

MODULE 5:

APACHE SPARK

5.1 SPARK

Apache Spark is a fast and general compute engine. Apache Spark powers the analytics applications up to 100 times faster. It supports HDFS compatible data. Spark has a simple and expressive programming model. Expressive program model implements a number of mathematical and logic operations with smaller and easier written codes which a compiler as well as programmer can understand easily. The model, therefore, gives programming ease for a wide range of applications. Applications of expressive codes are in analytics, Extract Transform Load (ETL), Machine Learning (ML), stream processing and graph computations. Spark runs on both Windows and UNIX-like systems, such as Linux and Mac OS. Java is essential for running Spark applications. Executing a spark application on a computer system therefore requires setting a JDK path using JAVA_HOME environment variable or system variable.

5.1.1 Introduction to Big Data Tool-Spark

Figure below shows the main components in the Apache software Foundation's Spark framework, which includes data storage, API's and resources management bonded with functions in Spark core.

Main components of Spark Architecture are:

1. Spark HDFS file system for data storage: Storage is at an HDFS or Hadoop compatible data source such as HDFS, HBase, Cassandra, Ceph or at the objectstore S3.
2. Spark standard API enables the creation of applications using Scala, Java, Python and R.
3. Spark resource management can be a stand-alone server or it can be on a distributed computing Framework, such as YARN or Mesos. Ceph refers to an open source, scalable object, block, file storing unified system. Ceph provides for dynamic replication and redistribution. Ceph provides HDFS compatible mechanisms for Big data in petabytes or exabytes. An application uniquely accesses the object, block and file stores in a system using Ceph. Ceph is highly reliable. Apache Mesos is an open source project developed at University of Berkeley, and now passed to Apache. It manages computing clusters. Its implementation is in C++. Apache released a stable version 1.3.0 of Mesos in 2017. Apache Mesos enables fine-grained sharing of CPU, RAM, IOs and other frameworks. Mesos offers them resource. Each resource contains a list of agents. Each agent has the Hadoop and Map

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

reduce executors. S3 (Simple Storage Service) refers to an Amazon web service on the cloud named S3 which provides Object stores for data.

