MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

D

SE-

3264 - Big Data Analytics

Laboratory Manual

Department	:	Data Science Engineering And Computer Applications
Course Name & code	:	DSE-3264 & Big Data Analytics Laboratory
Semester & branch	:	VI Sem & BTech Data Science & Engineering
Name of the faculty	:	Dr.Rashmi Laxmikant Malghan , Mrs. Shavantrevva S B
No of contact hours/week:		

L	T	P	C
0	0	3	1

What is HDFS??

Hadoop comes with a distributed file system called HDFS. In HDFS data is distributed over several machines and replicated to ensure their durability to failure and high availability to parallel application. It is cost effective as it uses commodity hardware. It involves the concept of blocks, data nodes and node name.

HDFS building Blocks:

Blocks: A Block is the minimum amount of data that it can read or write.HDFS blocks are 128 MB by default and this is configurable.Files n HDFS are broken into block-sized chunks,which are stored as independent units.Unlike a file system, if the file is in HDFS is smaller than block size, then it does not occupy full block?s size, i.e. 5 MB of file stored in HDFS of block size 128 MB takes 5MB of space only.The HDFS block size is large just to minimize the cost of seek.

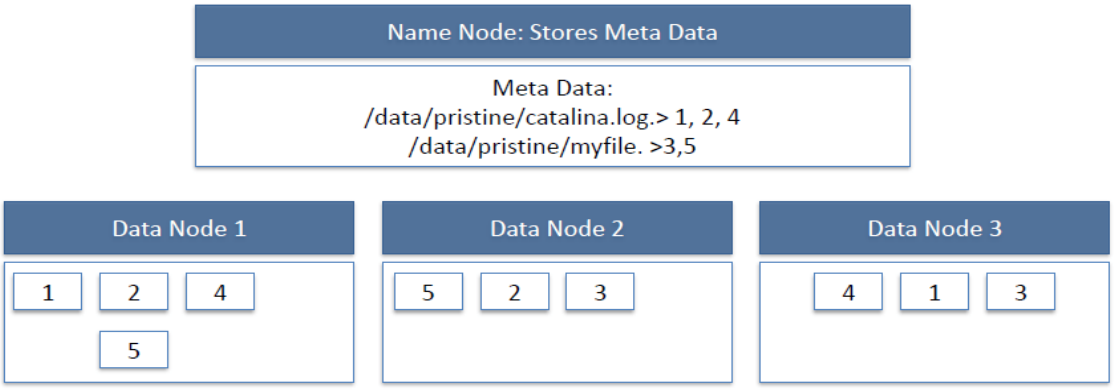
Name Node: HDFS works in master-worker pattern where the name node acts as master.Name Node is

controller and manager of HDFS as it knows the status and the metadata of all the files in HDFS; the metadata information being file permission, names and location of each block.The metadata are small, so it is stored in the memory of name node,allowing faster access to data. Moreover the HDFS cluster is accessed by multiple clients concurrently,so all this information is handled bya single machine. The file system operations like opening, closing, renaming etc. are executed by it.

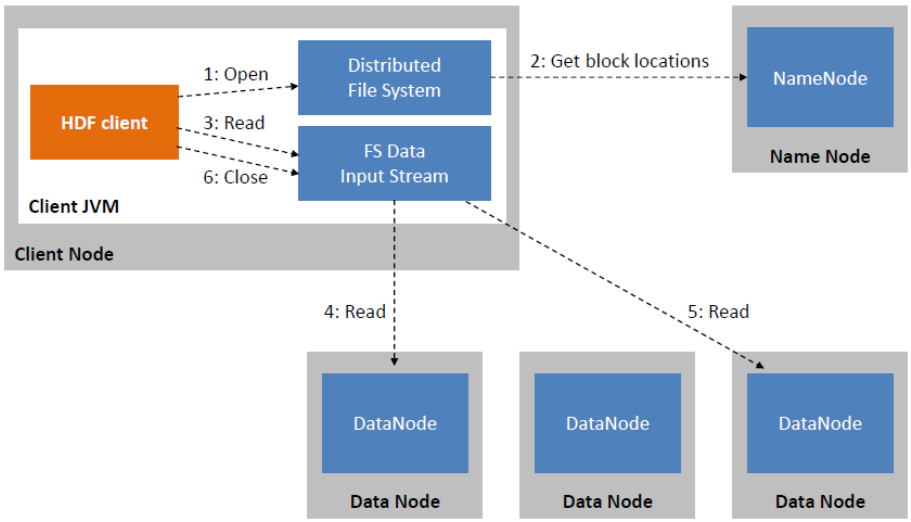
Data Node: They store and retrieve blocks when they are told to; by client or name node. They report back to name node periodically, with list of blocks that they are storing. The data node being a commodity hardware also does the work of block creation, deletion and replication as stated by the name node.

