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

DSE-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. Saraswati Koppad, Dr. Shavantrevva Sangappa Bilakeri

No of contact hours/week:

L	T	P	C
0	0	3	1

CONTENTS

Course Objectives ·

1. Gain practical experience using big data tools and platforms like Hadoop, Spark, Hive, Pig, and HBase to store, process, and analyze large datasets.
2. Implement Hadoop MapReduce for processing Big Data
3. Apply data analytics techniques to solve problems and extract meaningful insights using big data frameworks.

Course Outcomes:

At the end of this course, students will have the ability to

1. Demonstrate the ability to use big data frameworks such as Hadoop, Spark
2. Apply MapReduce techniques to process large data sets.
3. Demonstrate the ability to use big data tools such as Pig, Hive, and HBase to store and process Big Data
4. Apply analytical methods and techniques to solve data-driven problems.

Evaluation plan : (Tentative)

- Internal Assessment Marks: 60%
- End semester assessment of 2-hour duration: 40 %

Evaluation pattern

Internal Marks – 60 + End Sem - 40	
Internal Marks	Internal Assessment – 40 + Mid-sem - 20
Internal Assessment	Lab observations – 2*11(22) + Viva (18)
Lab Observation	Record (1*11) + Execution (1*11)
Viva	Quiz and mini-project (18)

INSTRUCTIONS TO THE STUDENTS

Pre-Lab Session Instructions

- Be on time, adhere to the institution's rules, and maintain decorum.
- Leave your mobile phones, pen drives, and other electronic devices in your bag and keep the bag in the designated place in the lab.
- Must Sign in to the log register provided.
- Make sure to occupy the allotted system and answer the attendance.

In-Lab Session Instructions

- Follow the instructions on the allotted exercises.
- Show the program and results to the instructors on completion of experiments.
- Copy the program and results for the lab record.
- Prescribed textbooks and class notes can be kept ready for reference if required.

General Instructions for the Exercises in Lab

- Academic honesty is required in all your work. You must solve all programming assignments independently, except where group work is authorized. This means you must not take, show, give, or otherwise allow others to take your program code, problem solutions, or other work.
- The programs should meet the following criteria:
    - Programs should be interactive with appropriate prompt messages, error messages if any, and descriptive messages for outputs.
    - Programs should perform input validation (Data type, range error, etc.), give appropriate error messages, and suggest corrective actions.
    - Comments should be used to give the statement of the problem, and every function should indicate the purpose of the function, inputs, and outputs.
    - Statements within the program should be properly indented.
    - Use meaningful names for variables and functions.
    - Make use of constants and type definitions wherever needed.
    - The exercises for each week are divided into three sets:
        - Solved solutions
        - Lab exercises - to be completed during lab hours
        - Additional Exercises - to be completed outside the lab or in the lab to enhance the skill

Questions for lab tests and examinations are not necessarily limited to the questions in the manual but may involve some variations and/or combinations of the questions.

THE STUDENTS SHOULD NOT

- Possess mobile phones or any other electronic gadgets during lab hours.
- Go out of the lab without permission.
- **Change/update any configuration in your allotted system. If so, the student will lose internal assessment marks**

Week 1 - Understanding Hadoop and HDFS Basic Commands

Introduction to the Hadoop Ecosystem and HDFS

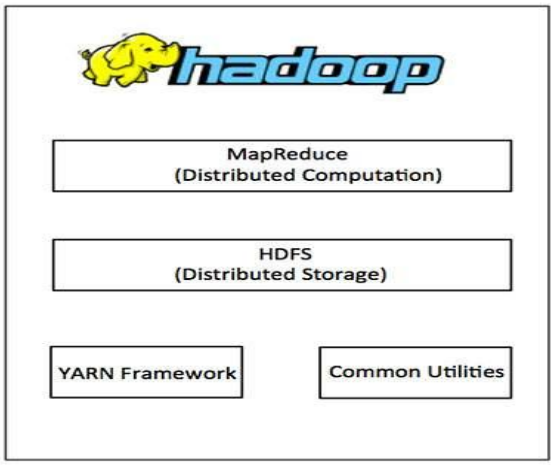
The Hadoop Ecosystem is a collection of tools and frameworks that work together to manage and process big data. HDFS (Hadoop Distributed File System) is the backbone of the Hadoop ecosystem, providing distributed storage and fault tolerance.

Hadoop Ecosystem Overview

The Hadoop Ecosystem is built around the core Hadoop components and includes tools for data storage, processing, querying, and analytics. Here's a breakdown:

Core Components:

1. **HDFS (Hadoop Distributed File System):**
   - Provides distributed storage.
   - Handles large files across multiple machines.
2. **YARN (Yet Another Resource Negotiator):**
   - Manages resources and job scheduling in the cluster.
3. **MapReduce:**
   - Programming model for data processing.



