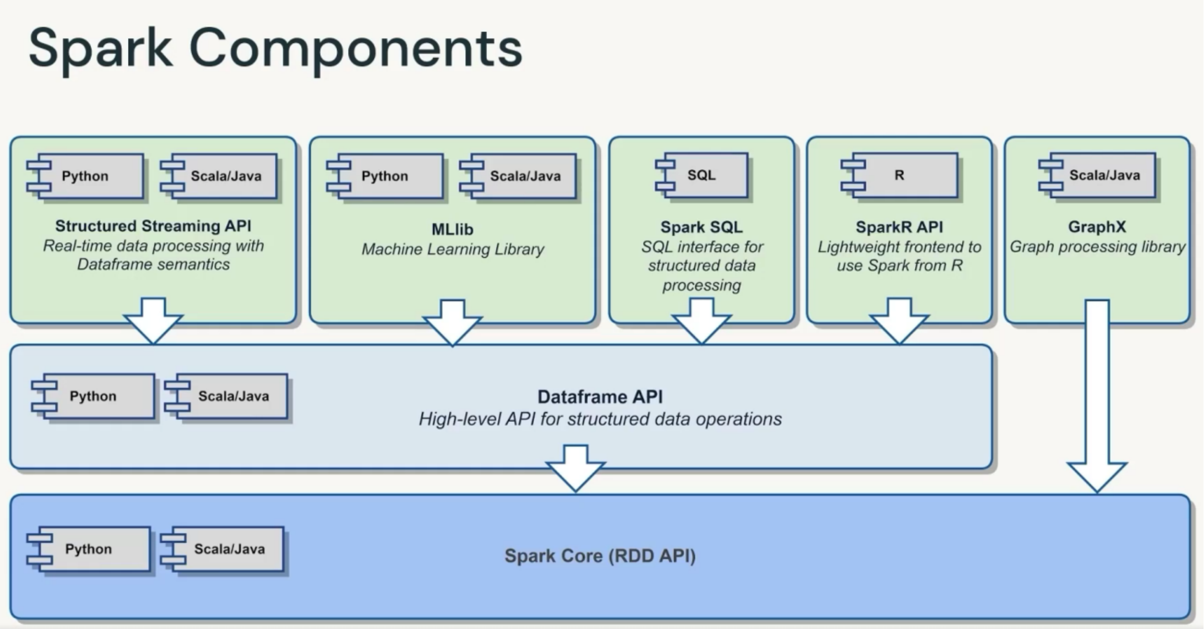
Apache Spark

Apache Spark is an open source, distributed, unified analytics engine for fast, last-scale data processing.

- Supports SQL, Structured Streaming, Machine Learning, and Graph Processing.
- Performs In memory computing.

