CORBA Based Distributed Framework for GPGPU processing

By

Yassser Khan
ID: 2011-2-60-035
&
Monirul Islam
ID: 2011-3-60-028

Supervised By

Dr. Md. Shamim Akhter
Assistant Professor
Department of Computer Science and Engineering
East West University

A Project Submitted in Partial Fulfillment of the Requirements for the Degree of Bachelors of Science in Computer Science and Engineering

to the

Department of Computer Science and Engineering
East West University
Dhaka, Bangladesh

Declaration

We hereby declare that, this report was done under CSE497 and has not been submitted elsewhere for requirement of any degree or diploma or for any purpose except for publication

Yasser Khan
ID: 2011-2-60-035
Department of Computer Science and Engineering
East West University

Monirul Islam
ID: 2012-3-60-028
Department of Computer Science and Engineering
East West University

Letter of Acceptance

I hereby declare that this thesis is from the student's own work and best effort of mine, and all other sources of information used have been acknowledged. This thesis has been submitted with our approval.

____

Dr. Md. Shamim Akhter **Supervisor**
Assistant Professor
Department of Computer Science and Engineering
East West University

____

Dr. Mozammel Huq Azad Khan **Chairperson**
Chairperson and Professor
Department of Computer Science and Engineering
East West University

Acknowledgement

We would like to thank and pay our sincere gratitude to our mentor and thesis Supervisor Dr. Md. Shamim Akhter for supporting us throughout our endeavor with his extensive knowledge of the related grounds, his tremendous efforts and robustness that compelled us succeeding endure the ordeals and his extraordinary visions that enthralled us to refine our work to the finest point.

We also would like to thank our fellow acquaintance Tawsif F. Rahman for providing us with a much clearer idea of the requirements that we are expected to achieve.

We thank our beloved family and friends, for supporting us throughout the time of our experiments that allowed us to be able to come up with what we really have today.

Abstract

We present a CORBA based distributed system that implemented to execute CUDA program from a remote GPU enable machine. CPU is unable to execute that program. So, we introduced CORBA based distributed system to provide the services so that CPU users can get the facilities of GPU based system. Where CPU users will act as client and GPU based system will act as server. Clients can request to the server to use its' GPU based system which is costly setup for users.

Table of Content

Contents

List of Figure

Chapter 1

INTRODUCTION

Introduction

1.1 Overview

GPU enabled machines are more costly compared to CPU enabled machines. . A simple way to understand the difference between a CPU and GPU is to compare how they process tasks. GPU can implement a lot of task at a single time where CPU is unable to do that. So, we have introduced CORBA based distributed system which is language and platform independent. CPU users can invoke to GPU enabled machines so that CPU users can use GPGPU processing.

1.2 Scope and limitation

Scope:

- Faster calculation of enormous size of data rather than serial implementations,
- lesser memory consumptions,
- Rendition of previously introduced algorithms which lacks in the fields of requirements to be accepted as new standard.

1.3 Objective:

- To introduce CORBA architecture with GPGPU
- Providing GPU services to clients
- Saving consumptions of time
- Drawing a standard design so that the oversized data might not result in retaliate in long run by adding overhead to the total execution time

1.4 Contribution Outlining

In next chapter there are described a concept of Distributed system and Middleware. Chapter 3 contains CORBA architecture and Chapter 4 has GPGPU based architecture. At last we have some Security in Chapter 5 and Chapter 6 contains Implementations of our project.

Chapter 2

Distributed System

Distributed System

2.1 Overview

Now-a-days, computer networks are almost in everywhere. All of these networks such as factory networks, mobile phone networks, home networks, campus networks, corporate networks etc. are connected through the internet. Two computer devices may communicate with each other which can be situated in a same room or two other continents. So, computer networks can be separated by any distance. The best-known computer network is the Internet.

2.2 What is distributed system?

We define distributed system as one in which software or hardware located at networked computers communicates or coordinate their actions by passing messages. [1] A distributed architecture is an architecture supporting the development of applications and services that can exploit a physical architecture consisting of multiple, autonomous processing elements. Those elements do not share primary memory but cooperate by sending messages over the network. A distributed system is a collection of networks that appears to the users of the system as a single coherent system.

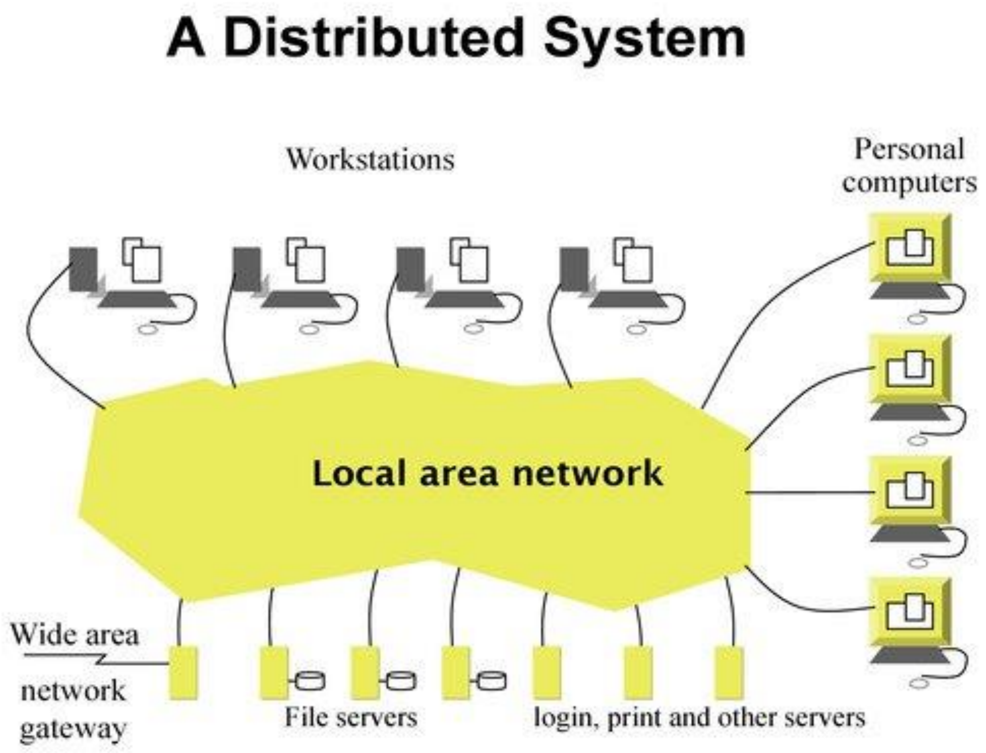


Figure 1 Distributed System Architecture

2.3 Middleware

A distributed system organized as middleware. A middleware layer runs on all machines, and offers a uniform interface to the system. Middleware is a general term for software that serves to "glue together" separate, often complex and already existing, programs. Some software components that are frequently connected with middleware include enterprise applications and Web services.[3]

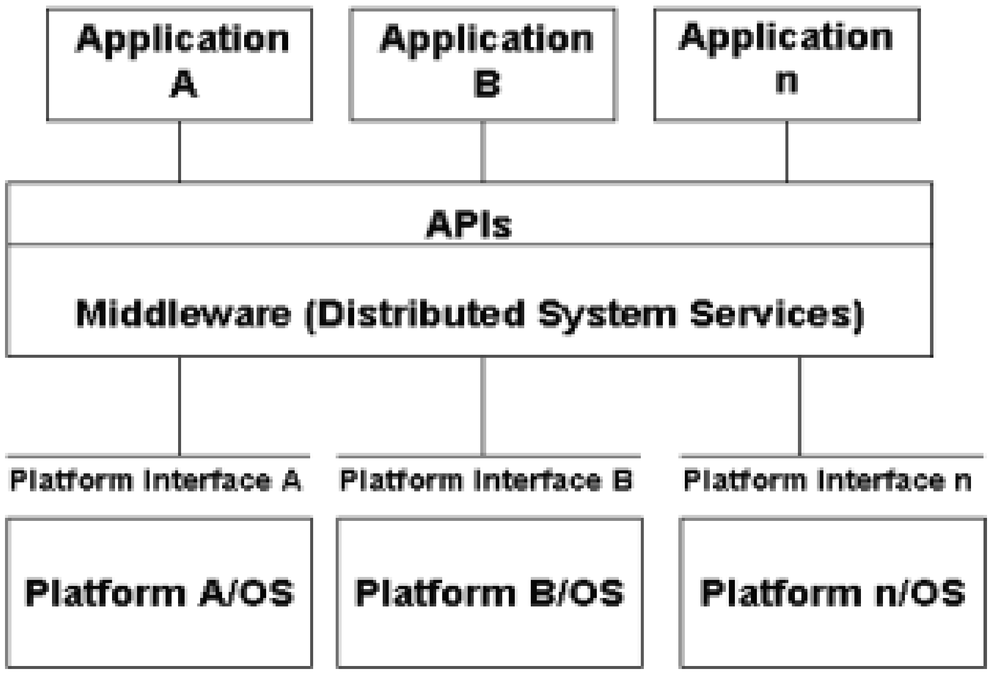


Figure 2 Middleware (Distributed system services)

