# Ex No: 3 Implementation of RMI across VMs and Date: 18.03.2022 Study of VM Performance

**Aim**

1. Study of Virtualization Platforms
2. RMI program Execution in Physical and Virtual platforms and compare the speed and efficiency metrics

# Study of Virtualization Platforms

A Virtual Machine (VM) is a compute resource that uses software instead of a physical computer to run programs and deploy apps. One or more virtual “guest” machines run on a physical “host” machine Code

# VirtualBox

Oracle VM VirtualBox is a type-2 hypervisor for x86 virtualization developed by Oracle Corporation.

VirtualBox was originally created by Innotek GmbH, which was acquired by Sun Microsystems in 2008, which was in turn acquired by Oracle in 2010.

# VMware Workstation

VMware Workstation Pro is a hosted hypervisor that runs on x64 versions of Windows and Linux operating systems; it enables users to set up virtual machines on a single physical machine and use them simultaneously along with the host machine.

# Virtual Iron

Virtual Iron Software, was located in Lowell, Massachusetts, sold proprietary software for the virtualization and management of a virtual infrastructure

# Microsoft Hyper-V

Microsoft Hyper-V, codenamed Viridian, and briefly known before its release as Windows Server Virtualization, is a native hypervisor; it can create virtual machines on x86-64 systems running Windows

# RMI program Execution in Physical and virtual platforms: Code:

**Server:**

import java.rmi.\*;

import java.rmi.registry.\*; public class MyServer{

public static void main(String args[]){ try{

Adder stub=new AdderRemote(); Naming.rebind("rmi://localhost:2228/navin",stub);

}catch(Exception e){System.out.println(e);}

}

}

# Client

import java.rmi.\*; import java.util.\*; public class MyClient{

public static void main(String args[]){ try{

Adder stub=(Adder)Naming.lookup("rmi://172.17.17.176:2228/navin"); Scanner s = new Scanner(System.in);

System.out.println("Enter number: "); int num1 = s.nextInt();

System.out.println("Enter another number: "); int num2 = s.nextInt();

System.out.println("1. Add/n2. Subtract/n3. Multiply/n4. Divide/n"); int ch = s.nextInt();

if(ch==1)

{

System.out.println(stub.add(num1,num2));

}

else if(ch==2)

{

System.out.println(stub.subtract(num1,num2));

}

else if(ch==3)

{

System.out.println(stub.multiply(num1,num2));

}

else if(ch==4)

{

System.out.println(stub.divide(num1,num2));

}

else{

System.out.println("Wrong option");

}

}catch(Exception e){System.out.println(e);}

}

}

# remote:

import java.rmi.\*; import java.rmi.server.\*;

public class AdderRemote extends UnicastRemoteObject implements Adder{ AdderRemote()throws RemoteException{

super();

}

public int add(int x,int y){return x+y;} public int subtract(int x,int y){return x-y;} public int multiply(int x,int y){return x\*y;} public int divide(int x,int y){return x/y;}

}

# Function:

import java.rmi.\*;