Secondary Name Node: It is a separate physical machine which acts as a helper of name node. It performs periodic check points.It communicates with the name node and take snapshot of meta data which helps minimize downtime and loss of data.

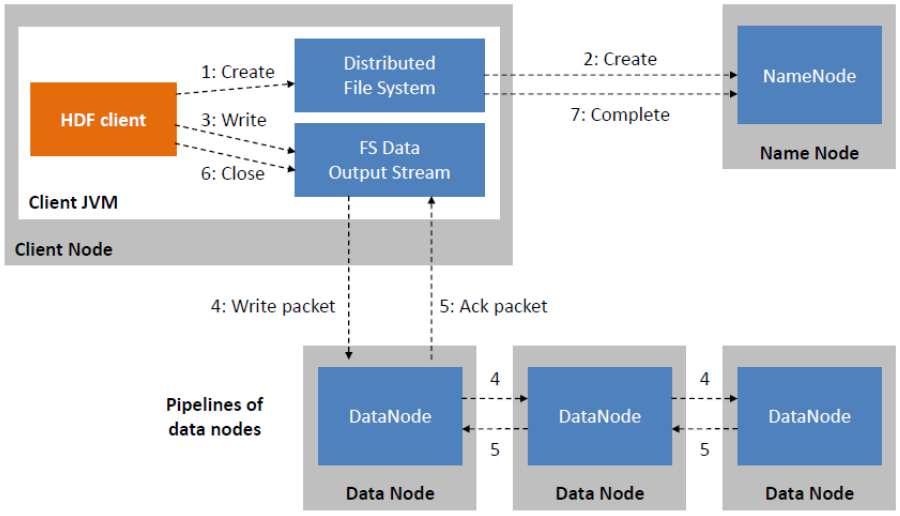
- **HDFS DataNode and NameNode Image:**



- **HDFS Read workflow**:



- **HDFS Write:**



- To use the HDFS commands, first you need to start the Hadoop services using the following command:

  $sbin/start-all.sh

- To check the Hadoop services are up and running use the following command:

  $jps

- Check the datanode service is up by running ***jps*** commond at client side.

```
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ jps
2546 SecondaryNameNode
2404 DataNode
2295 NameNode
2760 ResourceManager
2874 NodeManager
4251 Jps
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$
```

- To perform various file operations use the prefix ***Hadoop fs/ hdfs dfs*** as a prefix for each command

- To create a directory. In Hadoop *dfs* there is no home directory by default. So let's first create it sing ***mkdir*** commond.

Week 1 Exercise: Hadoop Distributed File System

1. List all possible Linux file operations and execute each one of them in Linux CLI.
2. Interact with HDFS using command line interface to understand the basic working structure of Hadoop cluster. Using Hadoop CLI, demonstrate the following commands to:
    - Create a directory in HDFS.
    - create an empty file
    - copy files/folders from local file system to hdfs store.
    - print file contents.
    - copy files/folders from hdfs store to local file system.
    - move file from local to hdfs
    - copy files within hdfs
    - move files within hdfs
    - size of each file in directory
    - total size of directory/file
    - last modified time of directory or path
    - change the replication factor of a file/directory in HDFS.
    - List the contents of a directory in HDFS.
    - Remove a file from HDFS.
    - Change File Permissions
    - Changing File Ownership
    - Checksum Calculation
    - File Concatenation
    - File Compression/Decompression
    - File Block Location Information
    - File Encryption/Decryption

3. Use web interface to monitor Name node manager, resource manager, and Data node status.

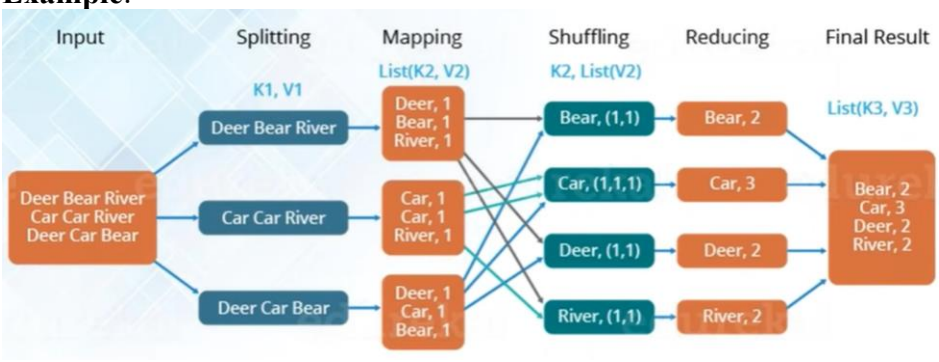
MapReduce Concept:

1. **OBJECTIVE:** Run a basic word count Map Reduce program to understand Map Reduce Paradigm.