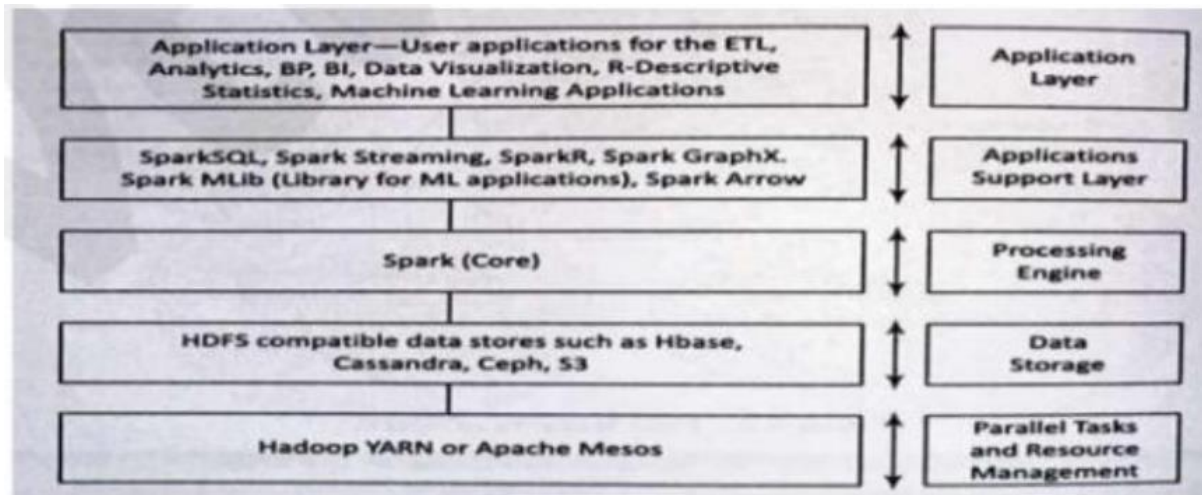
5.1.1.1 Features of Spark

1. Spark provisions for creating applications that use the complex data. In-memory Apache Spark computing engine enables up to 100 times performance with respect to Hadoop.
2. Execution engine uses both in-memory and on-disk computing. Intermediate results save in-memory and spill over to disk.
3. Data uploading from an Object Store for immediate use as a Spark object instance. Spark service interface sets up the Object Store.
4. Provides high performance when an application accesses memory cache from the disk.
5. Contains API to define Resilient Distributed Datasets (RDDs). RDD is a programming abstraction. RDD is the core concept in Spark framework. RDD represents a collection of Object Stores distributed across many compute nodes for parallel processing. Spark stores data in RDD on different partitions. A table has partitions into columns or rows. Similarly, an RDD can also be considered as a table in a database that can hold any type of data. RDDs are also fault tolerant.
6. Processes any kind of data, namely structured, semi-structured or unstructured data arriving from various sources.
7. Supports many new functions in addition to Map and Reduce functions.
8. Optimizes data processing performance by slowing the evaluation of Big Data queries.
9. Provides concise and consistent APIs in Scala, Java and Python. Spark codes are in Scala and run on JVM environment.
10. Provides powerful tool to analyze data interactively using shell which is available in either Scala (which runs on the Java VM and is thus a good way to use existing Java libraries) or Python. The tool also provides for learning the usages of API.

5.1.1.2 Spark Software Stack

Figure below shows a five-layer architecture for running applications when using Spark stack.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML



The main components of Spark stack are SQL, Streaming, R, GraphX, MLlib and Arrow at the applications support layer. Spark core is the processing engine. Data Store provides the data to the processing engine. Hadoop, YARN or Mesos facilitates the parallel running of the tasks and the management and scheduling of the resources.

Spark Stack

Spark stack imbibes generality to Spark. Grouping of the following forms Spark stack: Spark SQL for the structured data. The SQL runs the queries on Spark data in the traditional business HQL queries also run in Spark SQL. Runs UDFS for inline SQL, distributed DataFrames, Parquet, Hive and analytics and visualization applications. Spark SQL enables Spark datasets to use JDBC or ODBC APL Cassandra Data Stores. Spark Streaming is for processing real-time streaming data. Processing is based on micro-batches style of computing and processing. Streaming uses the DStream which is basically a series of RDDs, to process the real-time data. SparkR is an R package used as light-weight front end for Apache Spark from R. Spark API uses SparkR through the RDD class. A user can interactively run the jobs from the R shell on a cluster. An RDD API is in the distributed lists in R. Spark MLlib is Spark's scalable machine learning library. It consists of common learning algorithms and utilities. MLlib includes classification, regression clustering, collaborative filtering, dimensionality reduction and optimization primitives. MLlib applies in recommendation systems, clustering and classification using Spark. Spark GraphX is an API for graphs. GraphX extends the Spark RDD by introducing the Resilient Distributed Property. GraphX computations use fundamental operators (e.g., subgraph, join Vertices and aggregateMessages). GraphX uses a collection of graph algorithms for programming. Graph analytics tasks are created with ease using GraphX. Spark Arrow for columnar in-memory analytics and enabling usages of vectorised UDFS (VUDFS). The Arrow enables high performance Python UDFs for SerDe and data pipelines.

Characteristics of Spark

- Open Source
- Distributed Computing Framework
- Fault Tolerant
- High Speed
- Scalable
- In-memory Processing
- Multi-language Support

Components of Spark Architecture

1. Data Storage Layer

Stores data in:

- Hadoop HDFS
- HBase
- Cassandra
- Ceph
- Amazon S3

Spark can directly read data from these storage systems.

2. Spark APIs

Spark provides APIs for:

- Scala
- Java
- Python (PySpark)
- R

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Developers can build Spark applications using any of these languages.

