

## ASSIGNMENT 1

**PROBLEM STATEMENT:** Design a distributed application using RPC for remote computation where client submits an integer value to the server and server calculates factorial and returns the result to the client program.

### THOERY:

#### Remote Procedure Call (RPC):

Remote Procedure Call is a protocol that allows a program to request a service from a program located on another computer in a network. It abstracts the complexities of the network communication and enables programmers to write distributed software as if all operations were local.

The main goals of RPC are:

- **Transparency:** Allow remote procedure calls to appear as local calls, hiding the complexity of network communication.
- **Simplicity:** Simplify the development of distributed applications by abstracting low-level networking details.
- **Interoperability:** Enable communication between different systems and platforms through standardized protocols and data formats.

#### How RPC Works

RPC works in the following manner:

1. **Client Stub:** The client-side code that prepares the procedure call by packing the procedure name and parameters into a message.
2. **Transport Layer:** The message is sent over the network to the server.
3. **Server Stub:** The server-side code receives the request, unpacks the message, and calls the actual procedure.
4. **Execution:** The procedure is executed on the server.
5. **Response:** The result is packed into a response message and sent back to the client.

#### Application of RPC to compute Factorial of a Number:

The Remote Procedure Call (RPC) mechanism can be effectively used to compute the factorial of a number in a distributed application:

##### 1. Client-Server Model:

The application follows a client-server architecture. The client accepts an integer input from the user and sends it to the server using RPC. The server receives the request, computes the factorial of the number, and sends the result back to the client.

##### 2. Remote Computation:

The factorial computation is delegated to a remote server. This offloads the computational task from the client, making the system more scalable and modular. The client doesn't need to know how the computation is done—it just gets the result.

### **3. Real-Time Result Delivery**

RPC enables real-time invocation of the factorial function on the server. The response time is minimal, and results are returned instantly, enhancing user experience.

### **Components of RPC Application:**

#### **1.Client**

The client is responsible for:

- Accepting user input (an integer).
- Initiating an RPC to request factorial computation from the server.
- Receiving and displaying the result.

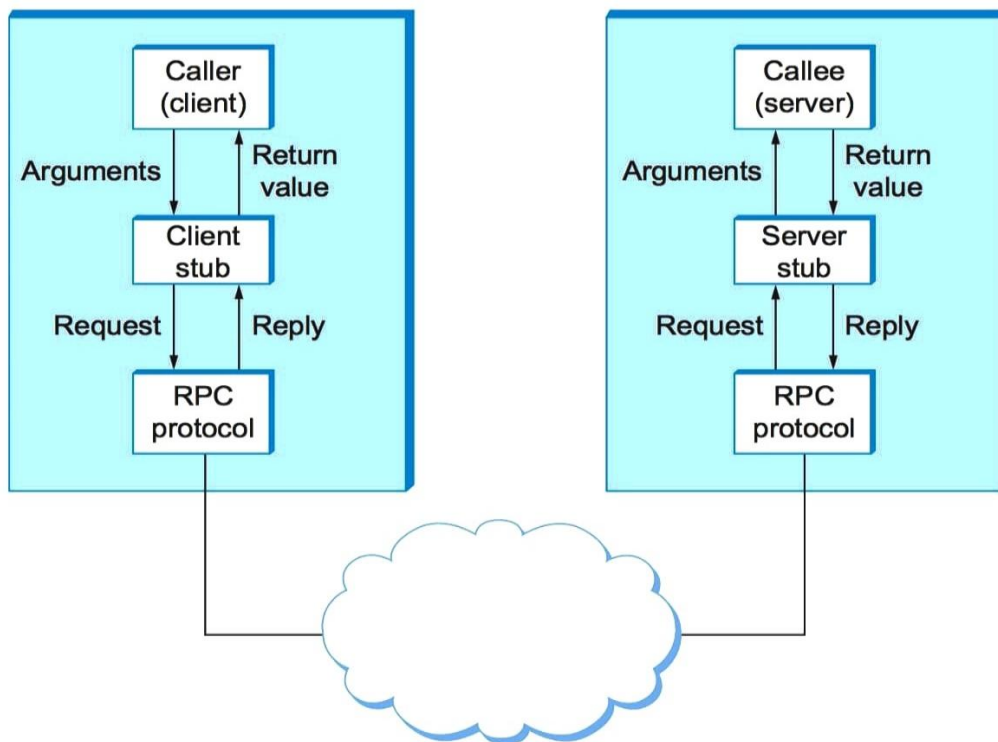
#### **2.Server**

The server:

- Listens for incoming RPC requests.
- Extracts the integer from the request.
- Computes the factorial using either an iterative or recursive method.
- Sends the result back to the client.

#### **3.RPC Stubs**

Stubs are auto-generated or manually written code that handles marshalling (converting data into a transmittable format) and unmarshalling between the client and server. The stubs hide the underlying communication protocols from the developer.



## Remote Procedure Call (RPC)

### Factorial of a Number:

The factorial of a non-negative integer  $n$  is the product of all positive integers less than or equal to  $n$ . It is denoted as  $n!$ .

### Advantages of RPC:

- **Transparency:** RPC makes remote calls appear like local ones, simplifying distributed program development greatly.
- **Simplified Communication:** Hides complex network protocols, enabling developers to focus on application logic, not data transfer.
- **Code Reusability:** Server-side functions can be reused by multiple clients without duplicating computation logic elsewhere.
- **Platform Independence:** RPC allows communication across different platforms and languages using standard data serialization formats.
- **Modularity:** Supports separation of concerns by dividing applications into independent client and server modules.

## Limitations of RPC:

- **Network Dependency:** Requires stable, fast network; performance is affected by latency, bandwidth, and packet loss.
- **Limited Error Handling:** Errors like timeouts or network failures may be hard to debug and recover from gracefully.
- **Security Issues:** RPC lacks built-in encryption; secure implementations require extra effort for authentication and data protection.
- **Complex Deployment:** Properly setting up and maintaining client-server RPC systems requires more configuration and monitoring effort.
- **Platform Compatibility Issues:** Different data formats or architectures between client and server may cause serialization/deserialization errors.

## CODE:

### Client:

```
import xmlrpc.client

proxy = xmlrpc.client.ServerProxy("http://localhost:8000/")

while True:
    user_input = input("Enter a number to compute its factorial (or type 'exit' to quit): ").strip()

    if user_input.lower() == 'exit':
        try:
            print(proxy.shutdown())
        except:
            print("Server already shut down.")
        break

    try:
        num = int(user_input)
        result = proxy.factorial(num)
        print(f"Factorial of {num} is {result}")
    except ValueError:
        print("Please enter a valid integer or 'exit'.")
```

### Server:

```
from xmlrpc.server import SimpleXMLRPCServer
import threading

def factorial(n):
    if n == 0 or n == 1:
```

```

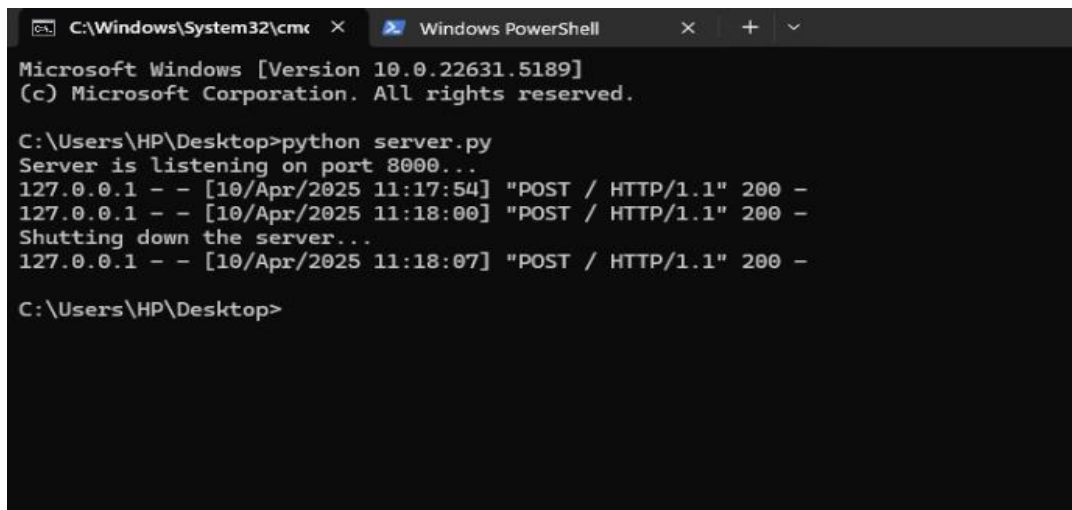
        return 1
    else:
        return n * factorial(n - 1)

def shutdown():
    print("Shutting down the server...")
    threading.Thread(target=server.shutdown).start()
    return "Server is shutting down."

server = SimpleXMLRPCServer(("localhost", 8000), allow_none=True)
print("Server is listening on port 8000...")
server.register_function(factorial, "factorial")
server.register_function(shutdown, "shutdown")
server.serve_forever()

```

## OUTPUT:



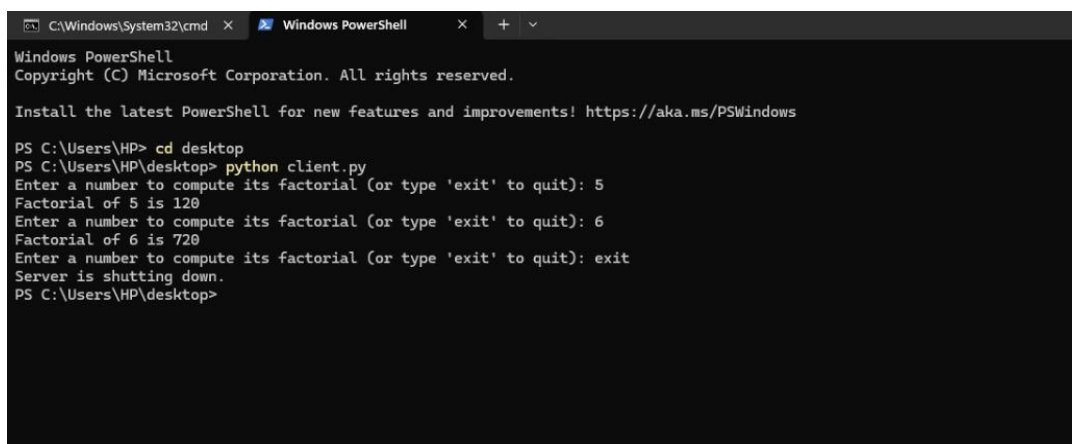
```

C:\Windows\System32\cmd X Windows PowerShell X + v
Microsoft Windows [Version 10.0.22631.5189]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP\Desktop>python server.py
Server is listening on port 8000...
127.0.0.1 - - [10/Apr/2025 11:17:54] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [10/Apr/2025 11:18:00] "POST / HTTP/1.1" 200 -
Shutting down the server...
127.0.0.1 - - [10/Apr/2025 11:18:07] "POST / HTTP/1.1" 200 -

C:\Users\HP\Desktop>

```



```

C:\Windows\System32\cmd X Windows PowerShell X + v
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\HP> cd desktop
PS C:\Users\HP\Desktop> python client.py
Enter a number to compute its factorial (or type 'exit' to quit): 5
Factorial of 5 is 120
Enter a number to compute its factorial (or type 'exit' to quit): 6
Factorial of 6 is 720
Enter a number to compute its factorial (or type 'exit' to quit): exit
Server is shutting down.
PS C:\Users\HP\Desktop>

```

## **CONCLUSION:**

The implementation of Remote Procedure Call (RPC) for computing the factorial of an integer successfully demonstrates the effectiveness of distributed computing in simplifying complex client-server interactions. By offloading the computation to a server, the system maintains a clear separation of responsibilities while enhancing performance and scalability. The client-server communication is abstracted to resemble local function calls, showcasing RPC's transparency and ease of use. The result is returned seamlessly, maintaining the accuracy and reliability of the output. This assignment highlights how RPC can be utilized to build efficient, modular, and maintainable distributed applications.

## ASSIGNMENT 2

**PROBLEM STATEMENT:** Design a distributed application using RMI for remote computation where client submits two strings to the server and server returns the concatenation of the given strings.

