**Assignment(22-12-2023)**

**Introduction to Apache**

* Apache Spark is an open-source, distributed processing system used for big data workloads.
* It utilizes in-memory caching, and optimized query execution for fast analytic queries against data of any size.
* Spark was introduced by Apache Software Foundation for speeding up the Hadoop computational computing software process.
* Spark uses Hadoop in two ways – one is **storage** and second is **processing**.
* Since Spark has its own cluster management computation, it uses Hadoop for storage purpose only.
* The Apache HTTP Server is a widely-used, open-source web server software that delivers web content across the internet.
* It plays a crucial role in serving static and dynamic web pages, handling user requests, and facilitating communication between clients (web browsers) and servers.

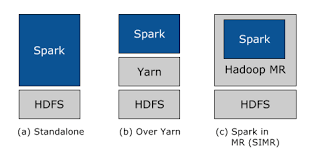
**Apache Spark**

* The main feature of Spark is its **in-memory cluster computing** that increases the processing speed of an application.
* Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming.
* Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate tools.

**Features of Spark**

* **Support Multiple Languages:**It provides multiple languages like Scala,Java,Python and R programming languages.However still multiple languages exist,The primary language is Scala.
* **Advanced Analytics** − Spark not only supports ‘Map’ and ‘reduce’. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

Spark Built on Hadoop



**Imagine a Neighbourhood:**

* Think of Spark as a worker who helps with tasks. In a standalone setup, Spark sets up shop right next to Hadoop (like living in the same neighborhood).
* They work together, and both Spark and Hadoop's MapReduce can do their jobs side by side.

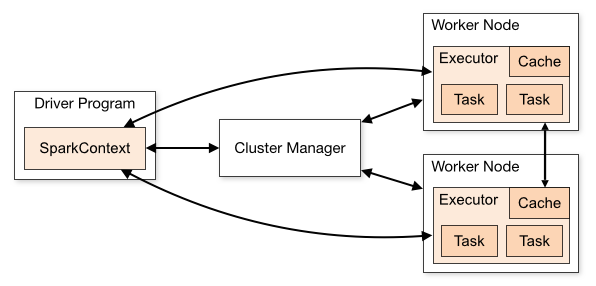
**2. Hadoop Yarn Deployment:**

* Spark at the Hadoop Party: In this scenario, Spark joins the Hadoop party but doesn't need a special invitation.
* It runs on Hadoop's Yarn, which is like the organizer making sure everything runs smoothly.
* This way, Spark becomes part of the Hadoop family, and other Hadoop components can join the fun too.

**3. Spark in MapReduce (SIMR):**

* Adding Spark to MapReduce's Team: SIMR is like inviting Spark to play on MapReduce's team.
* It's a way to use Spark alongside MapReduce, kind of like having two skilled players on the same team.
* Even if you don't have special powers (administrative access), you can still use Spark and its tools.

**PySpark Architecture**



PySpark work under the Master-Slave Architecture ,Master is the Super Node that means it is the head of the all Nodes. Slave is the worker node of all nodes.

**1. Spark Cluster:**

Imagine a big playground where a lot of tasks need to be done. This is your Spark cluster.

**2. Driver Program:**

Think of the "Driver Program" as the coach or manager. It's like the brain that plans and coordinates the tasks. In PySpark, this is your Python script or application.

**3. Spark Context:**

The "Spark Context" is like the coach's whistle. It's the communication channel between your Python program (driver) and the Spark cluster. It tells the cluster what tasks to perform.

**4. Executors:**

Now, think of "Executors" as the players on the field. These are the worker nodes in your cluster that do the actual work. They follow the coach's instructions (Driver Program) and perform tasks like calculations or data transformations.

**5. Resilient Distributed Datasets (RDD):**

In PySpark, data is represented as RDDs. These are like playbooks that tell the players (Executors) what actions to take on the data. RDDs are distributed across the cluster, allowing parallel processing.

**6. DataFrame API:**

Spark also offers a DataFrame API, which is like a more organized playbook. It's a higher-level abstraction over RDDs and makes it easier to work with structured data, similar to working with tables in a database.

**7. Cluster Manager:**

Picture a referee ensuring fair play. The "Cluster Manager" makes sure tasks are distributed evenly among the players (Executors) and manages their performance.

**8. Data Sources and Sinks:**

These are like the water fountains on the field. Data can flow into your Spark application from various sources (like a file or a database) and flow out to different destinations after processing.

**9. Application Master:**

In YARN (Yet Another Resource Negotiator) mode, there's an "Application Master" that negotiates resources with the cluster manager and coordinates the execution of tasks.

**10. Driver and Executor Communication:**

Communication between the Driver Program and Executors happens through the SparkContext. The Driver sends tasks to the Executors, and they report back with results.

In the PySpark we can execute the things as follows