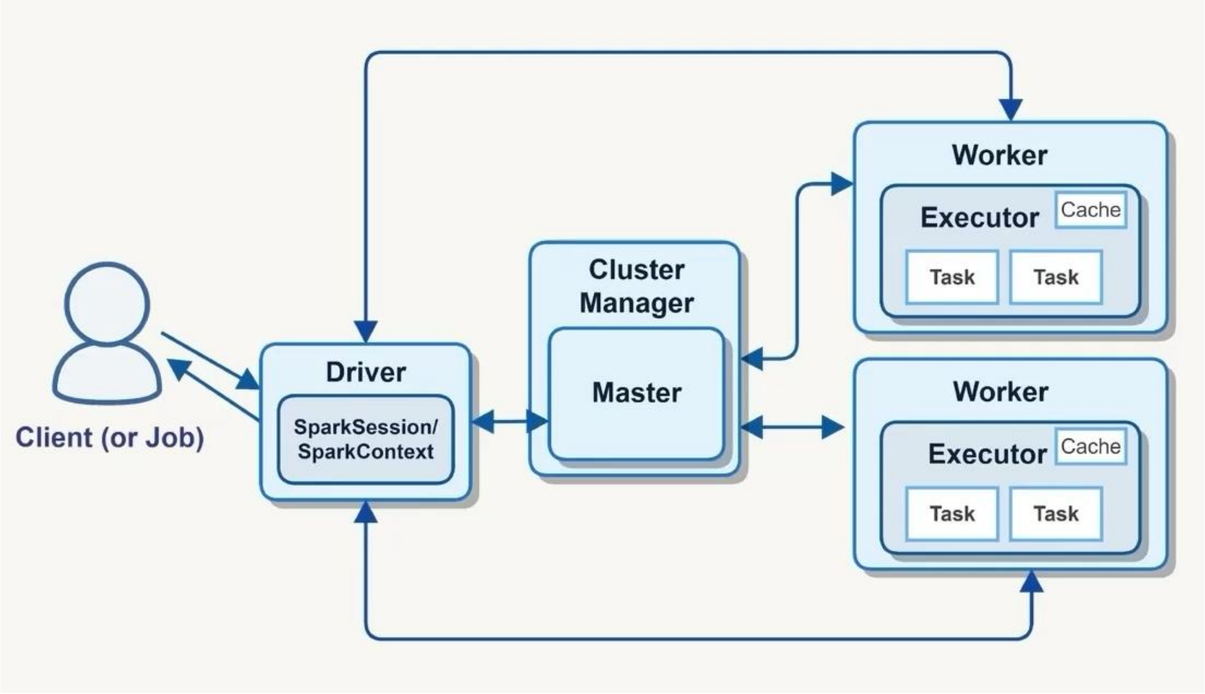
Components



The core engine provides foundation for all applications, this includes memory management, fault tolerance & recovery, and scheduling tasks distribution.

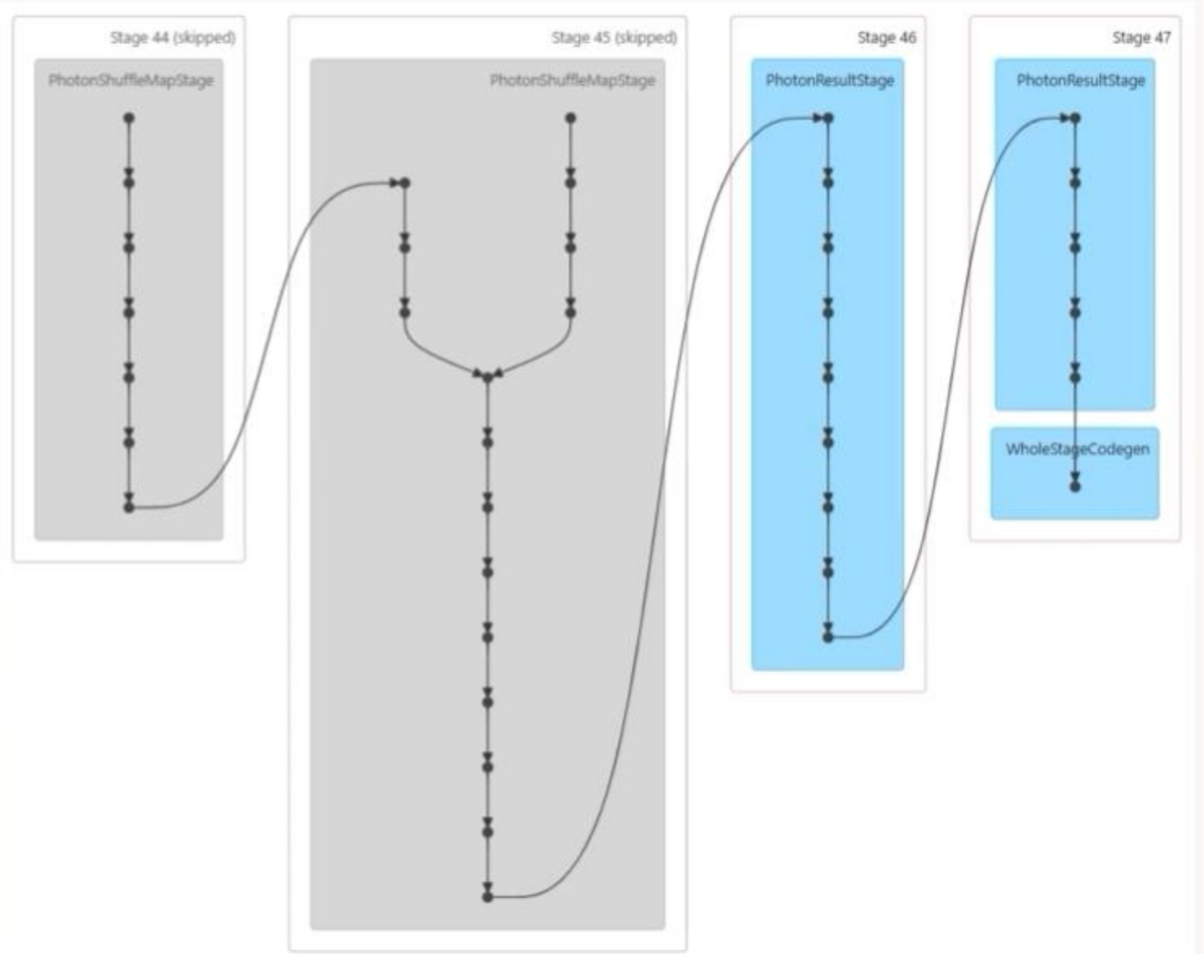
This core engine is exposed to higher level APIs like Dataframe, Structured Streaming, MLib, and GraphX which are then exposed to different languages like scala and python.

