%md

## Spack Intro

1. Apache Spark is the most actively developed open-source engine for data processing on computer clusters.

2. This engine is now a standard data processing tool for developers and data scientists who want to work with big data.

3. Spark supports a variety of common languages (Python, Java, Scala, and R), includes libraries for a variety of tasks, from ETL to streaming to machine learning to graph processing & analytics.

4. Spark can run on a laptop to a cluster of thousands of servers. Spark can be deployed on Mesos, Hadoop via YARN, or Spark’s own cluster manager can all be used to deploy it.

## What is spark

1. Spark is an open-source, framework-based component that processes a large amount of unstructured, semi-structured, and structured data for analytics.

2. Apart from Hadoop and map-reduce architectures for big data processing, Apache Spark’s architecture is regarded as an alternative.

3. The Resilient Distributed Dataset (RDD) and Directed Acyclic Graph (DAG), Spark’s data storage and processing framework, are utilized to store and process data, respectively.

4. Spark architecture consists of four components, including the spark driver, executors, cluster administrators, and worker nodes.

5. It uses the Dataset and data frames as the fundamental data storage mechanism to optimize the Spark process and big data computation.

## Why Spark?

Spark is used to apply transformations on Big Data

Two scenarios in which it is particularly useful:

When data is too large (Big Data)

When we want to accelerate a calculation

### The following are main features of Spark

1. Speed

Spark performs up to 100 times faster than MapReduce for processing large amounts of data. It is also able to divide the data into chunks in a controlled way.

2. Powerful Caching

Powerful caching and disk persistence capabilities are offered by a simple programming layer.

3. Deployment

Mesos, Hadoop via YARN, or Spark’s own cluster manager can all be used to deploy it.

4. Real-Time

Because of its in-memory processing, it offers real-time computation and low latency.

5. Polyglot

In addition to Java, Scala, Python, and R, Spark also supports all four of these languages. You can write Spark code in any one of these languages. Spark also provides a command-line interface in Scala and Python

## Spark v/s Map Reduce

The following are differences between Spark & Map Reduce:

1. Spark is faster as it processes data in RAM (memory) while Hadoop reads and writes files to HDFS (on disk)

2. Spark is optimized for better parallelism , CPU utilization , and faster startup

3. Spark has richer functional programming model

4. Spark is especially useful for iterative algorithms

## Spark Architecture

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

1. Driver

2. Cluster Manager

3. Worker Nodes

4. Executors

## Driver

1. The Driver Program is a process that runs the main() function of the application and creates the SparkContext object.

2. The purpose of SparkContext is to coordinate the spark applications, running as independent sets of processes on a cluster.

3. To run on a cluster, the SparkContext connects to a different type of cluster managers and then perform the following tasks: -

a. It acquires executors on nodes in the cluster.

b. Then, it sends your application code to the executors. Here, the application code can be defined by JAR or Python files passed to the SparkContext.

c. At last, the SparkContext sends tasks to the executors to run.

## Cluster Manager

1. The role of the cluster manager is to allocate resources across applications. The Spark is capable enough of running on a large number of clusters.

2. It consists of various types of cluster managers such as Hadoop YARN, Apache Mesos and Standalone Scheduler.

3. Here, the Standalone Scheduler is a standalone spark cluster manager that facilitates to install Spark on an empty set of machines.

## Worker Nodes

1. The worker node is a slave node

2. Its role is to run the application code in the cluster.

## Executer

1. An executor is a process launched for an application on a worker node.

2. It runs tasks and keeps data in memory or disk storage across them.

3. It read and write data to the external sources.

4. Every application contains its executor

## Spark Components

The Spark project consists of different types of tightly integrated components. At its core, Spark is a computational engine that can schedule, distribute and monitor multiple applications.