2.3.1 Examples of Middleware

Different time different Middleware layers are introduced. Some of the Middleware layers :

RPC (Request Procedure Call)

RMI (Remote Method Invocation)

CORBA (Common Object Request Broker)

DCOM (Distributed Common Object Model)

Chapter 3

CORBA

CORBA

3.1 What is CORBA?

CORBA is a software standard that is defined and maintained by the Object Management Group (OMG). Common Object Request Broker Architecture (CORBA) is an architecture and specification for creating, distributing, and managing distributed program objects in a network. It allows programs at different locations and developed by different vendors to communicate in a network through an "interface broker".

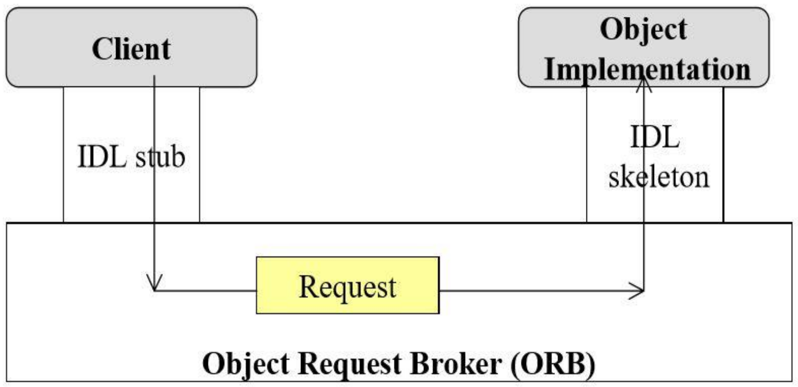


Figure 3 IDL and ORB

3.2 How CORBA architecture works

CORBA automates many common network programming tasks such as object registration, location, and activation; request demultiplexing; framing and error-handling; parameter marshalling and unmarshalling; and operation dispatching. The most essential concept of this architecture is Object Request Broker (ORB). ORB support in a network of clients and server on different computers that means a client program can request services from a server program or object without having to understand where the server is in a distributed network or what the interface to the server program looks like. To make requests or return replies between the ORBs, programs use the General Inter-ORB Protocol (GIOP) and, for the Internet, it's Internet Inter-ORB Protocol (IIOP). IIOP maps GIOP requests and replies to the Internet's Transmission Control Protocol (TCP) layer in each computer. CORBA is a software standard that is defined and maintained by the Object Management Group (OMG).

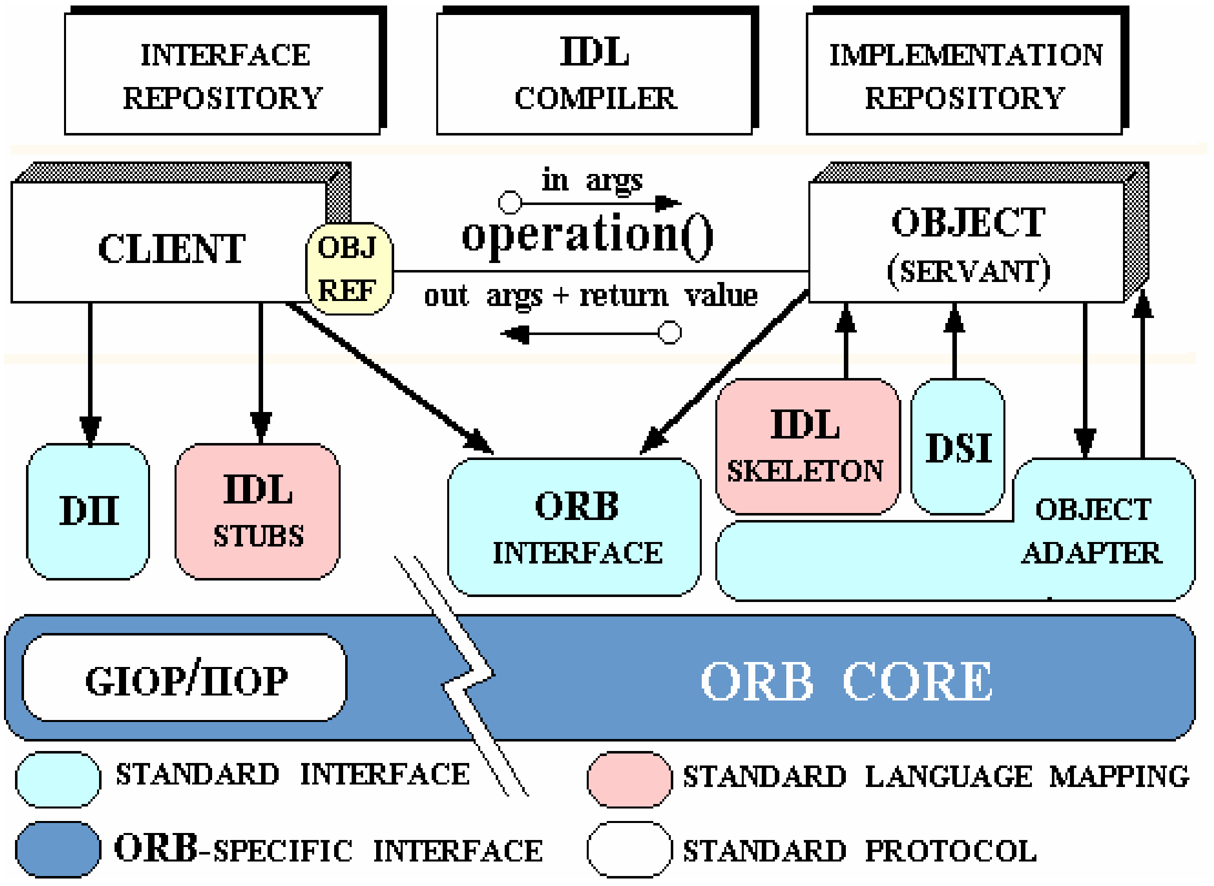


Figure 4 CORBA based Distributed Architecture

3.3 Security

- Provides a high-level security framework
- Supports authentication of remote users and services, access control for key objects and services, auditing functions, ability to establish secure communications channels between clients and object services
- Encryption functions are not included in the framework

Chapter 4

Basic introduction to GPU and CPU

Basic Introduction to CPU and GPU

Using GPU to compute general computational work is known as GPGPU (General purpose GPU). GPU has hundreds of cores that can execute multiple numbers of instructions in parallel and it is far greater than modern CPUs with 4 or 8 cores. GPU Processing Capacity limited to independent fragments. But these fragments can process in those cores in parallel. So, a programmer should choose those parts of a program that can be fragmented and those fragments are independent from each other. So, it can be said that, GPU can process a multiple times of a same operation on many records in a stream in parallel. A set of records needs similar computation is the Stream. We can say that, the parallelism is provided by Streams.

4.1 GPU based system architecture

The GPU multiprocessors are worked as co-processors for CPU. It more likes an acceleration device for CPU. When CPU invokes a kernel to GPU that kernel executes in parallel number of times in GPUs cores. So, how many tasks a GPU can complete at a time depends on its number of SM and cores per SM. Simple adding more SM can make a device completed more task.

Total no. of SM and SP per SM depends on different architecture and model of device.

