PROJECT WORK

Hadoop, Spark Installation and algorithm execution

Operating System and Cloud Computing DS203

Our Team Members :

Rishi Koushik : 21bds067
Nischay Kondai : 21bds045
Vamsi Madhav : 21bds051
Yuvraaj Bhama : 21bds071

What is Hadoop?

Hadoop is a software framework that is freely available for storing and processing big data sets across multiple computers. The aim is to allow the system to expand from one server to thousands of machines, each with local storage and computing power. Initially created by Doug Cutting and Mike Cafarella in 2005, the software was designed to manage the enormous quantities of data created by web crawlers.

What is Spark?

Apache Spark is an open-source distributed computing system that processes large data sets across clusters of computers. Developed at the University of California, Berkeley, in 2009 and made available to the public in 2010, Spark uses an in-memory computing engine to provide faster data processing than traditional disk-based systems. It offers different libraries and APIs for data processing, including SQL, machine learning, and graph processing. Spark's core engine uses RDDs to allow parallel processing across nodes in a cluster.

Steps for Installation:

Hadoop

1. Download Hadoop
2. Install Java
3. Configure SSH
4. Set up environment variables
5. Configure Hadoop
6. Format the Hadoop file system
7. Start Hadoop

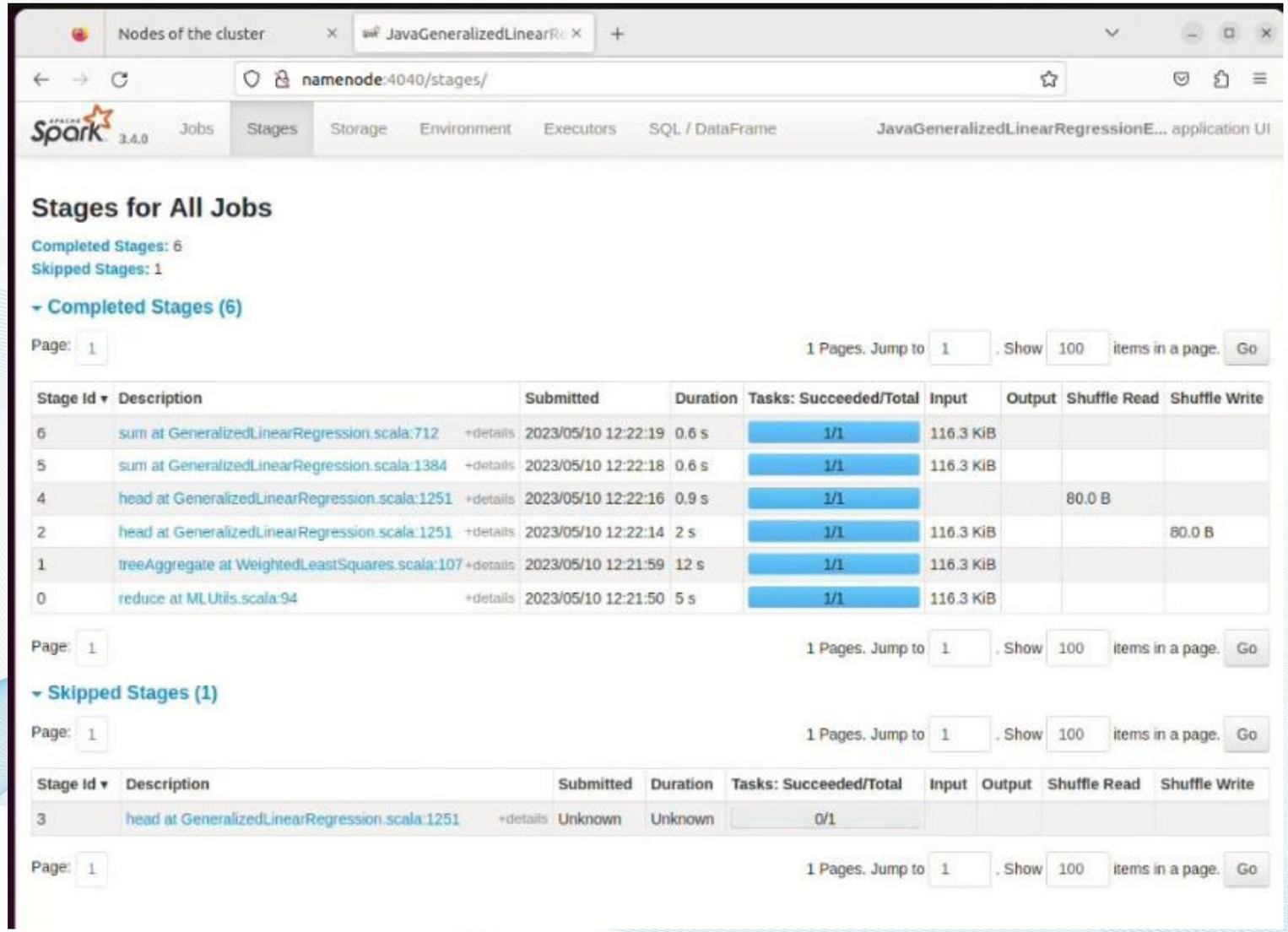
Spark

1. Ensure that your system meets the minimum requirements to run Apache Spark.
2. Download the latest version of Apache Spark from the official website.
3. Install Java if it's not already installed.
4. Set up environment variables for Java and Spark.
5. Configure Apache Spark by unzipping the downloaded file and adjusting it to your system specifications.
6. Start Apache Spark using the command line or launch it via a notebook interface.