2. **RESOURCES:** VMWare stack (Hadoop), __ GB RAM, Web browser, Hard Disk __ GB.

3. **PROGRAM LOGIC:** MapReduce (Wordcount): It consists of 4 phases (i.e. Partition or splitting, Mapping, Sorting or shuffling, Reducing).
   WordCount is a simple program which counts the number of occurrences of each word in a given text input data set. WordCount fits very well with the MapReduce programming model making it a great example to understand the Hadoop Map/Reduce programming style. Our implementation consists of three main parts:
   Mapper
   Reducer
   Driver

Example:



Step-1. Write a Mapper

Mapper.py: Initially the partition of content takes place based on line.split() function, number of partitions made = number of mapper class gets created. Mapper overrides the ―map‖ function which provides <key, value> pairs as the input. Even the key is repeated in same or different mapper class it doesnot matter as the default value for every key is assigned to be as "1". A Mapper implementation may output<key,value> pairs using the provided Context .

Input value of the WordCount Map task will be a line of text from the input data file and the key would be the line number <line_number, line_of_text> . Map task outputs <word, one> for each word in the line of text.

Pseudo-code : mapper.py

```
#!/usr/bin/python3
"mapper.py"
import sys
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
```

```
    # increase counters
    for word in words:
        print ('%s\t%s' % (word, 1))
```

Step-2. Write a Reducer

A Reducer collects the intermediate <key,value> output from multiple map tasks and assemble a single result. Here, the WordCount program will sum up the occurrence of each word to pairs as <word, occurrence>.

Pseudo-code: reducer.py

```
#!/usr/bin/python3
"reducer.py"
import sys
current_word = None
current_count = 0

for line in sys.stdin:
    # remove leading and trailing whitespaces
    line = line.strip()
    # parse the input we got from mapper.py
    word, count = line.split('\t')
    count = int(count)

    if current_word == word:
        current_count += count
    else:
        if current_word:
            print ('%s\t%s' % (current_word, current_count))
        current_count = count
```

4. **Input/Output**

TO run word count program locally use following 4 and 5 commands

hdoop@hadoop-client:~cat input.txt |pyhton3 mapper.py

Output:

```
hi      1
how     1
are     1
you     1
i       1
am      1
good    1
```

hope 1
you 1
doing 1
good 1
too 1
how 1
about 1
you. 1
i 1
am 1
in 1
manipal 1
studying 1
Btech 1
in 1
Data 1
science. 1

hdoop@hadoop-client:~$ cat input.txt |python3 mapper.py|sort|python3 reducer.py

Output:

about 1

am 2

are 1

Btech 1

Data 1

doing 1

good 2

hi 1

hope 1

how 2

i 2

in 2

manipal 1

science. 1

studying 1

too 1

you 2

you. 1

TO run word count program on Hadoop framework use following command:

hdoop@hadoop-client:~$ hadoop jar '/home/hdoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.6.jar' -file mapper.py -mapper mapper.py -file reducer.py -reducer reducer.py -input /bda1/input.txt -output /bda1/oup1

Output:2024-01-16 10:18:00,489 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.

packageJobJar: [mapper.py, reducer.py, /tmp/hadoop-unjar2657340332712108565/] [] /tmp/streamjob3503883941011863300.jar tmpDir=null

2024-01-16 10:18:01,069 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /192.168.159.101:8032

2024-01-16 10:18:01,343 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /192.168.159.101:8032

2024-01-16 10:18:01,544 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hdoop/.staging/job_1705376153146_0001

2024-01-16 10:18:02,354 INFO mapred.FileInputFormat: Total input files to process : 1

2024-01-16 10:18:02,425 INFO mapreduce.JobSubmitter: number of splits:2

2024-01-16 10:18:02,577 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1705376153146_0001

2024-01-16 10:18:02,577 INFO mapreduce.JobSubmitter: Executing with tokens: []

2024-01-16 10:18:02,786 INFO conf.Configuration: resource-types.xml not found

2024-01-16 10:18:02,786 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.

