RPC Implementation in Cloud Environment

Name:Nagasurya
Roll No:2023115106

Aim

To design and implement a **Remote Procedure Call (RPC) based distributed application** using **Python**, where the **server is hosted in a cloud environment (Microsoft Azure VM)** and the **client runs on a local machine**, enabling remote procedure invocation over a network.

Software & Tools Used

Component	Description
Programming Language	Python 3
RPC Mechanism	HTTP-based RPC
Cloud Platform	Microsoft Azure
Server OS	Ubuntu Linux (Azure VM)
Client OS	Windows 10

Libraries Used	http.server, json, requests

System Architecture

- The **RPC Server** is deployed on an **Azure Virtual Machine**
- The **Client** runs on a **local Windows system**
- Client sends requests using HTTP POST
- Server processes the request and returns results in JSON format

Client (Windows) → Azure VM (RPC Server)

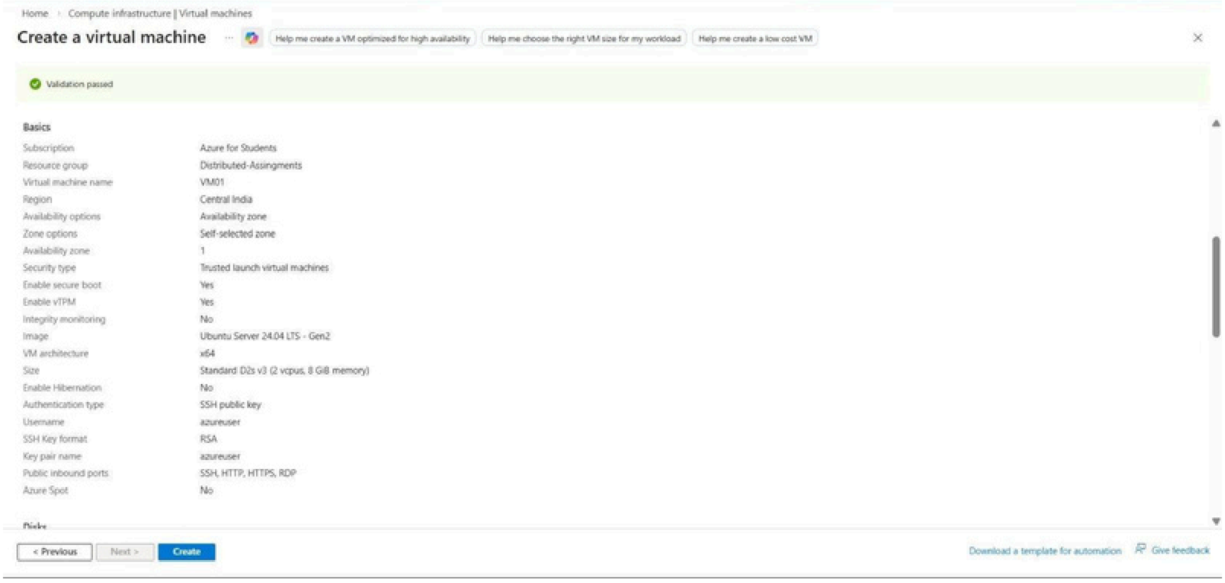
Remote Procedures Implemented

Procedure Name	**Description**
multipleyMatrix	Perform Matrix Multiplication
getPrimes	Print prime numbers in a range

Execution Steps

Step 1: Start Azure VM

- Login to Azure Portal
- Start Ubuntu Virtual Machine
- Enable inbound rule for port **8000**



Home › Compute infrastructure | Virtual machines
Create a virtual machine
Help me create a VM optimized for high availability
Help me choose the right VM size for my workload
Help me create a low cost VM
✓ Validation passed

Basics

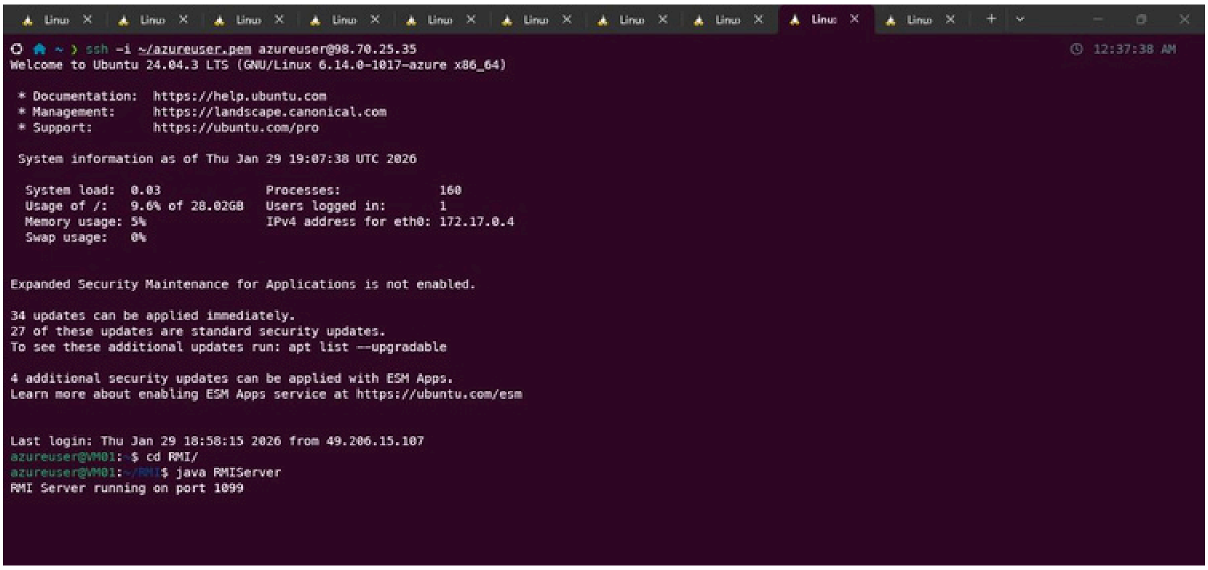
Subscription	Azure for Students
Resource group	Distributed-Assingments
Virtual machine name	VM01
Region	Central India
Availability options	Availability zone
Zone options	Self-selected zone
Availability zone	1
Security type	Trusted launch virtual machines
Enable secure boot	Yes
Enable vTPM	Yes
Integrity monitoring	No
Image	Ubuntu Server 24.04 LTS - Gen2
VM architecture	x64
Size	Standard D2s v3 (2 vcpus, 8 GiB memory)
Enable Hibernation	No
Authentication type	SSH public key
Username	azureuser
SSH Key format	RSA
Key pair name	azureuser
Public inbound ports	SSH, HTTP, HTTPS, RDP
Azure Spot	No

