

Remote Procedure Call (RPC) – Cloud-Based Calculator Application

1. Aim

To implement a **Remote Procedure Call (RPC)** based calculator application where the server is hosted on a **cloud platform (AWS EC2)** and the client remotely invokes arithmetic procedures.

2. Objective

- To understand RPC-based communication in distributed systems
 - To implement client–server interaction using RPC
 - To deploy the RPC server on a cloud environment
 - To verify remote procedure execution and result transfer
-

3. System Requirements

Hardware

- Computer with minimum 4 GB RAM
- Internet connection

Software

- Operating System: Ubuntu (Server), Windows / Kali Linux (Client)
 - Programming Language: Python
 - Cloud Platform: AWS EC2
-

4. RPC Architecture

- RPC follows a **procedure-oriented client–server model**
 - The client invokes remote procedures as normal function calls
 - The server executes the procedure and returns the result
 - Communication is handled transparently over the network
-

5. RPC Implementation Details

5.1 Remote Procedures

The server provides the following remote procedures:

- Addition
- Subtraction
- Multiplication
- Division

5.2 Working Principle

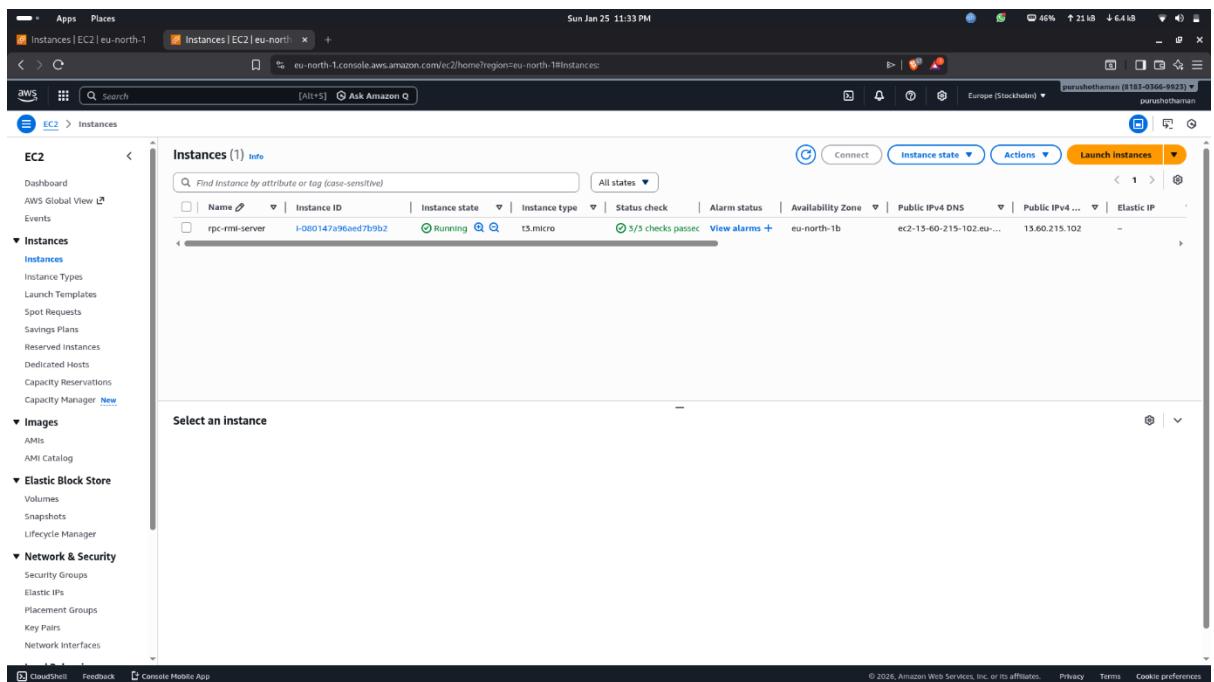
1. RPC server defines arithmetic procedures
2. Server listens on a specific port
3. Client connects using server IP address and port number
4. Client invokes procedures remotely
5. Server processes the request and returns the result