3. Resource Management

Spark manages computing resources using:

- Standalone Cluster Manager
- Hadoop YARN
- Apache Mesos

These allocate CPU, memory, and storage resources efficiently.

4. Spark Core

Spark Core is the heart of the Spark framework.

Functions include:

- Memory Management
- Task Scheduling
- Fault Recovery
- Job Execution
- Resource Allocation
- RDD Management

Spark Execution Flow

User Program



Spark Driver

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML



Spark Core



Cluster Manager



Worker Nodes



RDD Processing



Results

Features of Apache Spark

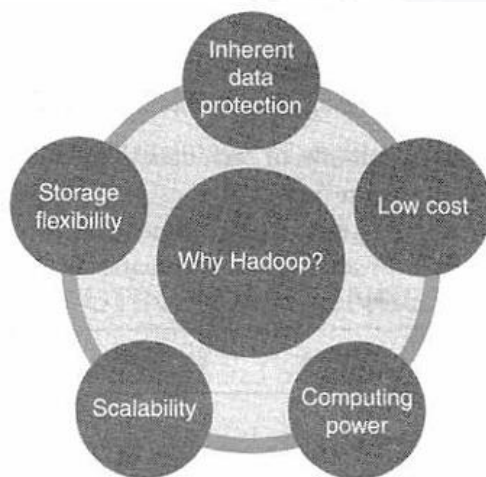


Figure 5.2 Key considerations of Hadoop.

1. In-Memory Computing

Intermediate results are stored in RAM.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Benefits:

- Faster execution
- Reduced disk access
- Better performance

2. High Performance

Spark executes applications much faster than Hadoop because of in-memory processing.

3. Fault Tolerance

RDDs automatically recover lost data using lineage information.

4. Resilient Distributed Dataset (RDD)

RDD is Spark's fundamental data abstraction.

Characteristics:

- Immutable
- Distributed
- Fault tolerant
- Parallel processing

5. Multi-language Support

Spark supports:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Scala
- Java
- Python
- R
- SQL

6. Supports Multiple Data Types

Processes:

- Structured Data
- Semi-Structured Data
- Unstructured Data

7. Rich APIs

Provides APIs for:

- SQL
- Machine Learning
- Streaming
- Graph Analytics

8. Interactive Processing

Supports interactive shell programming.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

9. Scalability

Spark easily scales from a single computer to thousands of nodes.

10. Ease of Programming

Provides concise and expressive APIs, reducing code complexity.

Advantages of Spark

- Extremely fast
 - Easy programming model
 - High fault tolerance
 - Supports machine learning
 - Supports graph analytics
 - Supports streaming analytics
 - Distributed processing
 - Efficient memory utilization
-

Limitations of Spark

- High memory requirement
 - Not suitable for very small datasets
 - Streaming uses micro-batches (in classic Spark Streaming)
 - Requires cluster resource management
-

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Spark Software Stack

Spark provides a layered software architecture.

Applications



Spark SQL

Spark Streaming

MLlib

GraphX

SparkR



Spark Core



Data Storage (HDFS, Cassandra,
Hive, S3, Parquet)



Cluster Manager
(YARN / Mesos / Standalone)

(Based on the Spark stack described in the module.)

Components of Spark Stack

Spark SQL

Used for:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Structured data processing
- SQL queries
- DataFrames
- JDBC/ODBC connectivity

Spark Streaming

Processes real-time streaming data using **micro-batches**.

SparkR

Allows R programmers to execute Spark programs.

MLlib

Spark's machine learning library.