The GPU architecture can be described using 3 key words [1]

- Memory(register, shared global)
- Multiprocessors (SM)
- Stream processors (SP) or cores

4.1.1 Streaming Multiprocessor (SM)

Streaming Multiprocessors(SM) are accumulation of multiple independent operators known as core/streaming processor. Upon receiving an execution command from CPU, the GPU SMs are awaken and distributed with equalized workload of “responsibilities” which are so referred as “kernel”.

Having two SMs on the delineated GPU , it is capable of executing both SMs at the same time that is where the parallelism kicks in ; meaning each workload of two different SMs ends at the same given moment ensued by very same initiating moment .

As each kernel is primarily composed of BLOCKS and THREADS, SMs are designed to execute the BLOCKS. Given (i.e.) out of two executable blocks each BLOCKs were accommodated by each SM starts execution at time=0; they are bound to terminate the both operations at the same latter time of time=3.

Though the bigger picture can be convenient enough considering SMs to be responsible for parallelism in GPU it would not be possible without the contribution of streaming processors.

4.1.2 Streaming Processor / CUDA core

Throughout the course of development of NVIDIA GPUs capacity enhancement marketing policy took a subtle turn of coining a term “core” , that has previously been always implied only to specify CPU configuration .

A CUDA core is actually a rendition of streaming processor, which aside from assisting the parallelism it actually fuels the core mechanism of the desired process (requirement to execute Kernel). Each SM is allotted with equal number of streaming processors (SPs). The role of SPs can sometimes be undermined by exploiting nature of SMs but it is the SPs that and only that enables the SMs to be exploited with such a high scale.

SPs are capable of handling threads only. If a kernel is thrown into SMs, the SMs would distribute all the instructions residing in the kernel to all available SPs. As the GPU is designated to execute the same kernel depending on the number of iterations that must be executed to suffice the requirement demanded by user , SMs would follow the same rule of distribution , meaning that same kernel would be distributed as threads recurrently to all available SPs (lesser if requirement is set to a lower level)

Considering the kernel would require the SMs to execute the same operations residing kernel for 20 times and the SMs are sufficed with 192 SPs, each set of operations (THREADS) would be distributed to each available SPs.

In short it could be expounded as each SP must execute at least one and more threads in parallel.

4.2 CPU VERSUS GPU

A simple way to understand the difference between a CPU and GPU is to compare how they process tasks. A CPU consists of a few cores optimized for sequential serial processing while a GPU has a massively parallel architecture consisting of thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously [5].

