Class Note:

Subject : .....BDT.....

Faculty : ....V. Divya.....

Topic: ...Apache Spark:

Unit No. : 4
Lecture No.: 1
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 1

Apache Spark:

- Apache Spark is a fast and general compute engine. and powers the analytics applications upto 100 times faster.
- It supports HDFS compatible data and Spark has a simple and expressive programming model.
- Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming.

Evolution of Apache Spark:

- Spark is one of Hadoop's sub project developed in 2009 in UC Berkley's AMPLab by Matei Zaharia. It was Open Sourced in 2010 under a BSD license.
- It was ~~founded~~ donated to Apache software foundation in 2013, and now Apache Spark has become a top level Apache project from Feb-2014.

Features of Apache Spark:

→ Speed - Spark helps to run an application in Hadoop cluster, upto 100 times faster in memory and 10 times

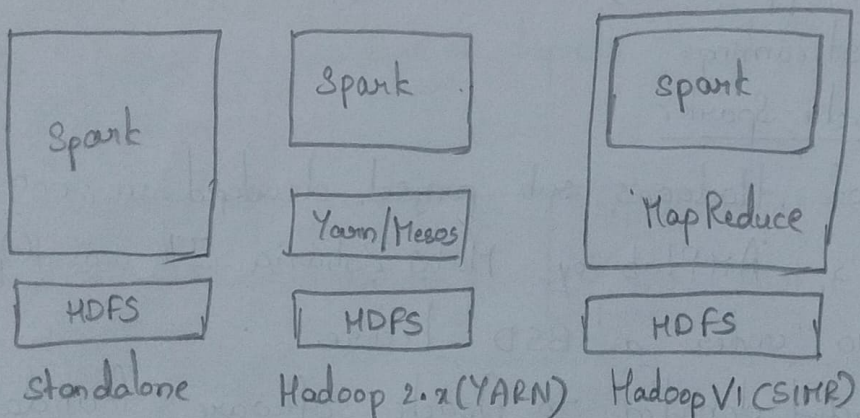
faster when running on disk. This is possible by reducing no. of read/write operations to disk. It stores the intermediate processing data in memory.

→ Supports multiple languages: Spark provides built in APIs in Java, Scala or Python. Spark comes up with 80 high level operators for interactive querying.

→ Advanced Analytics - Spark not only supports 'Map' and 'reduce'. It also supports SQL queries, Streaming data, Machine Learning and Graph algorithms.

Spark built on Hadoop

- Following diagram shows three ways of how Spark can be built with Hadoop components.



There are three ways of Spark deployment as explained below:

→ Standalone: Spark Standalone deployment means Spark occupies the place on top of HDFS and space is allocated for HDFS, explicitly. Here, Spark and MapReduce will run side by side to cover all spark jobs on cluster.

→ Hadoop Yarn: Hadoop Yarn deployment means, simply, spark runs on Yarn without any preinstallation or root access required.

Class Note:

Subject : ....BDT.....

Faculty : ...V. Divya.....

Topic: .....Apache...Spark.

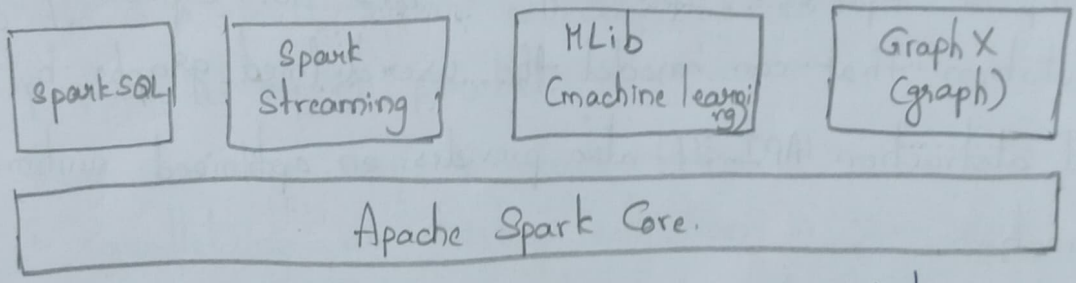
Unit No. : 4
Lecture No.: 1
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:

It helps to integrate Spark into Hadoop ecosystem or Hadoop stack. It allows other components to run on top of Stack.

→ Spark in MapReduce (SIMR) : Spark in MapReduce is used to launch spark job in addition to standalone deployment. With SIMR, user can start Spark and uses its shell without any administrative access.

Components of Spark:

The following illustration depicts the different components of Spark.



→ Apache Spark Core: Spark Core is the underlying general execution engine for Spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.

→ Spark SQL: Spark SQL is a components on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.

→ Spark Streaming: Spark Streaming leverages (use to max advantage) Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini batches and performs RDD (Resilient Distributed Datasets) transformations on those mini batches of data.

→ MLib (Machine Learning Library): MLib is a distributed machine learning framework above Spark because of the distributed memory based Spark architecture. It is, according to benchmarks done by MLib developers against Alternating Least Squares (ALS) implementations. Spark MLib is nine times as fast as Hadoop disk based version of Apache Mahout.

→ GraphX: GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

Class Note:

Subject : ....BDT.....

Faculty : ....V. Divya.....

Topic: .....RDDs.....