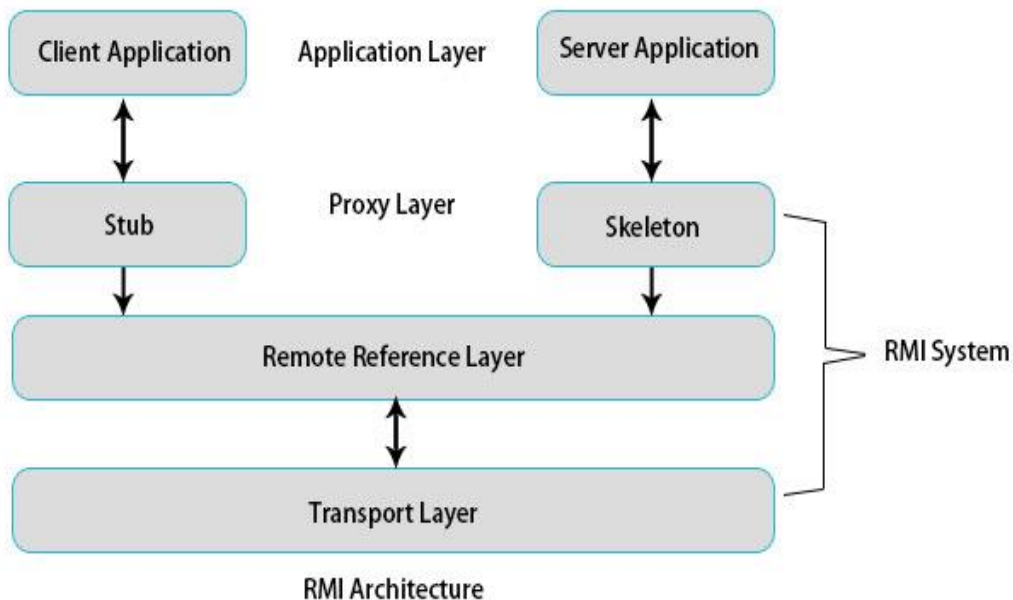
### THOERY:

#### Remote Method Invocation

Remote Method Invocation (RMI) is a Java API that allows an object residing in one Java Virtual Machine (JVM) to invoke methods on an object running in another JVM. It is Java's native solution for building distributed applications using object-oriented concepts.

RMI simplifies the process of calling remote methods by:

- Hiding the complexity of network communication.
- Allowing developers to focus on business logic.
- Providing a seamless way to invoke remote objects just like local ones.



#### Components of RMI

RMI architecture is composed of the following core components:

1. **Remote Interface:** A Java interface that declares the methods to be called remotely. It extends the `java.rmi.Remote` interface.

2. **Remote Object Implementation:** A class that implements the remote interface and provides the actual logic. It must extend `UnicastRemoteObject` and handle exceptions such as `RemoteException`.
3. **RMI Registry:** A simple server-side name service that allows clients to look up remote objects by name.
4. **Client:** The client program looks up the remote object from the registry and calls its methods as if it were local.

### Applications of RMI:

Java RMI (Remote Method Invocation) is a powerful mechanism that allows objects to invoke methods located on remote systems as if they were local. This simplifies the process of building distributed applications in Java.

1. **Distributed Applications:** RMI is widely used to build distributed systems where different components of an application are located on different machines. It allows seamless method calls between objects residing on different JVMs (Java Virtual Machines).
2. **Remote File Sharing Systems:** Using RMI, clients can request files stored on a remote server. The server reads the file and sends it back to the client, enabling centralized storage and controlled access.
3. **Remote Database Access:** Java RMI is also used in applications where remote clients need access to a central database. The database operations are implemented on the server-side, and clients use remote interfaces to perform actions like insert, delete, update, or retrieve data.
4. **Chat Applications and Messaging Services:** RMI is ideal for creating real-time chat systems where messages are passed between clients via a central server. Each client registers with the server and receives messages using remote method calls.

### String Concatenation:

String concatenation is the process of joining two or more strings together to form a single string. It is a fundamental operation in many programming languages and plays an important role in text processing, data formatting, message building, and user interface creation.

### Advantages of RMI:

- **Object-Oriented Communication:** Enables method calls between objects across JVMs, maintaining object-oriented design principles.
- **Simplifies Distributed Programming:** Hides network communication details, making distributed development easier and cleaner for developers.
- **Automatic Stub/Skeleton Generation:** Tools like `rmic` auto-generate network communication code, reducing manual effort and errors.



- **Strong Typing with Interfaces:** Uses Java interfaces to ensure type safety and consistency between client and server.
- **Built-in Security Support:** Supports security managers and policies, enhancing the safety of remote method invocations.

### Limitations of RMI:

- **Java-only Communication:** Works only with Java, limiting interoperability with applications written in other languages.
- **Tightly Coupled Systems:** Both client and server must have shared interfaces, leading to tighter integration and dependencies.
- **Performance Overhead:** Serialization and network latency can make RMI slower compared to local method calls.
- **Complex Debugging:** Debugging distributed systems is more challenging due to remote execution and network issues.
- **Requires Java Runtimes Everywhere:** Every participating machine must have compatible Java runtimes and RMI configurations installed.

### CODE:

#### Service:

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface ConcatService extends Remote {
    String concatenate(String str1, String str2) throws RemoteException;
}
```

#### Server:

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class ConcatServer extends UnicastRemoteObject implements ConcatService {
    protected ConcatServer() throws RemoteException {
        super();
    }

    @Override
    public String concatenate(String str1, String str2) throws RemoteException {
        return str1 + str2;
    }
}
```

```

public static void main(String[] args) {
    try {
        ConcatServer server = new ConcatServer();
        Naming.rebind("rmi://localhost/ConcatService", server);
        System.out.println("Server is running...");
    } catch (Exception e) {
        System.err.println("Server exception: " + e.getMessage());
        e.printStackTrace();
    }
}
}

```

## Client:

```

import java.rmi.Naming;
import java.util.Scanner;

public class ConcatClient {
    public static void main(String[] args) {
        try {
            ConcatService service = (ConcatService)
Naming.lookup("rmi://localhost/ConcatService");
            Scanner scanner = new Scanner(System.in);

            while (true) {
                System.out.println("Menu:");
                System.out.println("1. Concatenate Strings");
                System.out.println("2. Exit");
                System.out.print("Enter choice: ");

                int choice = scanner.nextInt();
                scanner.nextLine(); // Consume newline

                if (choice == 1) {
                    System.out.print("Enter first string: ");
                    String str1 = scanner.nextLine();

                    System.out.print("Enter second string: ");
                    String str2 = scanner.nextLine();

                    String result = service.concatenate(str1, str2);
                    System.out.println("Concatenated Result: " + result);
                } else if (choice == 2) {
                    System.out.println("Exiting...");
                    break;
                } else {
                    System.out.println("Invalid choice. Please try again.");
                }
            }

            scanner.close();
        } catch (Exception e) {
            System.err.println("Client exception: " + e.getMessage());
            e.printStackTrace();
        }
    }
}

```

```
}  
}  
}
```

## OUTPUT:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

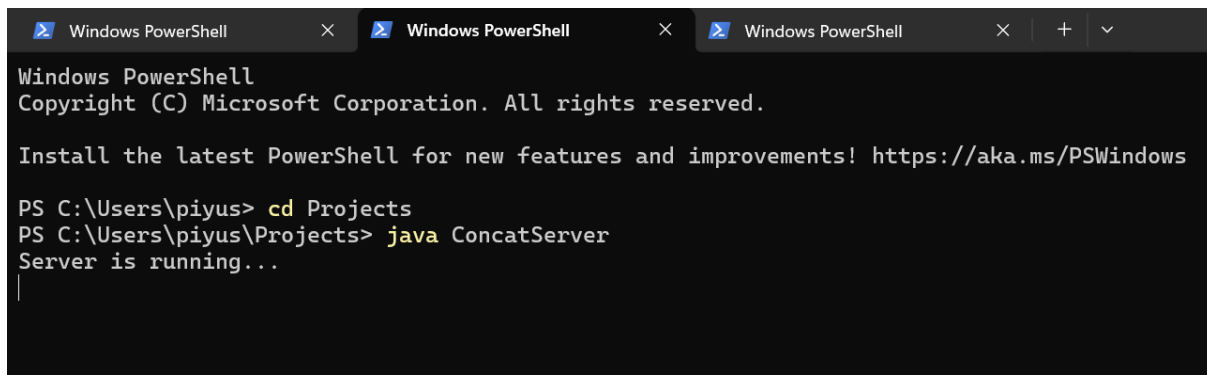
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\piyus> cd Projects
PS C:\Users\piyus\Projects> javac *.java
PS C:\Users\piyus\Projects> start "C:\Program Files\Java\jdk-24\bin\rmiregistry.exe"
PS C:\Users\piyus\Projects> |
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\piyus> cd Projects
PS C:\Users\piyus\Projects> java ConcatClient
Menu:
1. Concatenate Strings
2. Exit
Enter choice: 1
Enter first string: Hello
Enter second string: World
Concatenated Result: HelloWorld
Menu:
1. Concatenate Strings
2. Exit
Enter choice: 2
Exiting...
PS C:\Users\piyus\Projects>
```

A screenshot of a Windows PowerShell terminal window. The window has three tabs, all labeled 'Windows PowerShell'. The terminal text is as follows:  
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
Install the latest PowerShell for new features and improvements! <https://aka.ms/PSWindows>  
  
PS C:\Users\piyus> cd Projects  
PS C:\Users\piyus\Projects> java ConcatServer  
Server is running...  
|

## CONCLUSION:

The application of Java RMI for remote string concatenation demonstrates the ability to perform distributed operations in an object-oriented manner. By allowing the client to invoke methods on a remote server seamlessly, RMI abstracts the complexity of network communication. The task of string concatenation, though simple, effectively showcases the core concepts of RMI such as remote object access, interface-based communication, and object serialization. The system also highlights the scalability and reusability of distributed applications when designed with RMI. Overall, the assignment provides valuable insights into building and deploying Java-based distributed applications, reinforcing key principles of remote computing.

## ASSIGNMENT 3

**PROBLEM STATEMENT:** Design a distributed application using MapReduce under Hadoop for: a) Character counting in a given text file. b) Counting no. of occurrences of every word in a given text file.

### THEORY:

#### Introduction

In today's data-driven world, organizations generate and store massive volumes of data from various sources like social media, sensors, transactions, and logs. Analyzing such large-scale datasets using traditional single-machine systems is inefficient and often impossible due to limitations in storage and processing power. To tackle these challenges, frameworks like Apache Hadoop were introduced. Hadoop allows for the distributed storage and parallel processing of large datasets across a cluster of commodity hardware. One of the core components of Hadoop is MapReduce, a programming model and processing engine that enables parallel computation of data by dividing tasks into two fundamental phases: Map and Reduce. This practical is a classic example of using the MapReduce model to perform a word count operation, where the program scans a large text file and counts the frequency of each word. It is often considered the "Hello World" of MapReduce because it demonstrates the power of distributed processing through a simple and understandable task.

#### Hadoop Streaming

While Hadoop is typically used with Java, Hadoop Streaming is a powerful utility that allows users to write Mapper and Reducer code in any language that can read from standard input and write to standard output, such as Python, Perl, Bash, or Ruby. This enables greater flexibility and rapid development. In this practical, we use Python for its simplicity and readability.

#### A) Working of Word Count using MapReduce

The process follows these four key steps:

**1. Input File (HDFS):** The input to the MapReduce job is a plain text file stored in Hadoop Distributed File System (HDFS). This file is split into blocks and distributed across the cluster for parallel processing.

#### 2. Mapper Phase

The Mapper reads the input line by line. For each line, it performs the following:

- Splits the line into words based on whitespace and punctuation.
- For each word, emits a key-value pair in the form of: (word, 1)

### **Example**

#### **Input to Mapper:**

hello hadoop

hadoop program

hadoop world

hello world

#### **Output from Mapper:**

("hello", 1)

("hadoop", 1)

("hadoop", 1)

("program", 1)

("hadoop", 1)

("world", 1)

("hello", 1)

("world", 1)

### **3. Shuffle and Sort (Handled by Hadoop Framework)**

This is an internal and automatic phase managed by Hadoop.

- All intermediate key-value pairs are shuffled, meaning they are sent to the appropriate reducer based on the key.
- Hadoop then groups the values by key, which means it collects all values (1s) for the same word together.

("hadoop", [1, 1, 1])

("hello", [1, 1])

("program", [1])

("world", [1, 1])

## 4.Reducer Phase

The Reducer receives each unique key (word) and a list of its values (counts). It then:

- Sums up the values to get the total occurrences of each word.
- Emits the final key-value pair as: (word, total\_count)

**hadoop    3**

**hello     2**

**program   1**

**world     2**

This is the final output, which shows how many times each word appeared in the input file.

## B) Working of Character Count using MapReduce

**1. Input File (HDFS):** The input to the MapReduce job is a plain text file stored in the Hadoop Distributed File System (HDFS). This file is split into blocks and distributed across the cluster for parallel processing.

### 2. Mapper Phase

The Mapper reads the input line by line. For each line, it performs the following:

- Splits the line into characters.
- For each character, emits a key-value pair in the form of: (character, 1)

### Example

#### Input to Mapper:

abbc

ccba

aacb

#### Output from Mapper:

("a", 1), ("b", 1), ("b", 1), ("c", 1)

("c", 1), ("c", 1), ("b", 1), ("a", 1)

("a", 1), ("a", 1), ("c", 1), ("b", 1)

### 3. Shuffle and Sort (Handled by Hadoop Framework)

This is an internal and automatic phase managed by Hadoop:

- All intermediate key-value pairs are shuffled, meaning they are sent to the appropriate reducer based on the key (character).
- Hadoop then groups the values by key, collecting all 1s for the same character together.

("a", [1, 1, 1, 1])

("b", [1, 1, 1, 1])

("c", [1, 1, 1])

### 4.Reducer Phase

The Reducer receives each unique key (character) and a list of its values (counts). It then:

- Sums up the values to get the total occurrences of each character.
- Emits the final key-value pair as: (**character**, **total\_count**)

#### Final Output:

a 4

b 4

c 3

This is the final output, which shows how many times each word appeared in the input file.

### A) CODE: Word Count

#### 1. Open Terminal and switch to Hadoop user

