Apache Spark

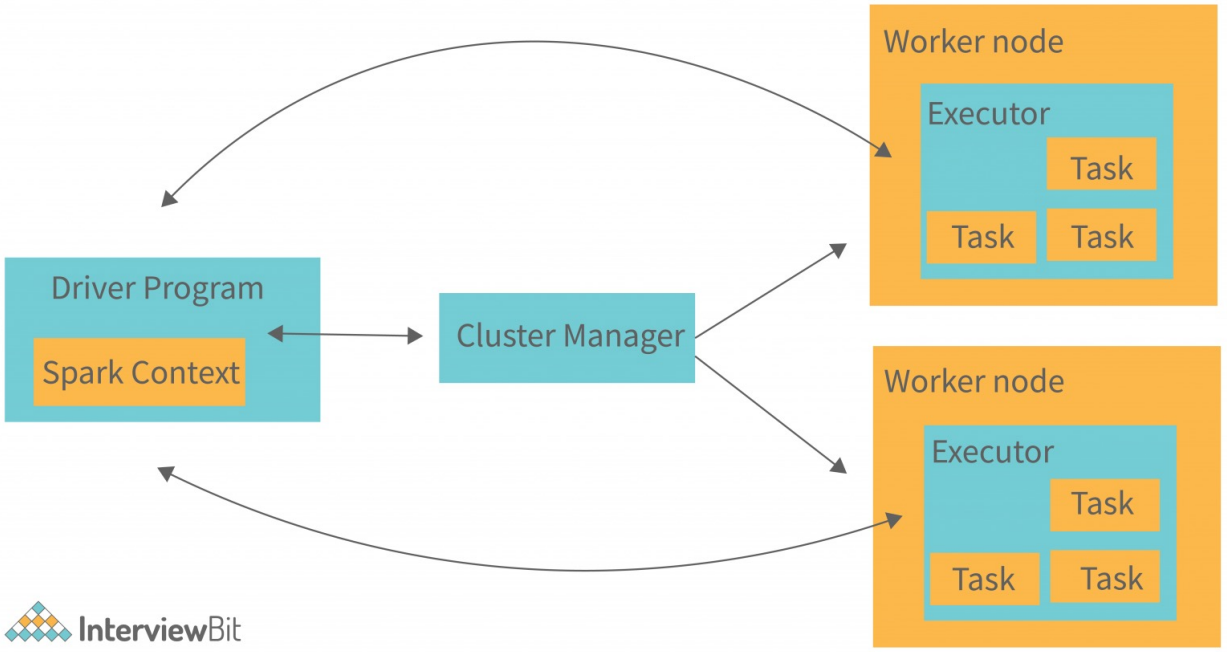
- Apache Spark™ is a multi-language engine for executing data engineering, data science, and machine learning on single-node machines or clusters.

Spark SQL engine

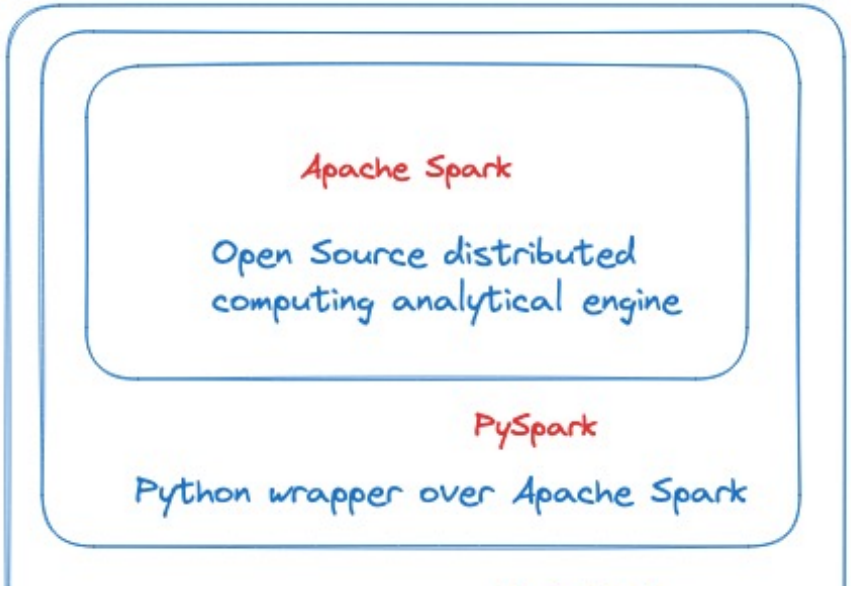
- Spark SQL engine: under the hood
  - Apache Spark™ is built on an advanced distributed SQL engine for large-scale data
- Adaptive Query Execution
  - Spark SQL adapts the execution plan at runtime, such as automatically setting the number of reducers and join algorithms.
- Support for ANSI SQL
  - Use the same SQL you're already comfortable with.
- Structured and unstructured data
  - Spark SQL works on structured tables and unstructured data such as JSON or images.

