

ASSIGNMENT

Course Code	19CSC313A
Course Name	Distributed and Cloud Computing
Programme	B.Tech.
Department	Computer Science and Engineering
Faculty	FET

Name of the Student	Subhendu Maji
Reg. No	18ETCS002121
Semester/Year	6 TH sem/ 2018
Course Leader/s	MS. Chaitra

Declaration Sheet			
Student Name	Subhendu Maji		
Reg. No	18ETCS002121		
Programme	B.Tech.	Semester/Year	6 th sem/ 2018
Course Code	19CSC313A		
Course Title	Distributed and Cloud Computing		
Course Date		to	
Course Leader	MS. Chaitra		
<p>Declaration</p> <p>The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.</p>			
Signature of the Student		Date	
Submission date stamp (by Examination & Assessment Section)			
Signature of the Course Leader and date		Signature of the Reviewer and date	

Declaration Sheet	ii
Contents	iii
Question No. A1.....	4
A1.1 Need for Distributed Systems in Modern Computing	4
A1.2 Select any conference paper on any one Distributed Systems application and give a summary .	5
A1.3 Stance taken and justification with the support of conference paper.....	8
Question No. B1.....	9
B1.1 Problem solving approach	9
B1.2 Implementation	10
B1.3 Results and Analysis.....	13

Solution to Question No. 1:

A1.1 Need for Distributed Systems in Modern Computing

A distributed system is a set of computing factors every being able to behave independently of every difference. A computing detail, which we can normally confer with as a node, can be both a hardware tool or a software manner. Another element is that users (be they people or packages) consider they are handling a single machine.

There are multiple needs for a distributed system in modern computing, some of them are as follows:

Resource Sharing:

An important goal of a distributed system is to make it easy for users (and applications) to access and share remote resources. Resources can be virtually anything, but typical examples include peripherals, storage facilities, data, files, services, and networks, to name just a few. Connecting users and resources also makes it easier to collaborate and exchange information, as is illustrated by the success of the Internet with its simple protocols for exchanging files, mail, documents, audio, and video.

Availability:

The availability of a monolithic system is limited to the availability of the piece of hardware it runs on. Modern hardware is pretty great and combined with a good data center and good management practices servers can be expected to fail with an annual failure rate (AFR) in the single-digit percentages.

Scalability:

Distributing a system over many machines gives a lot of flexibility about how to scale it. Stateless systems are relatively easy to scale, and basic techniques like HTTP load balancers are great for an awful lot of use-cases. Stateful systems are harder to scale, both because you need to decide how to spread the state around, and because you need to figure out how to send users to the right place to get the state.

Durability:

Like availability, the durability of single storage devices is pretty great these days. Distributed storage systems continuously make multiple copies of a piece of data, allowing a great deal of flexibility around cost, time-to-recovery, durability, and other factors. They can also be built to be extremely tolerant to correlated failures and avoid correlation outright.

Efficiency

Distributed systems make sure that the work is done efficiently no matter when the user wants to work. Distributed computing is a foundational model for cloud computing because cloud systems are distributed systems. Besides administrative tasks mostly connected to the accessibility of resources in the cloud, the extreme dynamism of cloud systems—where new nodes and services are provisioned on demand—constitutes the major challenge for engineers and developers. This characteristic is pretty peculiar to cloud computing solutions and is mostly addressed at the middleware layer of computing system.

A1.2 Select any conference paper on any one Distributed Systems application and give a summary

IJCSMC,

Vol. 4, Issue. 4, April 2015

Distributed System and its Role in HealthCare System

Introduction

It is seen that, the organizations have been interested in the decentralization of processing while achieving the integration of the information resources within their geographically distributed systems of database, applications and users. Web based systems and applications (called WEBAPPS) have evolved into sophisticated computing tools that not only provide standalone function to the users, but also have been integrated with corporate databases and real-world applications

[1] Primary Health Centers (PHC) are dispersed throughout the country and it is observed that, the communication techniques used today are not optimum and having lot of limitations. It is observed that, the health care (HC) environment in rural areas is inadequate in all aspects and the HC practitioners are more involved in collecting, preparing requisite reports rather than consulting the patients which is their main role. Due to their more involvement in administration, it is difficult to provide better services, consultation and provision of basic HC. Hence the Distributed web-based model be setup for such organizations in.