```
pvg@pvg-HP-ProDesk-400-G4-SFF:~$ su hduser
Password:
hduser@pvg-HP-ProDesk-400-G4-SFF:/home/pvg$ cd
```

#### 2. Create a text file to count words



```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ nano word_count.txt
```

#word\_count.txt file will open in nano text editor

#Add your text in this file

#Press Ctrl + X

#Press Y

#Press Enter key

### 3. Start HDFS

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ start-dfs.sh
```

Starting namenodes on [localhost]

Starting datanodes

Starting secondary namenodes [pvg-HP-ProDesk-400-G4-SFF]

2025-04-21 14:33:31,053 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ start-yarn.sh
```

Starting resourcemanager

Starting nodemanagers

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ jps
```

8275 Jps

7894 NodeManager

7563 SecondaryNameNode

7197 NameNode

7325 DataNode

7774 ResourceManager

### 4. Create an input directory and upload your file to HDFS:

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -ls /
```

2025-04-21 14:34:37,504 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

```
Found 3 items
drwxr-xr-x - hduser supergroup          0 2025-04-21 13:53 /input
drwxr-xr-x - hduser supergroup
drwxr-xr-x - hduser supergroup
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -rm -r /input
2025-04-21 14:35:47,488 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Deleted /input
```

**#Similarly, delete any previous output files if present using: `hdfs dfs -rm -r /output`**

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -mkdir -p /input
2025-04-21 14:35:55,794 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -ls /
2025-04-21 14:36:06,292 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Found 3 items
drwxr-xr-x - hduser supergroup          0 2025-04-21 14:35 /input
drwxr-xr-x - hduser supergroup
drwxr-xr-x - hduser supergroup
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -put word_count.txt /input/
2025-04-21 14:36:44,153 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -ls /input/
2025-04-21 14:36:55,653 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--  1 hduser supergroup          53 2025-04-21 14:36
/input/word_count.txt
```

## 5. Run the word count program:

**#we are using the inbuilt WordCount MapReduce program provided in the `hadoop-mapreduce-examples.jar` file**

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ whereis hadoop
hadoop: /usr/local/hadoop /usr/local/hadoop/bin/hadoop.cmd
/usr/local/hadoop/bin/Hadoop
```

**#copy the path /usr/local/Hadoop and rest of the path is in the following command**

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hadoop jar
/usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.3.4.jar
wordcount /input /output
2025-04-21 14:42:53,757 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
2025-04-21 14:42:54,199 INFO impl.MetricsConfig: Loaded properties from hadoop-
metrics2.properties
2025-04-21 14:42:54,270 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot
period at 10 second(s).
2025-04-21 14:42:54,270 INFO impl.MetricsSystemImpl: JobTracker metrics system
started
2025-04-21 14:42:54,486 INFO input.FileInputFormat: Total input files to
process : 1
2025-04-21 14:42:54,529 INFO mapreduce.JobSubmitter: number of splits:1
2025-04-21 14:42:54,671 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_local1306156686_0001
2025-04-21 14:42:54,671 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-04-21 14:42:54,785 INFO mapreduce.Job: The url to track the job:
http://localhost:8080/
2025-04-21 14:42:54,786 INFO mapreduce.Job: Running job: job_local1306156686_0001
2025-04-21 14:42:54,787 INFO mapred.LocalJobRunner: OutputCommitter set in config
null
2025-04-21 14:42:54,795 INFO output.FileOutputCommitter: File Output Committer
Algorithm version is 2
2025-04-21 14:42:54,796 INFO output.FileOutputCommitter: FileOutputCommitter skip
cleanup _temporary folders under output directory:false, ignore cleanup failures:
false
2025-04-21 14:42:54,797 INFO mapred.LocalJobRunner: OutputCommitter is
org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2025-04-21 14:42:54,835 INFO mapred.LocalJobRunner: Waiting for map tasks
2025-04-21 14:42:54,836 INFO mapred.LocalJobRunner: Starting task:
attempt_local1306156686_0001_m_000000_0
2025-04-21 14:42:54,857 INFO output.FileOutputCommitter: File Output Committer
Algorithm version is 2
2025-04-21 14:42:54,857 INFO output.FileOutputCommitter: FileOutputCommitter skip
cleanup _temporary folders under output directory:false, ignore cleanup failures:
false
2025-04-21 14:42:54,872 INFO mapred.Task: Using ResourceCalculatorProcessTree :
[ ]
2025-04-21 14:42:54,875 INFO mapred.MapTask: Processing split:
hdfs://localhost:54310/input/word_count.txt:0+53
2025-04-21 14:42:54,918 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2025-04-21 14:42:54,918 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2025-04-21 14:42:54,918 INFO mapred.MapTask: soft limit at 83886080
```

2025-04-21 14:42:54,918 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600  
2025-04-21 14:42:54,918 INFO mapred.MapTask: kvstart = 26214396; length = 6553600  
2025-04-21 14:42:54,921 INFO mapred.MapTask: Map output collector class =  
org.apache.hadoop.mapred.MapTask\$MapOutputBuffer  
2025-04-21 14:42:55,007 INFO mapred.LocalJobRunner:  
2025-04-21 14:42:55,008 INFO mapred.MapTask: Starting flush of map output  
2025-04-21 14:42:55,008 INFO mapred.MapTask: Spilling map output  
2025-04-21 14:42:55,008 INFO mapred.MapTask: bufstart = 0; bufend = 85; bufvoid =  
104857600  
2025-04-21 14:42:55,008 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend  
= 26214368(104857472); length = 29/6553600  
2025-04-21 14:42:55,023 INFO mapred.MapTask: Finished spill 0  
2025-04-21 14:42:55,031 INFO mapred.Task:  
Task:attempt\_local1306156686\_0001\_m\_000000\_0 is done. And is in the process of  
committing  
2025-04-21 14:42:55,035 INFO mapred.LocalJobRunner: map  
2025-04-21 14:42:55,035 INFO mapred.Task: Task  
'attempt\_local1306156686\_0001\_m\_000000\_0' done.  
2025-04-21 14:42:55,039 INFO mapred.Task: Final Counters for  
attempt\_local1306156686\_0001\_m\_000000\_0: Counters: 24

#### File System Counters

FILE: Number of bytes read=281168  
FILE: Number of bytes written=922756  
FILE: Number of read operations=0  
FILE: Number of large read operations=0  
FILE: Number of write operations=0  
HDFS: Number of bytes read=53  
HDFS: Number of bytes written=0  
HDFS: Number of read operations=5  
HDFS: Number of large read operations=0  
HDFS: Number of write operations=1  
HDFS: Number of bytes read erasure-coded=0

#### Map-Reduce Framework

Map input records=4  
Map output records=8  
Map output bytes=85  
Map output materialized bytes=57  
Input split bytes=108  
Combine input records=8  
Combine output records=4  
Spilled Records=4  
Failed Shuffles=0  
Merged Map outputs=0  
GC time elapsed (ms)=8  
Total committed heap usage (bytes)=195035136

#### File Input Format Counters

Bytes Read=53

2025-04-21 14:42:55,039 INFO mapred.LocalJobRunner: Finishing task:  
attempt\_local1306156686\_0001\_m\_000000\_0  
2025-04-21 14:42:55,039 INFO mapred.LocalJobRunner: map task executor complete.  
2025-04-21 14:42:55,042 INFO mapred.LocalJobRunner: Waiting for reduce tasks  
2025-04-21 14:42:55,042 INFO mapred.LocalJobRunner: Starting task:  
attempt\_local1306156686\_0001\_r\_000000\_0  
2025-04-21 14:42:55,047 INFO output.FileOutputCommitter: File Output Committer  
Algorithm version is 2  
2025-04-21 14:42:55,047 INFO output.FileOutputCommitter: FileOutputCommitter skip  
cleanup \_temporary folders under output directory:false, ignore cleanup failures:  
false  
2025-04-21 14:42:55,047 INFO mapred.Task: Using ResourceCalculatorProcessTree :  
[ ]  
2025-04-21 14:42:55,049 INFO mapred.ReduceTask: Using ShuffleConsumerPlugin:  
org.apache.hadoop.mapreduce.task.reduce.Shuffle@cf9471c  
2025-04-21 14:42:55,050 WARN impl.MetricsSystemImpl: JobTracker metrics system  
already initialized!  
2025-04-21 14:42:55,062 INFO reduce.MergeManagerImpl: MergerManager:  
memoryLimit=1437178240, maxSingleShuffleLimit=359294560, mergeThreshold=948537664,  
ioSortFactor=10, memToMemMergeOutputsThreshold=10  
2025-04-21 14:42:55,063 INFO reduce.EventFetcher:  
attempt\_local1306156686\_0001\_r\_000000\_0 Thread started: EventFetcher for fetching  
Map Completion Events  
2025-04-21 14:42:55,083 INFO reduce.LocalFetcher: localfetcher#1 about to shuffle  
output of map attempt\_local1306156686\_0001\_m\_000000\_0 decomp: 53 len: 57 to  
MEMORY  
2025-04-21 14:42:55,085 INFO reduce.InMemoryMapOutput: Read 53 bytes from map-  
output for attempt\_local1306156686\_0001\_m\_000000\_0  
2025-04-21 14:42:55,086 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-  
output of size: 53, inMemoryMapOutputs.size() -> 1, commitMemory -> 0, usedMemory  
->53  
2025-04-21 14:42:55,087 INFO reduce.EventFetcher: EventFetcher is interrupted..  
Returning  
2025-04-21 14:42:55,088 INFO mapred.LocalJobRunner: 1 / 1 copied.  
2025-04-21 14:42:55,088 INFO reduce.MergeManagerImpl: finalMerge called with 1  
in-memory map-outputs and 0 on-disk map-outputs  
2025-04-21 14:42:55,094 INFO mapred.Merger: Merging 1 sorted segments  
2025-04-21 14:42:55,094 INFO mapred.Merger: Down to the last merge-pass, with 1  
segments left of total size: 44 bytes  
2025-04-21 14:42:55,098 INFO reduce.MergeManagerImpl: Merged 1 segments, 53 bytes  
to disk to satisfy reduce memory limit  
2025-04-21 14:42:55,099 INFO reduce.MergeManagerImpl: Merging 1 files, 57 bytes  
from disk  
2025-04-21 14:42:55,099 INFO reduce.MergeManagerImpl: Merging 0 segments, 0 bytes  
from memory into reduce  
2025-04-21 14:42:55,099 INFO mapred.Merger: Merging 1 sorted segments