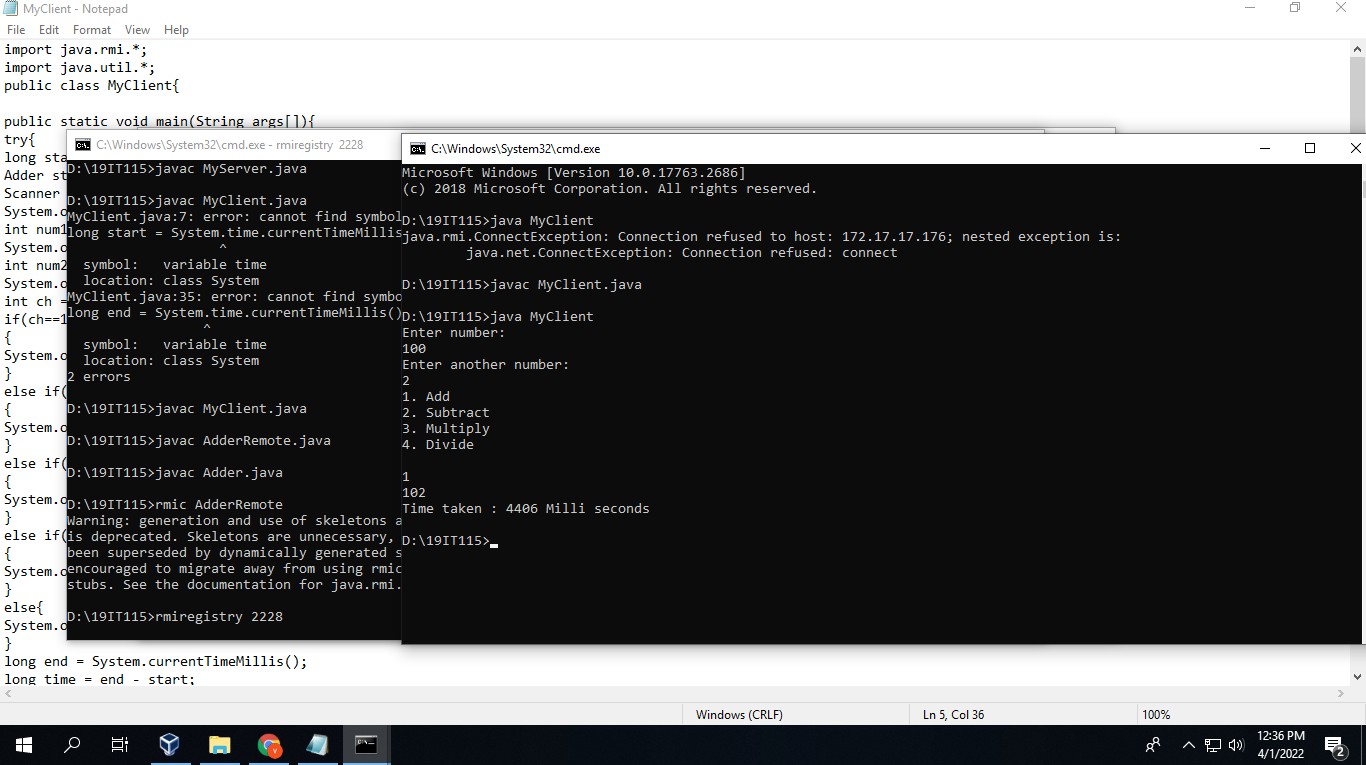