GPUs have thousands of cores to process parallel workloads efficiently

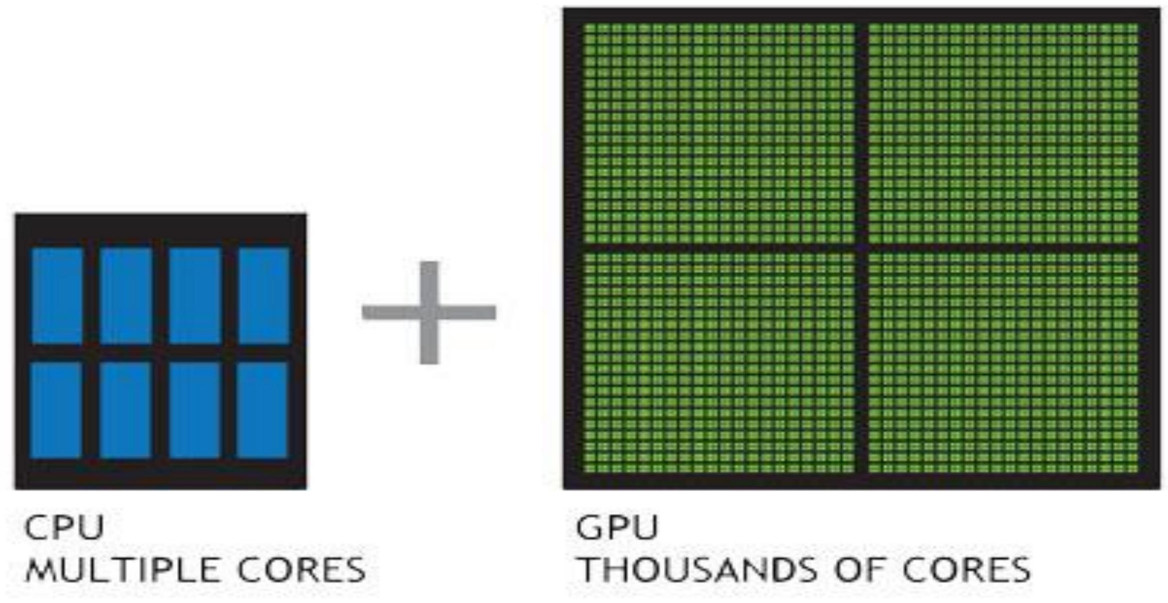


Figure 5 the difference between CPU and GPU

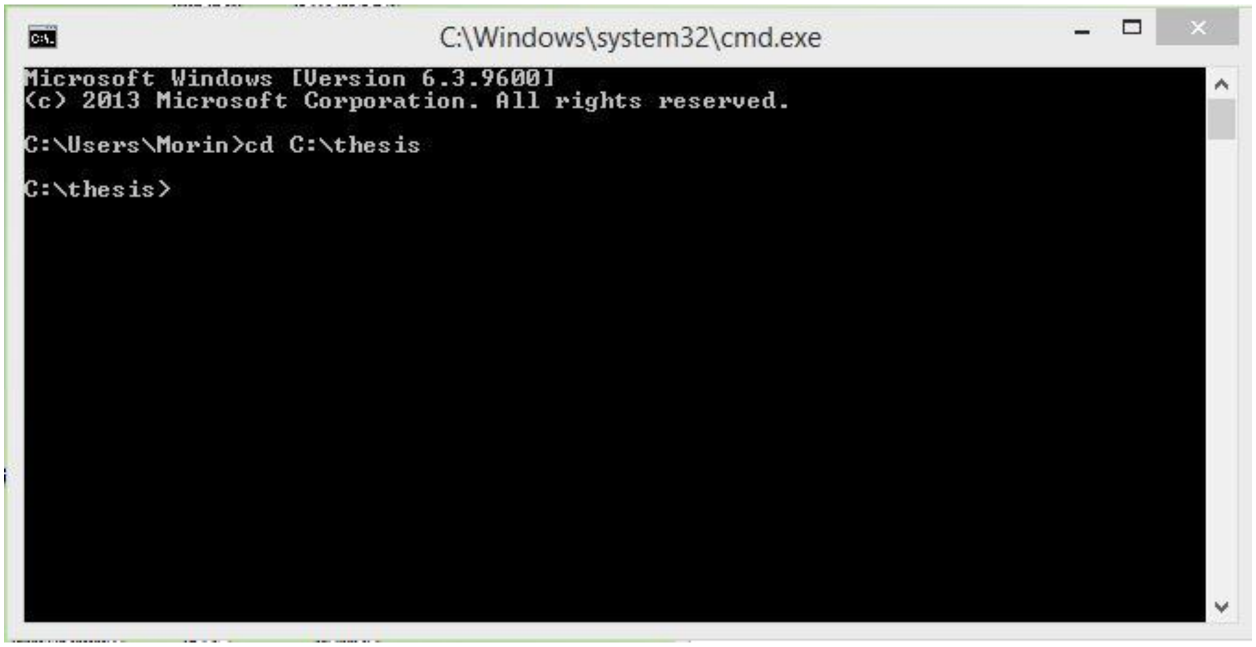
Chapter 5

Implementation

Implementation

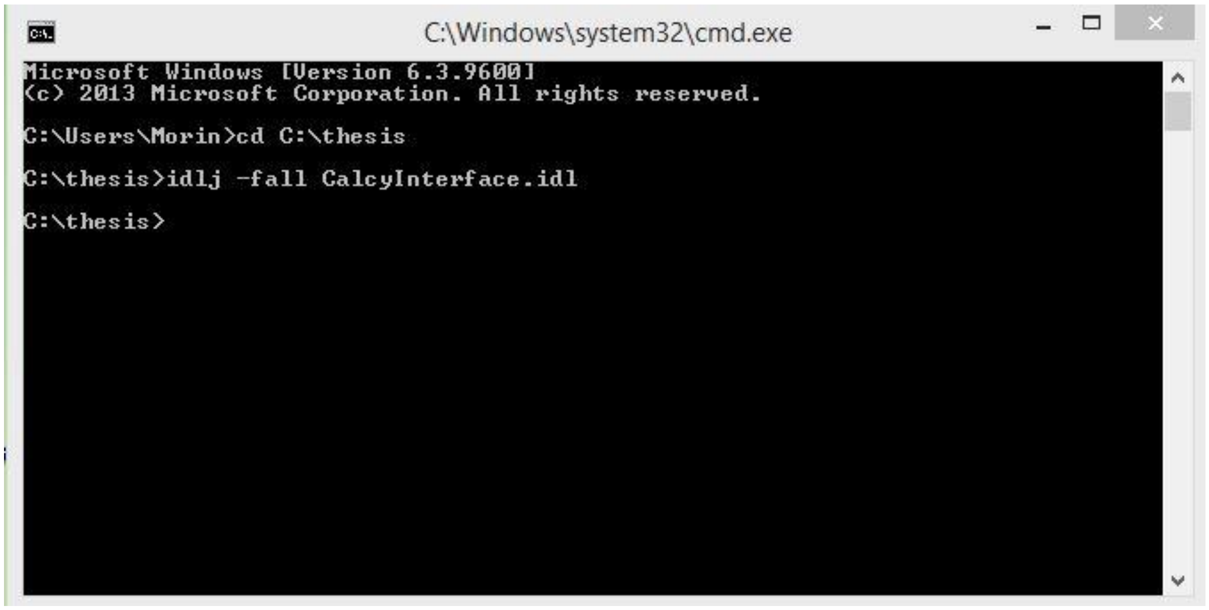
The complete execution process can be divided into two parts: first one is client side request and other one is client side acceptance.

Client side:



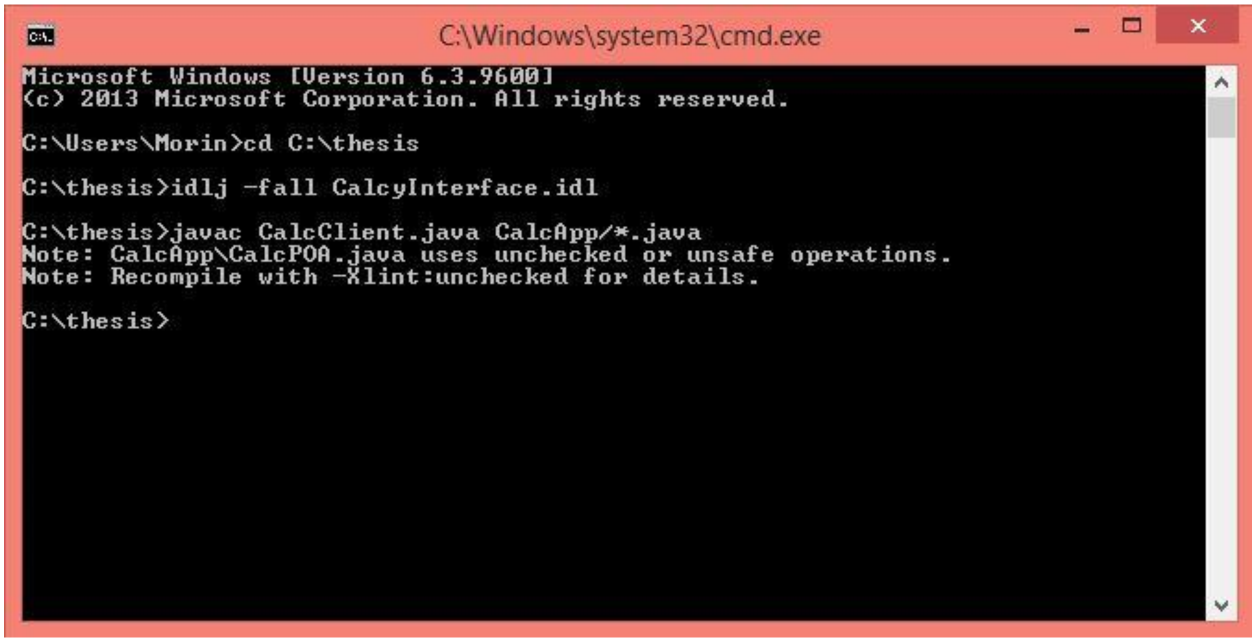
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Morin>cd C:\thesis

C:\thesis>
```

“cd C:\thesis” command entering into ‘thesis’ folder that containing the client code.



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Morin>cd C:\thesis

C:\thesis>idlj -fall CalcyInterface.idl

C:\thesis>
```

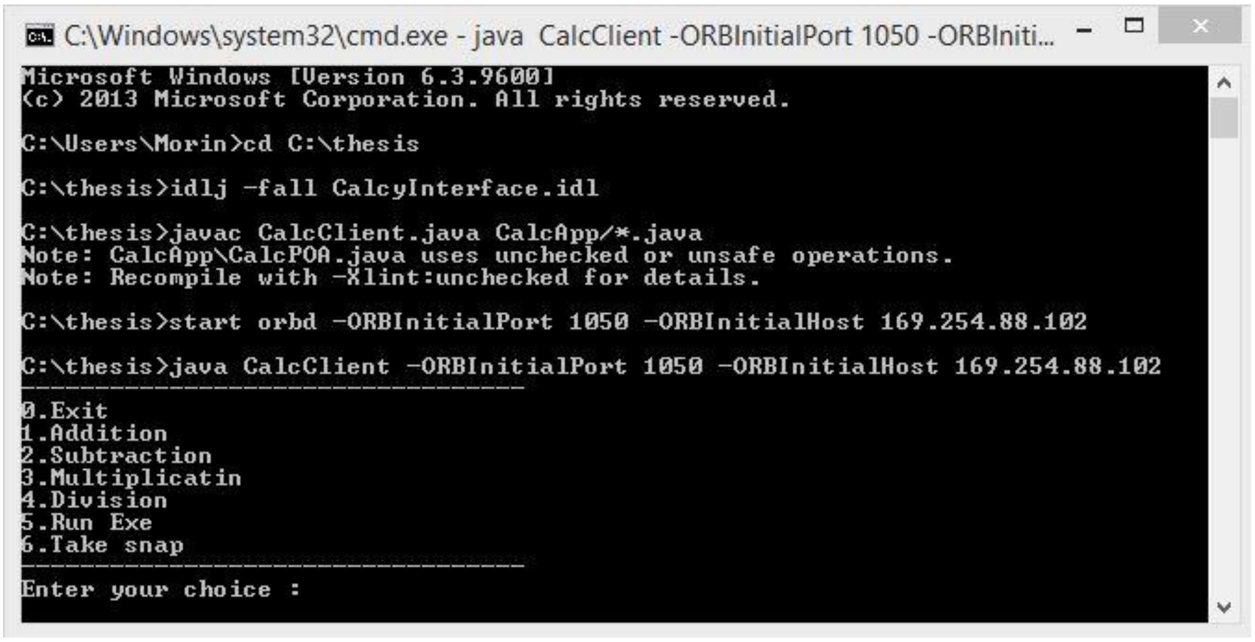
“idlj -fall CalcyInterface.idl” to create a java binding from a given idl file