The following are Spark components:

1. Spark Core

2. Spark SQL

3. Spark Streaming

4. MLlib

5. GraphX

## Spark Core

1. The Spark Core is the heart of Spark and performs the core functionality.

2. It holds the components for task scheduling, fault recovery, interacting with storage systems and memory management.

## Spark SQL

1. The Spark SQL is built on the top of Spark Core. It provides support for structured data.

2. It allows to query the data via SQL (Structured Query Language) as well as the Apache Hive variant of SQL called the HQL (Hive Query Language).

3. It supports JDBC and ODBC connections that establish a relation between Java objects and existing databases, data warehouses and business intelligence tools.

4. It also supports various sources of data like Hive tables, Parquet, and JSON.

## Spark Streaming

1. Spark Streaming is a Spark component that supports scalable and fault-tolerant processing of streaming data.

2. It uses Spark Core's fast scheduling capability to perform streaming analytics.

3. It accepts data in mini-batches and performs RDD transformations on that data.

4. Its design ensures that the applications written for streaming data can be reused to analyze batches of historical data with little modification.

5. The log files generated by web servers can be considered as a real-time example of a data stream.

## Mlib

1. The MLlib is a Machine Learning library that contains various machine learning algorithms.

2. These include correlations and hypothesis testing, classification and regression, clustering, and principal component analysis.

3. It is nine times faster than the disk-based implementation used by Apache Mahout.

## GraphX

1. The GraphX is a library that is used to manipulate graphs and perform graph-parallel computations.

2. It facilitates to create a directed graph with arbitrary properties attached to each vertex and edge.

3. To manipulate graph, it supports various fundamental operators like subgraph, join Vertices, and aggregate Messages.

## Spark Componets

Spark Jobs / Spark Tasks

1. A Spark Job(s) are composed of tasks

2. A Spark Task(s) are actual computation or transformation

Actions & Transformations

1. Actions & Transformations are output of a Task

2. If a task returns a DataFrame, Dataset, or RDD, it is a transformation.

3. If a task  returns anything else or does not return a value at all it is an action.

Lazy Evaluation

1. Lazy Evaluation is a trick commonly used for large data processing.

2. Lazy Evaluation means triggering processing only when a Spark action is run and not a Spark transformation.

3. This allows Spark to prepare a logical and physical execution plan to perform the action efficiently.

Wide and Narrow Transformations

The Spark transformations are divided into:

1. Wide Transformations: require data shuffle, are naturally the most expensive.

2. Narrow Transformations: does not require data shuffle

Maximize Parallelism In Spark

1. Spark’s efficiency is based on processing several tasks in parallel.

2. This is why optimizing a Spark job often means reading and processing as much data as possible in parallel.

3. And to achieve this goal, it is necessary to split a dataset into several partitions.

Partitions

1. Partition is a logical chunk of your DataFrame.

2. When reading the default size is 128 MB

3. Methods to modify partition : repartition and coalesce

4. Coalesce: only to reduce the number of partitions in a DataFrame, no shuffle

5. Repartition: to either increase or decrease the number of partitions, does a full shuffle

Caching

1. Spark can execute part of the DAG to store expensive intermediate results for downstream operations.

2. Like transformations, caching is applied when an action is executed

3. Caching is appropriate if you use the same DataFrame multiple times (EDA or ML model

4. Aside from this, you should not cache (performance degrades)

Resilient Distributed Datasets (RDD)

1. It is a key tool for data computation.

2. It enables you to recheck data in the event of a failure, and it acts as an interface for immutable data.

3. It helps in recomputing data in case of failures, and it is a data structure.

4. There are two methods for modifying RDDs: transformations and actions.

Directed Acyclic Graph (DAG)

1. The driver converts the program into a DAG for each job.

2. The Apache Spark Eco-system includes various components such as the API core, Spark SQL, Streaming and real-time processing, MLIB, and Graph X.

3. A sequence of connection between nodes is referred to as a driver.

4. As a result, you can read volumes of data using the Spark shell.

5.You can also use the Spark context  to  run a job / task or to stop a job / task.