2025-04-21 14:42:55,100 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 44 bytes  
2025-04-21 14:42:55,100 INFO mapred.LocalJobRunner: 1 / 1 copied.  
2025-04-21 14:42:55,116 INFO Configuration.deprecation: mapred.skip.on is deprecated. Instead, use mapreduce.job.skiprecords  
2025-04-21 14:42:55,172 INFO mapred.Task:  
Task:attempt\_local1306156686\_0001\_r\_000000\_0 is done. And is in the process of committing  
2025-04-21 14:42:55,175 INFO mapred.LocalJobRunner: 1 / 1 copied.  
2025-04-21 14:42:55,175 INFO mapred.Task: Task  
attempt\_local1306156686\_0001\_r\_000000\_0 is allowed to commit now  
2025-04-21 14:42:55,189 INFO output.FileOutputCommitter: Saved output of task  
'attempt\_local1306156686\_0001\_r\_000000\_0' to hdfs://localhost:54310/output  
2025-04-21 14:42:55,190 INFO mapred.LocalJobRunner: reduce > reduce  
2025-04-21 14:42:55,190 INFO mapred.Task: Task  
'attempt\_local1306156686\_0001\_r\_000000\_0' done.  
2025-04-21 14:42:55,190 INFO mapred.Task: Final Counters for  
attempt\_local1306156686\_0001\_r\_000000\_0: Counters: 30

#### File System Counters

FILE: Number of bytes read=281314  
FILE: Number of bytes written=922813  
FILE: Number of read operations=0  
FILE: Number of large read operations=0  
FILE: Number of write operations=0  
HDFS: Number of bytes read=53  
HDFS: Number of bytes written=35  
HDFS: Number of read operations=10  
HDFS: Number of large read operations=0  
HDFS: Number of write operations=3  
HDFS: Number of bytes read erasure-coded=0

#### Map-Reduce Framework

Combine input records=0  
Combine output records=0  
Reduce input groups=4  
Reduce shuffle bytes=57  
Reduce input records=4  
Reduce output records=4  
Spilled Records=4  
Shuffled Maps =1  
Failed Shuffles=0  
Merged Map outputs=1  
GC time elapsed (ms)=0  
Total committed heap usage (bytes)=195035136

#### Shuffle Errors

BAD\_ID=0  
CONNECTION=0  
IO\_ERROR=0

```
        WRONG_LENGTH=0
        WRONG_MAP=0
        WRONG_REDUCE=0
    File Output Format Counters
        Bytes Written=35
2025-04-21 14:42:55,191 INFO mapred.LocalJobRunner: Finishing task:
attempt_local1306156686_0001_r_000000_0
2025-04-21 14:42:55,191 INFO mapred.LocalJobRunner: reduce task executor complete.
2025-04-21 14:42:55,790 INFO mapreduce.Job: Job job_local1306156686_0001 running
in uber mode : false
2025-04-21 14:42:55,791 INFO mapreduce.Job: map 100% reduce 100%
2025-04-21 14:42:55,794 INFO mapreduce.Job: Job job_local1306156686_0001
completed successfully
2025-04-21 14:42:55,800 INFO mapreduce.Job: Counters: 36
    File System Counters
        FILE: Number of bytes read=562482
        FILE: Number of bytes written=1845569
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=106
        HDFS: Number of bytes written=35
        HDFS: Number of read operations=15
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=4
        HDFS: Number of bytes read erasure-coded=0
    Map-Reduce Framework
        Map input records=4
        Map output records=8
        Map output bytes=85
        Map output materialized bytes=57
        Input split bytes=108
        Combine input records=8
        Combine output records=4
        Reduce input groups=4
        Reduce shuffle bytes=57
        Reduce input records=4
        Reduce output records=4
        Spilled Records=8
        Shuffled Maps =1
        Failed Shuffles=0
        Merged Map outputs=1
        GC time elapsed (ms)=8
        Total committed heap usage (bytes)=390070272
    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=53
File Output Format Counters
  Bytes Written=35
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -ls /
2025-04-21 14:43:23,963 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Found 2 items
drwxr-xr-x  - hduser supergroup          0 2025-04-21 14:36 /input
drwxr-xr-x  - hduser supergroup          0 2025-04-21 14:42 /output
```

## 6. View Output:

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -ls /output/
2025-04-21 14:43:31,859 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--  1 hduser supergroup          0 2025-04-21 14:42 /output/_SUCCESS
-rw-r--r--  1 hduser supergroup        35 2025-04-21 14:42 /output/part-r-00000
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -cat /output/part-r-00000
2025-04-21 14:43:51,634 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
hadoop    3
hello     2
program   1
world     2
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ stop-dfs.sh
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [Ubuntu]
2025-04-21 14:44:17,461 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ stop-yarn.sh
Stopping nodemanagers
Stopping resourcemanager
```



## B) CODE: Character Count

### mapper.py

```
#!/usr/bin/env python3
import sys
for line in sys.stdin:
    for char in line.strip():
        print(f"{char}\t1")
```

### reducer.py

```
#!/usr/bin/env python3
import sys
from collections import defaultdict
counts = defaultdict(int)
for line in sys.stdin:
    key, val = line.strip().split("\t")
    counts[key] += int(val)
for key in sorted(counts):
    print(f"{key}\t{counts[key]}")
```

## 1. Open Terminal and switch to Hadoop user

```
pvg@pvg-HP-ProDesk-400-G4-SFF:~$ su hduser
Password:
hduser@pvg-HP-ProDesk-400-G4-SFF:/home/pvg$ cd
```

## 2. Start HDFS

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [pvg-HP-ProDesk-400-G4-SFF]
2025-04-23 15:07:21,649 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ jps
12466 NameNode
12871 SecondaryNameNode
13243 NodeManager
12652 DataNode
13598 Jps
```

### 3. Create an input directory

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -mkdir -p /input
2025-04-23 15:08:21,712 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -ls /
2025-04-23 15:08:24,979 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Found 1 items
drwxr-xr-x   - hduser supergroup          0 2025-04-23 15:08 /input
```

**#Delete any previous input or output files if present using: `hdfs dfs -rm -r /input /output`**

### 4. Create a text file and upload it to HDFS

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ nano character_count.txt
```

**#character\_count.txt file will open in nano text editor**

**#Add your text in this file**

**#Press Ctrl + X**

**#Press Y**

**#Press Enter key**

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -put character_count.txt
/input/
2025-04-23 15:09:21,442 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -ls /input/
2025-04-23 15:09:31,281 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--   1 hduser supergroup          15 2025-04-23 15:09
/input/character_count.txt
```

### 5. Similarly, create a mapper.py and reducer.py file

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ nano mapper.py
```

#This will open the mapper.py file in nano editor; write the Mapper python code here

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ nano reducer.py
```

#Write the Reducer python code

#Now, make the python script executable

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ chmod +x mapper.py
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ chmod +x reducer.py
```

## 6. Run Hadoop streaming jar using the mapper and reducer scripts

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ whereis hadoop
```

```
hadoop: /usr/local/hadoop /usr/local/hadoop/bin/hadoop.cmd  
/usr/local/hadoop/bin/Hadoop
```

# - is used to specify flags

# \ at the end of a line is used to split a long command into multiple lines

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hadoop jar  
/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.4.jar \  
> -input /input/character_count.txt \  
> -output /output/character_output \  
> -mapper mapper.py \  
> -reducer reducer.py \  
> -file mapper.py \  
> -file reducer.py
```

```
2025-04-23 15:12:38,863 WARN streaming.StreamJob: -file option is deprecated,  
please use generic option -files instead.
```

```
2025-04-23 15:12:38,954 WARN util.NativeCodeLoader: Unable to load native-hadoop  
library for your platform... using builtin-java classes where applicable
```

```
packageJobJar: [mapper.py, reducer.py] [] /tmp/streamjob14083921992577679215.jar  
tmpDir=null
```

```
2025-04-23 15:12:39,444 INFO impl.MetricsConfig: Loaded properties from hadoop-  
metrics2.properties
```

```
2025-04-23 15:12:39,514 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot  
period at 10 second(s).
```

```
2025-04-23 15:12:39,514 INFO impl.MetricsSystemImpl: JobTracker metrics system  
started
```

2025-04-23 15:12:39,526 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!

2025-04-23 15:12:39,710 INFO mapred.FileInputFormat: Total input files to process : 1

2025-04-23 15:12:39,768 INFO mapreduce.JobSubmitter: number of splits:1

2025-04-23 15:12:39,891 INFO mapreduce.JobSubmitter: Submitting tokens for job: job\_local333615659\_0001

2025-04-23 15:12:39,891 INFO mapreduce.JobSubmitter: Executing with tokens: []

2025-04-23 15:12:40,054 INFO mapred.LocalDistributedCacheManager: Localized file:/home/hduser/mapper.py as  
file:/app/hadoop/tmp/mapred/local/job\_local333615659\_0001\_c0fdc6e9-bdc9-48dd-b275-22476d2aa5ca/mapper.py

2025-04-23 15:12:40,074 INFO mapred.LocalDistributedCacheManager: Localized file:/home/hduser/reducer.py as  
file:/app/hadoop/tmp/mapred/local/job\_local333615659\_0001\_8af362e9-5351-4a5e-99a3-0bc92437819d/reducer.py

2025-04-23 15:12:40,120 INFO mapreduce.Job: The url to track the job: http://localhost:8080/

2025-04-23 15:12:40,121 INFO mapred.LocalJobRunner: OutputCommitter set in config null

2025-04-23 15:12:40,122 INFO mapreduce.Job: Running job: job\_local333615659\_0001

2025-04-23 15:12:40,126 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCommitter

2025-04-23 15:12:40,129 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2

2025-04-23 15:12:40,129 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup \_temporary folders under output directory:false, ignore cleanup failures: false

2025-04-23 15:12:40,170 INFO mapred.LocalJobRunner: Waiting for map tasks

2025-04-23 15:12:40,173 INFO mapred.LocalJobRunner: Starting task: attempt\_local333615659\_0001\_m\_000000\_0

2025-04-23 15:12:40,198 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2

2025-04-23 15:12:40,198 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup \_temporary folders under output directory:false, ignore cleanup failures: false

2025-04-23 15:12:40,207 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]

2025-04-23 15:12:40,216 INFO mapred.MapTask: Processing split: hdfs://localhost:54310/input/character\_count.txt:0+15

2025-04-23 15:12:40,230 INFO mapred.MapTask: numReduceTasks: 1

2025-04-23 15:12:40,261 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)

2025-04-23 15:12:40,261 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100

2025-04-23 15:12:40,261 INFO mapred.MapTask: soft limit at 83886080

2025-04-23 15:12:40,261 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600

2025-04-23 15:12:40,261 INFO mapred.MapTask: kvstart = 26214396; length = 6553600

2025-04-23 15:12:40,263 INFO mapred.MapTask: Map output collector class =  
org.apache.hadoop.mapred.MapTask\$MapOutputBuffer  
2025-04-23 15:12:40,273 INFO streaming.PipeMapRed: PipeMapRed exec  
[/home/hduser/./mapper.py]  
2025-04-23 15:12:40,276 INFO Configuration.deprecation: mapred.work.output.dir is  
deprecated. Instead, use mapreduce.task.output.dir  
2025-04-23 15:12:40,276 INFO Configuration.deprecation: mapred.local.dir is  
deprecated. Instead, use mapreduce.cluster.local.dir  
2025-04-23 15:12:40,277 INFO Configuration.deprecation: map.input.file is  
deprecated. Instead, use mapreduce.map.input.file  
2025-04-23 15:12:40,277 INFO Configuration.deprecation: map.input.length is  
deprecated. Instead, use mapreduce.map.input.length  
2025-04-23 15:12:40,277 INFO Configuration.deprecation: mapred.job.id is  
deprecated. Instead, use mapreduce.job.id  
2025-04-23 15:12:40,277 INFO Configuration.deprecation: mapred.task.partition is  
deprecated. Instead, use mapreduce.task.partition  
2025-04-23 15:12:40,278 INFO Configuration.deprecation: map.input.start is  
deprecated. Instead, use mapreduce.map.input.start  
2025-04-23 15:12:40,278 INFO Configuration.deprecation: mapred.task.is.map is  
deprecated. Instead, use mapreduce.task.ismap  
2025-04-23 15:12:40,278 INFO Configuration.deprecation: mapred.task.id is  
deprecated. Instead, use mapreduce.task.attempt.id  
2025-04-23 15:12:40,278 INFO Configuration.deprecation: mapred.tip.id is  
deprecated. Instead, use mapreduce.task.id  
2025-04-23 15:12:40,278 INFO Configuration.deprecation: mapred.skip.on is  
deprecated. Instead, use mapreduce.job.skiprecords  
2025-04-23 15:12:40,279 INFO Configuration.deprecation: user.name is deprecated.  
Instead, use mapreduce.job.user.name  
2025-04-23 15:12:40,354 INFO streaming.PipeMapRed: R/W/S=1/0/0 in:NA [rec/s]  
out:NA [rec/s]  
2025-04-23 15:12:40,356 INFO streaming.PipeMapRed: Records R/W=3/1  
2025-04-23 15:12:40,357 INFO streaming.PipeMapRed: MRErrorThread done  
2025-04-23 15:12:40,357 INFO streaming.PipeMapRed: mapRedFinished  
2025-04-23 15:12:40,359 INFO mapred.LocalJobRunner:  
2025-04-23 15:12:40,359 INFO mapred.MapTask: Starting flush of map output  
2025-04-23 15:12:40,359 INFO mapred.MapTask: Spilling map output  
2025-04-23 15:12:40,359 INFO mapred.MapTask: bufstart = 0; bufend = 48; bufvoid =  
104857600  
2025-04-23 15:12:40,359 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend  
= 26214352(104857408); length = 45/6553600  
2025-04-23 15:12:40,368 INFO mapred.MapTask: Finished spill 0  
2025-04-23 15:12:40,377 INFO mapred.Task:  
Task:attempt\_local333615659\_0001\_m\_000000\_0 is done. And is in the process of  
committing  
2025-04-23 15:12:40,379 INFO mapred.LocalJobRunner: Records R/W=3/1  
2025-04-23 15:12:40,379 INFO mapred.Task: Task  
'attempt\_local333615659\_0001\_m\_000000\_0' done.