Supporting Tools:

1. **Apache Hive:**
    - Data warehouse tool for querying data in HDFS using SQL-like language (HiveQL).
2. **Apache HBase:**
    - NoSQL database for real-time read/write access to large datasets.
3. **Apache Spark:**
    - Fast, in-memory data processing engine.
4. **Apache Pig:**
    - High-level platform for creating MapReduce programs using a scripting language (Pig Latin).
5. **Apache Sqoop:**
    - Tool for transferring data between Hadoop and relational databases.
6. **Apache Flume:**
    - Tool for collecting and transferring large amounts of log data into HDFS.
7. **ZooKeeper:**
    - Centralized service for managing distributed systems.
8. **Oozie:**
    - Workflow scheduler for Hadoop jobs.

HDFS Basics

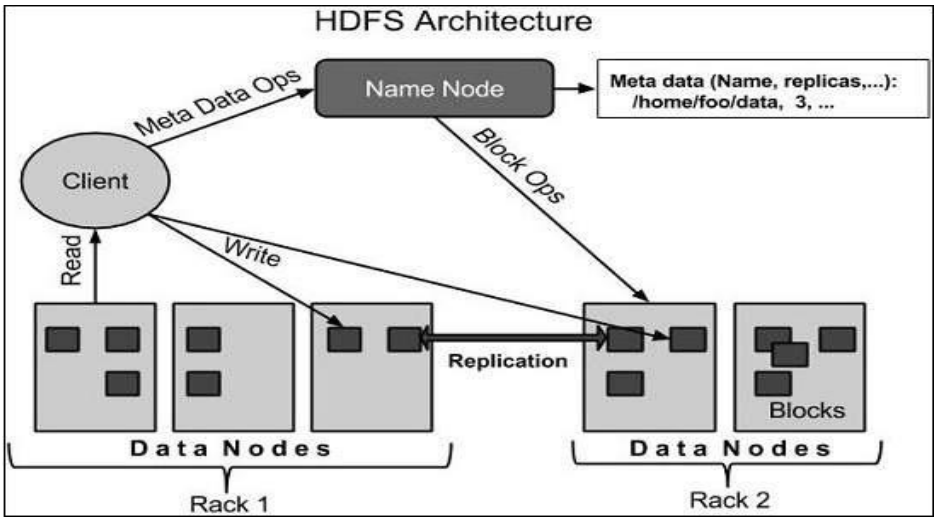
HDFS is the storage layer of Hadoop, designed for scalability and fault tolerance. It handles massive data volumes by breaking files into smaller chunks and distributing them across a cluster.

HDFS Features:

- Replication: Ensures fault tolerance by replicating data blocks (default: 3 copies).
- Write-Once, Read-Many: Optimized for batch processing.
- Scalability: Handles petabytes of data across thousands of nodes.

HDFS Architecture:

1. NameNode:
    - Maintains metadata (e.g., file structure, permissions).
    - Coordinates DataNodes but does not store data.
2. DataNodes:
    - Store actual data blocks.
    - Perform read/write operations as instructed by NameNode.
3. Secondary NameNode:
    - Periodically saves snapshots of NameNode metadata (not a backup).



Working with Hadoop:

- Open a new terminal and start the Hadoop service by following commands

```
start-dfs.sh
start-yarn.sh
```

This command initializes all the required Hadoop daemons for the cluster to become operational.

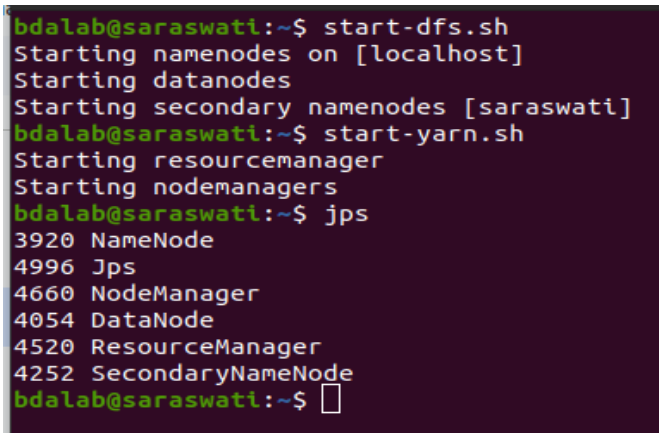
When you want to stop the services use the command:

```
stop-dfs.sh
stop-yarn.sh
```

- Check the Status of Hadoop Services (To confirm that all necessary Hadoop services are up and running, you can check their status using the 'jps' command)

```
jps
```

The 'jps' command displays the Java Virtual Machine (JVM) processes and should show a list of Hadoop services running, such as the NameNode, DataNode, ResourceManager, and NodeManager.