Spark Algorithm : Generalized Linear Regression

```java
/**
 * An example demonstrating generalized linear regression.
 * Run with
 * <pre>
 * bin/run-example ml.JavaGeneralizedLinearRegressionExample
 * </pre>
 */

public class JavaGeneralizedLinearRegressionExample {

  public static void main(String[] args) {
    SparkSession spark = SparkSession
      .builder()
      .appName("JavaGeneralizedLinearRegressionExample")
      .getOrCreate();

    // $example on$
    // Load training data
    Dataset<Row> dataset = spark.read().format("libsvm")
      .load("hdfs://namenode:9000/example_algo/sample_linear_regression_data.txt");

    GeneralizedLinearRegression glr = new GeneralizedLinearRegression()
      .setFamily("gaussian")
      .setLink("identity")
      .setMaxIter(10)
      .setRegParam(0.3);

    // Fit the model
    GeneralizedLinearRegressionModel model = glr.fit(dataset);

    // Print the coefficients and intercept for generalized linear regression model
    System.out.println("Coefficients: " + model.coefficients());
    System.out.println("Intercept: " + model.intercept());

    // Summarize the model over the training set and print out some metrics
    GeneralizedLinearRegressionTrainingSummary summary = model.summary();
    System.out.println("Coefficient Standard Errors: "
      + Arrays.toString(summary.coefficientStandardErrors()));
    System.out.println("T Values: " + Arrays.toString(summary.tValues()));
    System.out.println("P Values: " + Arrays.toString(summary.pValues()));
    System.out.println("Dispersion: " + summary.dispersion());
    System.out.println("Null Deviance: " + summary.nullDeviance());
    System.out.println("Residual Degree Of Freedom Null: " + summary.residualDegreeOfFreedomNull());
    System.out.println("Deviance: " + summary.deviance());
    System.out.println("Residual Degree Of Freedom: " + summary.residualDegreeOfFreedom());
    System.out.println("AIC: " + summary.aic());
    System.out.println("Deviance Residuals: ");
    summary.residuals().show();
    // $example off$

    spark.stop();
```

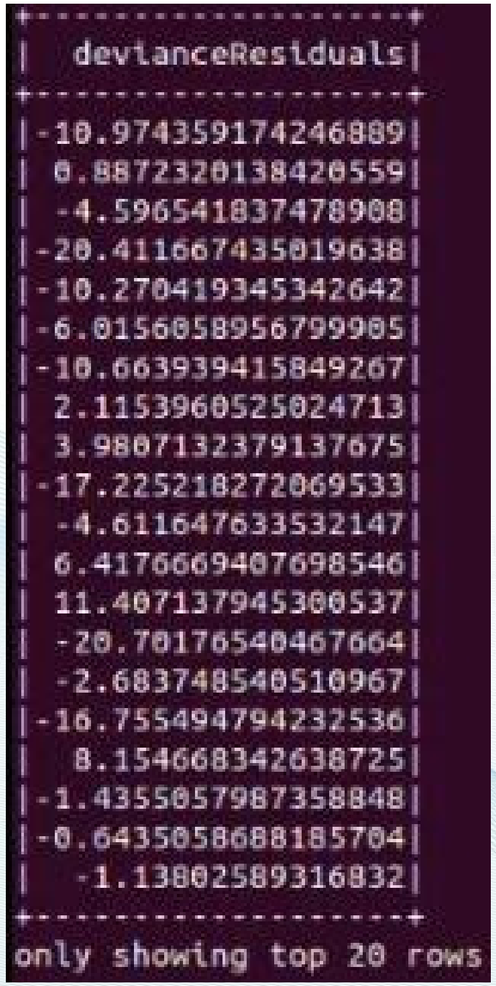
Algorithm Execution :