C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Morin>cd C:\thesis

C:\thesis>idlj -fall CalcyInterface.idl

C:\thesis>javac CalcClient.java CalcApp/*.java
Note: CalcApp\CalcPOA.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\thesis>
```

“javac CalcServer.java CalcApp/*.java” the ‘javac’ tool using on the command read class and interface definitions, written in the java programming language and compiles them into bytecode class files.



C:\Windows\system32\cmd.exe - java CalcClient -ORBInitialPort 1050 -ORBIniti...

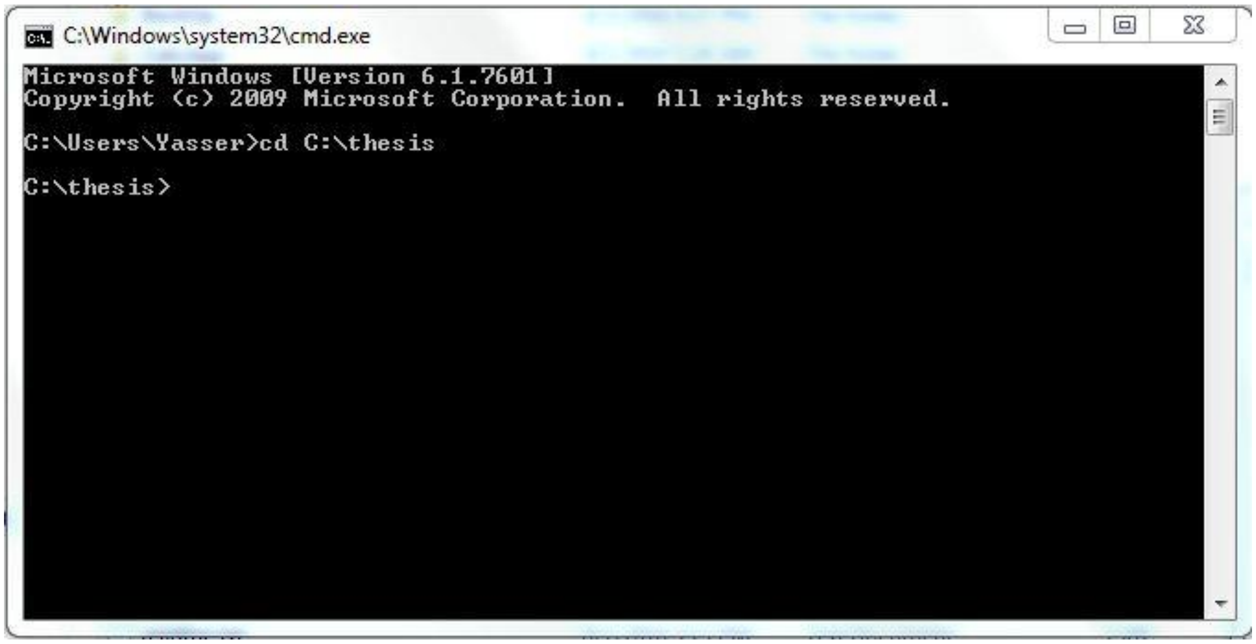
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Morin>cd C:\thesis

C:\thesis>idlj -fall CalcyInterface.idl

C:\thesis>javac CalcClient.java CalcApp/*.java
Note: CalcApp\CalcPOA.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\thesis>start orbd -ORBInitialPort 1050 -ORBInitialHost 169.254.88.102

C:\thesis>java CalcClient -ORBInitialPort 1050 -ORBInitialHost 169.254.88.102
----------------------------------
0.Exit
1.Addition
2.Subtraction
3.Multiplicatin
4.Division
5.Run Exe
6.Take snap
----------------------------------
Enter your choice :
```

If the connection established successfully the blank window will appear of an exe file named ORBD (Object request broker daemon) used to enable clients to transparently locate and invoke persistent objects on servers in the CORBA environment.

The port no. and IP address in the command are using to establish a connection from client to server.

The client will be will be shown options as showing above. The code will perform several operations like ADD, SUB, and Merging.

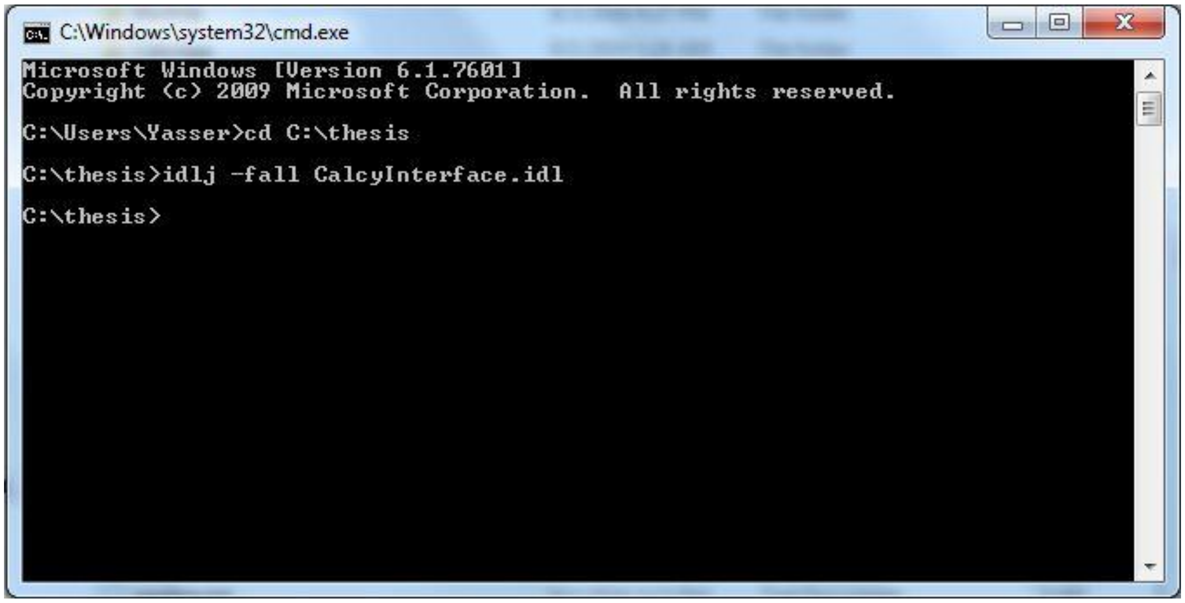
Server side:



C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Yasser>cd C:\thesis

C:\thesis>
```

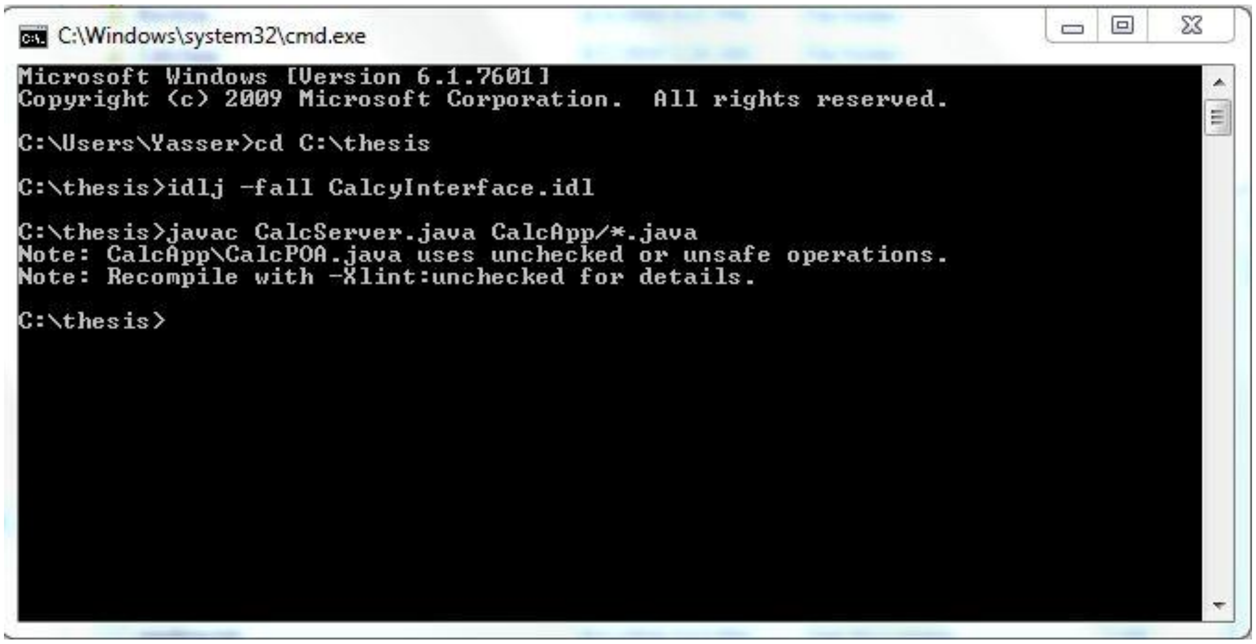
Open directory that contains the server codes.



C:\Windows\system32\cmd.exe

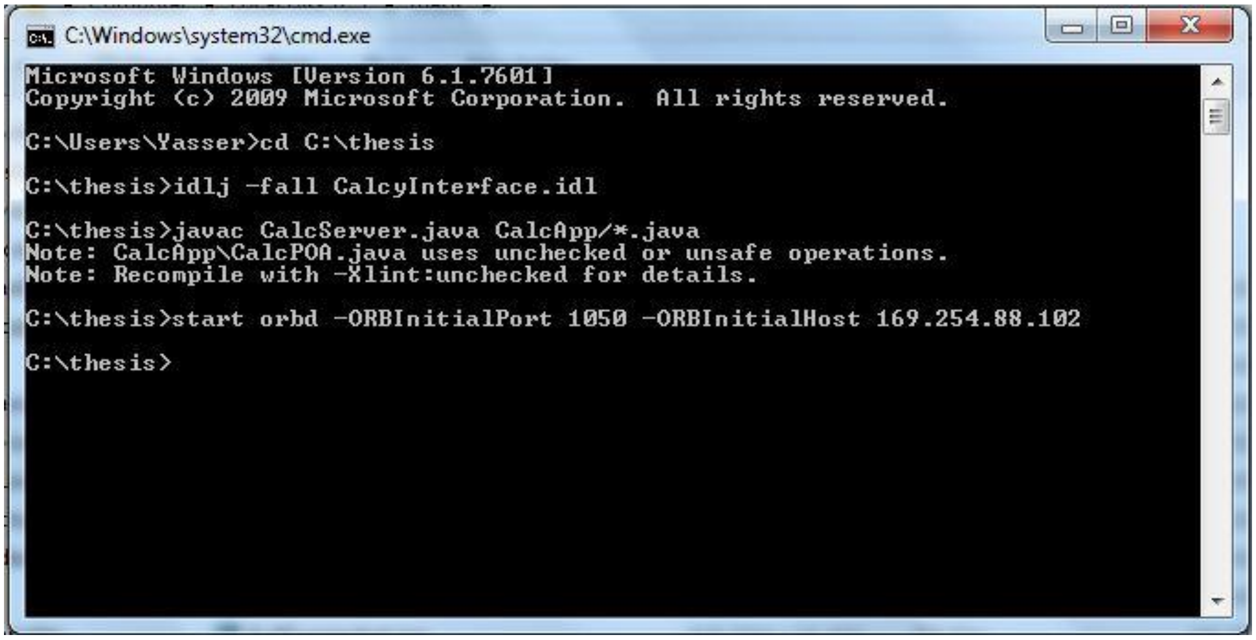
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Yasser>cd C:\thesis

C:\thesis>idlj -fall CalcyInterface.idl

C:\thesis>
```