Unit No. : 4
Lecture No.: 2
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 3

Resilient Distributed Datasets:

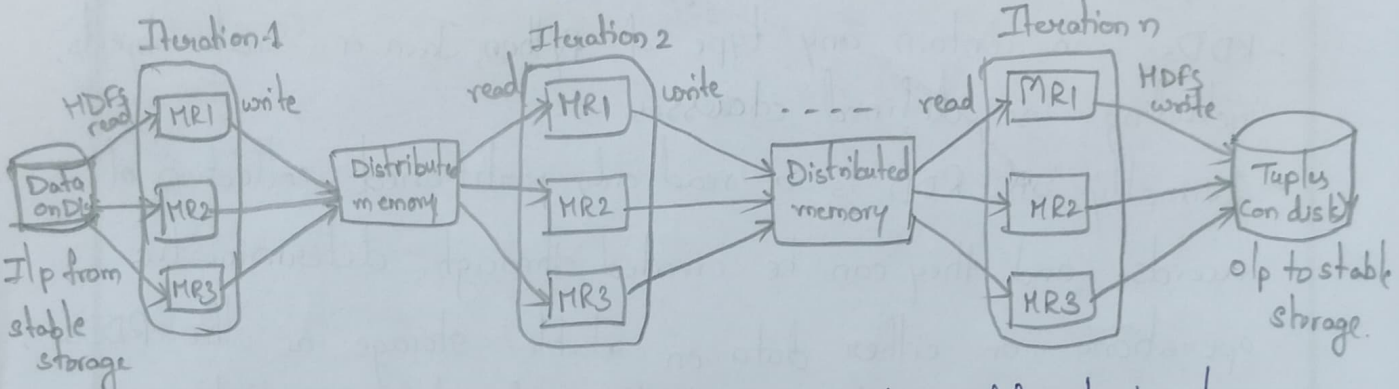
- RDD is a fundamental data structure of spark. which is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of cluster.
- RDDs can contain any type of Python, Java or Scala objects, including user defined classes.
- Formally, an RDD is a read only partitioned collection of records. and they can be created through deterministic operations on either data on stable storage or other RDDs.
- RDD is a fault tolerant collection of elements that can be operated on in parallel.
- There are two ways to create RDDs
  (i) Parallelizing - an existing collection in your driver pgm.
  (ii) Referencing a dataset - in an external storage system,
- Spark makes use of the concept of RDD to achieve faster and efficient MapReduce operations.

Data sharing using Spark RDD:

- Data sharing is slow in MapReduce due to replication, serialization and disk IO.
- The key idea of spark is Resilient Distributed Datasets (RDD), and it supports in-memory processing computation.
- It stores the state of memory as an object across the jobs and the object is sharable between those jobs.

→ Iterative Operations on Spark RDD:

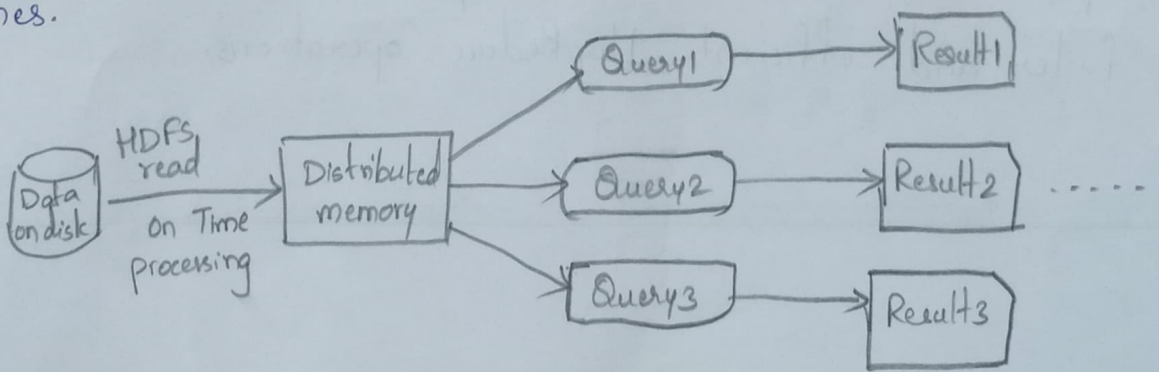
- It will store intermediate results in a distributed memory instead of stable storage (Disk) and make the system faster.



- If the Distributed Memory (RAM) is not sufficient to store intermediate results (State of Job), then it will store those results on disk.

→ Interactive Operations on Spark RDD:

- If different queries are run on the same set of data repeatedly, this particular data can be kept in memory for better execution times.



Class Note:

Subject : .....BDT.....

Faculty : .....V.Divya.....

Topic: Spark Programming.

Unit No. : 4
Lecture No.: 3
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 4

- By default, each transformed RDD may be recomputed each time you ~~can~~ run on action on it.
- You may also persist an RDD in memory, in which case Spark will keep the elements around on the cluster ~~around on the cluster~~ for much faster access, the next time you query it.

Spark Core Programming:

- Spark Core is the base of whole project and it provides distributed task dispatching, scheduling and basic I/o functionalities.
- Spark uses a specialized fundamental data structure known as RDD that is a logical collection of data partitioned across machines.
- RDD can be created in two ways:
  (i) referencing datasets in external storage systems
  (ii) applying transformation on existing RDDs.

→ Spark Shell:

- Spark provides an interactive shell - a powerful tool to analyze data interactively.

- It is available on either Scala or Python language.
- Spark's primary abstraction is a distributed collection of items called a RDD. which can be created from Hadoop Input formats or by transforming other RDDs.

→ Open Spark shell:

Following command is used to open Spark shell.

```
$ spark-shell.
```

→ Create a simple RDD:

create a simple RDD from text file and use following command to create a simple RDD.

```
scala> val inputfile = sc.textFile ("input.txt")
```

The o/p for above command is

```
inputfile: org.apache.spark.rdd.RDD[String] = input.txt MappedRDD[1]
                    at textfile at <console> :12
```

RDD Transformations:

- RDD transformations returns pointer to new RDD and allows you to create dependencies blw RDDs.
- Each RDD in dependency chain has a function for calculating its data and has a pointer to its parent RDD.
- Spark is lazy, so nothing will be executed unless you call some transformation or action that will trigger job creation and execution.
- Given below is a list of RDD transformations:

S.No	Transformations & Meaning.
1.	map(func) - returns a new distributed dataset, formed by passing each element of the source through a func.
2.	filter(func) - returns a new dataset formed by selecting those elements of source on which func returns true.