2025-04-23 15:12:40,384 INFO mapred.Task: Final Counters for  
attempt\_local333615659\_0001\_m\_000000\_0: Counters: 23

File System Counters

FILE: Number of bytes read=1047  
FILE: Number of bytes written=644740  
FILE: Number of read operations=0  
FILE: Number of large read operations=0  
FILE: Number of write operations=0  
HDFS: Number of bytes read=15  
HDFS: Number of bytes written=0  
HDFS: Number of read operations=5  
HDFS: Number of large read operations=0  
HDFS: Number of write operations=1  
HDFS: Number of bytes read erasure-coded=0

Map-Reduce Framework

Map input records=3  
Map output records=12  
Map output bytes=48  
Map output materialized bytes=78  
Input split bytes=100  
Combine input records=0  
Spilled Records=12  
Failed Shuffles=0  
Merged Map outputs=0  
GC time elapsed (ms)=7  
Total committed heap usage (bytes)=309329920

File Input Format Counters

Bytes Read=15

2025-04-23 15:12:40,385 INFO mapred.LocalJobRunner: Finishing task:

attempt\_local333615659\_0001\_m\_000000\_0

2025-04-23 15:12:40,385 INFO mapred.LocalJobRunner: map task executor complete.

2025-04-23 15:12:40,387 INFO mapred.LocalJobRunner: Waiting for reduce tasks

2025-04-23 15:12:40,387 INFO mapred.LocalJobRunner: Starting task:

attempt\_local333615659\_0001\_r\_000000\_0

2025-04-23 15:12:40,393 INFO output.FileOutputCommitter: File Output Committer  
Algorithm version is 2

2025-04-23 15:12:40,393 INFO output.FileOutputCommitter: FileOutputCommitter skip  
cleanup\_temporary folders under output directory:false, ignore cleanup failures:  
false

2025-04-23 15:12:40,393 INFO mapred.Task: Using ResourceCalculatorProcessTree :  
[ ]

2025-04-23 15:12:40,395 INFO mapred.ReduceTask: Using ShuffleConsumerPlugin:  
org.apache.hadoop.mapreduce.task.reduce.Shuffle@386a06de

2025-04-23 15:12:40,396 WARN impl.MetricsSystemImpl: JobTracker metrics system  
already initialized!

2025-04-23 15:12:40,409 INFO reduce.MergeManagerImpl: MergerManager:  
memoryLimit=1437178240, maxSingleShuffleLimit=359294560, mergeThreshold=948537664,  
ioSortFactor=10, memToMemMergeOutputsThreshold=10  
2025-04-23 15:12:40,410 INFO reduce.EventFetcher:  
attempt\_local333615659\_0001\_r\_000000\_0 Thread started: EventFetcher for fetching  
Map Completion Events  
2025-04-23 15:12:40,435 INFO reduce.LocalFetcher: localfetcher#1 about to shuffle  
output of map attempt\_local333615659\_0001\_m\_000000\_0 decomp: 74 len: 78 to MEMORY  
2025-04-23 15:12:40,437 INFO reduce.InMemoryMapOutput: Read 74 bytes from map-  
output for attempt\_local333615659\_0001\_m\_000000\_0  
2025-04-23 15:12:40,437 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-  
output of size: 74, inMemoryMapOutputs.size() -> 1, commitMemory -> 0, usedMemory  
->74  
2025-04-23 15:12:40,439 INFO reduce.EventFetcher: EventFetcher is interrupted..  
Returning  
2025-04-23 15:12:40,439 INFO mapred.LocalJobRunner: 1 / 1 copied.  
2025-04-23 15:12:40,439 INFO reduce.MergeManagerImpl: finalMerge called with 1  
in-memory map-outputs and 0 on-disk map-outputs  
2025-04-23 15:12:40,445 INFO mapred.Merger: Merging 1 sorted segments  
2025-04-23 15:12:40,445 INFO mapred.Merger: Down to the last merge-pass, with 1  
segments left of total size: 70 bytes  
2025-04-23 15:12:40,449 INFO reduce.MergeManagerImpl: Merged 1 segments, 74 bytes  
to disk to satisfy reduce memory limit  
2025-04-23 15:12:40,449 INFO reduce.MergeManagerImpl: Merging 1 files, 78 bytes  
from disk  
2025-04-23 15:12:40,450 INFO reduce.MergeManagerImpl: Merging 0 segments, 0 bytes  
from memory into reduce  
2025-04-23 15:12:40,450 INFO mapred.Merger: Merging 1 sorted segments  
2025-04-23 15:12:40,450 INFO mapred.Merger: Down to the last merge-pass, with 1  
segments left of total size: 70 bytes  
2025-04-23 15:12:40,450 INFO mapred.LocalJobRunner: 1 / 1 copied.  
2025-04-23 15:12:40,453 INFO streaming.PipeMapRed: PipeMapRed exec  
[/home/hduser/./reducer.py]  
2025-04-23 15:12:40,455 INFO Configuration.deprecation: mapred.job.tracker is  
deprecated. Instead, use mapreduce.jobtracker.address  
2025-04-23 15:12:40,457 INFO Configuration.deprecation: mapred.map.tasks is  
deprecated. Instead, use mapreduce.job.maps  
2025-04-23 15:12:40,489 INFO streaming.PipeMapRed: R/W/S=1/0/0 in:NA [rec/s]  
out:NA [rec/s]  
2025-04-23 15:12:40,489 INFO streaming.PipeMapRed: R/W/S=10/0/0 in:NA [rec/s]  
out:NA [rec/s]  
2025-04-23 15:12:40,490 INFO streaming.PipeMapRed: Records R/W=12/1  
2025-04-23 15:12:40,492 INFO streaming.PipeMapRed: MRErrorThread done  
2025-04-23 15:12:40,493 INFO streaming.PipeMapRed: mapRedFinished  
2025-04-23 15:12:40,544 INFO mapred.Task:  
Task:attempt\_local333615659\_0001\_r\_000000\_0 is done. And is in the process of  
committing

2025-04-23 15:12:40,547 INFO mapred.LocalJobRunner: 1 / 1 copied.  
2025-04-23 15:12:40,547 INFO mapred.Task: Task  
attempt\_local333615659\_0001\_r\_000000\_0 is allowed to commit now  
2025-04-23 15:12:40,564 INFO output.FileOutputCommitter: Saved output of task  
'attempt\_local333615659\_0001\_r\_000000\_0' to  
hdfs://localhost:54310/output/character\_output  
2025-04-23 15:12:40,565 INFO mapred.LocalJobRunner: Records R/W=12/1 > reduce  
2025-04-23 15:12:40,565 INFO mapred.Task: Task  
'attempt\_local333615659\_0001\_r\_000000\_0' done.  
2025-04-23 15:12:40,566 INFO mapred.Task: Final Counters for  
attempt\_local333615659\_0001\_r\_000000\_0: Counters: 30

File System Counters

FILE: Number of bytes read=1235  
FILE: Number of bytes written=644818  
FILE: Number of read operations=0  
FILE: Number of large read operations=0  
FILE: Number of write operations=0  
HDFS: Number of bytes read=15  
HDFS: Number of bytes written=12  
HDFS: Number of read operations=10  
HDFS: Number of large read operations=0  
HDFS: Number of write operations=3  
HDFS: Number of bytes read erasure-coded=0

Map-Reduce Framework

Combine input records=0  
Combine output records=0  
Reduce input groups=3  
Reduce shuffle bytes=78  
Reduce input records=12  
Reduce output records=3  
Spilled Records=12  
Shuffled Maps =1  
Failed Shuffles=0  
Merged Map outputs=1  
GC time elapsed (ms)=0  
Total committed heap usage (bytes)=309329920

Shuffle Errors

BAD\_ID=0  
CONNECTION=0  
IO\_ERROR=0  
WRONG\_LENGTH=0  
WRONG\_MAP=0  
WRONG\_REDUCE=0

File Output Format Counters

Bytes Written=12

2025-04-23 15:12:40,568 INFO mapred.LocalJobRunner: Finishing task:  
attempt\_local333615659\_0001\_r\_000000\_0



2025-04-23 15:12:40,569 INFO mapred.LocalJobRunner: reduce task executor complete.

2025-04-23 15:12:41,130 INFO mapreduce.Job: Job job\_local333615659\_0001 running in uber mode : false

2025-04-23 15:12:41,131 INFO mapreduce.Job: map 100% reduce 100%

2025-04-23 15:12:41,134 INFO mapreduce.Job: Job job\_local333615659\_0001 completed successfully

2025-04-23 15:12:41,140 INFO mapreduce.Job: Counters: 36

#### File System Counters

FILE: Number of bytes read=2282

FILE: Number of bytes written=1289558

FILE: Number of read operations=0

FILE: Number of large read operations=0

FILE: Number of write operations=0

HDFS: Number of bytes read=30

HDFS: Number of bytes written=12

HDFS: Number of read operations=15

HDFS: Number of large read operations=0

HDFS: Number of write operations=4

HDFS: Number of bytes read erasure-coded=0

#### Map-Reduce Framework

Map input records=3

Map output records=12

Map output bytes=48

Map output materialized bytes=78

Input split bytes=100

Combine input records=0

Combine output records=0

Reduce input groups=3

Reduce shuffle bytes=78

Reduce input records=12

Reduce output records=3

Spilled Records=24

Shuffled Maps =1

Failed Shuffles=0

Merged Map outputs=1

GC time elapsed (ms)=7

Total committed heap usage (bytes)=618659840

#### 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=15

#### File Output Format Counters

```
Bytes Written=12
2025-04-23 15:12:41,140 INFO streaming.StreamJob: Output directory:
/output/character_output
```

## 7. View Output

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -ls /output/character_output/
2025-04-23 15:13:37,279 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--  1 hduser supergroup          0 2025-04-23 15:12
/output/character_output/_SUCCESS
-rw-r--r--  1 hduser supergroup        12 2025-04-23 15:12
/output/character_output/part-00000
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -cat
/output/character_output/part-00000
2025-04-23 15:14:21,607 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
a  4
b  4
c  4
```

## 8. Stop HDFS

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ stop-dfs.sh
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [Ubuntu]
2025-04-23 15:15:32,461 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ stop-yarn.sh
Stopping nodemanagers
Stopping resourcemanager
```

## **CONCLUSION:**

This assignment helped understand the MapReduce programming model by implementing a simple yet powerful word and character frequency counter. It demonstrates the use of Hadoop Streaming with Python, and how big data problems can be broken into parallel tasks efficiently. By splitting the tasks into Mapper and Reducer phases, we counted occurrences of each word or character in parallel across the cluster.

## ASSIGNMENT 4

**PROBLEM STATEMENT:** Write code to simulate requests coming from clients and distribute them among the servers using the load balancing algorithms.

### THEORY:

#### Load Balancing

Load balancing is the method of evenly distributing network traffic or tasks across multiple servers, computers, or other resources to ensure no single server becomes overwhelmed. It ensures that no single server handles too much workload, which could lead to slow performance or crashes.

