**Assignment(23-12-2023)**

Apache Spark is a powerful open-source distributed computing system that provides fast and general-purpose cluster computing for big data processing. Spark provides a high-level API in Java, Scala, Python, and R, and it supports SQL, streaming data, machine learning, and graph processing.

Here are some **basic Spark commands** that we can use

**Starting Spark Shell:**

To start Spark Shell with Scala: spark-shell

To start Spark Shell with Python: pyspark

**Loading Data:**

Load data from a text file into an RDD (Resilient Distributed Dataset)

val textFile = sc.textFile("path/to/textfile.txt")

**Creating RDDs (Resilient Distributed Datasets):**

Create an RDD from a collection:

val data = Array(1, 2, 3, 4, 5)

val rdd = sc.parallelize(data)

**Actions:**

**Prdd.collect()** which returns all elements of the RDD

**rdd.take(3)** which returns the first 3 elements of the RDD

**rdd.count()** which returns the number of elements in the RDD perform actions to retrieve results from RDDs

**Transformations:**

Apply transformations to create new RDDs:

val squaredRDD = rdd.map(x => x \* x)

val filteredRDD = rdd.filter(x => x > 3)

Here are some **Intermediate Spark commands** that we can use

**Caching:**

Persist RDD in memory for faster access:

rdd.persist(StorageLevel.MEMORY\_ONLY)

**Joins:**

Perform joins on Pair RDDs:

val rdd1 = sc.parallelize(Seq(("A", 1), ("B", 2)))

val rdd2 = sc.parallelize(Seq(("A", "apple"), ("B", "banana")))

val joinedRDD = rdd1.join(rdd2)

**Union and Distinct:**

Combine two RDDs or get distinct elements:

val unionRDD = rdd

**Grouping and Aggregation:**

Group data by key and perform aggregations:1.union(rdd2)

val distinctRDD = rdd.distinct()

Here are some **Advanced Spark commands** that we can use

**Window Operations:**

Perform operations on sliding windows in time-series data:

val windowedRDD = rdd.window(windowSize, slideInterval)

**Spark SQL:**

Execute SQL queries on Spark data:

val dataFrame = spark.read.json("path/to/json")

dataFrame.createOrReplaceTempView("myTable")

val result = spark.sql("SELECT \* FROM myTable WHERE age > 21")

**Machine Learning with MLlib:**

Train machine learning models with Spark's MLlib:

import org.apache.spark.ml.feature.VectorAssembler

import org.apache.spark.ml.regression.LinearRegression

// Prepare data and features

val assembler = new VectorAssembler().setInputCols(Array("feature1", "feature2")).setOutputCol("features")

val assembledData = assembler.transform(data)