Class Note:

Subject : BDT

Faculty : V. Divya

Topic: Spark Programming

Unit No. : 4
Lecture No.: 3
Link to Session
Planner (SP) S.No. ..... of SP
Date Conducted:
Page No. 5

3. flatMap (func) - Similar to map, but each input item can be mapped to 0 or more output items.
4. mapPartitions (func) - Similar to map, but runs separately on each partition of RDD, so func must be of type Iterator<T> ⇒ Iterator<U> when running on RDD of type T.
5. mapPartitionsWithIndex (func) - Similar to mapPartitions, but also provides func with an integer value representing the index of the partition, so func must be of type (Int, Iterator<T>) ⇒ Iterator<U> when running on an RDD of type T.
6. sample (withReplacement, fraction, seed) - Sample a fraction of data, with or without replacement, using a given random number generator seed.
7. union (otherDataset) - Returns a new dataset that contains the union of elements in the source dataset and the argument.
8. intersection (otherDataset) - Returns a new RDD that contains the intersection of elements in the source dataset and the argument.

9. distinct([numTasks]) - Returns a new dataset that contains the distinct elements of the source dataset.
10. groupByKey([numTasks]) - When called on a dataset of (K,V) pairs, returns a dataset of (K, Iterable<V>) pairs.

RDD actions:

1. reduce(func) - Aggregate the elements of dataset using a function func. function should be commutative and associative so that it can be computed correctly in ||el.
2. collect() - Returns all the elements of dataset as an array at the driver program.
3. count() - returns the no. of elements in dataset.
4. first() - returns the first element of dataset.
5. take(n) - returns an array with first 'n' elements of dataset.
6. takeSample(withReplacement, num, [seed]) - returns an array with a random sample of num elements of the dataset, with or w/o replacement, optionally pre-specifying a random no. generator seed
7. takeOrdered(n, [ordering]) - returns the first n elements of RDD using either their natural order or a custom comparator.
8. saveAsTextFile(path) - writes the elements of dataset as a text file in a given directory in the local filesystem, HDFS & any other Hadoop supported file system.
9. saveAsObjectSequenceFile(path)(Java and Scala) - writes the elements of dataset in a simple format using Java serialization, which can then be loaded using SparkContext.objectFile().
10. countByKey() - only available on RDDs of type (K,V). Returns a hashmap of (K, Int) pairs with the count of each key.
11. foreach(func) - runs a function func on each element of dataset.

Class Note:

Subject : ....BDT.....

Faculty : ....V.Divya....

Topic: .....Dataframes.

Unit No. : 4
Lecture No.: 4
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 6

Datasets and Dataframes:

- A dataset is a distributed collection of data. It is a new interface added in Spark 1.6 that provides the benefits of RDDs with the benefits of Spark SQL's optimized execution engine.
- A dataframe is a dataset organized into named columns.
- It is conceptually equivalent to a table in a relational database or a data-frame in R|Python, but with richer optimization under the hood.
- Dataframes can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing RDDs.
- The dataframe API is available in Scala, Java, Python and R.
- In Scala, and Java, a dataframe is represented by a dataset of Rows.
- In the Scala API, dataframe is simply a type alias of Dataset[Row].

- While in Java API, users need to use Dataset<Row> to represent a Dataframe.

Features of Dataframes:

→ Use of ilp optimization engine: Dataframes make use of ilp optimizatio engines to process data efficiently. We can use the same engine for all Python, Java, Scala and R Dataframes APIs.

→ Handling of structured Data: Dataframes provide a schematic view of data. Here, the data has some meaning to it when it is being stored.

→ Custom memory management: In RDDs, the data is stored in memory, whereas dataframes store data off-heap (outside main Java Heap space, but still inside RAM), which in turn reduces the garbage collection overload.

→ Flexibility: dataframe like RDDs, can support various formats of data, such as csv, Cassandra etc.

→ Scalability: dataframes can be integrated with various other Big data tools and they allow processing megabytes to petabytes of data at once.

Applications of dataframes:

- Spark dataframes are used in Advanced Analytics and Mlc learning
- Data scientists are using these dataframes for increasingly sophisticated techniques to get their job done.
- Dataframes can be used directly in MLib's ML pipeline API.

Class Note:

Subject : ....B.D.T.....

Faculty : ....V.Divya.....

Topic: .....Spark SQL:....

Unit No. : 4
Lecture No.: 5
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 7

Spark SQL:

- Spark introduces a programming module for structured data processing called Spark SQL.
- It provides a programming called Dataframe and can act as distributed SQL query engine.

features of Spark SQL:-

→ Integrated - Seamlessly mix SQL queries with Spark pgms. Spark SQL lets us query structured data as a distributed dataset (RDD) in Spark, with integrated APIs in Python, Scala and Java. This tight integration makes it easy to run SQL queries alongside complex analytic algorithms.

→ Unified data access: Load and query data from a variety of sources. Schema-RDDs provide a single interface for efficiently working with structured data, including Hive tables, parquet files and JSON files.