The main goals of load balancing are:

1. **Optimized Resource Utilization** – Ensure all servers share the workload evenly.
2. **High Availability & Reliability** – Prevent service disruptions by rerouting traffic if a server fails.
3. **Scalability** – Allow seamless addition of servers to handle increased demand.
4. **Reduced Latency** – Route requests to the fastest or least busy server to minimize response time.
5. **Fault Tolerance** – Maintain system performance even during hardware or software failures.

There are several algorithms to achieve load balancing, each with its pros and cons.

#### 1. Round Robin

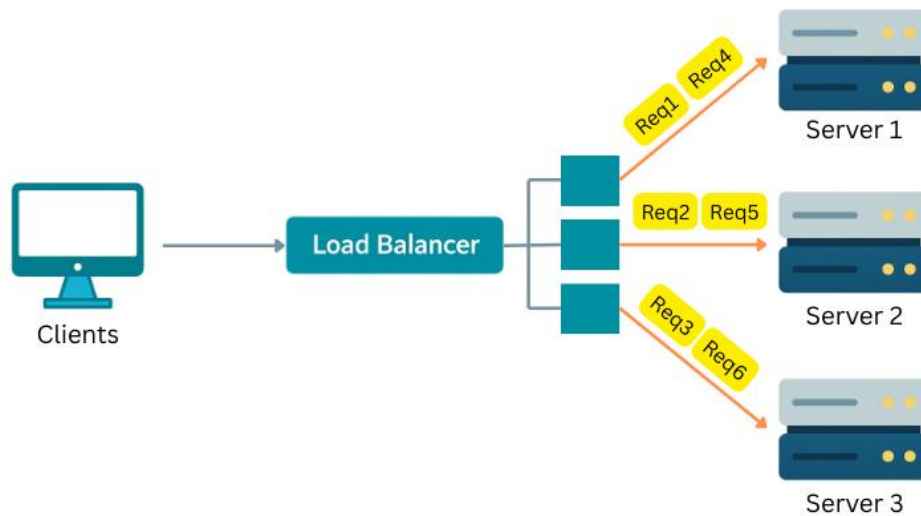
This algorithm distributes incoming requests to servers in a cyclic order. It assigns a request to the first server, then moves to the second, third, and so on, and after reaching the last server, the algorithm loops back to the first server. It is used when all servers have similar processing capabilities and are equally capable of handling requests.

##### Pros:

- Ensures even distribution of traffic
- Easy to implement and understand.

##### Cons:

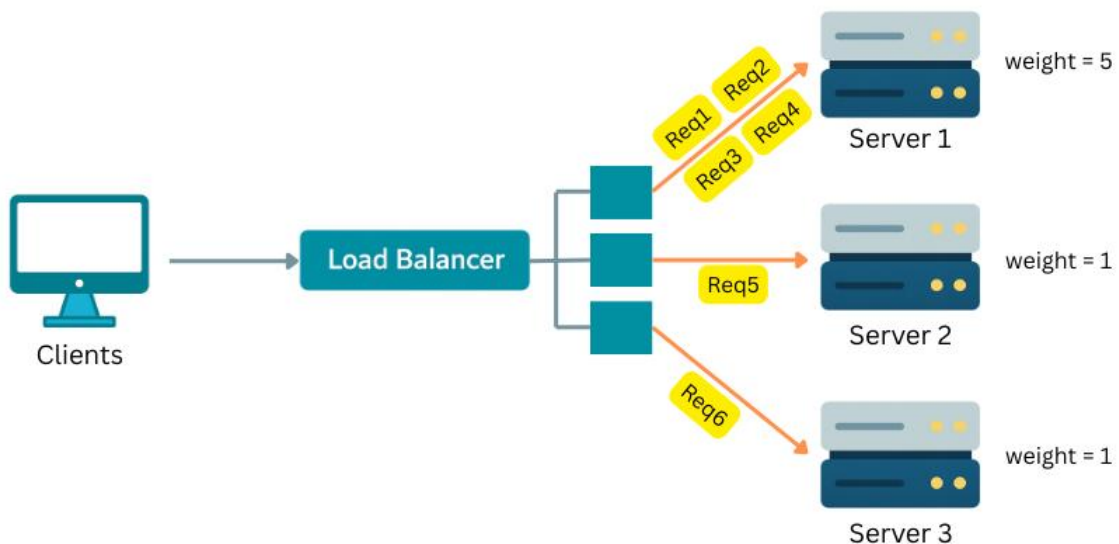
- Does not consider server load or response time.
- Can lead to inefficiencies if servers have different processing capabilities.



1.Round Robin

## 2. Weighted Round Robin

Weighted Round Robin is an enhanced version of the Round Robin load balancing algorithm. It assigns weights to each server based on their capacity or performance, distributing incoming requests proportionally according to these weights. This ensures that more powerful servers handle a larger share of the load, while less powerful servers handle a smaller share.



2.Weighted Round Robin

### Pros

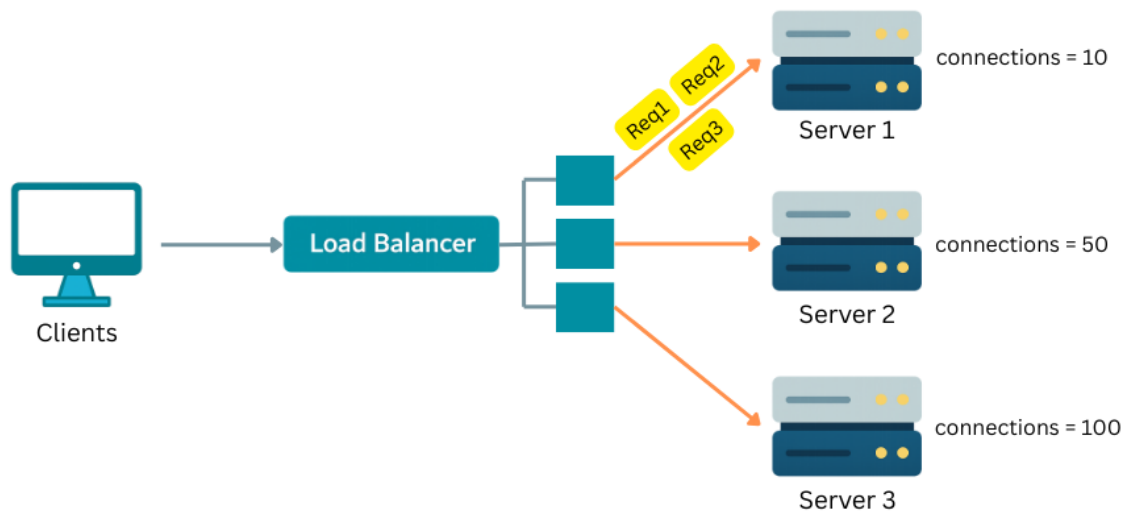
- Balances load according to server capabilities.
- More efficient use of server resources.

### Cons

- Complexity in weight assignment
- Does not consider real-time server load or response time.

## 3. Least Connections

Least Connections takes into account the number of active connections on each server. It assigns incoming requests to the server with the least number of active connections.



3.Least Connections

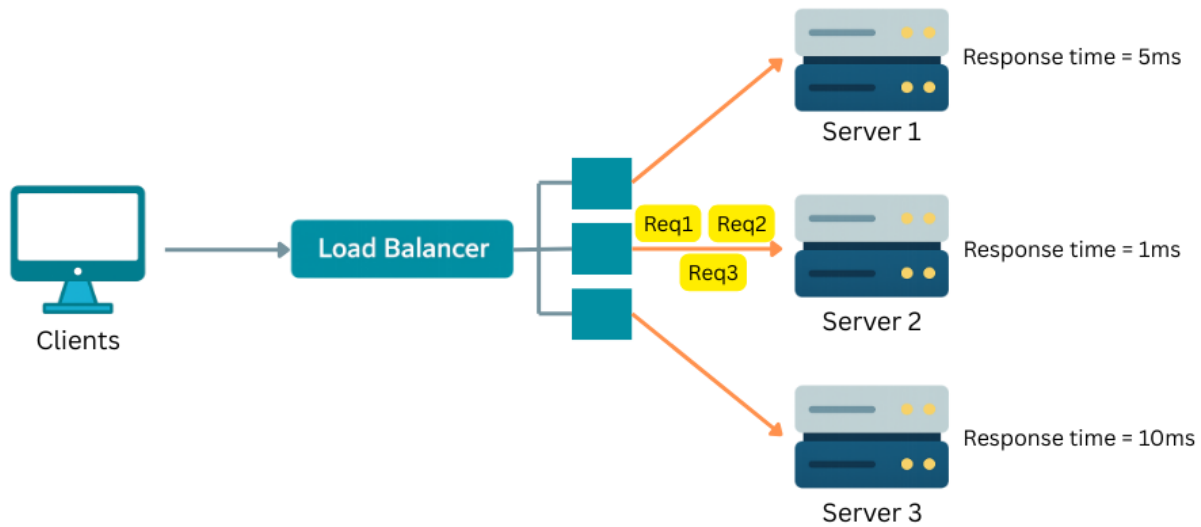
### Pros:

- Balances load more dynamically based on current server load.
- Helps prevent any server from becoming overloaded with a high number of active connections.

### Cons:

- May not be optimal if servers have different processing capabilities.
- Requires tracking active connections for each server.

## 4. Least Response Time



4. Least Response Time

Least Response Time load balancing is a dynamic algorithm that assigns incoming requests to the server with the lowest response time, ensuring efficient utilization of server resources and optimal client experience. This approach aims to direct traffic to the server that can handle the request the fastest, based on recent performance metrics.

#### Pros

- Minimizes overall latency by selecting the server with the fastest response time.
- Can adapt dynamically to changes in server response times.

#### Cons

- More complex as it requires continuous monitoring of server performance.
- Monitoring response times and dynamically adjusting the load can introduce additional overhead.
- Response times can vary in the short term due to network fluctuations or transient server issues, potentially causing frequent rebalancing.

#### CODE:

```
import time
import random
```

```

class RoundRobin:
    def __init__(self, servers):
        self.servers = servers
        self.current_index = -1

    def get_next_server(self):
        self.current_index = (self.current_index + 1) % len(self.servers)
        return self.servers[self.current_index]

class WeightedRoundRobin:
    def __init__(self, servers, weights):
        self.servers = servers
        self.weights = weights
        self.current_index = -1
        self.current_weight = 0

    def get_next_server(self):
        while True:
            self.current_index = (self.current_index + 1) % len(self.servers)
            if self.current_index == 0:
                self.current_weight -= 1
                if self.current_weight <= 0:
                    self.current_weight = max(self.weights)
            if self.weights[self.current_index] >= self.current_weight:
                return self.servers[self.current_index]

class LeastConnections:
    def __init__(self, servers):
        self.servers = {server: 0 for server in servers}

    def get_next_server(self):
        # Find the minimum number of connections
        min_connections = min(self.servers.values())
        # Get all servers with the minimum number of connections
        least_loaded_servers = [server for server, connections in
self.servers.items() if connections == min_connections]
        # Select a random server from the least loaded servers
        selected_server = random.choice(least_loaded_servers)
        self.servers[selected_server] += 1
        return selected_server

    def release_connection(self, server):
        if self.servers[server] > 0:
            self.servers[server] -= 1

class LeastResponseTime:
    def __init__(self, servers):

```



```

        self.servers = servers
        self.response_times = [0] * len(servers)

    def get_next_server(self):
        min_response_time = min(self.response_times)
        min_index = self.response_times.index(min_response_time)
        return self.servers[min_index]

    def update_response_time(self, server, response_time):
        index = self.servers.index(server)
        self.response_times[index] = response_time

    def simulate_response_time():
        # Simulating response time with random delay
        delay = random.uniform(0.1, 1.0)
        time.sleep(delay)
        return delay

    def demonstrate_algorithm(algorithm_name, load_balancer, iterations=6,
                             use_response_time=False, use_connections=False):
        print(f"\n---- {algorithm_name} ----")

        for i in range(iterations):
            server = load_balancer.get_next_server()
            print(f"Request {i + 1} -> {server}")

            if use_response_time:
                response_time = simulate_response_time()
                load_balancer.update_response_time(server, response_time)
                print(f"Response Time: {response_time:.2f}s")

            if use_connections:
                load_balancer.release_connection(server)

if __name__ == "__main__":
    servers = ["Server1", "Server2", "Server3"]

    # Round Robin
    rr = RoundRobin(servers)
    demonstrate_algorithm("Round Robin", rr)

    # Weighted Round Robin
    weights = [5, 1, 1]
    wrr = WeightedRoundRobin(servers, weights)
    demonstrate_algorithm("Weighted Round Robin", wrr, iterations=7)

    # Least Connections

```

```
lc = LeastConnections(servers)
demonstrate_algorithm("Least Connections", lc, use_connections=True)

# Least Response Time
lrt = LeastResponseTime(servers)
demonstrate_algorithm("Least Response Time", lrt, use_response_time=True)
```

## OUTPUT:

---- Round Robin ----

```
Request 1 -> Server1
Request 2 -> Server2
Request 3 -> Server3
Request 4 -> Server1
Request 5 -> Server2
Request 6 -> Server3
```

---- Weighted Round Robin ----

```
Request 1 -> Server1
Request 2 -> Server1
Request 3 -> Server1
Request 4 -> Server1
Request 5 -> Server1
Request 6 -> Server2
Request 7 -> Server3
```

---- Least Connections ----

```
Request 1 -> Server2
Request 2 -> Server2
Request 3 -> Server1
Request 4 -> Server3
Request 5 -> Server2
Request 6 -> Server3
```

---- Least Response Time ----

```
Request 1 -> Server1
Response Time: 0.87s
Request 2 -> Server2
Response Time: 0.79s
Request 3 -> Server3
Response Time: 0.61s
Request 4 -> Server3
Response Time: 0.81s
Request 5 -> Server2
Response Time: 0.81s
Request 6 -> Server2
```

Response Time: 0.30s

## **CONCLUSION:**

Thus, we explored and implemented various load balancing algorithms including Round Robin, Weighted Round Robin, Least Connections, and Least Response Time. Each algorithm offers a unique approach to distributing client requests across multiple servers to optimize performance and resource utilization. By comparing these strategies, we gained practical insights into how load balancing enhances system scalability, minimizes response time, and prevents server overload.