“idlj -fall CalcyInterface.idl” to create a java binding from a given idl file


C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Yasser>cd C:\thesis

C:\thesis>idlj -fall CalcyInterface.idl

C:\thesis>javac CalcServer.java CalcApp/*.java
Note: CalcApp\CalcPOA.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\thesis>
```

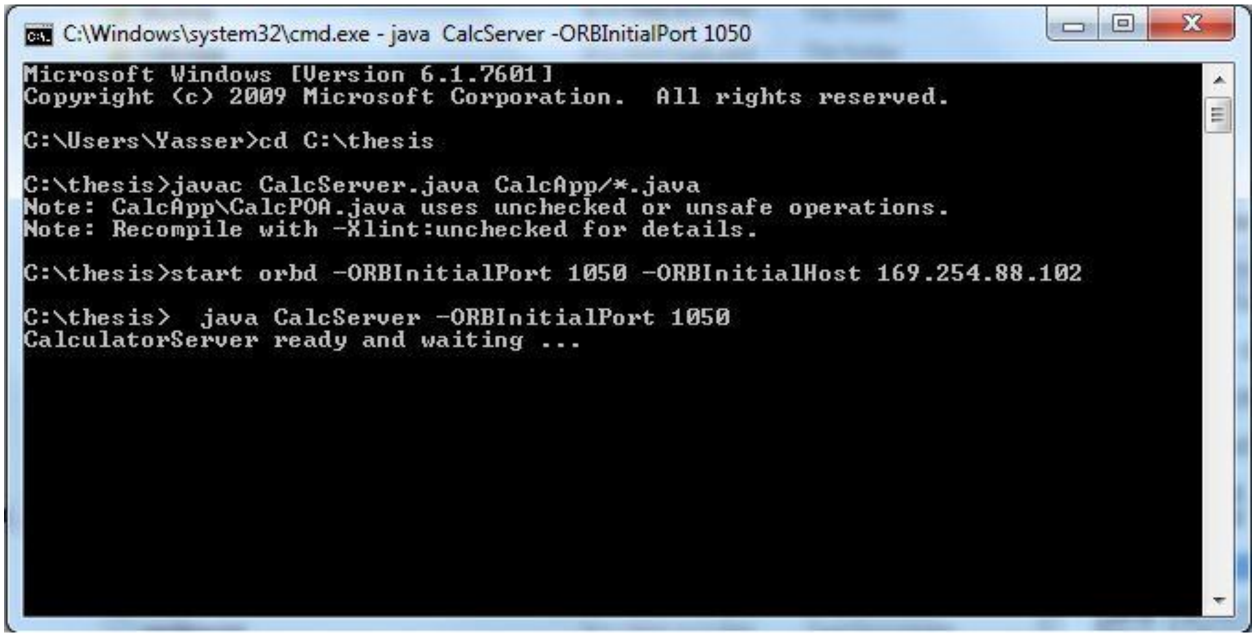
“javac CalcServer.java CalcApp/*.java” the ‘javac’ tool using on the command read class and interface definitions, written in the java programming language and compiles them into bytecode class files.


C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Yasser>cd C:\thesis

C:\thesis>idlj -fall CalcyInterface.idl

C:\thesis>javac CalcServer.java CalcApp/*.java
Note: CalcApp\CalcPOA.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\thesis>start orbd -ORBInitialPort 1050 -ORBInitialHost 169.254.88.102

C:\thesis>
```

Start ORBD application and accept the connection request from server and completed the connection.

C:\Program Files\Java\jdk1.8.0_101\bin\orbd.exe



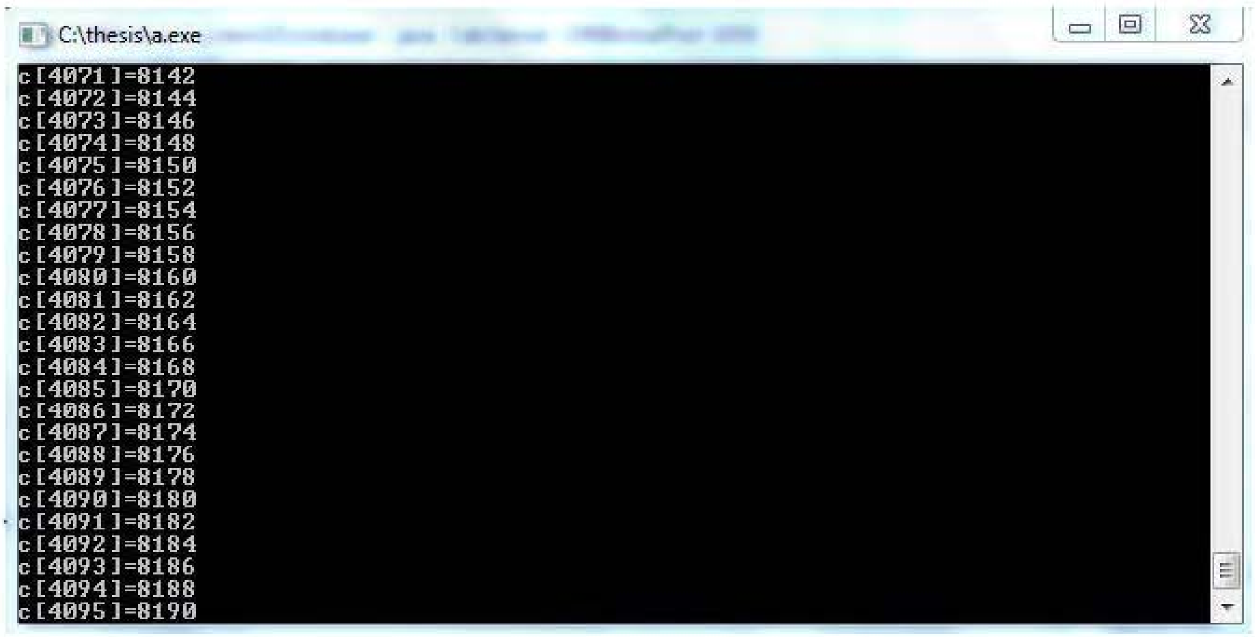
C:\Windows\system32\cmd.exe - java CalcServer -ORBInitialPort 1050