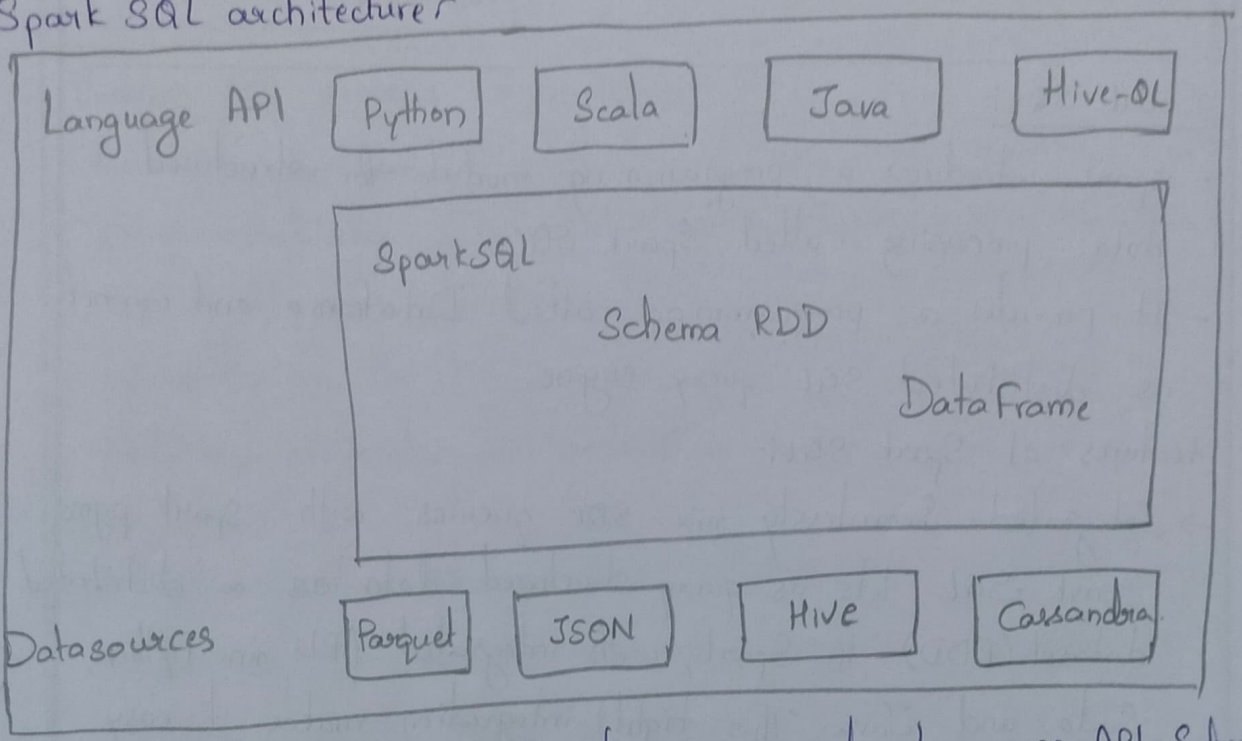
→ Hive compatibility: Run unmodified Hive queries on existing warehouses. Spark SQL reuses the Hive frontend and MetaStore, giving you full compatibility with existing Hive data, queries and UDFs. Simply install it alongside Hive.

→ Standard Connectivity: Connect through JDBC or ODBC. Spark SQL includes a server mode with industry

standard JDBC and ODBC connectivity.

→ Scalability: Use the same engine for both interactive and long queries. Spark SQL takes advantage of RDD model to support mid query fault tolerance, letting it scale to large jobs too.

Spark SQL architecture:



- This architecture contains 3 layers namely, Language API, Schema RDD, & data sources.

→ Language API:- Spark is compatible with different languages and Spark SQL. It is also supported by these languages- API (Python, Scala, Java, HiveQL).

→ Schema RDD: Spark core is designed with special data structure called RDD. Generally, Spark SQL works on schemas, tables and records. Therefore, we can use the schema RDD as temporary table. We can call this Schema RDD as dataframe.

→ Data sources: Usually the datasources for spark core is a text file, Avro file, etc. However the datasources for Spark SQL is different. Those are Parquet file, JSON document, HIVE tables and Cassandra database.

Class Note:

Subject : .....BDT.....

Faculty : ....V.Divya.....

Topic: .....Spark SQL.....

Unit No. : 4
Lecture No.: 5
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 8

Running SQL queries programmatically:

- An SQLContext enables applications to run SQL queries programmatically while running SQL functions and returns the result as a dataframe.
- Generally, in the background, Spark SQL supports two different methods for converting existing RDDs into dataframes -

(i) Inferring the schema using reflection:- This method uses reflection to generate the schema of an RDD that contains specific types of objects.

(ii) Programmatically specifying the schema:- This method is for creating dataframe is through programmatic interface that allows ~~you~~ to construct a schema and then apply it to an existing RDD.

Spark SQL - Datasources:

- A dataframe interface allows different Datasources to work on Spark SQL. It is a temporary table and can be operated as a normal RDD. Registering a dataframe as a table allows us to run SQL queries over its data.

- There are different types of data sources available in Spark SQL, such as:

1. JSON datasets: Spark SQL can automatically capture the schema of a JSON dataset and load it as a dataframe.
2. Hive tables: Hive comes bundled with the spark library as HiveContext, which inherits from SQLContext.
3. Parquet files: Parquet is a columnar format, supported by many data processing systems.

Class Note:

Subject : .....BDT.....

Faculty : ....V. Divya.....

Topic: .....PySpark.....

Unit No. : 4
Lecture No.: 6
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 9

PySpark:

- Apache Spark is written in Scala programming language. To support Python with Spark, Apache Spark Community released a tool, PySpark.
- Using PySpark, we can work with RDDs in python programming lang. also. and it is because of a library called Py4j that they are able to achieve this.
- PySpark offers PySpark Shell which links the Python API to the spark core and initializes the Spark context.
- Majority of data scientists and analytics experts today use Python because of its rich library set.
- Integrating Python with Spark is a boon to them.