Architecture



Sparks follows Client-Server Architecture and have the following components

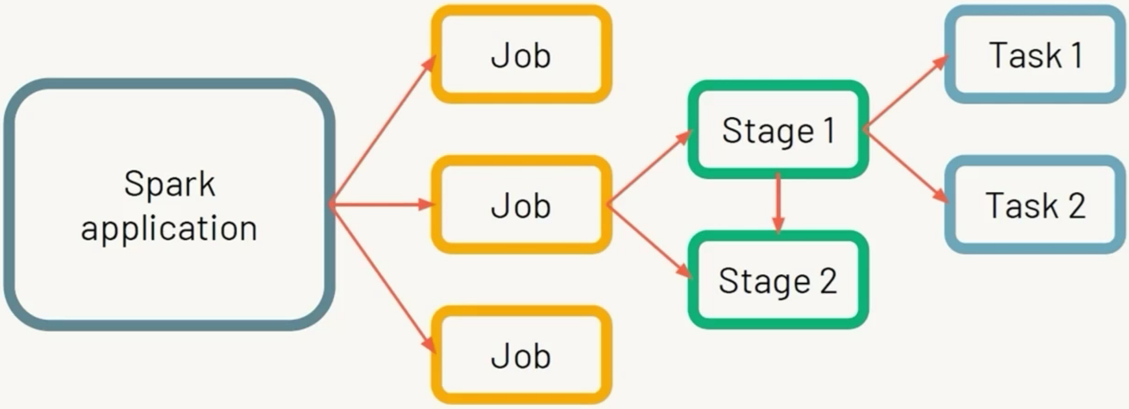
1. **Driver:** Driver is the centralized entry point for the spark application it is responsible for managing and optimizing the execution. (this can be a query or a job).
    a. This creates Spark Session.
    b. Analyses the spark application and constructs a DAG.
    c. Schedules and distributes tasks to executors for execution.
    d. It also monitors these executions and handles failures.
    e. Finally returns the results to the client.
2. **Cluster Manager:** Manages resources and allocates them to the Driver.
3. **Workers:** are the Nodes (VMs) that host executors. A worker can have multiple executors.
4. **Executors:** run tasks and cache data. So, these are ones which do heavy lifting.
    a. These can be configured based on requirements like number of CPU cores and memory.
    b. These stores intermediate and final results in memory or disk.



Spark application is implemented using DAGs meaning tasks flow in one direction.

Spark Jobs are broken down into stages. Stages are the group of tasks that can be executed in parallel.

Spark Application Execution



1. An application can trigger multiple jobs.
2. Jobs are high level operations which will be trigged by actions.

3. Jobs are implemented by DAGs.
4. These jobs are divided into stages.
5. Stages are groups of tasks that can be executed in parallel.
6. Stages communicate with each other with a process called shuffle process.

The tasks within a stage run in parallel, Known as **Shared Nothing** mode.