[2] Today in healthcare system, telemedicine provides the good role. The advantages of telemedicine are providing improved health care to the people in inaccessible areas. It reduces cost and improves quality of health care. It mostly reduces the isolation of users viz. specialists, nurses and allied health professionals. The term tele-medicine means the use of telecommunication and computer information technologies with

medical expertise to facilitate by providing information for remote health care delivery, medical services to remote areas or across great distances on the globe.

Traditional Systems versus Distributed Systems for Communication:

Early database systems moved towards centralization and resulted into complex databases. The database systems are single user, multiuser with centralized computing at server side only. These systems are traditional systems viz. single tier or host systems and two-tier Architecture viz. Traditional systems having no. of limitations. Today the need of users is not limited and not based upon single location but the user wants to have the global information access and to update regularly. In the traditional systems, information is only read by the users. In these Client-side computing environments cannot be made available logically to server side through networks globally or remotely.

As compared to the traditional systems, distributed systems prove to be better. Thus, in healthcare application these systems play a vital role. The important merits of distributed systems are:

Openness

- Resource sharing
- Concurrency
- Fault tolerance
- Scalability

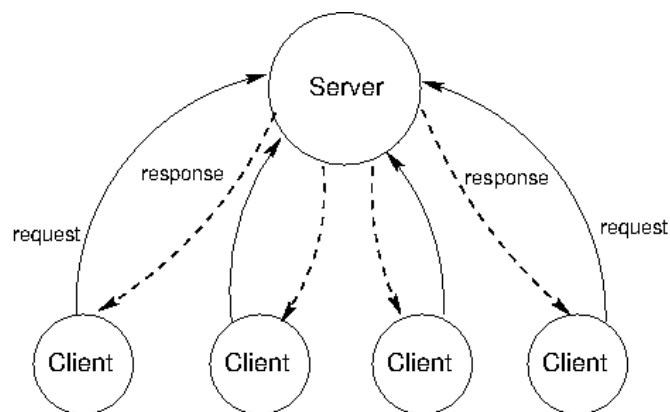


Figure 1 Client Server Architecture

HEALTHCARE SYSTEM.

[3] The recent advances in the growth of medical sciences, engineering studies, communications and information technologies have been supported by the growth of internet technology. Internet technology provides us effective, efficient and improved health care information about the patients and their health-related problems.

A) Interoperability In healthcare system: Interoperability is the ability of two or more components, applications or systems to exchange and use information. There is currently a major challenge for the healthcare industry in achieving interoperability among applications provided by different vendors each hospital department or medical clinic may use multiple applications to share clinical and administrative information among applications.

For health professionals, it improves access to health record data. For patients, quality and safety of care is improved. For health managers, data collection is improved. For health researchers, availability of medical data is increased.

B) Distributed Patient Record: Distributed systems have a great importance in handling distributed patient records. The patient record has a distributed architecture and each client computer have local database. In distributed database architecture the data is not stored entirely at a single physical location instead it is spread across a network of computers and connected via communication links. The most important advantage is that a distributed database allows faster local queries and can reduce network traffic.

C) Important Processes of Healthcare and Use of Distributed Systems: It is observed that, the information is collected on paper and used at healthcare centers i.e., PHC (Primary Health Centers), THO (Tahsil Health Officer) and DHO (District Health Officer) units and its allied units. In the view of role and responsibilities performed by these authorities, it seems that Distributed Web Based model in the form of Distributed Web Based System is one of the suitable computing solutions which is helpful in many applications like creation of database, deletion of databases and any specific entries, modifications within the existing data, searching specific data items etc.

Conclusion

This paper presented a research perspective on distributed system in healthcare sector. By the survey it is seen that, the current operations of health services are inadequate due to lack of communication facilities between them. Designing of Distributed System based on Network technology, solution in the form of computing tool, providing solution to handle different query issues with healthcare system remains future work.

A1.3 Stance taken and justification with the support of conference paper

Upon stating all the facts, I stand for motion “Is Distributed Systems Knowledge Essential for Modern Computing Professionals?”

Distributed systems are one of the most important aspects for modern computing and will continue to be the same in the future.