PySpark Environment Setup:

Step 1:- Go to official Apache Spark download page and download the latest version of Apache Spark available there. by using spark-2.1.0-bin-hadoop2.7

Step 2 - Now extract the downloaded Spark tar file. By default, it will get downloaded in Downloads directory.

#tar -xvf Downloads/Spark-2.1.0-bin-hadoop2.7.tgz

It will create a directory spark-2.1.0-bin-hadoop2.7. Before starting PySpark, we need to set the following environments to set the Spark path and the Py4j path.

```
export SPARK_HOME=/home/hadoop/spark-2.1.0-bin-hadoop2.7
export PATH=$PATH:/home/hadoop/spark-2.1.0-bin-hadoop2.7/bin
export PYTHONPATH = $SPARK_HOME/python:$SPARK_HOME/python/lib/py4j-0.10.4-src.zip:$PYTHONPATH.
export PATH = $SPARK_HOME/python:$PATH
```

or, to set the above environments globally, put them in .bashrc file. Then run the following command for the environments to work.

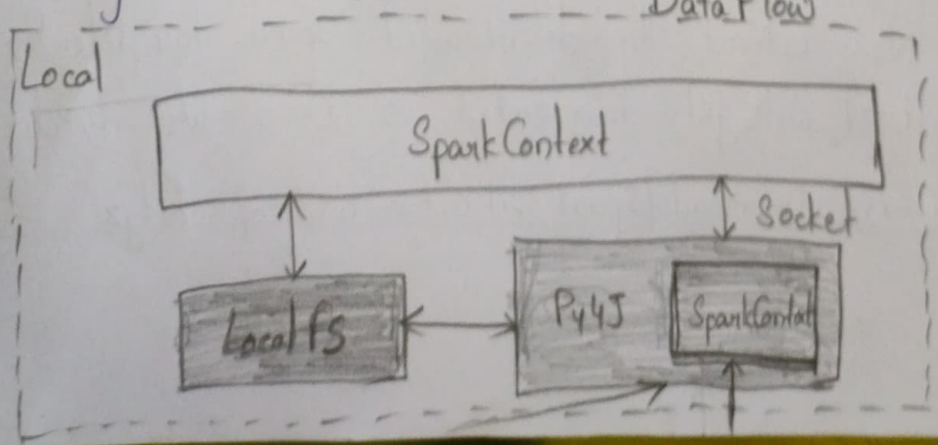
```
# source .bashrc
```

Now that we have all the environments set, let us go to Spark directory and invoke PySpark shell by running following command-

```
# ./bin/pyspark
```

Spark Context:-

- SparkContext is the entry point to any spark functionality. When we run any Spark application, a driver program starts, which has main function and SparkContext gets initiated here. The driver program then runs the operations inside the executors on worker nodes.
- SparkContext uses Py4J to launch a JVM and creates a JavaSparkContext. By default, PySpark has SparkContext available as 'sc', so creating a new SparkContext won't work.



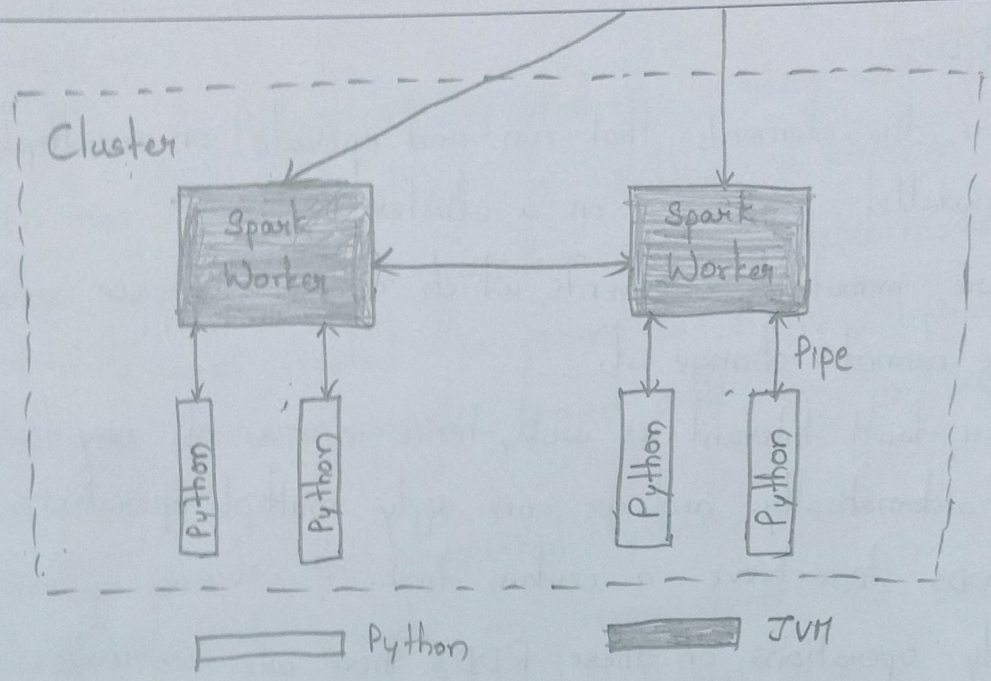
Class Note:

Subject : ....BDT....

Faculty : ....V. Divya....

Topic: ....Py.Spark....

Unit No. : 4
Lecture No.: 6
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 10



Parameters:-

- Following are parameters of a SparkContext.

→ Master - It is the URL of a cluster it connects to.

→ appName - Name of your job.

→ sparkHome - Spark installation directory.

→ pyFiles - The .zip or .py files to send to cluster and add to the PYTHONPATH.

→ Environment - worker nodes environment variables.

→ batchSize - The number of python objects represented as a single Java object.

→ Serializer - RDD serializer.