Step 2: Run RPC Server on Azure VM

python3 rpc_server.py

Output:

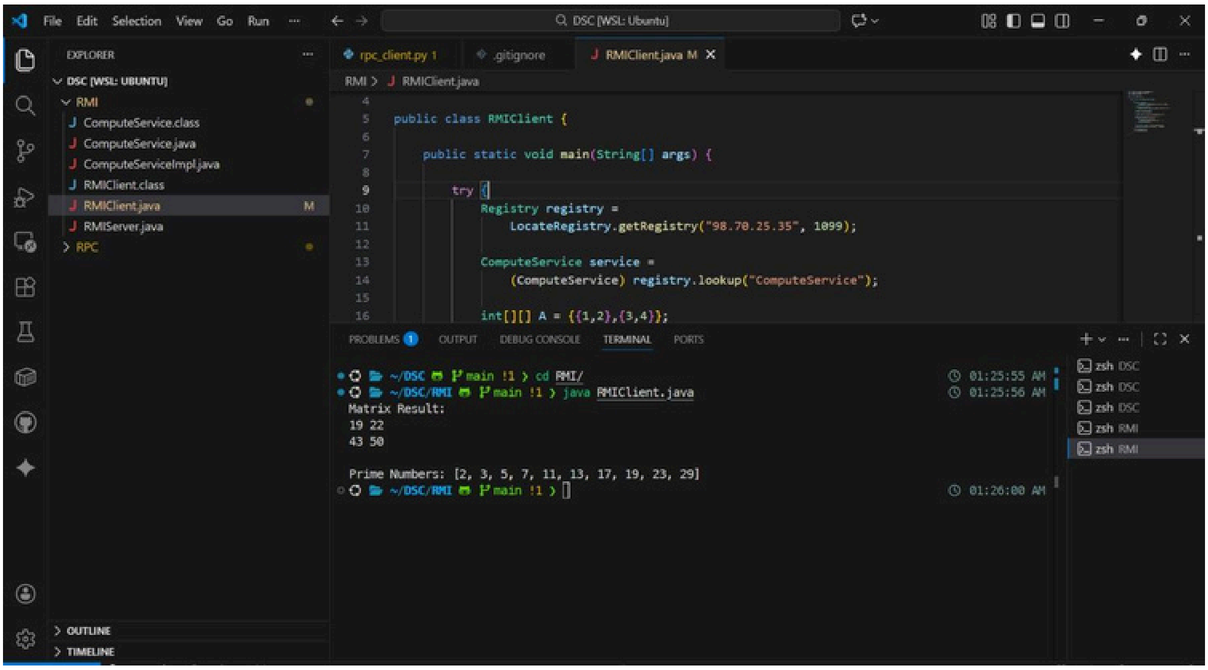
RPC Server running on Azure VM at port 8000



```
~ › ssh -i ~/azureuser.pem azureuser@98.70.25.35
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1017-azure x86_64)

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

 System information as of Thu Jan 29 19:07:38 UTC 2026

  System load:  0.03              Processes:             160
  Usage of /:   9.6% of 28.02GB   Users logged in:       1
  Memory usage: 5%                IPv4 address for eth0: 172.17.0.4
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

34 updates can be applied immediately.
27 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

4 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


Last login: Thu Jan 29 18:58:15 2026 from 49.206.15.107
azureuser@VM01:~$ cd RMI/
azureuser@VM01:~/RMI$ java RMIServer
RMI Server running on port 1099
```

Step 3: Run Client on Local Machine

python rpc_client.py



```java

public class RMIClient {

    public static void main(String[] args) {

        try {
            Registry registry =
                LocateRegistry.getRegistry("98.70.25.35", 1099);

            ComputeService service =
                (ComputeService) registry.lookup("ComputeService");

            int[][] A = {{1,2},{3,4}};
```

```
~/DSC  main !1 > cd RMI/                                    01:25:55 AM
~/DSC/RMI  main !1 > java RMIClient.java                    01:25:56 AM
Matrix Result:
19 22
43 50

Prime Numbers: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
~/DSC/RMI  main !1 >                                        01:26:00 AM
```

Error Handling

- Invalid RPC endpoints return **404 error**
- JSON parsing errors handled by server
- Network connectivity verified using Azure NSG rules

Result

Thus, a **Remote Procedure Call (RPC) based distributed application** was successfully implemented and executed in a **cloud environment using Microsoft Azure**, allowing a remote client to invoke procedures hosted on the server and receive correct results.

Conclusion

The experiment demonstrates how RPC enables transparent communication between distributed systems. Hosting the server in the cloud allows scalability and remote access, making RPC suitable for real-world distributed applications.