Practical No: 01

Title: 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

Objective:

1. Implement a distributed application using Remote Procedure Call (RPC) to allow a client to submit an integer value to the server.

2. Enable the server to calculate the factorial of the received integer and return the result to the client program.

3. Demonstrate the functionality and efficiency of RPC in remote computation tasks.

Outcome:

- Successfully implement an RPC-based client-server architecture for remote factorial
- calculation.
- Demonstrate the ease of implementing distributed applications using RPC.
- Python (3.x recommended)
- Jupyter Notebook or any Python IDE

Hardware Requirement:

A machine with sufficient RAM and processing power for model training (8GB RAM recommended)

- Basic understanding of Python programming
- Familiarity with the concepts of RPC

What is RPC?

Remote Procedure Call (RPC) is an interprocess communication technique. The Full form of RPC is Remote Procedure Call. It is used for client-server applications. RPC mechanisms are used when a computer program causes a procedure or subroutine to execute in a different address space, which is coded as a normal procedure call without the programmer specifically coding the details for the remote interaction.

This procedure call also manages low-level transport protocol, such as User Datagram Protocol, Transmission Control Protocol/Internet Protocol etc. It is used for carrying the message data between programs.

Types of RPC

Three types of RPC are:

- Callback RPC

- Broadcast RPC
- Batch-mode RPC

Callback RPC

This type of RPC enables a P2P paradigm between participating processes. It helps a process to be both client and server services.

Functions of Callback RPC:

- Remotely processed interactive application problems
- Offers server with clients handle
- Callback makes the client process wait
- Manage callback deadlocks
- It facilitates a peer-to-Peer paradigm among participating processes.

Broadcast RPC

Broadcast RPC is a client's request, that is broadcast on the network, processed by all servers which have the method for processing that request.

Functions of Broadcast RPC:

- Allows you to specify that the client's request message has to be broadcasted.
- You can declare broadcast ports.
- It helps to reduce the load on the physical network

Batch-mode RPC

Batch-mode RPC helps to queue, separate RPC requests, in a transmission buffer, on the clientside, and then send them on a network in one batch to the server.

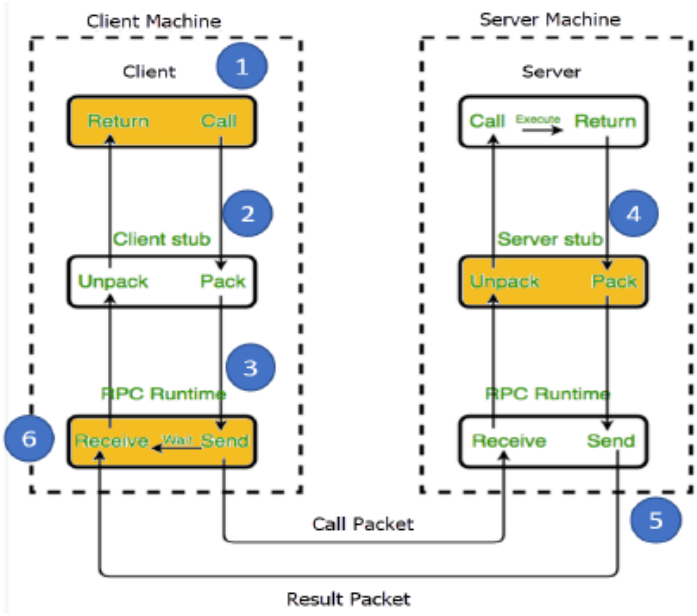
Functions of Batch-mode RPC:

- It minimizes overhead involved in sending a request as it sends them over the network in one batch to the server.
- This type of RPC protocol is only efficient for the application that needs lower call rates.
- It needs a reliable transmission protocol.

RPC Architecture

RPC architecture has mainly five components of the program:

1. Client
2. Client Stub
3. RPC Runtime
4. Server Stub
5. Server



How RPC Works?

Following steps take place during the RPC process:

Step 1) The client, the client stub, and one instance of RPC run time execute on the client machine.