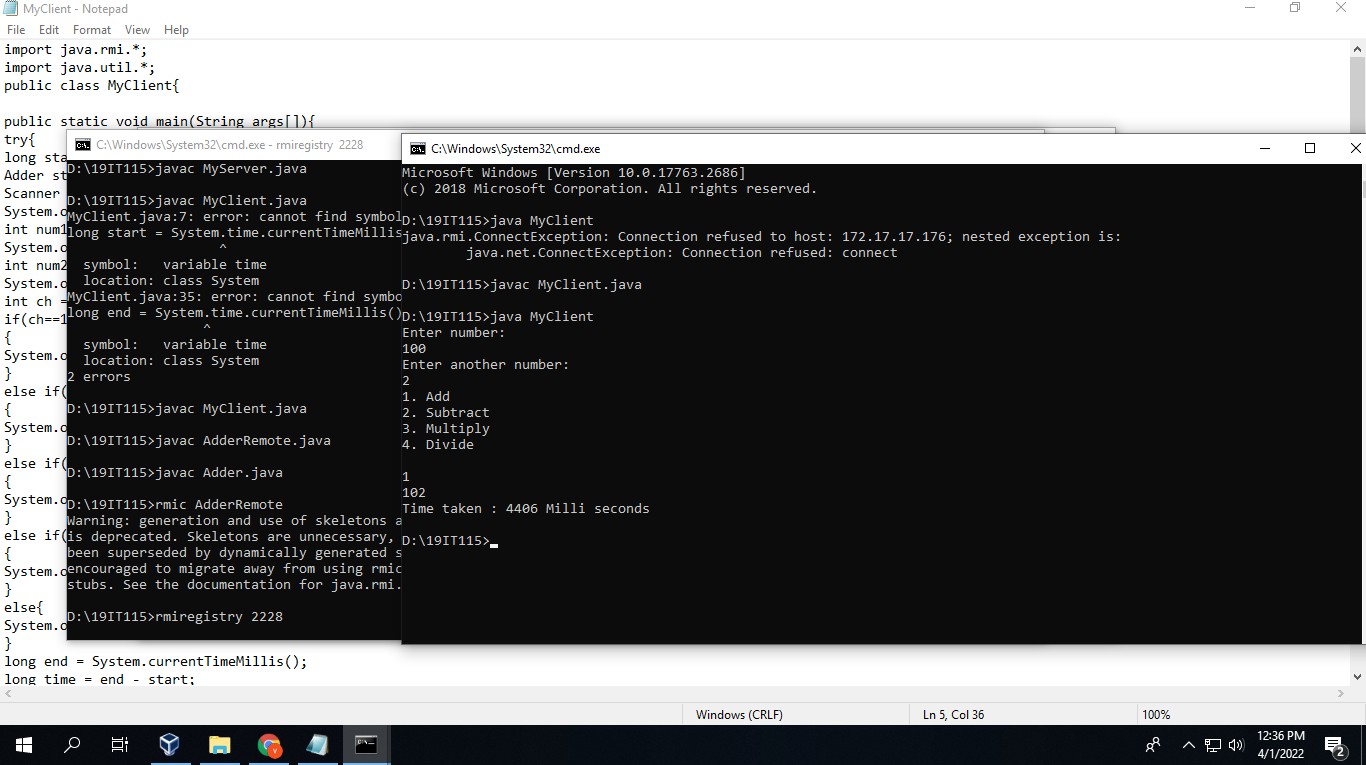
public interface Adder extends Remote{