## ASSIGNMENT 5

**PROBLEM STATEMENT:** Design and develop a distributed application to find the coolest/hottest year from the available weather data. Use weather data from the Internet and process it using MapReduce.

### **THEORY:**

#### **MapReduce:**

- MapReduce is a programming model used in distributed systems to process and generate large datasets in parallel across a cluster of computers. It breaks down tasks into two main functions: Map and Reduce.
- Distributed systems have multiple nodes (computers) working together. MapReduce leverages this architecture by:
  - Dividing tasks across nodes
  - Processing in parallel, improving speed and scalability
  - Handling faults automatically

#### **Hadoop Streaming**

While Hadoop is typically used with Java, Hadoop Streaming is a powerful utility that allows users to write Mapper and Reducer code in any language that can read from standard input and write to standard output, such as Python, Perl, Bash, or Ruby. This enables greater flexibility and rapid development. In this practical, we use Python for its simplicity and readability.

#### **Workflow of finding hottest/coolest year from the data**

##### **Input File (HDFS):**

The dataset is stored in HDFS and split across multiple blocks to be processed in parallel by the Hadoop framework.

##### **1. Mapper Phase:**

The mapper reads each line of the input, extracts the year and the temperature, and emits a key-value pair:

(date, (min\_temperature, max\_temperature))

**2. Shuffle and Sort:** (Handled by Hadoop)

Hadoop groups all values (temperatures) associated with the same year and sends them to the same reducer.

**3. Reducer Phase:**

For each year, the reducer calculates:

- The minimum and maximum temperature (or max/min if required)
- Emits:

(year, (min\_temperature, max\_temperature))

**4. Final Output:**

From the output file, the highest and lowest average temperatures of every year can be determined, showing the hottest and coolest years respectively.

## CODE:

### mapper.py

```
#!/usr/bin/env python3
import sys
for line in sys.stdin:
    line = line.strip()
    if not line:
        continue

    parts = line.split()
    if len(parts) == 3:
        date_str, min_temp, max_temp = parts
        if "-" in date_str:
            year = date_str.split("-")[0]
            print(f"{year} {min_temp} {max_temp}")
```

### reducer.py

```
#!/usr/bin/env python3
import sys
from collections import defaultdict

temps_by_year = defaultdict(list)

for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    parts = line.split()
    if len(parts) != 3:
        continue
    year, min_temp, max_temp = parts
    try:
        temps_by_year[year].append((int(min_temp), int(max_temp)))
    except:
        continue

for year in sorted(temps_by_year.keys()):
    mins, maxs = zip(*temps_by_year[year])
    print(f"{year}\t{min(mins)}\t{max(maxs)}")
```

## 1. Open Terminal and switch to Hadoop user

```
pvg@pvg-HP-ProDesk-400-G4-SFF:~$ su hduser
Password:
hduser@pvg-HP-ProDesk-400-G4-SFF:/home/pvg$ cd
```

## 2. Start HDFS

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [pvg-HP-ProDesk-400-G4-SFF]
2025-04-23 15:17:37,545 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ jps
16580 NodeManager
16932 Jps
15815 NameNode
16216 SecondaryNameNode
16425 ResourceManager
15998 DataNode
```

## 3. Create an input directory

```
#Delete any previous input or output files if present using: hdfs dfs -rm
-r /input /output
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -mkdir -p /input
2025-04-23 15:20:45,988 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -ls /
2025-04-23 15:21:00,439 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Found 1 items
drwxr-xr-x  - hduser supergroup          0 2025-04-23 15:20 /input
```

## 4. Create a text file, paste the weather data and upload it to HDFS

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ nano weather_data.txt
```

```
#weather_data.txt file will open in nano text editor
```

```
#Copy the text from the provided weather data text file and paste it here
```

**#Press Ctrl + X**

**#Press Y**

**#Press Enter key**

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -put weather_data.txt /input/
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -ls /input/
2025-04-23 15:21:32,321 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Found 1 items
-rw-r--r--  1 hduser supergroup      1685 2025-04-23 15:21
/input/weather_data.txt
```

## **5. Similary, create a mapper.py and reducer.py file**

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ nano mapper.py
```

**#This will open the mapper.py file in nano editor; write the Mapper python code here**

**#Here's the code for mapper.py**

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ nano reducer.py
```

**#Write the reducer python code**

**#Now, make the python script executable**

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ chmod +x mapper.py
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ chmod +x reducer.py
```

## **6. Run Hadoop streaming jar using the mapper and reducer scripts**

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ whereis hadoop
hadoop: /usr/local/hadoop /usr/local/hadoop/bin/hadoop.cmd
/usr/local/hadoop/bin/Hadoop
```

**# - is used to specify flags**

**# \ at the end of a line is used to split a long command into multiple lines**

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hadoop jar
/usr/local/hadoop/share/hadoop/tools/lib/hadoop-streaming-3.3.4.jar \
> -input /input/weather_data.txt \
```



```
> -output /output/weather_output \  
> -mapper mapper.py \  
> -reducer reducer.py \  
> -file mapper.py \  
> -file reducer.py  
2025-04-23 15:24:51,122 WARN streaming.StreamJob: -file option is deprecated,  
please use generic option -files instead.  
2025-04-23 15:24:51,217 WARN util.NativeCodeLoader: Unable to load native-hadoop  
library for your platform... using builtin-java classes where applicable  
packageJobJar: [mapper.py, reducer.py] [] /tmp/streamjob9268891282553717103.jar  
tmpDir=null  
2025-04-23 15:24:51,701 INFO impl.MetricsConfig: Loaded properties from hadoop-  
metrics2.properties  
2025-04-23 15:24:51,769 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot  
period at 10 second(s).  
2025-04-23 15:24:51,769 INFO impl.MetricsSystemImpl: JobTracker metrics system  
started  
2025-04-23 15:24:51,784 WARN impl.MetricsSystemImpl: JobTracker metrics system  
already initialized!  
2025-04-23 15:24:51,993 INFO mapred.FileInputFormat: Total input files to  
process : 1  
2025-04-23 15:24:52,053 INFO mapreduce.JobSubmitter: number of splits:1  
2025-04-23 15:24:52,176 INFO mapreduce.JobSubmitter: Submitting tokens for job:  
job_local1388892213_0001  
2025-04-23 15:24:52,176 INFO mapreduce.JobSubmitter: Executing with tokens: []  
2025-04-23 15:24:52,326 INFO mapred.LocalDistributedCacheManager: Localized  
file:/home/hduser/mapper.py as  
file:/app/hadoop/tmp/mapred/local/job_local1388892213_0001_3cc73aa2-b9eb-4451-  
bd53-20464abdb17f/mapper.py  
2025-04-23 15:24:52,360 INFO mapred.LocalDistributedCacheManager: Localized  
file:/home/hduser/reducer.py as  
file:/app/hadoop/tmp/mapred/local/job_local1388892213_0001_1494c792-fedb-4543-  
be63-f5178e81a6a0/reducer.py  
2025-04-23 15:24:52,404 INFO mapreduce.Job: The url to track the job:  
http://localhost:8080/  
2025-04-23 15:24:52,405 INFO mapred.LocalJobRunner: OutputCommitter set in config  
null  
2025-04-23 15:24:52,408 INFO mapreduce.Job: Running job: job_local1388892213_0001  
2025-04-23 15:24:52,410 INFO mapred.LocalJobRunner: OutputCommitter is  
org.apache.hadoop.mapred.FileOutputCommitter  
2025-04-23 15:24:52,414 INFO output.FileOutputCommitter: File Output Committer  
Algorithm version is 2  
2025-04-23 15:24:52,414 INFO output.FileOutputCommitter: FileOutputCommitter skip  
cleanup _temporary folders under output directory:false, ignore cleanup failures:  
false  
2025-04-23 15:24:52,450 INFO mapred.LocalJobRunner: Waiting for map tasks
```

2025-04-23 15:24:52,452 INFO mapred.LocalJobRunner: Starting task:  
attempt\_local1388892213\_0001\_m\_000000\_0  
2025-04-23 15:24:52,476 INFO output.FileOutputCommitter: File Output Committer  
Algorithm version is 2  
2025-04-23 15:24:52,476 INFO output.FileOutputCommitter: FileOutputCommitter skip  
cleanup \_temporary folders under output directory:false, ignore cleanup failures:  
false  
2025-04-23 15:24:52,484 INFO mapred.Task: Using ResourceCalculatorProcessTree :  
[ ]  
2025-04-23 15:24:52,488 INFO mapred.MapTask: Processing split:  
hdfs://localhost:54310/input/weather\_data.txt:0+1685  
2025-04-23 15:24:52,503 INFO mapred.MapTask: numReduceTasks: 1  
2025-04-23 15:24:52,536 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)  
2025-04-23 15:24:52,536 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100  
2025-04-23 15:24:52,536 INFO mapred.MapTask: soft limit at 83886080  
2025-04-23 15:24:52,536 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600  
2025-04-23 15:24:52,536 INFO mapred.MapTask: kvstart = 26214396; length = 6553600  
2025-04-23 15:24:52,538 INFO mapred.MapTask: Map output collector class =  
org.apache.hadoop.mapred.MapTask\$MapOutputBuffer  
2025-04-23 15:24:52,545 INFO streaming.PipeMapRed: PipeMapRed exec  
[/home/hduser/./mapper.py]  
2025-04-23 15:24:52,547 INFO Configuration.deprecation: mapred.work.output.dir is  
deprecated. Instead, use mapreduce.task.output.dir  
2025-04-23 15:24:52,548 INFO Configuration.deprecation: mapred.local.dir is  
deprecated. Instead, use mapreduce.cluster.local.dir  
2025-04-23 15:24:52,549 INFO Configuration.deprecation: map.input.file is  
deprecated. Instead, use mapreduce.map.input.file  
2025-04-23 15:24:52,549 INFO Configuration.deprecation: map.input.length is  
deprecated. Instead, use mapreduce.map.input.length  
2025-04-23 15:24:52,549 INFO Configuration.deprecation: mapred.job.id is  
deprecated. Instead, use mapreduce.job.id  
2025-04-23 15:24:52,550 INFO Configuration.deprecation: mapred.task.partition is  
deprecated. Instead, use mapreduce.task.partition  
2025-04-23 15:24:52,551 INFO Configuration.deprecation: map.input.start is  
deprecated. Instead, use mapreduce.map.input.start  
2025-04-23 15:24:52,551 INFO Configuration.deprecation: mapred.task.is.map is  
deprecated. Instead, use mapreduce.task.ismap  
2025-04-23 15:24:52,551 INFO Configuration.deprecation: mapred.task.id is  
deprecated. Instead, use mapreduce.task.attempt.id  
2025-04-23 15:24:52,552 INFO Configuration.deprecation: mapred.tip.id is  
deprecated. Instead, use mapreduce.task.id  
2025-04-23 15:24:52,552 INFO Configuration.deprecation: mapred.skip.on is  
deprecated. Instead, use mapreduce.job.skiprecords  
2025-04-23 15:24:52,553 INFO Configuration.deprecation: user.name is deprecated.  
Instead, use mapreduce.job.user.name  
2025-04-23 15:24:52,631 INFO streaming.PipeMapRed: R/W/S=1/0/0 in:NA [rec/s]  
out:NA [rec/s]