→ Conf - An object of L{SparkConf} to set all the Spark properties.
→ Gateway - Use an existing gateway and JVM, otherwise initializing a new JVM.
→ JSC - The JavaSparkContext instance.
→ profiler_cls - A class of custom Profiler used to do profiling.

PySpark-RDDs

- RDDs are the elements that run and operate on multiple nodes to do parallel processing on a cluster.
- RDDs are immutable elements, which means once we create an RDD we cannot change it.
- RDDs are fault tolerant as well, hence in case of any failure, they recover automatically. and we can apply multiple operations on these RDDs to achieve a certain task.
- To apply operations on these RDD's there are two ways-
  → Transformation
  → Action
- The following code block has the detail of a PySpark RDD class-

```
class pyspark.RDD(
    jrdd,
    ctx,
    jrdd_deserializer = AutoBatchedSerializer(PickleSerializer())
)
```

PySpark-Broadcast & Accumulator:-

- There are two types of shared variables supported by Apache Spark:-
  → Broadcast → Accumulator.

Class Note:

Subject : ....RDT.....

Faculty : ....V. Divya.....

Topic: ....Numpy.....

Unit No. : 4
Lecture No.: 7
Link to Session
Planner (SP) S.No. ..... of SP
Date Conducted:
Page No. 11

Numpy :

- Numpy is a Python Library used for working with arrays.
- It has also functions for working in domain of linear algebra, fourier transform and matches. matrices.
- Numpy was created in 2005 by Travis Oliphant which is an open source project and we can use it freely.
- Numpy stands for Numerical Python.
- Numpy aims to provide an array object that is upto 50x faster than traditional Python lists. where the array object in Numpy is called "ndarray", it provides a lot of supporting functions that make working with "ndarray" very easy.
- Numpy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. and this behavior is called locality of reference.
- If we have Python and PIP already installed on a system, then installation of Numpy is very easy.

Install it using this command:

C:\Users\Your Name> pip install numpy.

- Once Numpy is installed, import it in our applications by adding the "import" keyword:

```
import numpy.
```

- Numpy is usually imported under the "np" alias.

```
import numpy as np.
```

- Numpy is used to work with arrays and array object in Numpy is called "ndarray". and we can create a Numpy "ndarray" object by using the array() function.

```
import numpy as np
arr = np.array([1,2,3,4,5])
print(arr)
print(type(arr))
```

type() - this is built in python function which tells us the type of object passed to it.

- Array indexing is the same as accessing an array element. and we can access an array element by referring to its index number.
- The indexes in Numpy array start with 0, meaning that the first element has index 0, and second has index 1.

```
import numpy as np
arr = np.array([1,2,3,4])
print(arr[0])
print(arr[2] + arr[3])
```

Class Note:

Subject : .....BDT.....

Faculty : .....V. Divya.....

Topic: .....Numpy.....

Unit No. : 4
Lecture No.: 7
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 12

- Slicing in Python means taking elements from one given index to another given index. and we pass slice instead of index like this: [start:end]
and we can also define the step, like this: [start:end:step]
If we don't pass start it's considered 0. and
If we don't pass end it's considered length of array in that dimension.
If we don't pass step its considered 1.
- Use minus operator to refer to an index from the end.

Datatypes:

- NumPy supports a much greater variety of numerical types than Python does.

	Datatypes	Description
1.	bool_	Boolean (True or False) stored as a byte.
2.	int_	default integer type (same as C long; normally either int64 or int32)
3.	intc	identical to C int (normally int32 or int64)
4.	intp	integer used for indexing (same as C ssize_t; normally either int32 or int64).

5. float_ - shorthand for float64.
6. complex_ - shorthand for complex128.

- A datatype object describes interpretation of fixed block of memory corresponding to an array, depending on following aspects-
  - → Type of data (integer, float or Python object)
  - → Size of data
  - → Byte order (little-endian or big-endian)
  - → in case of structured type, the names of fields, datatypes of each field and part of memory block taken by each field.
  - → If datatype is a subarray, its shape and datatype.
- A dtype object is constructed using the following syntax-

numpy.dtype(object, align, copy)

where

object - to be converted to datatype object
align - if true, adds padding to field to make it similar to C-struct
copy - makes a new copy of dtype object. If false, the result is reference to builtin datatype object.

Creating NumPy Array:

- We can create a NumPy array using various function provided by the Python NumPy library. This package provides a multidimensional array object and various other required objects, routines, for efficient functionality.
- Following are the functions using which we can create a NumPy array-
  - (i) Using numpy.array()
  - (ii) Using numpy.zeros()

Class Note:

Subject : ....BDT.....

Faculty : ....V.Divya.....

Topic: ....Numpy.....

Unit No. : 4
Lecture No.: 7
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 13

(iii) Using numpy.one( )
(iv) Using numpy.arange( )
(v) Using numpy.linspace( )
(vi) Using numpy.random.rand( )
(vii) Using numpy.empty( )
(viii) Using numpy.full( )

- Contents of ndarray object can be accessed and modified by indexing or slicing, just like Python's in-built container objects.
- Three types of indexing methods are available - field access, basic slicing and advanced indexing.
- There are two types of advanced indexing - Integer and Boolean.
- In Numpy, attributes are properties of array objects that provide important information about the arrays and their data. and these attributes are used to access various details regarding the structure and configuration of the arrays without modifying them.

Numpy Array Shape:-

- The shape of a Numpy array is a tuple of integers. Each integer in the tuple represents the size of the array along a particular dimension or axis.

Handling missing data in arrays:-

- Handling missing data is a common challenge in data analysis and processing. Missing data in arrays can arise due to various reasons, such as incomplete data collection, errors during data entry, or intentional omission.
- In Numpy and data analysis, dealing with missing values involves identifying, handling and processing them effectively to ensure data integrity and accurate results.