public int add(int x,int y)throws RemoteException; public int subtract(int x,int y)throws RemoteException; public int multiply(int x,int y)throws RemoteException; public int divide(int x,int y)throws RemoteException;

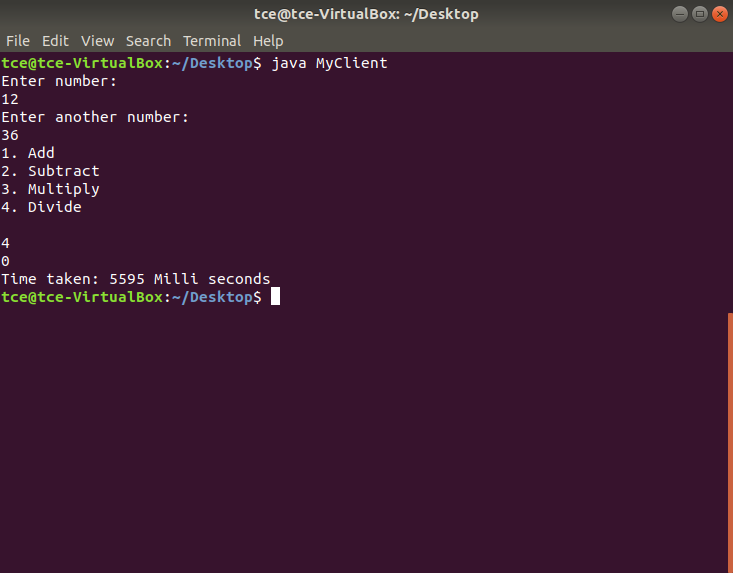
}

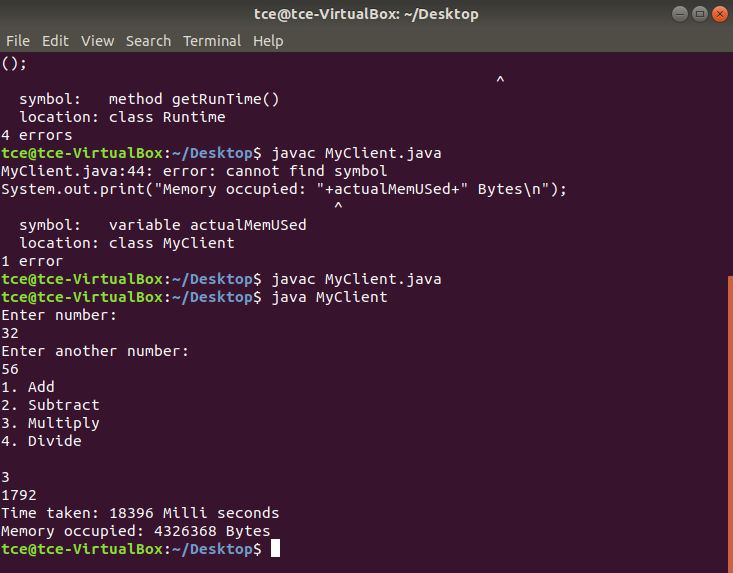
# Output of RMI program:

In Physical machine:

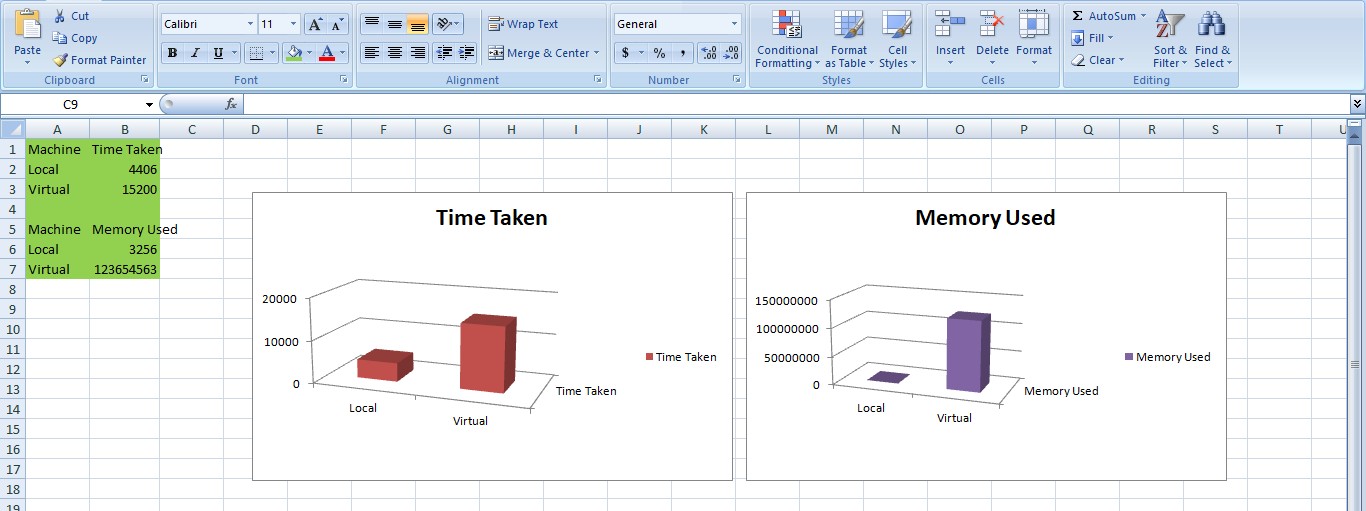
Memory consumed by the physical machine:

# In virtual Machines:





**Graph – Time taken comparison between Virtual and Local machines**



# Result:

Thus, various virtual machine platforms were studied and comparison was made on the speed of RMI across virtual and physical machine