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Yasser>cd C:\thesis

C:\thesis>javac CalcServer.java CalcApp/*.java
Note: CalcApp\CalcPOA.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\thesis>start orbd -ORBInitialPort 1050 -ORBInitialHost 169.254.88.102

C:\thesis>  java CalcServer -ORBInitialPort 1050
CalculatorServer ready and waiting ...
```

Both of the system is now ready for the operations.

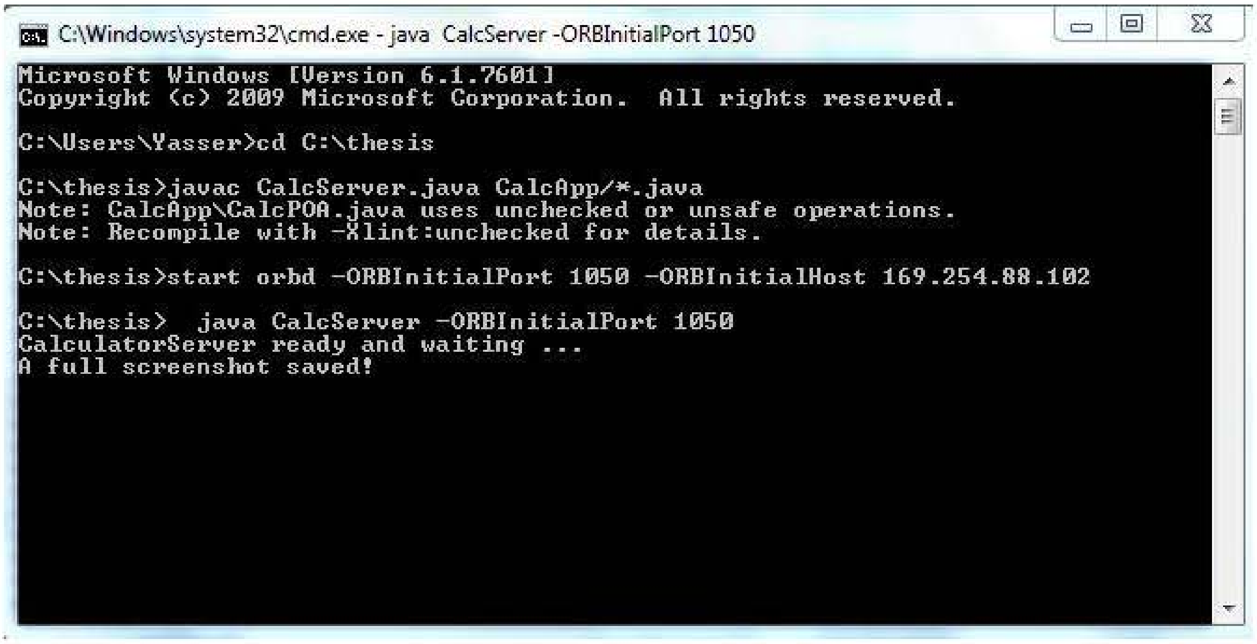
Outputs:

Output of a complete CUDA program that adds 2 arrays of 4096 integer each on server side and the operation has been done using GPU cores and take a screenshot of the outputs and save in on server computer.



C:\thesis\a.exe

```
c[4071]=8142
c[4072]=8144
c[4073]=8146
c[4074]=8148
c[4075]=8150
c[4076]=8152
c[4077]=8154
c[4078]=8156
c[4079]=8158
c[4080]=8160
c[4081]=8162
c[4082]=8164
c[4083]=8166
c[4084]=8168
c[4085]=8170
c[4086]=8172
c[4087]=8174
c[4088]=8176
c[4089]=8178
c[4090]=8180
c[4091]=8182
c[4092]=8184
c[4093]=8186
c[4094]=8188
c[4095]=8190
```



C:\Windows\system32\cmd.exe - java CalcServer -ORBInitialPort 1050

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Yasser>cd C:\thesis

C:\thesis>javac CalcServer.java CalcApp/*.java
Note: CalcApp\CalcPOA.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\thesis>start orbd -ORBInitialPort 1050 -ORBInitialHost 169.254.88.102

C:\thesis>  java CalcServer -ORBInitialPort 1050
CalculatorServer ready and waiting ...
A full screenshot saved!
```

Chapter 6

Conclusion

Conclusion

6.1 Conclusion

In this project we have developed a server-client program that can send request to the sever end for GPGPU processing. Whether the client is using CPU enabled machine, the user can get the GPU computational result from server end. CORBA is platform and language independent architecture. This program is developed through CORBA based architecture.

6.2 Future Plan

Our future plan is to establish a GPU enabled server machine and provide service to the client. 3D video rendering need a highly computational GPU which might have a costly setup for most of the people. Future games also needed highly computational GPU. So, we want to provide the GPU services through GPU this GPU enabled machine server.

Appendix

Code for Client-Server CORBA Architecture

Client Code:

```
import CalcApp.*;

import org.omg.CosNaming.*;

import org.omg.CosNaming.NamingContextPackage.*;

import org.omg.CORBA.*;

import org.omg.PortableServer.*;

import org.omg.PortableServer.POA;

import java.util.Properties;

import java.awt.Desktop;

import java.io.File;

import java.io.IOException;

import java.io.PrintWriter;

import java.io.FileWriter;

import java.io.DataInputStream;

import java.io.BufferedReader;

import java.io.BufferedInputStream;

import java.io.BufferedWriter;

import java.io.FileInputStream;

import java.io.InputStreamReader;

import java.io.Reader;

import java.io.*;

import java.io.InputStream;

import java.awt.AWTException;
import java.awt.Rectangle;

import java.awt.Robot;

import java.awt.Toolkit;

import java.awt.image.BufferedImage;

import javax.imageio.ImageIO;

import java.net.*;

import java.io.*;



class CalcImpl extends CalcPOA

{

  private ORB orb;


  public void setORB(ORB orb_val)

  {

    orb = orb_val;

  }

  public float add(float val1, float val2)

  {

    float res = val1 + val2;

    return res;

  }

  public float sub(float val1, float val2)

  {

    float res = val1 - val2;

    return res;
  }
  public float multi(float val1, float val2)
  {
    float res = val1 * val2;
    return res;
  }
  public float div(float val1, float val2)
  {
    float res = val1 / val2;
    return res;
  }
  public void shutdown()
  {
    orb.shutdown(false);
  }
      public void runMyOwnExe(){
              Desktop desktop = Desktop.getDesktop();
                                          try
                                          {
                                                  desktop.open(new
File("c:\\thesis\\a.exe"));

                                          } catch (IOException e) {

                                                  e.printStackTrace();
                                          }
      }
```

```
             public void snapshot() {
          try {
              Robot robot = new Robot();
              String format = "jpg";
              String fileName = "FullScreenshot." + format;


              Rectangle screenRect = new Rectangle(Toolkit.getDefaultToolkit().getScreenSize());
              BufferedImage screenFullImage = robot.createScreenCapture(screenRect);
              ImageIO.write(screenFullImage, format, new File(fileName));


              System.out.println("A full screenshot saved!");
          } catch (AWTException | IOException ex) {
              System.err.println(ex);
          }
      }
  }



public class CalcServer
{

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

```
      ORB orb = ORB.init(args, null);


      POA rootpoa = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));

      rootpoa.the_POAManager().activate();

      CalcImpl calcImpl = new CalcImpl();

      calcImpl.setORB(orb);

      org.omg.CORBA.Object ref = rootpoa.servant_to_reference(calcImpl);

      Calc href = CalcHelper.narrow(ref);

      org.omg.CORBA.Object objRef =

      orb.resolve_initial_references("NameService");

      NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

      String name = "Calc";

      NameComponent path[] = ncRef.to_name( name );

      ncRef.rebind(path, href);


      System.out.println("CalculatorServer ready and waiting ...");


                              orb.run();


}


catch (Exception e)

{

    System.err.println("ERROR: " + e);

    e.printStackTrace(System.out);