Class Note:

Subject : .....BDT.....

Faculty : .....V. Divya.....

Topic: .....Scipy.....

Unit No. : 4
Lecture No.: 8
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 14

Scipy:

- It is pronounced as Sigh Pi, is a scientific python open source, distributed under the BSD licensed library to perform Mathematical, Scientific and Engineering Computations.
- SciPy library depends on Numpy, which provides convenient and fast N-dimensional array manipulation. and is built to work with Numpy arrays and provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization.

SciPy Subpackages:

- scipy.cluster → vector quantization / Kmeans.
- scipy.constants → physical and mathematical constants
- scipy.fftpack → Fourier transform.
- scipy.integrate → integration routines.
- scipy.interpolate → interpolation.
- scipy.io → data input and output.
- scipy.linalg → linear algebra routines.
- scipy.ndimage → n-dimensional image package.

Scipy - Environment Setup:-

- A lightweight alternative is to install Scipy using the popular Python package installer,

  pip install pandas.

- If we install the Anaconda Python package, Pandas will be installed by default.
- Following are the packages and links to install them in different operating systems:

→ Windows:

- Anaconda is a free Python distribution for the Scipy stack. It is also available for Linux and Mac.
- Canopy is a available free, as well as for commercial distribution with a full Scipy stack for Windows, Linux & Mac.
- Python (x,y) - is a free Python distribution with Scipy stack and Spyder IDE for Windows OS.

→ Linux:

- Package managers of respective Linux distributions are used to install one or more packages in the Scipy stack.

Scipy Cluster:

- K means clustering is a method for finding clusters and cluster centers in a set of unlabelled data.
- Given an initial set of K centers, the K-means algorithm iterates the following two steps:

→ For each center, the subset of training points (its cluster) that is closer to it is identified than any other center.

→ The mean of each feature for the data points in each cluster

Class Note:

Subject : ....BDT.....

Faculty : ....V.Divya.....

Topic: .....Scipy.....

Unit No. : 4
Lecture No.: 8
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:

are computed, and this mean vector becomes the new center for that cluster.

- Scipy constants package provides a wide range of constants, which are used in the general scientific area.

- 1. pi - pi
  2. golden - golden ratio. } mathematical constants.

1. c - speed of light in vaccum
2. speed_of_light - speed of light in vaccum.
3. h
4. Planck
5. G
6. e
7. R
8. Avogadro
9. K
10. electron_mass / m_e
11. proton_mass / m_p
12. neutron_mass / m_n

} physical constants

- Scipy FFTpack:

- Fourier transformation is computed on a time domain signal to check its behavior in the frequency domain. and it find its application in disciplines such as signal and noise processing, etc.

- Interpolation is the process of finding a value b/w two points on a line or a curve.

```
import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
x = np.linspace(0,4,12)
y = np.cos(x**2/3+4)
print x,y.
```

-

Class Note:

Subject : ...BDT.....

Faculty : .....V.Divya.....

Topic: ....Code....Optimization

Unit No. : 4
Lecture No.: 9
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 15

Code optimization:-

- PySpark performance can be optimized by using techniques like data serialization and caching, and the Spark Dataframe API, which provides optimized execution plans and automatic optimization for many data processing tasks.

(i) Use Dataframe/Dataset over RDD.
(ii) Avoid UDFs (User Defined Functions)
(iii) Disable DEBUG and INFO Log Levels.
(iv) Use Small Scripts and Multiple Environments in PySpark
(v) Number of Partitions and Partition Size in PySpark.

(i) Use Dataframe/Dataset over RDD:-

- Spark's Resilient Distributed Datasets (RDDs) are a data structure that enables distributed data processing in Spark.
- RDDs have a no. of limitations, including the fact that they are low-level and donot provide an optimized execution plan for many data processing tasks.
- To address these limitations, Spark introduced the Dataframe and Dataset APIs, which provide

a higher level, more optimized interface for distributed data processing.

- Data Frames and Datasets are built on top of RDDs and provide a number of benefits over RDDs, including:
  i, Improved performance
  ii, Strong typing
  iii, Better integration with other tools
- Dataframe/Dataset API also leverages two technologies that help improve the performance;
  i, Project Tungsten
  ii, Catalyst Optimizer

ii, Avoid UDFs:

- UDFs are functions that are defined by the user and can be used to extend the functionality of Spark's SQL and Dataframe APIs.
- While UDFs can be useful in some cases, they can also have have a negative impact on the performance of PySpark applications.
- One of the main reasons UDFs can impact performance is that they are executed on driver node of a Spark application rather than on the worker nodes where data is distributed.
- To avoid the negative performance impact of UDFs, it is generally recommended to use Spark's built in functions and APIs whenever possible. and these functions and APIs are optimized for distributed data processing and can often be used to achieve the same results as UDFs with better performance.

Class Note:

Subject : ...BDT.....

Faculty : ...V.Divya.....

Topic: ...Optimization techniques.

Unit No. : 4
Lecture No.: 19
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 16

- There are also a no. of alternatives to UDFs that can be used to extend the functionality of Spark's SQL & Dataframe APIs including:
  a, Spark SQL functions
  b, Window functions
  c, Aggregate functions.

iii, Disable DEBUG & INFO Log levels:

- PySpark uses the Log4j library for logging and can o/p messages at different levels of severity, including DEBUG, INFO, WARN & error.
- This can be useful for debugging and troubleshooting, but it can also have a negative impact on the performance of the application, especially for large datasets and complex data processing tasks.
- To optimize the performance of PySpark applications, it is often recommended to disable or reduce the amount of logging output.

iv, Use small scripts and multiple environments in PySpark:

- Small scripts are easier to maintain, test and troubleshoot than large scripts, and they can be faster to execute.