2025-04-23 15:24:52,631 INFO streaming.PipeMapRed: R/W/S=10/0/0 in:NA [rec/s]  
out:NA [rec/s]  
2025-04-23 15:24:52,632 INFO streaming.PipeMapRed: R/W/S=100/0/0 in:NA [rec/s]  
out:NA [rec/s]  
2025-04-23 15:24:52,633 INFO streaming.PipeMapRed: Records R/W=100/1  
2025-04-23 15:24:52,634 INFO streaming.PipeMapRed: MRErrorThread done  
2025-04-23 15:24:52,636 INFO streaming.PipeMapRed: mapRedFinished  
2025-04-23 15:24:52,638 INFO mapred.LocalJobRunner:  
2025-04-23 15:24:52,638 INFO mapred.MapTask: Starting flush of map output  
2025-04-23 15:24:52,638 INFO mapred.MapTask: Spilling map output  
2025-04-23 15:24:52,638 INFO mapred.MapTask: bufstart = 0; bufend = 1185; bufvoid  
= 104857600  
2025-04-23 15:24:52,638 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend  
= 26214000(104856000); length = 397/6553600  
2025-04-23 15:24:52,648 INFO mapred.MapTask: Finished spill 0  
2025-04-23 15:24:52,659 INFO mapred.Task:  
Task:attempt\_local1388892213\_0001\_m\_000000\_0 is done. And is in the process of  
committing  
2025-04-23 15:24:52,662 INFO mapred.LocalJobRunner: Records R/W=100/1  
2025-04-23 15:24:52,662 INFO mapred.Task: Task  
'attempt\_local1388892213\_0001\_m\_000000\_0' done.  
2025-04-23 15:24:52,666 INFO mapred.Task: Final Counters for  
attempt\_local1388892213\_0001\_m\_000000\_0: Counters: 23  
File System Counters  
FILE: Number of bytes read=1736  
FILE: Number of bytes written=649844  
FILE: Number of read operations=0  
FILE: Number of large read operations=0  
FILE: Number of write operations=0  
HDFS: Number of bytes read=1685  
HDFS: Number of bytes written=0  
HDFS: Number of read operations=5  
HDFS: Number of large read operations=0  
HDFS: Number of write operations=1  
HDFS: Number of bytes read erasure-coded=0  
Map-Reduce Framework  
Map input records=100  
Map output records=100  
Map output bytes=1185  
Map output materialized bytes=1391  
Input split bytes=97  
Combine input records=0  
Spilled Records=100  
Failed Shuffles=0  
Merged Map outputs=0  
GC time elapsed (ms)=6  
Total committed heap usage (bytes)=196083712

## File Input Format Counters

Bytes Read=1685

```
2025-04-23 15:24:52,667 INFO mapred.LocalJobRunner: Finishing task:
attempt_local1388892213_0001_m_000000_0
2025-04-23 15:24:52,667 INFO mapred.LocalJobRunner: map task executor complete.
2025-04-23 15:24:52,669 INFO mapred.LocalJobRunner: Waiting for reduce tasks
2025-04-23 15:24:52,669 INFO mapred.LocalJobRunner: Starting task:
attempt_local1388892213_0001_r_000000_0
2025-04-23 15:24:52,675 INFO output.FileOutputCommitter: File Output Committer
Algorithm version is 2
2025-04-23 15:24:52,675 INFO output.FileOutputCommitter: FileOutputCommitter skip
cleanup _temporary folders under output directory:false, ignore cleanup failures:
false
2025-04-23 15:24:52,675 INFO mapred.Task: Using ResourceCalculatorProcessTree :
[ ]
2025-04-23 15:24:52,677 INFO mapred.ReduceTask: Using ShuffleConsumerPlugin:
org.apache.hadoop.mapreduce.task.reduce.Shuffle@77b2b11f
2025-04-23 15:24:52,678 WARN impl.MetricsSystemImpl: JobTracker metrics system
already initialized!
2025-04-23 15:24:52,691 INFO reduce.MergeManagerImpl: MergerManager:
memoryLimit=1437178240, maxSingleShuffleLimit=359294560, mergeThreshold=948537664,
ioSortFactor=10, memToMemMergeOutputsThreshold=10
2025-04-23 15:24:52,692 INFO reduce.EventFetcher:
attempt_local1388892213_0001_r_000000_0 Thread started: EventFetcher for fetching
Map Completion Events
2025-04-23 15:24:52,714 INFO reduce.LocalFetcher: localfetcher#1 about to shuffle
output of map attempt_local1388892213_0001_m_000000_0 decomp: 1387 len: 1391 to
MEMORY
2025-04-23 15:24:52,715 INFO reduce.InMemoryMapOutput: Read 1387 bytes from map-
output for attempt_local1388892213_0001_m_000000_0
2025-04-23 15:24:52,716 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-
output of size: 1387, inMemoryMapOutputs.size() -> 1, commitMemory -> 0,
usedMemory ->1387
2025-04-23 15:24:52,717 INFO reduce.EventFetcher: EventFetcher is interrupted..
Returning
2025-04-23 15:24:52,717 INFO mapred.LocalJobRunner: 1 / 1 copied.
2025-04-23 15:24:52,717 INFO reduce.MergeManagerImpl: finalMerge called with 1
in-memory map-outputs and 0 on-disk map-outputs
2025-04-23 15:24:52,723 INFO mapred.Merger: Merging 1 sorted segments
2025-04-23 15:24:52,723 INFO mapred.Merger: Down to the last merge-pass, with 1
segments left of total size: 1373 bytes
2025-04-23 15:24:52,726 INFO reduce.MergeManagerImpl: Merged 1 segments, 1387
bytes to disk to satisfy reduce memory limit
2025-04-23 15:24:52,726 INFO reduce.MergeManagerImpl: Merging 1 files, 1391 bytes
from disk
2025-04-23 15:24:52,726 INFO reduce.MergeManagerImpl: Merging 0 segments, 0 bytes
from memory into reduce
```

2025-04-23 15:24:52,726 INFO mapred.Merger: Merging 1 sorted segments  
2025-04-23 15:24:52,726 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 1373 bytes  
2025-04-23 15:24:52,727 INFO mapred.LocalJobRunner: 1 / 1 copied.  
2025-04-23 15:24:52,729 INFO streaming.PipeMapRed: PipeMapRed exec [/home/hduser/./reducer.py]  
2025-04-23 15:24:52,731 INFO Configuration.deprecation: mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address  
2025-04-23 15:24:52,734 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapreduce.job.maps  
2025-04-23 15:24:52,766 INFO streaming.PipeMapRed: R/W/S=1/0/0 in:NA [rec/s] out:NA [rec/s]  
2025-04-23 15:24:52,766 INFO streaming.PipeMapRed: R/W/S=10/0/0 in:NA [rec/s] out:NA [rec/s]  
2025-04-23 15:24:52,767 INFO streaming.PipeMapRed: R/W/S=100/0/0 in:NA [rec/s] out:NA [rec/s]  
2025-04-23 15:24:52,769 INFO streaming.PipeMapRed: Records R/W=100/1  
2025-04-23 15:24:52,771 INFO streaming.PipeMapRed: MRErrorThread done  
2025-04-23 15:24:52,771 INFO streaming.PipeMapRed: mapRedFinished  
2025-04-23 15:24:52,809 INFO mapred.Task:  
Task:attempt\_local1388892213\_0001\_r\_000000\_0 is done. And is in the process of committing  
2025-04-23 15:24:52,812 INFO mapred.LocalJobRunner: 1 / 1 copied.  
2025-04-23 15:24:52,812 INFO mapred.Task: Task  
attempt\_local1388892213\_0001\_r\_000000\_0 is allowed to commit now  
2025-04-23 15:24:52,826 INFO output.FileOutputCommitter: Saved output of task 'attempt\_local1388892213\_0001\_r\_000000\_0' to  
hdfs://localhost:54310/output/weather\_output  
2025-04-23 15:24:52,827 INFO mapred.LocalJobRunner: Records R/W=100/1 > reduce  
2025-04-23 15:24:52,827 INFO mapred.Task: Task  
'attempt\_local1388892213\_0001\_r\_000000\_0' done.  
2025-04-23 15:24:52,827 INFO mapred.Task: Final Counters for  
attempt\_local1388892213\_0001\_r\_000000\_0: Counters: 30  
File System Counters  
FILE: Number of bytes read=4550  
FILE: Number of bytes written=651235  
FILE: Number of read operations=0  
FILE: Number of large read operations=0  
FILE: Number of write operations=0  
HDFS: Number of bytes read=1685  
HDFS: Number of bytes written=120  
HDFS: Number of read operations=10  
HDFS: Number of large read operations=0  
HDFS: Number of write operations=3  
HDFS: Number of bytes read erasure-coded=0  
Map-Reduce Framework  
Combine input records=0

```
Combine output records=0
Reduce input groups=100
Reduce shuffle bytes=1391
Reduce input records=100
Reduce output records=10
Spilled Records=100
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=0
Total committed heap usage (bytes)=196083712
Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0
File Output Format Counters
  Bytes Written=120
2025-04-23 15:24:52,828 INFO mapred.LocalJobRunner: Finishing task:
attempt_local1388892213_0001_r_000000_0
2025-04-23 15:24:52,828 INFO mapred.LocalJobRunner: reduce task executor complete.
2025-04-23 15:24:53,412 INFO mapreduce.Job: Job job_local1388892213_0001 running
in uber mode : false
2025-04-23 15:24:53,414 INFO mapreduce.Job: map 100% reduce 100%
2025-04-23 15:24:53,416 INFO mapreduce.Job: Job job_local1388892213_0001
completed successfully
2025-04-23 15:24:53,435 INFO mapreduce.Job: Counters: 36
File System Counters
  FILE: Number of bytes read=6286
  FILE: Number of bytes written=1301079
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=3370
  HDFS: Number of bytes written=120
  HDFS: Number of read operations=15
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=4
  HDFS: Number of bytes read erasure-coded=0
Map-Reduce Framework
  Map input records=100
  Map output records=100
  Map output bytes=1185
  Map output materialized bytes=1391
  Input split bytes=97
```

```

Combine input records=0
Combine output records=0
Reduce input groups=100
Reduce shuffle bytes=1391
Reduce input records=100
Reduce output records=10
Spilled Records=200
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=6
Total committed heap usage (bytes)=392167424

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=1685
File Output Format Counters
  Bytes Written=120
2025-04-23 15:24:53,435 INFO streaming.StreamJob: Output directory:
/output/weather_output

```

## 7. View Output

```

hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -ls /output/weather_output/
2025-04-23 15:25:17,254 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Found 2 items
-rw-r--r--  1 hduser supergroup          0 2025-04-23 15:24
/output/weather_output/_SUCCESS
-rw-r--r--  1 hduser supergroup       120 2025-04-23 15:24
/output/weather_output/part-00000

```

```

hduser@pvg-HP-ProDesk-400-G4-SFF:~$ hdfs dfs -cat
/output/weather_output/part-00000
2025-04-23 15:26:12,583 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
1950      -18    43
1951      -17    44
1952      -12    32
1953      -20    41
1954      -13    40
1955      -16    45
1956      -14    33

```

1957	-19	38
1958	-20	28
1959	-19	40

## 8. Stop HDFS

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ stop-dfs.sh
```

```
Stopping namenodes on [localhost]
```

```
Stopping datanodes
```

```
Stopping secondary namenodes [Ubuntu]
```

```
2025-04-23 15:26:40,461 WARN util.NativeCodeLoader: Unable to load native-hadoop  
library for your platform... using builtin-java classes where applicable
```

```
hduser@pvg-HP-ProDesk-400-G4-SFF:~$ stop-yarn.sh
```

```
Stopping nodemanagers
```

```
Stopping resourcemanager
```



## **CONCLUSION:**

The MapReduce programming model simplifies the processing of large datasets across distributed systems by dividing tasks into two main phases: Map and Reduce. In this assignment, we implemented a MapReduce-based application to process weather data and extract useful insights, such as the minimum and maximum temperatures for each year.