}
```

```
        System.out.println("CalculatorServer Exiting ...");
      }



}
```

Client Code :

```
import CalcApp.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
import java.io.*;
import java.awt.Desktop;
import java.io.File;
import java.io.IOException;
import java.net.*;
import java.io.*;

public class CalcClient
{
    public void showMenu()
    {
        System.out.println("-----------------------------------");
        System.out.println("0.Exit");
        System.out.println("1.Addition");
    System.out.println("2.Subtraction");
    System.out.println("3.Multiplicatin");
    System.out.println("4.Division");
    System.out.println("5.Run Exe");
                    System.out.println("6.Take snap");
    System.out.println("----------------------------------");
    System.out.print("Enter your choice : ");

}
public float getValue() throws IOException
{
    float val=0;
    try
    {
        System.out.print("Enter the value : ");
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String s = br.readLine();
        val = Float.parseFloat(s);
    }
    catch (IOException e)
    {
        System.out.println(e);
    }
    return val;
}
public int getChoice() throws IOException
```

```
{
    int val=0;
    try
    {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        val = Integer.parseInt(br.readLine());
        System.out.println("----------------------------------");
    }
    catch (IOException e)
    {
        System.out.println(e);
    }
    return val;
}




public static void main(String args[])
{

    Calc calcImpl = null;
    CalcClient cc = new CalcClient();

    try
    {
```

```
ORB orb = ORB.init(args, null);
    org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
     String name = "Calc";
calcImpl = CalcHelper.narrow(ncRef.resolve_str(name));
float val1 = 0;
float val2 = 0;
float res = 0;
int ch = 1;

while (ch != 0 )
{
    cc.showMenu();
    ch = cc.getChoice();



    switch (ch)
    {
        case 1:
                                                              val1 = cc.getValue();
                                                              val2 = cc.getValue();

            res = calcImpl.add(val1, val2);

            break;
```

```
case 2:
                                                val1 = cc.getValue();
                                                         val2 = cc.getValue();
    res = calcImpl.sub(val1, val2);
    break;
case 3:
                                                val1 = cc.getValue();
                                                         val2 = cc.getValue();
    res = calcImpl.multi(val1, val2);
    break;
case 4:
                                                val1 = cc.getValue();
                                                         val2 = cc.getValue();
    res = calcImpl.div(val1, val2);
    break;
case 5:
                                            try{
                                                         calcImpl.runMyOwnExe();
                                            }
                                            catch(Exception e){
                                                         e.printStackTrace();
                                            }
      break;
                                            case 6:
                                            try{
                                                         calcImpl.snapshot();
```

```
                                    }
                                    catch(Exception e){
                                        e.printStackTrace();
                                    }
                                    break;

                }

                System.out.println("----------------------------------");
                System.out.println("Result : "+res);
                System.out.println("----------------------------------");
              }

                calcImpl.shutdown();
            }
            catch (Exception e)
            {
              System.out.println("ERROR : " + e) ;
              e.printStackTrace(System.out);
            }
        }

    }
```

Server Code:

```
import CalcApp.*;

import org.omg.CosNaming.*;

import org.omg.CosNaming.NamingContextPackage.*;

import org.omg.CORBA.*;

import org.omg.PortableServer.*;

import org.omg.PortableServer.POA;

import java.util.Properties;

import java.awt.Desktop;

import java.io.File;

import java.io.IOException;

import java.io.PrintWriter;

import java.io.FileWriter;

import java.io.DataInputStream;

import java.io.BufferedReader;

import java.io.BufferedInputStream;

import java.io.BufferedWriter;

import java.io.FileInputStream;

import java.io.InputStreamReader;

import java.io.Reader;

import java.io.*;

import java.io.InputStream;

import java.awt.AWTException;

import java.awt.Rectangle;

import java.awt.Robot;

import java.awt.Toolkit;
import java.awt.image.BufferedImage;

import javax.imageio.ImageIO;

import java.net.*;

import java.io.*;



class CalcImpl extends CalcPOA

{

    private ORB orb;


    public void setORB(ORB orb_val)

    {

        orb = orb_val;

    }

    public float add(float val1, float val2)

    {

        float res = val1 + val2;

        return res;

    }

    public float sub(float val1, float val2)

    {

        float res = val1 - val2;

        return res;

    }

    public float multi(float val1, float val2)

    {
        float res = val1 * val2;

        return res;

    }

    public float div(float val1, float val2)

    {

        float res = val1 / val2;

        return res;

    }

    public void shutdown()

    {

        orb.shutdown(false);

    }

            public void runMyOwnExe(){

                        //Runtime.getRuntime().exec("C:\\thesis\\a.exe", null, new File("c:\\thesis"));

                                                            Desktop desktop = Desktop.getDesktop();

                                                                    try

                                                                    {

                                                                            desktop.open(new
File("c:\\thesis\\a.exe"));

                                                                            //BufferedReader in = new
BufferedReader(new InputStreamReader(System.in));



                                                                            //PrintWriter out = new PrintWriter(new
FileWriter("output.txt"));

                                                                            //String m = in.readLine();

                                                                             // out.println(m);

                                                                            //out.close();

                                                                    } catch (IOException e) {
```

```
                                                                    // TODO Auto-generated catch block
                                                                    e.printStackTrace();
                                                            }
       }
       public void snapshot() {
     try {
       Robot robot = new Robot();
       String format = "jpg";
       String fileName = "FullScreenshot." + format;

       Rectangle screenRect = new Rectangle(Toolkit.getDefaultToolkit().getScreenSize());
       BufferedImage screenFullImage = robot.createScreenCapture(screenRect);
       ImageIO.write(screenFullImage, format, new File(fileName));

       System.out.println("A full screenshot saved!");
     } catch (AWTException | IOException ex) {
       System.err.println(ex);
     }
   }
}



public class CalcServer
{

   public static void main(String args[])
```

```
{
  try
  {

      ORB orb = ORB.init(args, null);



      POA rootpoa = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
      rootpoa.the_POAManager().activate();
      CalcImpl calcImpl = new CalcImpl();
      calcImpl.setORB(orb);
      org.omg.CORBA.Object ref = rootpoa.servant_to_reference(calcImpl);
      Calc href = CalcHelper.narrow(ref);
      org.omg.CORBA.Object objRef =
      orb.resolve_initial_references("NameService");
      NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
      String name = "Calc";
      NameComponent path[] = ncRef.to_name( name );
      ncRef.rebind(path, href);

      System.out.println("CalculatorServer ready and waiting ...");




                    //////////////////////////////////////////////TODO
```

```
//while(true)
/*{
        ServerSocket serverSocket = new ServerSocket(25);
        Socket socket = serverSocket.accept();
        System.out.println("Server Connected.");

        System.out.println("Accepted connection : " + socket);
        File transferFile = new File
("C:\\thesis\\FullScreenshot.jpg");
        byte [] bytearray = new byte [(int)transferFile.length()];
        FileInputStream fin = new FileInputStream(transferFile);
        BufferedInputStream bin = new
BufferedInputStream(fin);
        bin.read(bytearray,0,bytearray.length);
        OutputStream os = socket.getOutputStream();
        System.out.println("Sending Files...");
        os.write(bytearray,0,bytearray.length);
        os.flush();
        socket.close();
        System.out.println("File transfer complete");
}       */

        ////////////////////////////////////////////
```

```
                    orb.run();

        }

        catch (Exception e)
        {
          System.err.println("ERROR: " + e);
          e.printStackTrace(System.out);
        }

        System.out.println("CalculatorServer Exiting ...");
      }


}
```

<u>**Interface Code :**</u>

```
module CalcApp
{
  interface Calc
  {
    float add(in float value1,in float value2);
    float sub(in float value1,in float value2);
    float multi(in float value1,in float value2);
    float div(in float value1,in float value2);
```

```
                        oneway void runMyOwnExe();

                        oneway void snapshot();


              oneway void shutdown();

           };


        };
```

Reference

[1] Distributed System Concepts and Design by George Coulouris, Jean Dollimore, Tim Kindberg

[2] http://books.cs.luc.edu/distributedsystems/issues.html

[3] http://searchsoa.techtarget.com/definition/middleware

[4] CUDA C PROGRAMMING GUIDE By Nvidia corporation

[5] https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/