6. Cloud Deployment

- RPC server is hosted on **AWS EC2**
- Public IP address is used for client access
- Required ports are enabled in the EC2 security group

Attach the following screenshots:

- AWS EC2 instance running
- Security group inbound rule configuration



7. Error Handling

- Invalid inputs are handled safely
 - Division by zero is checked and prevented
 - Network-related exceptions are handled gracefully
-

8. Output

- RPC server running on AWS EC2

```
Sun Jan 25 10:10 PM
ubuntu@ip-172-31-39-106:~>

[+]purushoth@kali:[-]
$ ssh -i rpc-rm1-key.pem ubuntu@13.60.215.102
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.6-1040-aws x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/pro

System information as of Sun Jan 25 16:38:24 UTC 2024

System load: 0.08      Processes:          106
Usage of /: 40.9% of 7.57GB   Users logged in:    0
Memory usage: 27%        IPv4 address for ens5: 172.31.39.106
Swap usage: 0%          Swap usage: 0%

* Ubuntu Pro delivers the most comprehensive open source security and
  compliance features.

  https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

:3 updates can be applied immediately.

:6 see ESM Apps to receive additional future security updates.

Enable ESM Apps to receive additional future security updates.

See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.3 LTS' available.

Run 'do-release-upgrade' to upgrade to it.

** System restart required **

Last login: Sun Jan 25 15:01:39 2024 from 14.139.161.3
ubuntu@ip-172-31-39-106:~$ python3 rpc_server.py
RPC Server running on port 8000
14.139.161.3 - - [25/Jan/2024 16:40:14] "POST /RPC2 HTTP/1.1" 200 -
14.139.161.3 - - [25/Jan/2024 16:40:42] "POST /RPC2 HTTP/1.1" 200 -
14.139.161.3 - - [25/Jan/2024 16:40:53] "POST /RPC2 HTTP/1.1" 200 -
14.139.161.3 - - [25/Jan/2024 16:41:00] "POST /RPC2 HTTP/1.1" 200 -
```