Step 2) A client starts a client stub process by passing parameters in the usual way. The client stub stores within the client's own address space. It also asks the local RPC Runtime to send back to the server stub.

Step 3) In this stage, RPC accessed by the user by making regular Local Procedural Cal. RPC Runtime manages the transmission of messages between the network across client and server. It also performs the job of retransmission, acknowledgment, routing, and encryption.

Step 4) After completing the server procedure, it returns to the server stub, which packs (marshalls) the return values into a message. The server stub then sends a message back to the transport layer.

Step 5) In this step, the transport layer sends back the result message to the client transport layer, which returns back a message to the client stub.

Step 6) In this stage, the client stub de marshalls (unpack) the return parameters, in the resulting packet, and the execution process returns to the caller.

Characteristics of RPC

Here are the essential characteristics of RPC:

The called procedure is in another process, which is likely to reside in another machine.

- The processes do not share address space.
- Parameters are passed only by values.
- RPC executes within the environment of the server process.
- It doesn't offer access to the calling procedure's environment.

Features of RPC

Here are the important features of RPC:

- Simple call syntax
- Offers known semantics
- Provide a well-defined interface
- It can communicate between processes on the same or different machines

Advantages of RPC

Here are Pros/benefits of RPC:

RPC method helps clients to communicate with servers by the conventional use of procedure calls in high-level languages.

- RPC method is modeled on the local procedure call, but the called procedure is most likely to be executed in a different process and usually a different computer.
- RPC supports process and thread-oriented models.
- RPC makes the internal message passing mechanism hidden from the user.
- The effort needs to re-write and re-develop the code is minimum.
- Remote procedure calls can be used for the purpose of distributed and the local environment.
- It commits many of the protocol layers to improve performance.

RPC provides abstraction. For example, the message-passing nature of network communication remains hidden from the user.

RPC allows the usage of the applications in a distributed environment that is not only in the local environment.

- With RPC code, re-writing and re-developing effort is minimized.
- Process-oriented and thread-oriented models support by RPC.

Disadvantages of RPC

Here are the cons/drawbacks of using RPC:

- Remote Procedure Call Passes Parameters by values only and pointer values, which is not allowed.
- Remote procedure calling (and return) time (i.e., overheads) can be significantly lower than that for a local procedure.
- This mechanism is highly vulnerable to failure as it involves a communication system, another machine, and another process.
- RPC concept can be implemented in different ways, which is can't standard.
- Not offers any flexibility in RPC for hardware architecture as It is mostly interaction-based.
- The cost of the process is increased because of a remote procedure call.

RPC Architecture To implement this in Python, we can use the xmlrpc library, which provides support for writing RPC servers and clients. The server program will create an XML-RPC server using SimpleXMLRPCServer, register a function to compute the factorial, and then start the server to listen for incoming requests. The client program will create an XML-RPC proxy object to communicate with the server, then call the remote procedure with the integer value as an argument to request the factorial calculation.

1st file:Factserver.py

```
from xmlrpc.server import SimpleXMLRPCServer
from xmlrpc.server import SimpleXMLRPCRequestHandler
class FactorialServer:
    def calculate_factorial(self, n):
        if n < 0:
            raise ValueError("Input must be a non-negative integer.")
        result = 1
        for i in range(1, n + 1):
            result *= i
        return result
# Restrict to a particular path.
class RequestHandler(SimpleXMLRPCRequestHandler):
    rpc_paths = ('/RPC2',)
# Create server
with SimpleXMLRPCServer(('localhost', 8000),
                        requestHandler=RequestHandler) as server:
    server.register_introspection_functions()
    # Register the FactorialServer class
    server.register_instance(FactorialServer())
    print("FactorialServer is ready to accept requests.")
    # Run the server's main loop
    server.serve_forever()
```

2nd file: Factclient.py

```
import xmlrpc.client
# Create an XML-RPC client
```

```
with xmlrpc.client.ServerProxy("http://localhost:8000/RPC2") as proxy:
    try:
        # Replace 5 with the desired integer value
        input_value = 5
        result = proxy.calculate_factorial(input_value)
        print(f"Factorial of {input_value} is: {result}")
    except Exception as e:
        print(f"Error: {e}")
```