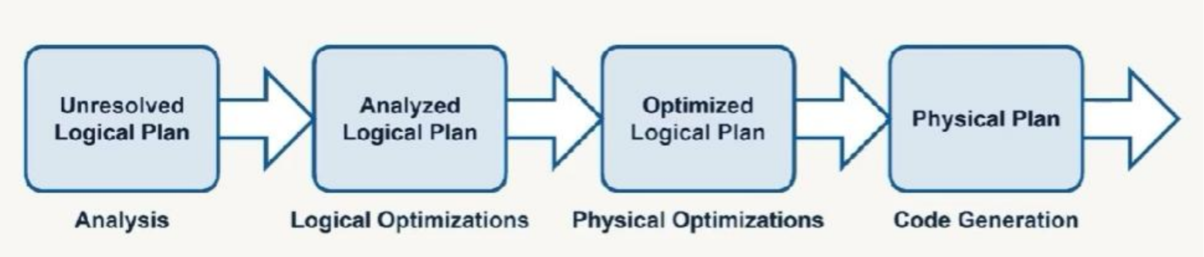
Databricks Compute types

1. **All purpose:** For developers running experiments in notebooks
2. **Job clusters:** For ephemeral tasks like ETL.
3. **SQL warehouse:** For BI End points.

Dataframe API

Dataframes are distributed collection of records with defined schema.

- It is built on top of core API designed for SQL-like operations.
- Dataframes are evaluated using DAGs, using lazy evaluations. This provides lineage and fault tolerance.



When Driver creates an optimized execution plans it went through a series of transformation steps to minimize the use of resources and execution time.

Dataframes can be created from JSONs, Parquets, CSVs, external databases etc.

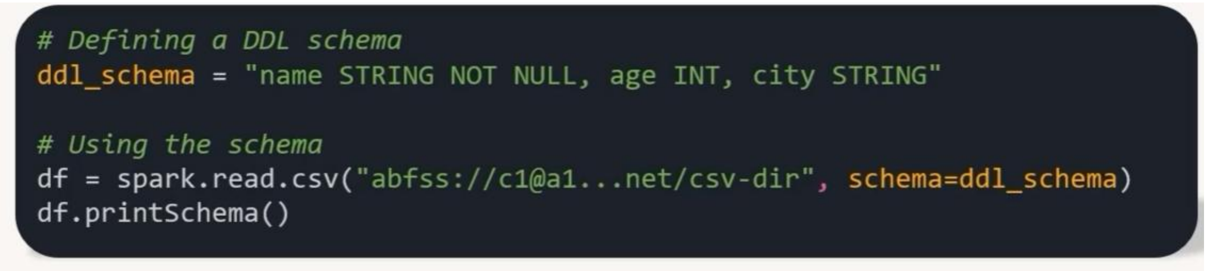
Dataframe Optimizations:

- Adaptive Query Optimization:
  - **Cost based** optimization during runtime based on actual data characteristics and execution patterns.
- In-Memory Columnar Storage:
  - Stores data in columnar format (Tungsten) for efficient query performance.
- Catalyst Optimizer:
  - Applies **rule based**, **Cost based** optimization to improve query performance.
- Photon engine:
  - Photon engine is a vectorized query engine that processes data in batches rather than row-by-row processing. (Runs by default on SQL warehouses and serverless compute need to enable on all-purpose job clusters)

We can use spark.read and spark.write for reading and writing data.

When reading the data from non-self-describing schema formats like CSVs use StructType to define data types to make reading much more efficient. No need for formats like Parquets.

We can also use SQL ddl to define schema



```
# Defining a DDL schema
ddl_schema = "name STRING NOT NULL, age INT, city STRING"

# Using the schema
df = spark.read.csv("abfss://c1@a1...net/csv-dir", schema=ddl_schema)
df.printSchema()
```

Transformations and Actions

Dataframes are immutable. Once they are created, they cannot be modified. Instead, we apply

Transformations To create new Dataframe from and existing ones (called coarse grained transformations).

Ex: select(), filter(), withColumn(), groupBy(), agg()

Actions: Actions are computation triggers like showing or saving output.

Ex: show(), count(), first(), write()

We can call multiple transformations but the job is only created when the action is requested.

SparkSQL

SparkSQL is built on top of Dataframe API supporting full SQL to enable SQL queries on Dataframe. This is usually how we interact with data when we run code in cells.

SQL Metastores

A metastore is responsible for defining table schemas and locations, partitions, UDFs etc.

Unity Catalog: is a centralized metadata service in databricks account and assigned to a Databricks workspace that enables fine-grained security and governance across all datasets.

Distributed Programming Core Principles

- Each Node works independently and manages its own resources (CPU, disk, memory)
- Adding more nodes will improve performance linearly
- Fault tolerance should be isolated to individual nodes and should not affect entire system.
- Resources should be partitioned across nodes.

Spark performs map shuffle reduce when triggered by

- Wide transformations
- Key based operations
- repartitioning