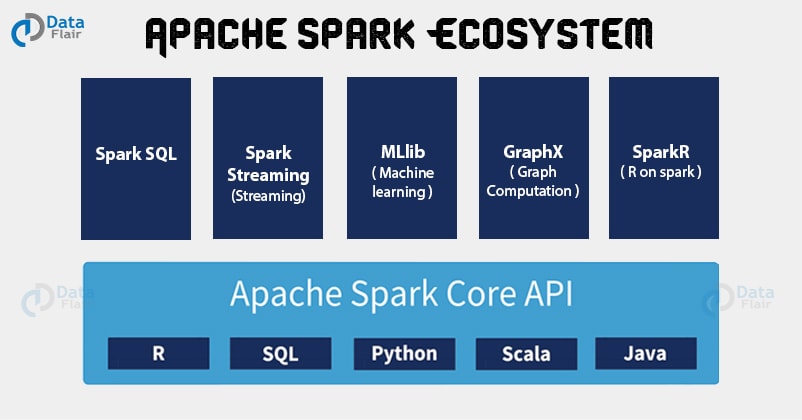
**Apache Spark Core:** Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.

**Spark SQL:** Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.

**Spark Streaming**: Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

**MLlib (Machine Learning Library):** MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of Apache Mahout (before Mahout gained a Spark interface).

**GraphX:** GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.



**Note: Apache Spark does not offer Cluster Management and Storage Management.**

## **Spark Architecture**

The Spark architecture depends upon two abstractions:

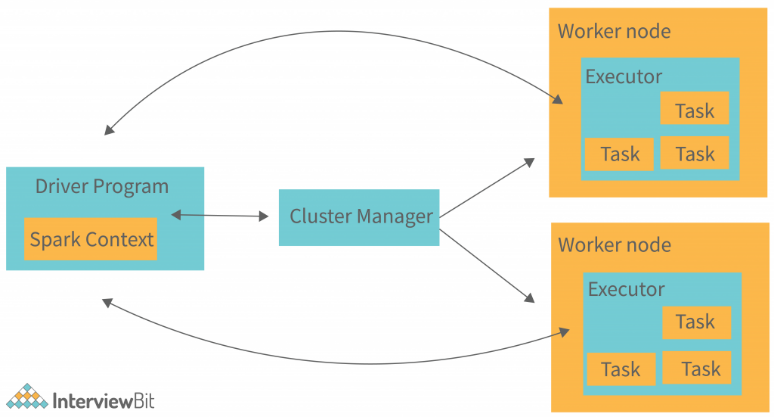
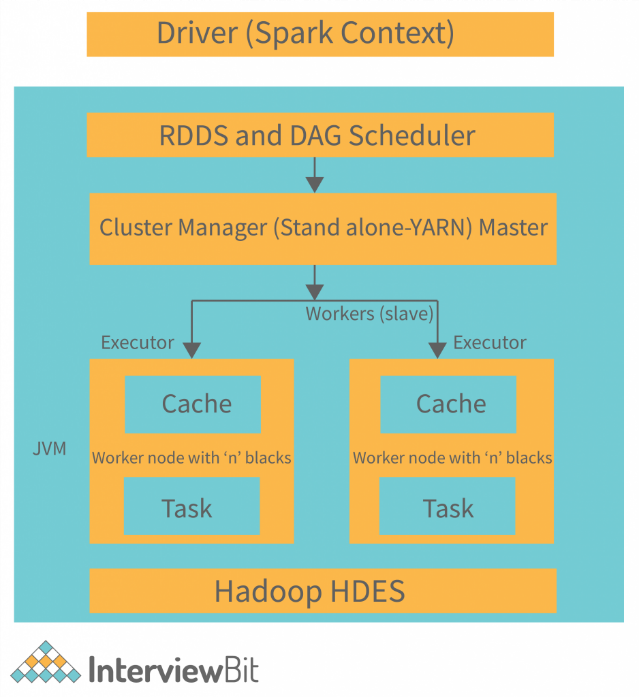
* Resilient Distributed Dataset (RDD)
* Directed Acyclic Graph (DAG)

**Resilient Distributed Dataset (RDD)**

**Directed Acyclic Graph (DAG)**

## **Spark Architecture Applications**

A high-level view of the architecture of the Apache Spark application is as follows:



**Spark Driver: The Spark driver is the master process that coordinates everything. It breaks down jobs into smaller tasks and schedules them to run on different machines (executors) in the cluster. It also creates a Spark context, which is like a controller, to monitor and connect to the cluster. The driver program starts the main application, sets up the Spark context, and manages the execution.**

**Spark Executors: Executors are responsible for running tasks and storing data in memory (cache). They start by registering with the driver. Executors handle tasks dynamically in Java processes, adding and removing resources as needed. The driver keeps an eye on their performance throughout the task execution.**

**Cluster Manager: The cluster manager takes care of resource allocation within the cluster. It controls how jobs are executed and where data is stored, dynamically assigning executors as needed. It monitors executors to ensure they are working efficiently and using resources well.**

**Worker Nodes: Worker nodes are the machines that actually process tasks and return the results. They get tasks from the Spark context and handle them in parallel by splitting them into smaller sub-tasks. Each worker node processes one Spark task, with parts of the task assigned to different executors for processing.qax**

# **PySpark**

PySpark is the Python API for Apache Spark. It is an open-source distributed system that is used for big data processing. It also supports Spark’s features like Spark DataFrame, Spark SQL, Spark Streaming, Spark MLlib and Spark Core.

## **What are the characteristics of PySpark?**

There are 4 characteristics of PySpark:

**Abstracted Nodes:** This means that the individual worker nodes can not be addressed.

**Spark API:** PySpark provides APIs for utilizing Spark features.

**Map-Reduce Model:** PySpark is based on Hadoop’s Map-Reduce model this means that the programmer provides the map and the reduce functions.

**Abstracted Network:** Networks are abstracted in PySpark which means that the only possible communication is implicit communication.