- **Accessing Hadoop Namenode and Resource Manager:**

  To access the Hadoop Namenode, open a web browser and enter the following URL:

  http://localhost:9870

  Similarly, to access the Hadoop Resource Manager, open a web browser and enter the following URL:

  http://localhost:8088

- **Verifying the Hadoop Cluster:**

  You can check the Hadoop version by following command

  ```
  hadoop version
  ```

  Create Directories in HDFS:

  ```
  hdfs dfs -mkdir /user/dir_name
  ```

  List Directories in HDFS:

  ```
  hdfs dfs -ls /
  hdfs dfs -ls /user/
  ```

  Transfer Files to the Hadoop File System:

  ```
  hdfs dfs -put source-filename /destination_folder/
  ```

Week-1 Exercise:

1. Practice Linux commands (No need to write in the record book)
    a. What command would you use to display the current directory?
    b. How do you list all files, including hidden ones, in a directory?
    c. Which command is used to create a new directory?
    d. How can you check the currently logged-in user?
    e. What does the pwd command do?
    f. How do you copy a file from one location to another?
    g. What is the command to move a file?
    h. How can you delete a directory and all its contents?
    i. How do you rename a file in Linux?
    j. What is the difference between `rm` and `rmdir`?
    k. What command is used to change the permissions of a file?
    l. How can you view the permissions of a file?
    m. Which command changes the owner of a file?
    n. How do you add execute permission for the owner of a file?
    o. What is the command to display all currently running processes?
    p. How can you check the IP address of your system?
    q. How do you display the contents of a file?
    r. How can you display the current date and time in Linux?

2. Explore Basic HDFS Commands
    a. Remove directory
    b. View/Read File Contents
    c. Download/copy a file from HDFS to the local system
    d. Copy/move files within HDFS
    e. Remove file from HDFS
    f. View file permission

Week 2: Advanced HDFS Commands and MapReduce

Week 2 Exercises (Advanced HDFS Commands):

Create a directory in Hadoop (/user/your_reg_no/lab2). Use the same directory to keep all your files.

hdfs dfs -mkdir /user/your_reg_no/lab2

(a).

1. Upload a new file (Ex. File1.txt) into Hadoop. Demonstrate the hdfs commands for the following:
    a. Check the permission of the file
    b. Change the permission of the file.
    c. Check ownership of the file
2. Upload a large file (Ex: largefile.txt) into Hadoop. Demonstrate the hdfs commands for following.
    a. Check block details of the file
    b. View the current replication factor for file
    c. Modify the replication factor of the file
3. Demonstrate hdfs commands for the following tasks
    a. Check directory size for the directory you have created (Disk usage analysis)
    b. Find the total capacity and usage of HDFS
4. Upload a text file with sample data to the Hadoop (Ex: analytics.txt). Using HDFS commands, find
    a. Number of lines in the file
    b. Number of occurrences of a specific word (Ex. Hadoop) in the file.

(b). Executing sample MapReduce jobs:

What is Hadoop MapReduce?

Hadoop MapReduce is a programming model and processing framework for processing large datasets in a distributed environment. It divides the job into smaller tasks, processes them in parallel, and consolidates the results.

Key Components

1. **Map Phase:**
    - Processes input data and generates intermediate key-value pairs.
    - Example: Counting words in a document; the mapper emits each word as a key and the count 1 as a value.
2. **Shuffle and Sort Phase:**
    - Intermediate key-value pairs are grouped and sorted by key.
    - Ensures all values for a specific key are brought together.
3. **Reduce Phase:**
    - Processes grouped data from the shuffle phase to generate final output.
    - Example: Aggregating word counts.

MapReduce Workflow

1. **Input Data:**
    - Stored in HDFS and divided into splits.
2. **Mapper:**
    - Processes each split and outputs key-value pairs.
3. **Shuffle and Sort:**
    - Intermediate outputs are shuffled and sorted by the framework.
4. **Reducer:**
    - Processes sorted key-value pairs and generates the final result.
5. **Output:**
    - Written back to HDFS.

Run bellow commands to install python

wget https://www.python.org/ftp/python/3.6.5/Python-3.6.5.tgz
tar -xvf Python-3.6.5.tgz

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
```

1. **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 '/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.4.1.jar' -file mapper.py -mapper mapper.py -file reducer.py -reducer reducer.py -input /user/input.txt -output /user/out1

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.