2024-01-16 10:18:02,975 INFO impl.YarnClientImpl: Submitted application application_1705376153146_0001

2024-01-16 10:18:03,028 INFO mapreduce.Job: The url to track the job: http://hadoop-master:8088/proxy/application_1705376153146_0001/

2024-01-16 10:18:03,029 INFO mapreduce.Job: Running job: job_1705376153146_0001

2024-01-16 10:18:09,113 INFO mapreduce.Job: Job job_1705376153146_0001 running in uber mode : false

2024-01-16 10:18:09,115 INFO mapreduce.Job: map 0% reduce 0%

2024-01-16 10:18:14,186 INFO mapreduce.Job: map 100% reduce 0%

2024-01-16 10:18:18,219 INFO mapreduce.Job: map 100% reduce 100%

2024-01-16 10:18:19,248 INFO mapreduce.Job: Job job_1705376153146_0001 completed successfully

2024-01-16 10:18:19,322 INFO mapreduce.Job: Counters: 54

File System Counters

FILE: Number of bytes read=214

FILE: Number of bytes written=843282

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=356

HDFS: Number of bytes written=127

HDFS: Number of read operations=11

HDFS: Number of large read operations=0

HDFS: Number of write operations=2

HDFS: Number of bytes read erasure-coded=0

Job Counters

Launched map tasks=2

Launched reduce tasks=1

Data-local map tasks=2

Total time spent by all maps in occupied slots (ms)=6466

Total time spent by all reduces in occupied slots (ms)=1625

Total time spent by all map tasks (ms)=6466

Total time spent by all reduce tasks (ms)=1625

Total vcore-milliseconds taken by all map tasks=6466

Total vcore-milliseconds taken by all reduce tasks=1625

Total megabyte-milliseconds taken by all map tasks=6621184

Total megabyte-milliseconds taken by all reduce tasks=1664000

Map-Reduce Framework

Map input records=6

Map output records=24

Map output bytes=160

Map output materialized bytes=220

Input split bytes=188

Combine input records=0

Combine output records=0

Reduce input groups=18

Reduce shuffle bytes=220

Reduce input records=24

Reduce output records=18

Spilled Records=48

Shuffled Maps =2

Failed Shuffles=0

Merged Map outputs=2

GC time elapsed (ms)=1469

CPU time spent (ms)=3500

Physical memory (bytes) snapshot=1203916800

Virtual memory (bytes) snapshot=7675027456

Total committed heap usage (bytes)=1232601088

Peak Map Physical memory (bytes)=477769728

Peak Map Virtual memory (bytes)=2556215296

Peak Reduce Physical memory (bytes)=250322944

Peak Reduce Virtual memory (bytes)=2562641920

Shuffle Errors

BAD_ID=0

CONNECTION=0

IO_ERROR=0

WRONG_LENGTH=0

WRONG_MAP=0

WRONG_REDUCE=0

File Input Format Counters

Bytes Read=168

File Output Format Counters

Bytes Written=127

2024-01-16 10:18:19,322 INFO streaming.StreamJob: Output directory: /bda1/oup1

If the above 7th command runs successfully: Then in local host -browse utilities-specific folder : 2 files will get created (status: successful and Part-00000)

To view that : hdfs dfs -ls/bda1/output

To display : hdoop@hadoop-client:~$ hdfs dfs -cat /bda1/oup1/part-00000

Output:

Btech 1

Data 1

about 1

am 2

are 1

doing 1

good 2

hi 1

hope 1

how 2

i 2

in 2

manipal 1

science. 1

studying 1

too 1

you 2

you. 1

hdoop@hadoop-client:~$ hdfs dfs -get /bda/output/part-00000 /home/hdoop

hdoop@hadoop-client:~$ Cat part-0000

Week 2: Exercise: MapReduce with Python

1. Consider the text file (consider larger file size) of your choice and perform word count using MapReduce technique.
2. Perform Matrix operations using MapReduce by considering 3 * 3 matrix and perform following operations:
    i. Matrix addition and subtraction
    ii. Matrix Multiplication
    iii. Matrix transpose

Note: Consider 3*3 matrix content as shown below

a,0,0,10

a,0,1,20
a,0,2,30
a,1,0,40
a,1,1,50
a,1,2,60
a,2,0,70
a,2,1,80
a,2,2,90

b,0,0,1
b,0,1,2
b,0,2,3
b,1,0,4
b,1,1,5
b,1,2,6
b,2,0,7
b,2,1,8
b,2,2,9

3. Create a text file containing the 20 student details such as registration number, name and marks (ex: 1001, john,45) .Write a MapReduce program to sort data by student name.