Distributed

- Apache Spark has its architectural foundation in the resilient distributed dataset (RDD), a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way.



PySpark


Apache Spark
Open Source distributed computing analytical engine
PySpark
Python wrapper over Apache Spark

PySpark DataFrame

- PySpark DataFrames are lazily evaluated. They are implemented on top of RDDs. When Spark transforms data, it does not immediately compute the transformation but plans how to compute later.
- PySpark applications start with initializing SparkSession.

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()
```

PySpark DataFrame

- Create a PySpark DataFrame with **pyspark.sql.SparkSession.createDataFrame**
- typically passing data as:
  - a list of lists
  - a list of tuples
  - a list of dictionaries
  - a list of pyspark.sql.Rows
  - a pandas DataFrame
  - an RDD consisting of such a list.
- createDataFrame takes a schema argument to specify the schema of the DataFrame. When it is omitted, PySpark infers the corresponding schema by taking a sample from the data.

DataFrame Exploration

- df.show()
- df.printSchema()
- df.columns
- df.describe()

DataFrame Methods

- df.show()
- df.collect()
- df.take()
- df.select()
- df.toPandas()
- df.<column name>
  - PySpark DataFrame is lazily evaluated and simply selecting a column does not trigger the computation but it returns a Column instance.
- df.withColumn()
- df.filter()
-

Data In and Out

CSV

- spark.read.csv('foo.csv', header=**True**).show()
- df.write.csv('foo.csv', header=**True**)

Parquet

- df.write.parquet('bar.parquet')
- spark.read.parquet('bar.parquet').show()

SQL

- df.createOrReplaceTempView("tableA")
- spark.sql("SELECT count(*) from tableA").show()

Session

```
from pyspark.sql import SparkSession

SparkSession.builder.master("local[*]").getOrCreate().stop()
```

```
spark = SparkSession.builder.remote("sc://localhost:15002").getOrCreate()
```

Spark SQL API

- Core Classes
- Spark Session
- Configuration
- Input/Output
- DataFrame
- Column
- Data Types
- Row
- Functions
- Window
- Grouping
- Catelog
- Avro
- Observation
- UDF
- UDTF
- Protobuf

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/index.html

Pandas API on Spark

- A pandas DataFrame can be converted to a pandas-on-Spark DataFrame

```
psdf = ps.from_pandas(pdf)
```

- Creating pandas-on-Spark DataFrame from Spark DataFrame.

```
import pandas as pd
import numpy as np
import pyspark.pandas as ps
from pyspark.sql import SparkSession

spark = SparkSession.builder.getOrCreate()

sdf = spark.createDataFrame(pdf)
psdf = sdf.pandas_api()
```

Pandas on Spark

- Input/Output
- General functions
- Series
- DataFrame
- Index objects
- Index
- Window
- GroupBy
- Resampling
- Machine Learning utilities
- Extensions

https://spark.apache.org/docs/latest/api/python/user_guide/pandas_on_spark/supported_pandas_api.html

Best Practices

General

- DRY
- SOLID
- Encapsulation
- Similar Level of Detail

Big Data

- Streaming (Generators)
- Category Types (instead of strings)

PySpark

- Chain transformations together and avoid unnecessary intermediate operations.
- Use DataFrame/Dataset over RDD
- Use coalesce() over repartition()
- Use mapPartitions() over map()
- Use Serialized data format's
- Avoid UDF's (User Defined Functions)
- Caching data in memory
- Reduce expensive Shuffle operations
- Disable DEBUG & INFO Logging

https://sparkbyexamples.com/spark/spark-performance-tuning/