Supports:

- Classification
- Regression
- Clustering
- Recommendation Systems
- Dimensionality Reduction

GraphX

Used for:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Graph Processing
- Social Network Analysis
- Graph Algorithms
- PageRank

Spark Core

Provides:

- Scheduling
- Memory Management
- Fault Recovery
- Resource Allocation

Spark vs Hadoop MapReduce

Spark

Hadoop MapReduce

In-memory processing Disk-based processing

Very fast Comparatively slow

Supports streaming Batch processing only

Supports SQL No native SQL

Supports MLlib External ML libraries

Easy APIs More programming effort

Interactive processing Non-interactive

Applications of Spark

- Business Intelligence
- Banking Analytics
- Healthcare Analytics
- Fraud Detection
- Recommendation Systems
- IoT Analytics
- Social Media Analytics
- Machine Learning
- ETL Pipelines
- Data Warehousing

5.2 Spark SQL

Introduction

Spark SQL is a Spark module used for processing **structured and semi-structured data**. It allows users to query data using **SQL statements** and provides support for **DataFrames** and **Datasets**. Spark SQL integrates easily with Hive and various data sources.

Definition

Spark SQL is a component of Apache Spark that enables users to process structured data using SQL queries, DataFrames, and integrated data sources.

Features of Spark SQL

- SQL query support

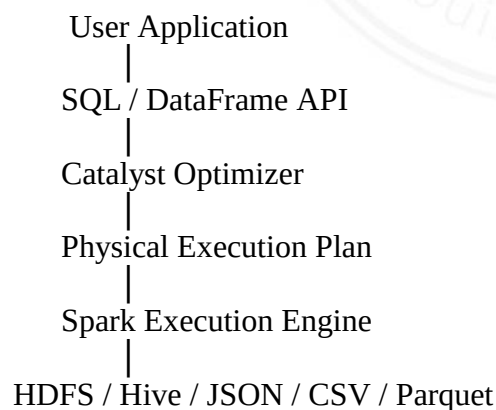
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- DataFrame API
- Hive integration
- Optimized query execution
- Supports multiple data sources
- High-speed in-memory processing
- Automatic optimization using Catalyst Optimizer

Advantages

- Familiar SQL syntax
- Faster query execution
- Easy integration with Hive
- Supports large datasets
- Suitable for Business Intelligence applications

Spark SQL Architecture



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Components

1. SQL Interface

Allows execution of SQL queries.

Example

```
SELECT * FROM Employee;
```

2. Catalyst Optimizer

Optimizes SQL queries automatically by:

- Logical optimization
- Physical optimization
- Query plan generation

This reduces execution time.

3. Execution Engine

Executes optimized query plans using Spark Core.

4. Data Sources

Spark SQL can read data from:

- Hive
- HDFS
- JSON
- CSV
- Parquet

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- ORC
- JDBC databases

DataFrames

Introduction

A **DataFrame** is Spark's primary data abstraction for structured data. It organizes data into **rows and named columns**, similar to a relational database table. Spark SQL uses DataFrames extensively for efficient analytics.

Definition

A **DataFrame** is a distributed collection of structured data organized into named columns.

Characteristics

- Distributed
- Immutable
- Fault tolerant
- Optimized
- Schema-based
- Supports SQL queries

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Advantages

- Easy to use
- High performance
- Memory efficient
- Automatic optimization
- Easy integration with Spark SQL

Creating a DataFrame

DataFrames can be created from:

- CSV files
- JSON files
- Parquet files
- Hive tables
- Existing RDDs
- Python collections

Example

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("Student").getOrCreate()

data = [
    ("Rahul",85),
    ("Anita",90),
    ("Priya",88)
]

df = spark.createDataFrame(data,["Name","Marks"])

df.show()
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Displaying Data

Show all records

```
df.show()
```

Display Schema

```
df.printSchema()
```

Display Column Names

```
df.columns
```

Display Data Types

```
df.dtypes
```

Common DataFrame Operations

Selecting Columns

```
df.select("Name").show()
```

Filtering Records

```
df.filter(df.Marks>80).show()
```

Sorting Data

```
df.orderBy("Marks").show()
```

Descending order

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

```
df.orderBy(df.Marks.desc()).show()
```

Group By

```
df.groupBy("Department").count()
```

Aggregate Functions

```
df.groupBy("Department").avg("Salary")
```

Distinct Values

```
df.select("Department").distinct()
```

SQLContext

Definition

SQLContext is a Spark interface used to execute SQL queries on structured data.