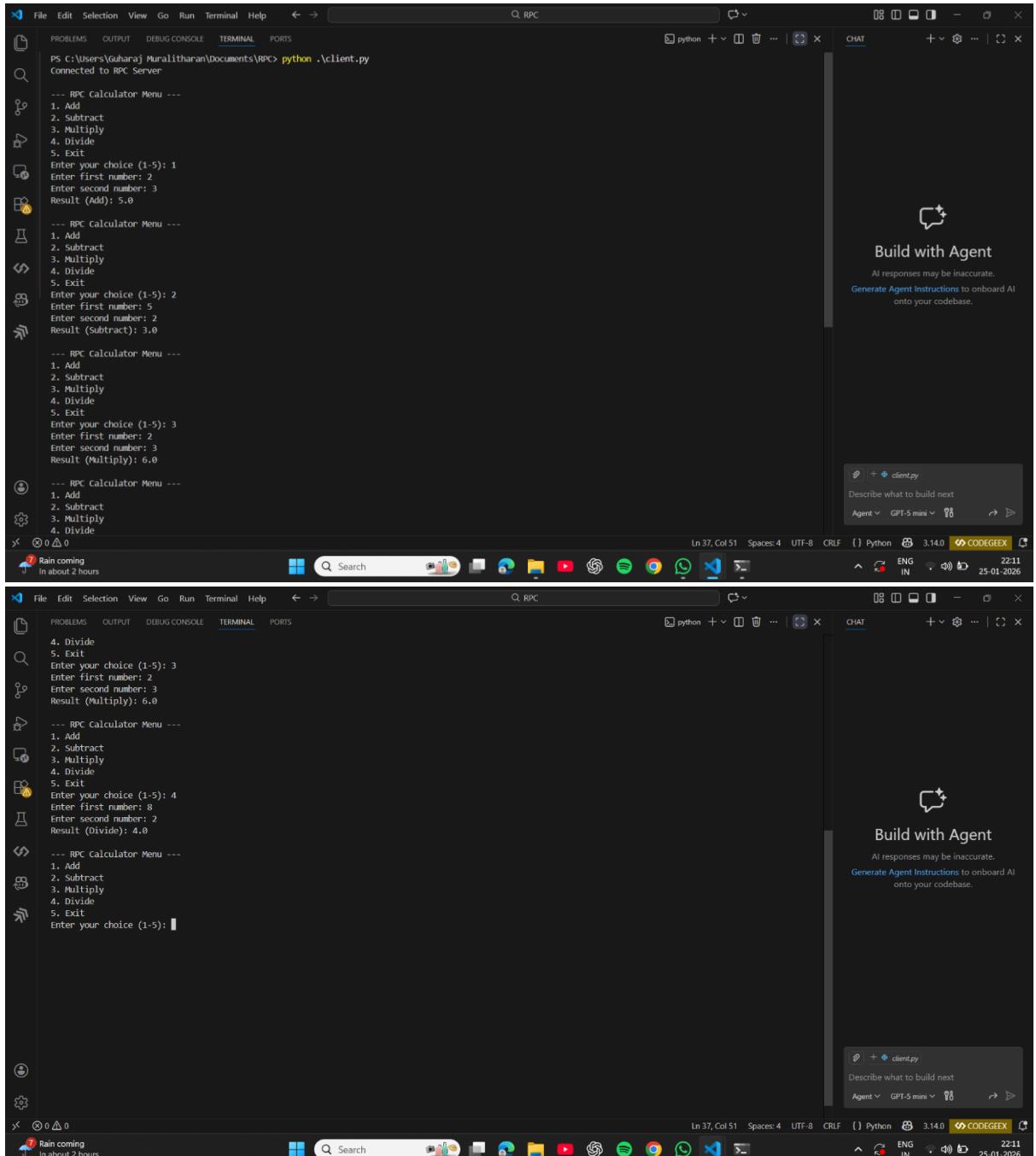
- Client invoking remote procedures

```
Mon Jan 26 1:18 PM
ubuntu@ip-172-31-39-106:~>

[+]purushoth@kali:[-]
$ python3 RPC_Server.py
RPC_Server.py

[GNOME Terminal 8.2]
from xmlrpc.server import SimpleXMLRPCServer
def add(a, b):
    return a + b
def subtract(a, b):
    return a - b
def multiply(a, b):
    return a * b
def divide(a, b):
    if b == 0:
        return "Error: Division by zero"
    return a / b
server = SimpleXMLRPCServer(("0.0.0.0", 8000))
print("RPC Server running on port 8000")
server.register_function(add)
server.register_function(subtract)
server.register_function(multiply)
server.register_function(divide)
server.serve_forever()
```

- Correct arithmetic results displayed



```

File Edit Selection View Go Run Terminal Help ↵ → 🔍 RPC
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Guhana\Documents\RPC> python ./client.py
Connected to RPC Server
--- RPC Calculator Menu ---
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter your choice (1-5): 1
Enter first number: 2
Enter second number: 3
Result (Add): 5.0

--- RPC Calculator Menu ---
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter your choice (1-5): 2
Enter first number: 5
Enter second number: 2
Result (Subtract): 3.0

--- RPC Calculator Menu ---
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter your choice (1-5): 3
Enter first number: 2
Enter second number: 3
Result (Multiply): 6.0

--- RPC Calculator Menu ---
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter your choice (1-5): 4
Enter first number: 8
Enter second number: 2
Result (Divide): 4.0

--- RPC Calculator Menu ---
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter your choice (1-5): 5

```

The screenshot shows two terminal windows side-by-side. Both windows are running Python 3.14.0 on a Windows 10 desktop. The top window displays the execution of `python ./client.py`, which connects to an RPC server and performs four operations: addition, subtraction, multiplication, and division. The bottom window shows the continuation of the client's menu selection, specifically choosing to exit. A sidebar on the right of both windows includes a "Build with Agent" feature and a "client.py" file entry.

9. Result

The RPC-based calculator application was successfully implemented. The client remotely invoked procedures hosted on the cloud server and received correct computation results.

10. Conclusion

This experiment demonstrated how RPC simplifies distributed computing by allowing remote functions to be invoked as local procedures. Deploying the server on AWS EC2 provided practical experience with cloud-based distributed systems