Week 3: Exercise: MapReduce with Python

1. Write a MapReduce program to find unit wise salary for the bellow given data.

EmpNo	EmpName	Unit	Designation	Salary
1001	John	IMST	TA	30000
1002	Jack	CLOUD	PM	80000
1003	Joshi	FNPR	TA	35000
1004	Jash	ECSSAP	PM	75000
1005	Yash	FSADM	SPM	60000
1006	Smith	ICS	TA	24000
1007	Lion	IMST	SPM	56000
1008	kate	FNPR	PM	76000
1009	cassy	MFGADM	TA	40000
1010	ronald	ECSSAP	SPM	65000

2. Consider the following sample text file to compute the the average, minimum and maximum recorded temperature by year wise using concept of Map Reduce.

Temperature.txt

2014 44

2013 42

2012 30

2013 44

2010 45

2014 38

2011 42

2010 44

PIG TOOL: Pig Latin

1. **OBJECTIVE:** HOW TO EXECUTE & RUN THE PROGRAM LOCALLY & TEST IT ON HADOOP

2. **RESOURCES:** VMWare stack (Hadoop), __ GB RAM, Web browser, Hard Disk __ GB.

3. **PROGRAM LOGIC: PIG (Wordcount, Identify Most Popular Movie)**
   Focus on the data transformations rather than the underlying MapReduce implementation. Apache Pig's high-level dataflow engine simplifies the development of large-scale data processing tasks on Hadoop clusters by providing an abstraction layer and leveraging the power of MapReduce without requiring users to write complex Java code.

   Execution Modes:
   MapReduce Mode: This is default mode, which needs access to a Hadoop cluster and HDFS installation. The input and the output files both are present on the HDFS environment.

   Command : pig –x mapreduce
   Or
   pig

   Local Mode: With access to a single machine, all files are installed and run using a local host and file system. The local mode is specified using "-x flag" (i.e. pig –x local). The input and output files are present on local file system
   Command : pig –x local

   Running Modes:
   Interactive mode: Run pig in interactive mode by invoking grunt shell.
   Batch mode: Create pig script to run in batch mode. Write pig latin statements in a file and save it with .pig extension

   Executing pig in "Batch Mode":
   While executing Apache Pig statements (commands) in batch mode , perform the steps as below:

   Step 1: Write all the pig statements in single file and save with .pig extension. (Ex: pigcript.pig)

Step 2: Execute the pig script choosing local or MapReduce mode.

Execution using Grunt Mode:
grunt>exec /pigscript.pig

Executing a Pig Script from HDFS
We can also execute a Pig script that resides in the HDFS.
Suppose there is a Pig script with the name pigscript.pig in the HDFS directory named /pig_data/. We can execute it as shown below
$ pig -x mapreduce hdfs://localhost:9000/pig_data/pigscript.pig

Procedure to execute Pig Progrsm:
Step1: Create input.txt file

Step2: Transfer to HDFS
hdfs dfs put /home/hdoop/input.txt bda1/
Step3: Create Pigscript file
sudo gedit pigscript.pig
OR
vi pigscript.pig

Code to be typed in pigscript.pig
record = load '/bda1/input.txt/';
store record into '/bda1/out';
Step4: **Run pigscript in mapreduce mode**
pig -x mapreduce pigscript.pig
Step5: **Check the status of execution**

Output: Found 2 items
-rw-r--r-- 2 hdoop supergroup 0 2024-01-12 15:05 /bda1/out/_SUCCESS
-rw-r--r-- 2 hdoop supergroup 112 2024-01-12 15:05 /bda1/out/part-m-00000

Step 6: View the output file
hdfs dfs -cat /bda1/out/part-m-00000

Sample Program with Execution: (Input and Output)
PIG EXECUTION : NORMAL TEXT FILE (HOW TO EXECUTE & RUN THE PROGRAM LOCALLY & TEST IT ON HADOOP)

1) Create input.txt file :
   Content :
   hi how are you
   i am good
   hope you doing good too
   how about you.
   i am in manipal
   studying Btech in Data science.