- By breaking down a large PySpark application into smaller, more focused scripts, it is often possible to improve the performance and maintainability of the application.
- Multiple environments, such as development, staging and production environments can also help to optimize the performance of PySpark applications.

V, No. of Partitions and Partition size in PySpark:-

- The no. of partitions and the size of each partition in a PySpark application can have a significant impact on performance of the application.
- In general, using a large no. of smaller partitions can lead to better performance, while using a smaller no. of larger partitions can lead to worse performance.
- To optimize the no. of partitions and partition size, it is often recommended to:

a, Use a larger no. of smaller partitions
b, Experiment with different partition sizes
c, Use the Spark configuration settings
d, Consider the Spark recommended partition size.

Class Note:

Class Note:

Subject : BDT

Faculty : V. Divya

Topic: Cluster configurations.

Unit No. : 4
Lecture No.: 10
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 18

Cluster configuration in PySpark:-

- Configuring a cluster in PySpark involves setting up the environment and parameters to connect to a Spark cluster, so you can execute distributed computations.
- Here are the keysteps and settings:

a, Configure the SparkSession:

- In PySpark, the SparkSession is the entry point for accessing Spark features. To configure it for a cluster, specify the cluster master URL and other configuration details.

b, Cluster Modes:

- There are different modes we can use for setting up the cluster:
  - → Standalone mode
  - → Yarn cluster
  - → Mesos cluster
  - → Kubernetes.

c, Key configuration parameters:

- Here are essential configuration options for tuning the cluster setup:
  - → spark.executor.memory
  - → spark.executor.cores.

→ spark.executor.instances
→ spark.driver.memory
→ spark.dynamicAllocation.enabled

iv. Submitting jobs to a cluster:
- Once configured, we can use spark-submit to run the job on the cluster. Save script and run.

v. Tuning for cluster resources:-
- Adjust these settings based on your cluster's resources and the job requirements:
  → Memory
  → Cores
  → Executors
- Cluster configurations vary, so use our cluster's specifications & requirements to finetune these settings.

Class Note:

Subject : .....BDT.....

Faculty : ....V.Divya.....

Topic: ..Linear Algebra computation in Large scale.

Unit No. : 4
Lecture No.: 11
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 19

Linear algebra computation in large scale in PySpark:

- PySpark, with its distributed processing capabilities, is well suited for large scale linear algebra computations, especially when handling massive datasets.
- While Pyspark doesn't directly offer a full linear algebra library, we can use Spark's MLib library for many linear algebra operations, such as matrix operations, decompositions and other algebraic manipulations.

i, Create and manipulate large matrices in PySpark:

- Spark's MLib MLlib provides a 'Matrices' class for working with dense and sparse matrices.

ii, Matrix operations:-

- Common operations on matrices, such as transpose, dot product and basic arithmetic can be performed on distributed matrices in PySpark.

a, Transpose and Dot product:

To perform matrix multiplication, we can use Dataframes and the MLib library

→ Column and row statistics:

- The RowMatrix and IndexedRowMatrix classes provide methods for computing column summary statistics, row normalization and other operations.

(iii) Singular Value Decomposition (SVD):

- Spark's MLlib provides support for Singular Value Decomposition on large matrices.
- This is useful for dimensionality reduction, recommendation systems and other machine learning applications.

(iv) Principal Component Analysis (PCA)

- PCA is a common linear algebra technique used for dimensionality reduction, which can also be performed in PySpark.

(v) Distributed Matrix Inversion:

- If we need matrix inversion at scale, we may need to write a custom implementation,
- Inverting large matrices across distributed nodes can be challenging due to memory and data shuffling costs.
- Instead, approximate methods like pseudoinverse or iterative solvers are often more practical in a distributed environment.

(vi) Gramian Matrix Computation:

- Spark also provides an efficient method for computing the Gramian matrix.

Class Note:

Subject : .....B.D.T.....

Faculty : .....V.Divya.....

Topic: .....Distributed file Storage systems.

Unit No. : 4
Lecture No.: 12
Link to Session
Planner (SP) S.No. .....of SP
Date Conducted:
Page No. 20

* Distributed File Storage systems in PySpark:

- It supports several distributed file storage systems for reading and writing large datasets, commonly used for big data processing. and these storage systems are designed to store and process data in parallel, supporting both high-throughput and fault tolerance.
- Here are some of the most widely used distributed file storage systems with PySpark:

i) HDFS:

- It is the default file system used in most Spark clusters, designed for reliable, scalable storage on large clusters.
- PySpark can directly access HDFS data and works seamlessly with it.

→ Setup
→ Reading from HDFS
→ Writing to HDFS.

ii) Amazon S3:

- It is an object storage system that allows you to store and retrieve large amounts of data and is commonly used with Spark on AWS EMR clusters.

→ Setup
→ Reading from S3
→ Writing to S3.

iii, Google Cloud Storage (GCS):
- GCS is another object storage service often used in Spark clusters running on Google Cloud Dataproc.

→ Setup
→ Reading from GCS
→ Writing to GCS.

iv, Azure Blob Storage:
- It is an object storage system optimized for massive amounts of unstructured data.
- It is frequently ~~data~~ used with Spark clusters on Azure databricks.

→ Setup
→ Reading from Azure Blob storage
→ Writing to Azure Blob storage.

v, Databricks Delta Lake:
- Delta Lake is an open source storage layer that brings ACID transactions, scalable metadata handling, and unified streaming and batch data processing to Apache Spark.
- Delta Lake can be layered on top of cloud storage systems such as S3, GCS or Azure blob.

→ Reading from Delta lake
→ Writing to Delta lake.

vi, Alluxio:
- It is a memory speed distributed files system often used to accelerate data access in big data systems. and it's compatible with various storage backends, including HDFS, S3 & GCS.