## Modes of execution

You can choose from three different execution modes.

These determine where your app’s resources are physically located when you run your app.

You can decide where to store resources locally, in a shared location, or in a dedicated location.

Execution Modes

2. Cluster Mode

3. Client Mode

4. Local Mode

## Cluster Mode

1. Cluster mode is the most frequent way of running Spark Applications.

2. In cluster mode, a user delivers a pre-compiled JAR, Python script, or R script to a cluster manager.

3. Once the cluster manager receives the pre-compiled JAR, Python script, or R script, the driver process is launched on a worker node inside the cluster, in addition to the executor processes.

4. This means that the cluster manager is in charge of all Spark application-related processes.

## Client Mode

1. In contrast to cluster mode, where the Spark driver remains on the client machine that submitted the application, the Spark driver is removed in client mode

2. Therefore, it is the responsibility the client machine to maintain the Spark driver process.

3. These machines, usually referred to as gateway machines or edge nodes, are maintained on the client machine.

## Local Mode

1. Local mode runs the entire Spark Application on a single machine, as opposed to the previous two modes, which parallelized the Spark Application through threads on that machine.

2. As a result, the local mode uses threads instead of parallelized threads.

3. This is a common way to experiment with Spark, try out your applications, or experiment iteratively without having to make any changes on Spark’s end.

4. In practice, it is not recommended to use local mode for running production applications.

## Cluster Manager Types

There are several cluster managers supported by the system:

1. Standalone

2. Apache Mesos

3. Hadoop YARN

4. Kubernetes

## Standalone

1. A Spark cluster manager is included with the software package to make setting up a cluster easy.

2. The Resource Manager and Worker are the only Spark Standalone Cluster components that are independent.

3. There is only one executor that runs tasks on each worker node in Standalone Cluster mode.

4. When a client establishes a connection with the Standalone Master, requests resources, and begins the execution process, a Standalone Clustered master starts the execution process.

5. The client here is the application master, and it wants the resources from the resource manager. We have a Web UI to view all clusters and job statistics in the Cluster Manager.

## Apache Mesos

It can run Hadoop MapReduce and service apps as well as be a general cluster manager. Apache Mesos contributes to the development and management of application clusters by using dynamic resource sharing and isolation. It enables the deployment and administration of applications in large-scale cluster environments.

The Mesos framework includes three components:

1. Mesos Master:A Mesos Master cluster provides fault tolerance (the capability to operate and recover from loss when a failure occurs). Because of the Mesos Master design, a cluster contains many Mesos Masters.

2. Mesos Slave: A Mesos Slave is an instance that delivers resources to the cluster. When a Mesos Master assigns a task, Mesos Slave does not assign resources.

3. Mesos Frameworks: Applications can request resources from the cluster so that the application can perform the tasks. Mesos Frameworks allow for this.

## Hadoop Yarn

A key feature of Hadoop 2.0 is the improved resource manager.

The Hadoop ecosystem relies on YARN to handle resources. It consists of the following two components:

1. Resource Manager: It controls the allocation of system resources on all applications. A Scheduler and an Application Manager are included. Applications receive resources from the Scheduler.

2. Node Manager: Each job or application needs one or more containers, and the Node Manager monitors these containers and their usage. Node Manager consists of an Application Manager and a Container Manager. Each task in the MapReduce framework runs in a container. The Node Manager monitors the containers and resource usage, and this is reported to the Resource Manager. A

## Kubernets

Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

1. Planet Scale:

Designed on the same principles that allow Google to run billions of containers a week, Kubernetes can scale without increasing your operations team.

2. Never Outgrow:

Whether testing locally or running a global enterprise, Kubernetes flexibility grows with you to deliver your applications consistently and easily no matter how complex your need is.

3. Run K8s Anywhere: Kubernetes is open source giving you the freedom to take advantage of on-premises, hybrid, or public cloud infrastructure, letting you effortlessly move workloads to where it matters to you.