It enables:

- DataFrame creation
 - SQL query execution
 - Reading structured files
-

Example

```
from pyspark.sql import SQLContext
```

```
sqlContext=SQLContext(sc)
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

HiveContext

Definition

HiveContext extends SQLContext and provides additional support for **Hive databases, Hive tables, and HiveQL queries**.

It enables Spark applications to access Hive metadata directly.

Example

```
from pyspark.sql import HiveContext  
  
hiveContext=HiveContext(sc)
```

SQLContext vs HiveContext

SQLContext

Executes SQL queries

No Hive metadata

Basic SQL support

Reads structured files

HiveContext

Executes HiveQL queries

Supports Hive metadata

Full Hive support

Reads Hive tables directly

Reading Files using Spark SQL

Read CSV

```
df=spark.read.csv("student.csv",header=True)
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Read JSON

```
df=spark.read.json("student.json")
```

Read Parquet

```
df=spark.read.parquet("student.parquet")
```

Creating Temporary View

A DataFrame can be converted into a temporary SQL table.

```
df.createOrReplaceTempView("Student")
```

Now SQL queries can be executed.

```
spark.sql("SELECT * FROM Student").show()
```

Spark SQL Functions

Common functions include:

- select()
 - where()
 - filter()
 - groupBy()
 - orderBy()
 - count()
 - sum()
 - avg()
 - max()
 - min()
-

Data Analysis Using Spark

Spark provides fast analytical operations such as:

- Data filtering
- Sorting
- Aggregation
- Statistical analysis
- Grouping
- Joins
- Data summarization

The textbook also lists common analysis tasks such as filtering, top-N lists, subtotals, unique lists, duplicate detection, sorting, and database functions.

Common Data Analysis Operations

Filtering

```
df.filter(df.Marks>80)
```

Sorting

```
df.orderBy("Marks")
```

Counting

```
df.count()
```

Maximum

```
df.agg({"Marks":"max"})
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Average

```
df.agg({"Marks":"avg"})
```

Joining DataFrames

```
student.join(department,  
student.DeptID==department.DeptID)
```

Spark SQL Workflow

Data Source



DataFrame



SQL Query



Catalyst Optimizer



Execution Engine



Results

Applications of Spark SQL

- Business Intelligence
- Data Warehousing
- Banking Analytics

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Healthcare Analytics
- Customer Analysis
- Sales Reporting
- ETL Pipelines
- Financial Analytics

Advantages of Spark SQL

- High-speed execution
- SQL support
- Optimized query execution
- Easy integration with Hive
- Supports multiple file formats
- Easy DataFrame programming
- Distributed processing

Limitations

- Memory intensive
- Requires Spark cluster
- Not ideal for small datasets
- Schema management required

5.3 PySpark

Introduction

PySpark is the Python API for Apache Spark. It enables developers to write Spark applications using the Python programming language while utilizing Spark's distributed computing capabilities. PySpark is widely used in Big Data Analytics because of Python's simplicity and extensive data science libraries.

Definition

PySpark is the Python programming interface of Apache Spark that enables distributed processing of large datasets.

Features of PySpark

- Easy to learn
- Open source
- Supports distributed computing
- Fault tolerant
- High-speed processing
- Compatible with Python libraries
- Supports SQL, Machine Learning and Streaming

Advantages

- Simple Python syntax
- High performance
- Supports Big Data Analytics

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Easy integration with Hadoop
- Extensive Machine Learning support

PySpark Architecture

Python Program

|

PySpark API

|

Spark Driver

|

Spark Executors

|

Hadoop / HDFS / Cloud

|

Results

SparkSession

SparkSession is the entry point for every PySpark application.

Example

```
from pyspark.sql import SparkSession
```

```
spark = SparkSession.builder \  
    .appName("Student") \  
    .getOrCreate()
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Reading Data

Read CSV

```
df = spark.read.csv(  
    "student.csv",  
    header=True)
```

Read JSON

```
df = spark.read.json(  
    "student.json")
```

Display Data

```
df.show()
```

NumPy

Introduction

NumPy (Numerical Python) is a Python library used for scientific computing and numerical operations. It provides fast multidimensional arrays and mathematical functions.