2) Transfer to HDFS: hdfs dfs -put /home/hdoop/input.txt bda1/

3) Create Pigscript file: sudo gedit pigscript.pig
Content : record = load '/bda1/input.txt/';
store record into '/bda1/out';

4) hdfs dfs -ls bda1/input

For compilation of Program:

5) hdoop@hadoop-master:~$ pig pigscript.pig

For execution of Program:

6) hdoop@hadoop-master:~$ run pigscript.pig

```
HadoopVersion   PigVersion   UserId StartedAt          FinishedAt         Features
3.3.6   0.17.0  hdoop  2024-01-12 15:05:13  2024-01-12 15:06:55  UNKNOWN
```

Output:
Success!

Job Stats (time in seconds):

JobId	Maps	Reduces	MaxMapTime	MinMapTime	AvgMapTime	MedianMapTime	MaxReduceTime	MinReduceTime	AvgReduceTime	MedianReducetime	Alias	FeatureOutputs
job_1705045524875_0010	1	0	n/a	n/a	n/a	0	0	0	0	0	record	MAP_ONLY /bda1/out,

Input(s):
Successfully read 0 records from: "/bda1/input.txt"

Output(s):
Successfully stored 0 records in: "/bda1/out"

Counters:
Total records written : 0
Total bytes written : 0
Spillable Memory Manager spill count : 0
Total bags proactively spilled: 0
Total records proactively spilled: 0

Job DAG:
job_1705045524875_0010

7) hdoop@hadoop-master:~$ hdfs dfs -ls /bda1/out
Output: Found 2 items
-rw-r--r-- 2 hdoop supergroup 0 2024-01-12 15:05 /bda1/out/_SUCCESS
-rw-r--r-- 2 hdoop supergroup 112 2024-01-12 15:05 /bda1/out/part-m-00000

8) hdoop@hadoop-master:~$ hdfs dfs -cat /bda1/out/part-m-00000
Output:
hi how are you

i am good
hope you doing good too
how about you.
i am in manipal
studying Btech in Data science.

Week 4 A): Pig Execution

A. Consider normal text file to learn the pig running modes and execution modes. Run the program locally and test it on Hadoop.
B. Write a pig program to count the number of word occurrences using python in different modes (local mode, MapReduce mode)
C. Execute the pig script to find the "most popular moive in the dataset". In this example we will be dealing with 2 files (ratings.data and movies.item). Consider the dataset:
wget https://raw.githubusercontent.com/ashaypatil11/hadoop/main/movies.item,
wget https://raw.githubusercontent.com/ashaypatil11/hadoop/main/ratings.data

Week5 Exercise

1. Create the dataset of your choice and perform word count program using spark tool.
2. Given a dataset of employee records containing (name, age, salary), use map transformation to transform each record into a tuple of (name, age * 2, salary)?

Reg.No	EmpName	Age	Salary
24	John	26	30000
34	Jack	40	80000
61	Joshi	25	35000
45	Jash	35	75000
34	Yash	40	60000
67	Smith	20	24000
42	Lion	42	56000
62	kate	50	76000
21	cassy	51	40000
10	ronald	57	65000
24	John	26	30000
67	Smith	20	24000
45	Jash	35	75000
21	cassy	51	40000

3. From the same employee dataset, filter out employees whose salary is greater than 50000 using the filter transformation.
4. Create a text file that will have few sentences, use flatMap transformation to split each sentence into words.
5. Create a dataset having student details such as (name, subject, score), from this dataset group students by subject using the groupBy transformation.
6. From the employee dataset, collect the first 5 records as an array using the collect action.

7. Demonstrate the creation of RDD using Parallelized collection, existing RDD by finding the sum of all elements in an RDD1(which holds array elements). Also, create an RDD from external sources.

A. Consider the dataset given in Question B. Perform the following operations.
sortByKey()
groupByKey()
countBykey()

Week 7 : Spark Execution and Scala

1) Assume you have a CSV file named **clickstream_data.csv** with the following columns: **user_id , page_id, timestamp, action** (e.g., 'click', 'view', 'purchase').

- **Load the data into a PySpark DataFrame.**
- **Display the schema and the first 5 rows of the DataFrame.**
- **Calculate the total number of clicks, views, and purchases for each user.**
- **Identify the most common sequence of actions performed by users (e.g., click -> view -> purchase).**

2) Consider a scenario of Web Log Analysis. Assume you have a log file named web_logs.txt with the columns: **Timestamp, user_id, page_id, action (e.g., 'click', 'view', 'purchase').** Identify the most engaged users by calculating the total time spent on the website for each user. Implement the mentioned case with "PySpark Scala"