**Day 11 Notes**

**Name : Podutur Lahari**

**Date : 20-11-2024**

**Introduction to PySpark RDDs**

* Resilient Distributed Dataset (RDD):
  + A core data structure in PySpark designed for distributed data processing.
  + Immutable and fault-tolerant.
  + Partitions data across the cluster for parallel computation.
* Key Characteristics:
  + Immutable: Once created, RDDs cannot be modified but can be transformed into new RDDs.
  + Fault-Tolerant: RDDs automatically recover lost data due to partitioning.
  + Lazily Evaluated: Transformations are not executed until an action is triggered.

**Types of Operations in PySpark RDDs**

1. Transformations:
   * Perform operations on an existing RDD to create a new one.
   * Lazy Evaluation: Transformations generate a computation plan (DAG) but do not execute immediately.
   * Examples: map(), filter(), union(), flatMap().
2. Actions:
   * Trigger computation and return results to the driver program.
   * Actions are eagerly executed, finalizing the transformations.
   * Examples: collect(), count(), first(), take().

**Transformations in PySpark RDDs**

* Definition: Operations that transform data in an RDD and return a new RDD.
* Key Transformations:
  + map():
    - Applies a function to each element of an RDD.
    - Example: Adding 10 to all elements.
  + filter():
    - Selects elements satisfying a given condition.
    - Example: Filtering even numbers.
  + union():
    - Merges two RDDs.
    - Example: Combining RDDs of numbers.
  + flatMap():
    - Splits and flattens data.
    - Example: Breaking sentences into words.
* Execution:
  + Transformations create a DAG for computation, which is executed only when an action is called.

**Actions in PySpark RDDs**

* Definition: Operations that trigger execution of the transformations and return final results.
* Key Actions:
  1. collect():
     + Returns all elements of the RDD to the driver program.
     + Example: Displaying all elements.
  2. count():
     + Counts the number of elements in the RDD.
  3. first():
     + Returns the first element of the RDD.
  4. take(n):
     + Fetches the first n elements.
  5. reduce():
     + Aggregates elements using a function.
     + Example: Summing elements.
  6. saveAsTextFile():
     + Saves the RDD to the specified file path.

**Pair RDDs in PySpark**

* Definition: Specialized RDDs with key-value pair elements.
* Purpose:
  + Used in scenarios requiring grouped or hierarchical data processing.
  + Example: Customer purchases (Customer ID as key, Purchases as values).
* Creating Pair RDDs:
  + Use a list of tuples to define key-value pairs.
  + Example: [("Alice", 23), ("Bob", 45)].

**Transformations in Pair RDDs**

1. reduceByKey():
   * Aggregates values for each key.
   * Example: Summing scores for each student.
2. sortByKey():
   * Sorts data by keys in ascending or descending order.
   * Example: Sorting names alphabetically.
3. groupByKey():
   * Groups values sharing the same key into an iterable.
   * Example: Grouping scores for each student.

**Actions in Pair RDDs**

* Specialized Actions:
  + countByKey():
    - Counts the number of values for each key.
    - Example: Counting occurrences of words in a dataset.
* Other Actions:
  + Pair RDDs also support general actions like collect(), take(), and reduce().

**Benefits of Using PySpark RDDs**

1. Scalability:
   * Handles large datasets by distributing computations across multiple nodes.
2. Fault-Tolerance:
   * Automatically reconstructs lost partitions in case of node failures.
3. Flexibility:
   * Supports a variety of transformations and actions.
4. Efficiency:
   * Lazy evaluation optimizes execution by minimizing data shuffling.

**Applications of PySpark RDDs**

1. Data Cleaning:
   * Example: Removing null values and filtering data.
2. Data Aggregation:
   * Example: Summing or averaging numerical fields.
3. Log Analysis:
   * Example: Parsing and extracting meaningful insights from log files.
4. Key-Value Data Processing:
   * Example: Analyzing grouped data such as sales by region.

**Conclusions**

* PySpark RDDs offer a powerful abstraction for distributed data processing.
* Transformations allow flexible data manipulation, while actions trigger the computation of results.
* Pair RDDs are particularly useful for key-value data, a common format in real-world datasets.

This overview provides a comprehensive understanding of PySpark RDDs and their operations. For practical application, setting up a SparkContext and using PySpark in tools like Google Colab or local environments is essential.