Definition

NumPy is a numerical computing library that supports large multidimensional arrays and mathematical operations.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Features

- Multi-dimensional arrays
- Mathematical operations
- Linear algebra
- Matrix operations
- Random number generation
- High computational speed

Example

```
import numpy as np  
  
a = np.array([10,20,30])  
  
print(a)
```

Applications

- Scientific Computing
- Machine Learning
- Data Analytics
- Matrix Computations
- Image Processing

SciPy

Introduction

SciPy extends NumPy by providing advanced scientific and mathematical algorithms.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Definition

SciPy is an open-source Python library for scientific and technical computing.

Features

- Optimization
 - Statistics
 - Signal Processing
 - Numerical Integration
 - Linear Algebra
 - Scientific Computing
-

Example

```
from scipy import stats  
  
stats.mean([10,20,30])
```

Applications

- Engineering
 - Statistics
 - Research
 - Scientific Analysis
 - Machine Learning
-

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Pandas

Introduction

Pandas is the most widely used Python library for data manipulation and analysis. It provides two important data structures:

- Series
- DataFrame

The textbook highlights Pandas as an important component for Spark-based analytics and for implementing vectorized UDFs.

Definition

Pandas is a Python library that provides high-performance data manipulation and analysis tools.

Features

- Data Cleaning
- Data Transformation
- Missing Value Handling
- Statistical Analysis
- Import/Export Files
- Data Visualization Support

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Pandas Data Structures

Series

One-dimensional data structure.

Example

```
import pandas as pd
```

```
s = pd.Series(  
[10,20,30])
```

```
print(s)
```

DataFrame

Two-dimensional table.

Example

```
import pandas as pd
```

```
df = pd.DataFrame(  
{  
'Name':['Rahul','Anita'],  
'Marks':[85,90]  
})
```

```
print(df)
```

Advantages of Pandas

- Easy data manipulation
- High performance
- Easy filtering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Statistical analysis
- Supports CSV, Excel, JSON

User Defined Functions (UDF)

Introduction

Spark provides several built-in functions. When custom operations are required, programmers create **User Defined Functions (UDFs)**.

Definition

A User Defined Function (UDF) is a custom function created by the programmer to perform application-specific operations on Spark DataFrames.

Features

- Custom logic
- Reusable
- DataFrame support
- SQL integration
- Python-based implementation

General Syntax

```
from pyspark.sql.functions import udf
```

```
udf_function = udf(  
    function_name,
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

returnType)

Example

```
from pyspark.sql.functions import udf
```

```
def bonus(x):
```

```
    return x + 1000
```

Applications

- Salary computation
- Grade calculation
- Data transformation
- Custom business rules

Vectorized UDF (VUDF)

Introduction

Normal UDFs process one record at a time.

Vectorized UDFs (VUDFs) process an entire **Pandas Series** simultaneously, making execution much faster.

Spark implements Vectorized UDFs using **Apache Arrow** for efficient in-memory columnar data transfer between Spark and Python.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Definition

A Vectorized UDF is a User Defined Function that processes an entire Pandas Series instead of one row at a time.

Working of VUDF

Spark DataFrame

|

Apache Arrow

|

Pandas Series

|

Vectorized UDF

|

Processed Series

|

Spark DataFrame

Advantages

- Faster execution
- Reduced serialization overhead
- Efficient memory utilization
- Better Python performance
- Suitable for large datasets

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Example

The textbook demonstrates a vectorized UDF that increases puzzle cost values using **pandas_udf**.

Simplified example:

```
from pyspark.sql.functions import pandas_udf

@pandas_udf("float")
def addTenPercent(v):
    return v * 1.10
```

UDF vs Vectorized UDF

UDF

Processes one row at a time

Slower

Higher serialization overhead

Python function

Suitable for small datasets

Vectorized UDF

Processes an entire Pandas Series

Faster

Lower serialization overhead

Pandas function

Suitable for Big Data

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Grouped Vectorized UDF (GVUDF)

Introduction

Grouped Vectorized UDF extends Vectorized UDF by applying operations **group-wise** using the **split-apply-combine** strategy of Pandas.

The textbook explains that Spark first splits the DataFrame into groups, applies the function to each group, and finally combines the results into a new Spark DataFrame.

Definition

A Grouped Vectorized UDF is a Pandas UDF that operates on grouped DataFrames and returns processed grouped results.

Working

Spark DataFrame

|

Group By

|

Split Data

|

Apply Pandas Function

|

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Combine Results

|

Output DataFrame

Steps

1. Split the DataFrame into groups.
2. Apply a vectorized function to each group.
3. Combine all processed groups.
4. Return the final DataFrame.

Advantages

- Fast grouped computation
- Efficient aggregation
- Better parallel execution
- Reduced execution time

Applications

- Department-wise salary analysis
- Product-wise sales analysis
- Customer analytics
- Banking reports
- Business Intelligence dashboards

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Comparison of UDF Types

Feature	UDF	Vectorized UDF	Grouped UDF	Vectorized
Input	Single Row	Pandas Series	Pandas (Group)	DataFrame
Output	Single Value	Pandas Series	Pandas DataFrame	
Speed	Slow	Fast	Fastest for grouped operations	
Uses Apache Arrow	No	Yes	Yes	
Best Use	Custom operations	row Column operations	Group analytics	

Applications of PySpark Ecosystem

- Data Cleaning
- ETL Processing
- Financial Analytics
- Machine Learning
- Recommendation Systems
- Healthcare Analytics
- Business Intelligence
- Fraud Detection
- Customer Behaviour Analysis

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

BUSINESS INTELLIGENCE SYSTEMS (BIS701)

Module 5 – Part 4 Notes

Data Analysis Operations, Outlier Removal, RDD
Programming, MLlib & VTU Exam Preparation
(VTU Semester Exam-Oriented)

5.4 Data Analysis Operations in Spark

Introduction

Apache Spark provides powerful analytical operations to process, transform, summarize, and visualize large datasets efficiently. Spark performs these operations in parallel on distributed systems, making it suitable for Business Intelligence and Big Data Analytics.

The textbook lists several commonly used analysis operations such as filtering, sorting, duplicate removal, unique value counting, subtotals, and database functions.

Common Data Analysis Operations

1. Filtering Data

Filtering retrieves only those records that satisfy a specified condition.

Example:

```
df.filter(df.Salary > 50000)
```

Applications:

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- High salary employees
- Students scoring above distinction
- Premium customers

2. Multiple Column Filtering

Example

```
df.filter(  
(df.Department=="CSE") &  
(df.Salary>60000)  
)
```

3. Sorting

Sorting arranges records in ascending or descending order.

```
df.orderBy("Salary")
```

Descending

```
df.orderBy(df.Salary.desc())
```

4. Creating Top-N Lists

Used to display the highest or lowest values.

Example

Top 10 students

```
df.orderBy(df.Marks.desc()).limit(10)
```

5. Grouping Data

Groups records having common values.

Example

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

```
df.groupBy("Department").count()
```

6. Aggregate Functions

Common functions include

- COUNT()
- SUM()
- AVG()
- MIN()
- MAX()

Example