Execution Steps

1. Open Command Prompt:

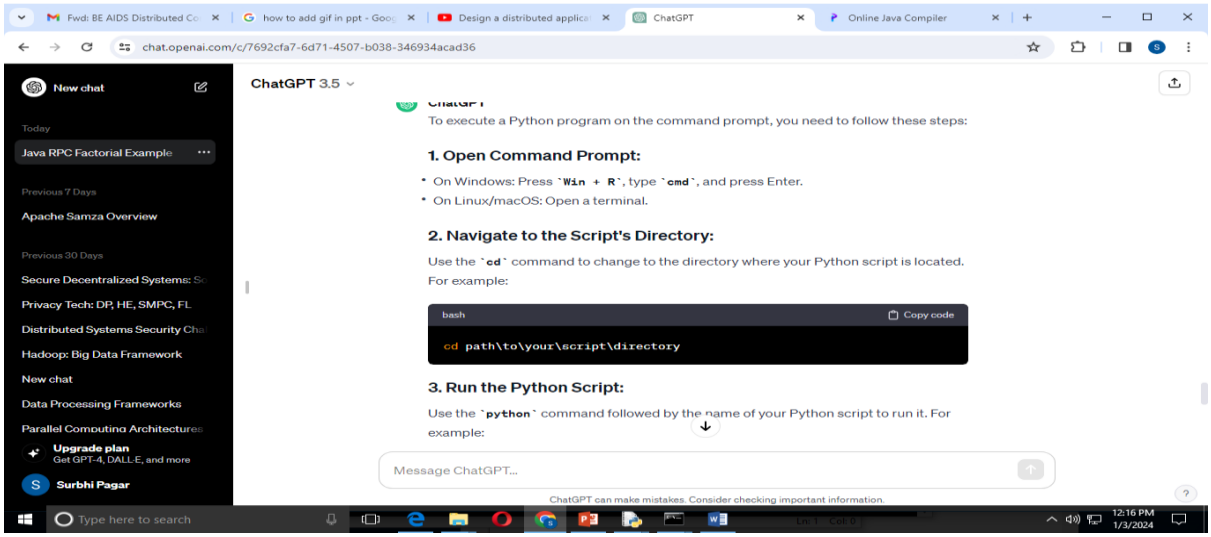
On Windows: Press Win + R, type cmd, and press Enter.

On Linux/macOS: Open a terminal.

2. Navigate to the Script's Directory:

Use the **cd command** to change to the directory where your Python script is located.

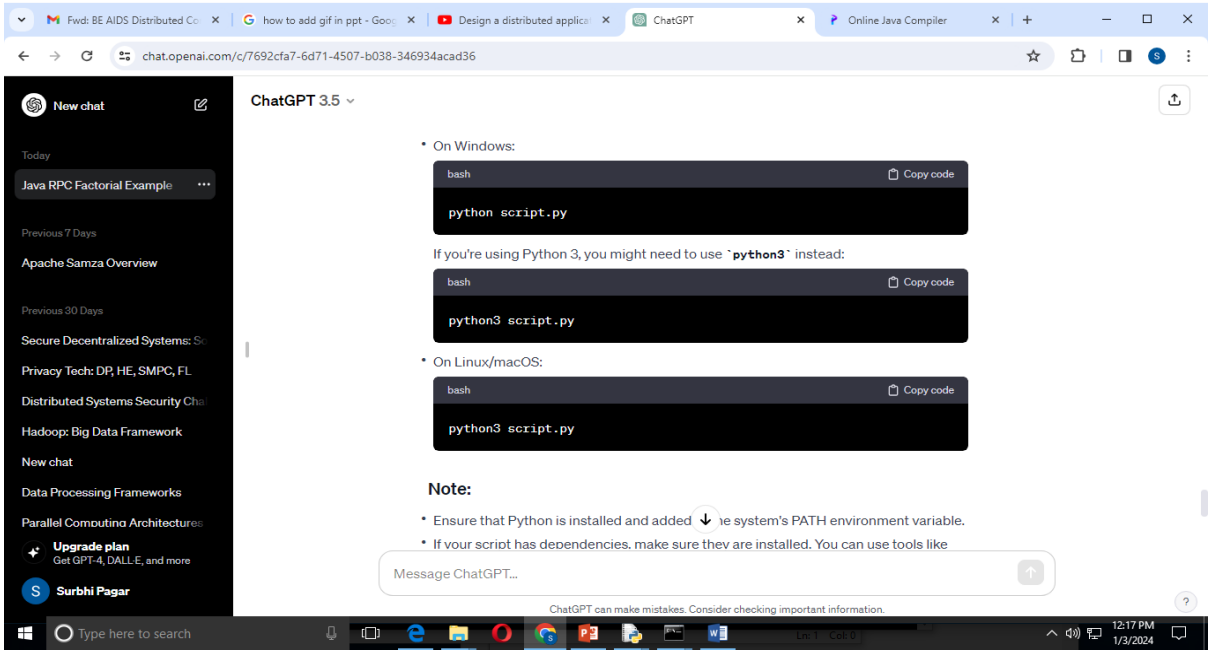
For example:



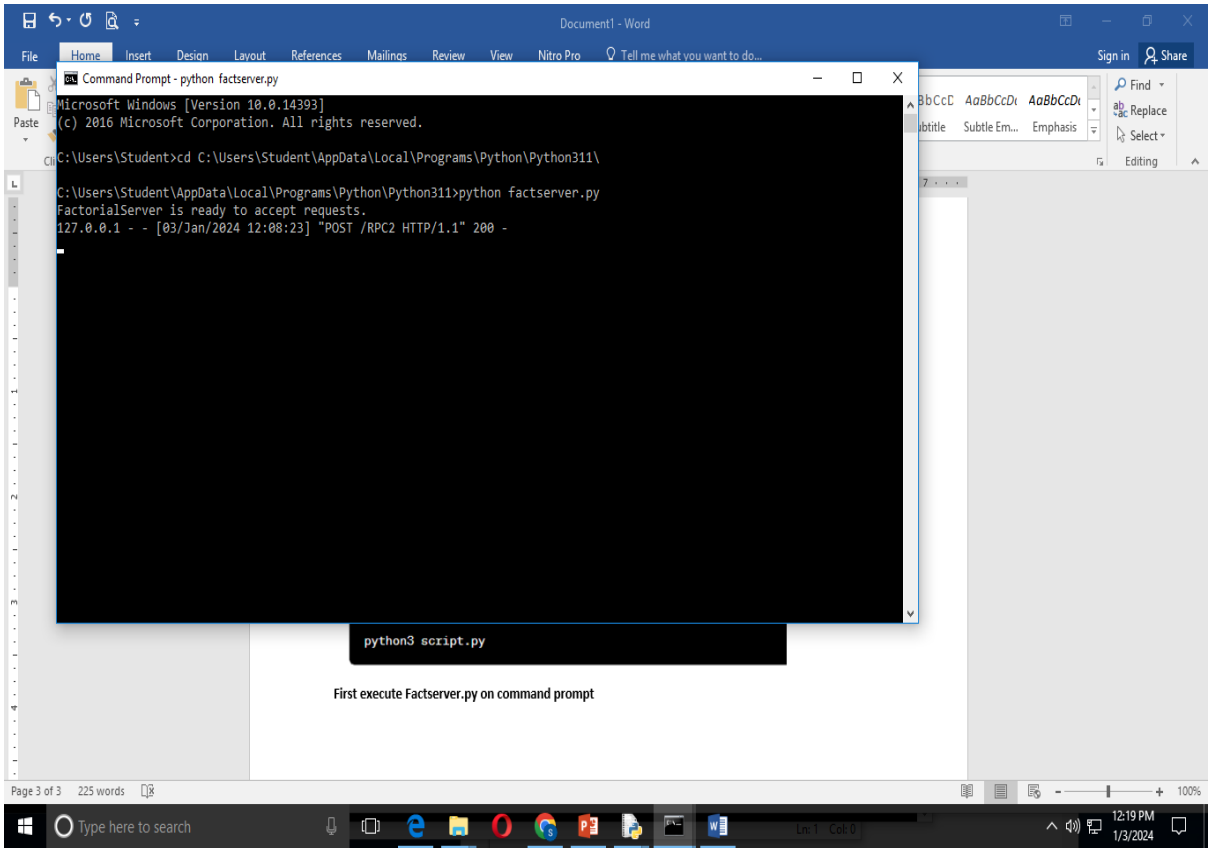
3. Run the Python Script:

Use the python command followed by the name of your Python script to run it.

