PySpark

Pandas vs PySpark

- PySpark is a library for working with large datasets in a distributed computing environment, while pandas is a library for working with smaller, tabular datasets on a single machine.
- PySpark is built on top of the Apache Spark framework and uses the Resilient Distributed Datasets (RDD) data structure, while pandas uses the DataFrame data structure.
- PySpark is designed to handle data processing tasks that are not feasible with pandas due to memory constraints, such as iterative algorithms and machine learning on large datasets.
- PySpark allows for parallel processing of data, while pandas does not.
- PySpark can read data from a variety of sources, including Hadoop Distributed File System (HDFS), Amazon S3, and local file systems, while pandas is limited to reading data from local file systems.
- PySpark can be integrated with other big data tools like Hadoop and Hive, while pandas is not.
- PySpark has a steeper learning curve than pandas, due to the additional concepts and technologies involved (e.g. distributed computing, RDDs, Spark SQL, Spark Streaming, etc.).
- Pandas is sufficnent in most of the small cases

In summary, use PySpark for large datasets and complex tasks that are not feasible with pandas, and use pandas for small datasets and simple tasks that can be handled on a single machine.

SparkSession()

The entry point to programming Spark with the Dataset and DataFrame API.

A SparkSession can be used create DataFrame, register DataFrame as tables, execute SQL over tables, cache tables, and read parquet files.

we used .getOrCreate() which will create and instantiate SparkSession into our object spark. Using the .getOrCreate() method would use an existing SparkSession if one is already present else will create a new one.

- Spark DataFrames are excellent for building a scalable application. Spark DataFrame is Immutable.

- most of the functionality of pandas df is present in spark df but not all, also spark df have some additional functionalities of their own. Complex operations are difficult to perform as compared to Pandas DataFrame.
- We can also convert the PySpark DataFrame into a Pandas DataFrame. This enables the functionality of Pandas methods on our DataFrame which can be very useful : .toPandas()

Documentation for dataframe

attributes and information
https://spark.apache.org/docs/latest/api/python/reference/pyspark.pandas/frame.html
functions
https://spark.apache.org/docs/3.1.1/api/python/reference/api/pyspark.sql.DataFrame.html

PySpark RDD

RDDs ia an important component of PySpark. PySpark RDD is one of the fundamental data structures for handling both structured and unstructured data and lacks any schema. Compared to network and disc sharing, PySpark RDD speeds up in-memory data sharing by 10 to 100 times.

Resilient Distributed Datasets, often known as RDDs, are the components used in a cluster's parallel processing that run and operate across numerous nodes. Since RDDs are immutable elements, you cannot alter them after creation.

Partitioning in PySpark
Data partitioning is an important concept in Spark and understanding how Spark deals with partitions allow one to control parallelism. A partition in Spark is the division of the large dataset with each part being stored in multiple locations across the cluster. By default, Spark partitions the data at the time of creating RDD based on several factors such as available resources, external datasets etc, however, this behavior can be controlled by passing a second argument called minPartitions which defines the minimum number of partitions to be created for an RDD.

Benefits of PySpark RDD
- Performance.
- RDDs are effective and quick due to parallel processing and data storage.
- Consistency.
- An RDD is immutable and unchangeable contents guarantee data stability.
- Tolerance for errors.
- Users can specify which RDDs they plan to reuse and select a storage method (memory or disc) for them.