```
df.groupBy("Department").avg("Salary")
```

7. Removing Duplicate Records

```
df.dropDuplicates()
```

8. Counting Unique Values

```
df.select("Department").distinct()
```

9. Database Functions

Spark supports operations similar to:

- SUM
- MAX
- MIN
- COUNT
- AVG

for analytical reporting.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Data Analysis Workflow

Raw Data



Cleaning



Filtering



Grouping



Aggregation



Visualization



Decision Making

Removing Outliers

Introduction

An **outlier** is a data value that differs significantly from other observations. Outliers may occur because of:

- Data entry mistakes
- Programming errors
- Sensor faults
- Exceptional events

Outliers can distort averages and analytical results, so they should be detected and handled appropriately.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Definition

An outlier is an observation whose value lies far away from the majority of the dataset.

Causes of Outliers

- Human errors
- Typing mistakes
- Equipment malfunction
- Experimental errors
- Genuine rare events

Steps for Removing Outliers

1. Convert values to numeric form.
2. Calculate the **mean**.
3. Calculate the **standard deviation**.
4. Compute the distance of each value from the mean.
5. Remove values beyond a chosen threshold (commonly $3 \times \text{standard deviation}$).

Outlier Detection Flow

Dataset



Calculate Mean



Calculate Standard Deviation



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Find Distance

↓

Is Distance $> 3 \times SD$?

↓

Yes → Outlier

↓

No → Valid Data

Advantages of Removing Outliers

- Improves data quality
- Better prediction accuracy
- Reliable statistical analysis
- Improved Machine Learning performance

Resilient Distributed Dataset (RDD)

Introduction

RDD (Resilient Distributed Dataset) is Spark's **fundamental data structure**.

It stores data across multiple cluster nodes while supporting parallel processing.

Definition

RDD is an immutable distributed collection of objects processed in parallel.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Characteristics

- Immutable
- Distributed
- Fault tolerant
- Parallel
- Lazy evaluation

Creating RDD

```
rdd = sc.parallelize(  
[10,20,30,40]  
)
```

Reading Text File

```
rdd = sc.textFile(  
"student.txt"  
)
```

Common RDD Operations

Map

```
rdd.map(lambda x:x*2)
```

Filter

```
rdd.filter(lambda x:x>20)
```

Reduce

```
rdd.reduce(lambda x,y:x+y)
```

Collect

```
rdd.collect()
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Count

rdd.count()

First

rdd.first()

RDD Execution Flow

Input Data



RDD Creation



Transformation



Action



Output

MLlib

Introduction

MLlib is Apache Spark's **Machine Learning Library**.

It provides scalable algorithms for predictive analytics and intelligent applications.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Definition

MLlib is Spark's distributed machine learning framework for implementing algorithms on large datasets.

MLlib Components

- Classification
- Regression
- Clustering
- Recommendation
- Dimensionality Reduction
- Collaborative Filtering

Machine Learning Algorithms

Classification

Examples

- Decision Tree
- Logistic Regression
- Naïve Bayes

Regression

Predicts continuous values.

Example

House price prediction

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Clustering

Groups similar objects.

Example

K-Means

Recommendation

Suggests products.

Example

Amazon Recommendation System

Dimensionality Reduction

Algorithms

- PCA
 - SVD
-

Applications of MLlib

- Fraud Detection
- Recommendation Systems
- Medical Diagnosis
- Credit Risk Analysis
- Customer Segmentation
- Image Classification
- Social Media Analytics
- Banking Analytics

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Advantages of MLlib

- Distributed processing
- High speed
- Fault tolerant
- Easy integration with Spark
- Large algorithm library

Spark Programming Workflow

Create Spark Session



Load Dataset



Create RDD/DataFrame



Transform Data



Perform Analysis



Machine Learning



Generate Results

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Apache Spark Applications

- Business Intelligence
- Healthcare
- Banking
- Finance
- Social Networks
- IoT
- Manufacturing
- E-Commerce
- Scientific Computing
- Government Analytics

Spark Advantages

- Fast processing
- Scalable
- In-memory computation
- Supports SQL
- Supports Machine Learning
- Streaming support
- Fault tolerant

Spark Limitations

- High memory usage
- Complex cluster configuration
- Less suitable for very small datasets
- Performance depends on cluster resources