Benefits of HDFS cluster for Generalized Linear Regression

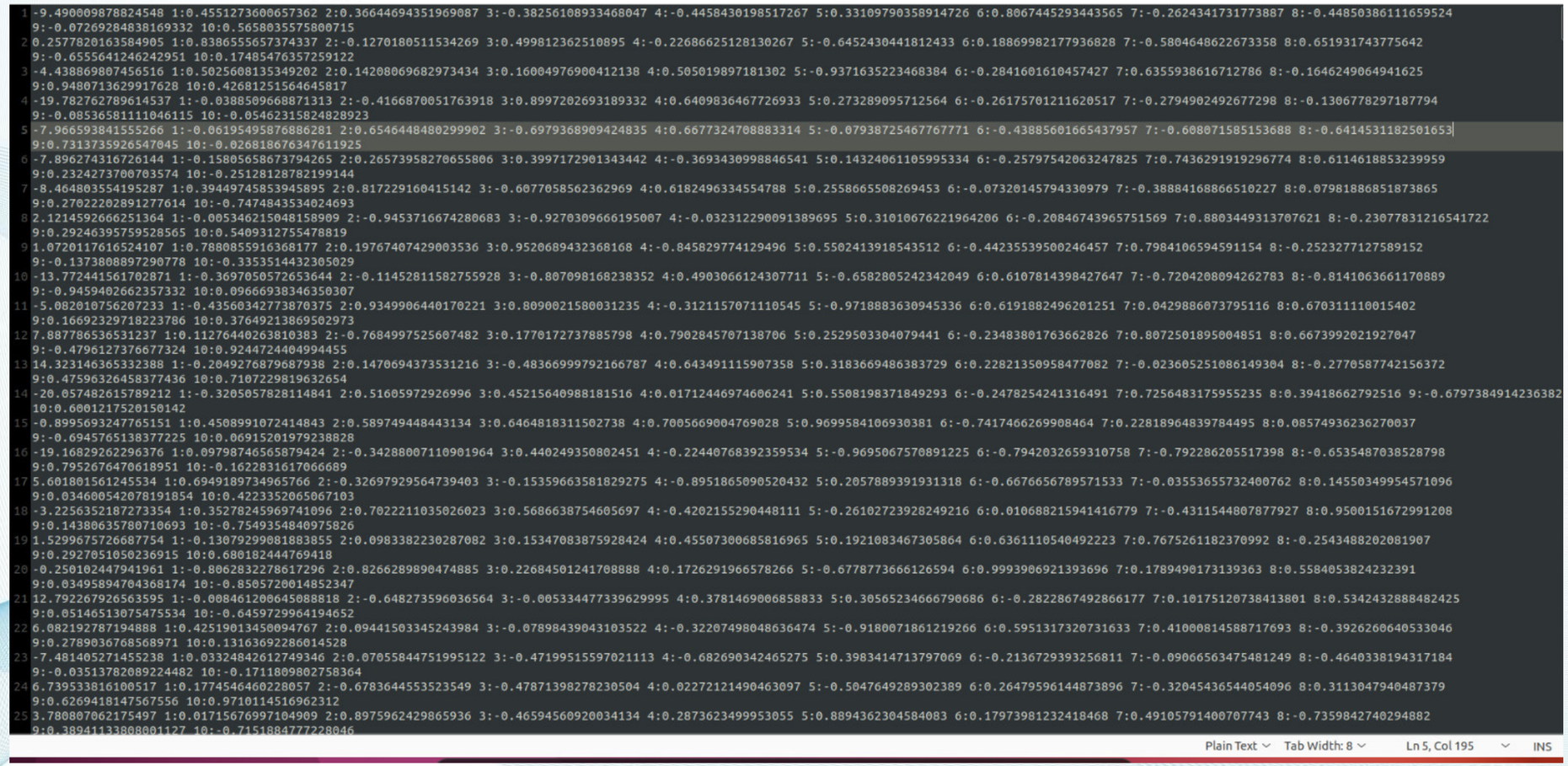
- Capability to handle large data volumes, which is critical for GLM algorithms that require substantial data to train and develop models accurately. HDFS clusters store and process large datasets across many nodes in a distributed manner, making them ideal for large-scale machine learning.
- Fault tolerance and high availability features : HDFS stores data redundantly across numerous nodes, ensuring data remains accessible even if some nodes fail.
- Scalable storage solution for GLM, which enables organizations to add more nodes as their data requirements expand. A scalable storage solution allows organizations to store and analyze vast amounts of data efficiently, which is crucial for machine learning that demands training models using large datasets.

Output Achieved



```
+--------------------+
|   devianceResiduals|
+--------------------+
|-10.974359174246889|
|  0.8872320138420559|
|  -4.5965418374789 08|
|-20.411667435019638|
|-10.270419345342642|
|-6.0156058956799905|
|-10.663939415849267|
|  2.1153960525024713|
|  3.9807132379137675|
|-17.225218272069533|
|  -4.611647633532147|
|  6.4176669407698546|
|  11.407137945300537|
|  -20.701765404 67664|
|  -2.683748540510967|
|-16.755494794232536|
|   8.154668342638725|
|-1.4355057987358848|
|-0.6435058688185704|
|   -1.13802589316832|
+--------------------+
only showing top 20 rows
```

Generalized Linear Regression on Sample_Linear_Regression_data:



Thank You

SLIDE PRESENTATIONS DESIGN