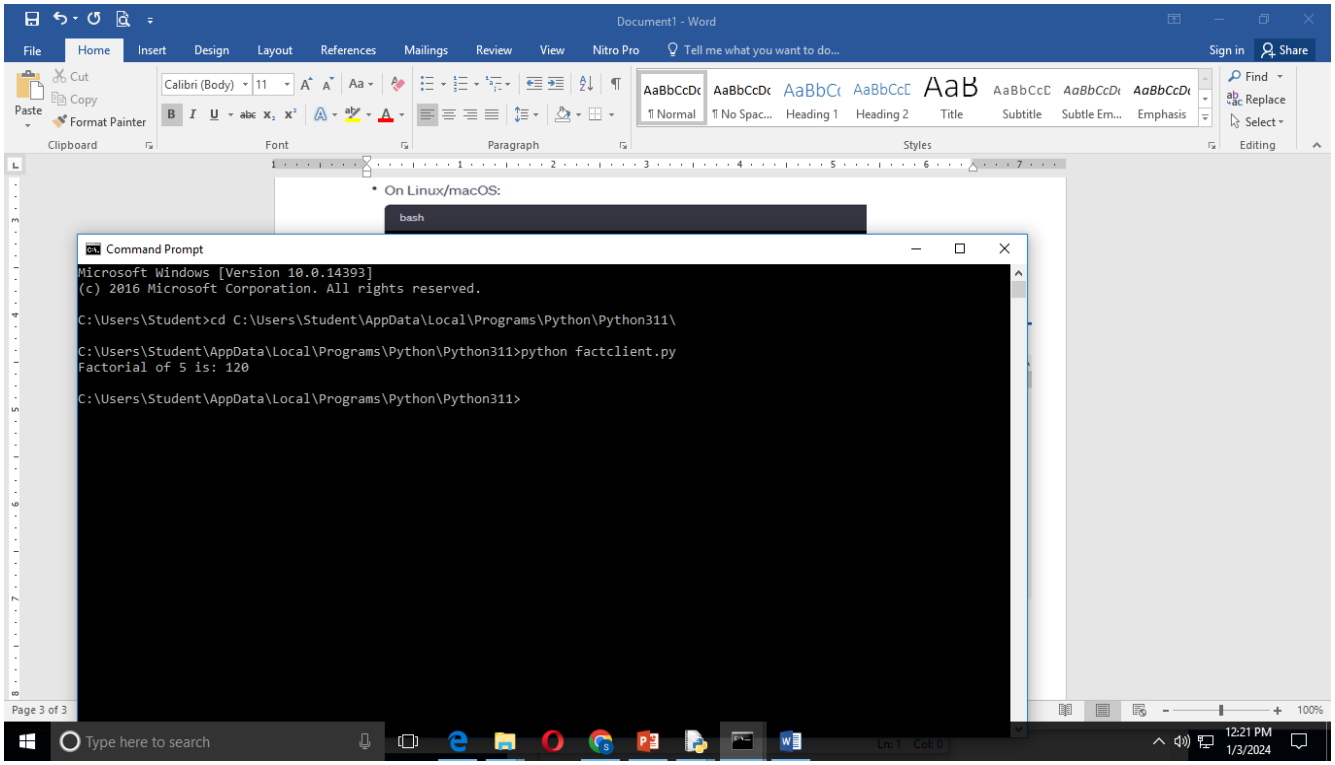
For example:



First execute factserver.py on command prompt



Then open another Command Prompt window and execute factclient.py



Conclusion:

Thus Implemented 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