Justification

In this paper, we are presented with a research perspective on distributed system in healthcare sector. By the survey it is seen that, the current operations of health services are inadequate due to lack of communication facilities between them. Also, the present data recording procedures in most of health centers leave much to be desired. Manual systems employed are inaccurate requires time, space, and cost. Due to lack of well computing facilities, unavailability of current data and recent information, there is a great need of modern communication system that reduces wastage of resources, time and cost. In order to minimize these drawbacks, there has to exist a distributed system based on network technology, logger methodology and web-based technology. In the view of role and responsibilities performed by these authorities, it seems that Distributed Web Based model in the form of Distributed Web Based System is one of the suitable computing solutions which is helpful in many applications like creation of database, deletion of databases and any specific entries, modifications within the existing data, searching specific data items etc.

Some of the examples of how Distributed systems will be useful in healthcare systems:

- Reliable, high fault tolerant: A system crash on one hospital server won't affect the other servers.
- Scalable: In distributed computing systems one can always add more machines.
- Flexible: It makes it easy to install, implement and debug new services.
- Fast: A distributed computer system would provide the quickness which is a requirement of all the healthcare facilities.
- Open: Patients can access their records from anywhere in the world

Solution to Question No. B1:

B1.1 Problem solving approach

Algorithm (Client Side)

Step 1. START

Step 2. Create 'Operation' class consisting of operand1, operand2 and operator attributes

Step 3. Setup socket connection on fixed port

Step 4. Create object of 'Operation' class

Step 5. Read values for operand1, operand2, and operator from user and store values in newly created object of 'Operation' class.

Step 6. Using ObjectOutputStream, send object to server.

Step 7. Using ObjectInputStream, read response from server and display.

Step 8. if user calls 'exit' => close all resources

Step 9. END

Algorithm (Server Side)

Step 1. START

Step 2. Create 'Operation' class consisting of operand1, operand2, and operator attributes;

Step 3. Create instance of ServerSocket class on fixed port and call accept() to wait for connection from client.

Step 4. Create object of 'Operation' class

Step 5. Once connection from client is established, read the object sent from client using

ObjectInputStream and deconstruct the object to get the operands and operation to perform.

Step 6. Carry out required calculation and send answer to client using ObjectOutputStream.

Step 7. if client calls 'exit' => close all resources

Step 8. END

B1.2 Implementation

To achieve client-server communication, we have made use of sockets.

Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server.

When the connection is made, the server creates a socket object on its end of the communication. The client and the server can now communicate by writing to and reading from the socket.

The `java.net.Socket` class represents a socket, and the `java.net.ServerSocket` class provides a mechanism for the server program to listen for clients and establish connections with them.

The following steps occur when establishing a TCP connection between two computers using sockets –

- The server instantiates a `ServerSocket` object, denoting which port number communication is to occur on.
- The server invokes the `accept()` method of the `ServerSocket` class. This method waits until a client connects to the server on the given port.
- After the server is waiting, a client instantiates a `Socket` object, specifying the server's name and the port number to connect to.
- The constructor of the `Socket` class attempts to connect the client to the specified server and the port number. If communication is established, the client now has a `Socket` object capable of communicating with the server.
- On the server side, the `accept()` method returns a reference to a new socket on the server that is connected to the client's socket.

After the connections are established, communication can occur using I/O streams. Each socket has both an `OutputStream` and an `InputStream`. The client's `OutputStream` is connected to the server's `InputStream`, and the client's `InputStream` is connected to the server's `OutputStream`.

TCP is a two-way communication protocol; hence data can be sent across both streams at the same time.

Source code (Server Side)

```
import java.io.*;
import java.net.*;

class Operation implements Serializable {

    String operator;
    int operand1;
    int operand2;

    // Constructor
    public Operation() {
    }
}

public class App implements java.io.Serializable {

    // Static ServerSocket variable
    private static ServerSocket server;
    // Socket server port on which it will listen
    private static int port = 5555;

    public static void main(String args[]) throws IOException, ClassNotFoundException {
        // Creating the socket server object
        server = new ServerSocket(port);
        System.out.println("listening on port " + port);

        // Creating socket and waiting for client connection
        Socket socket = server.accept();
        // Read from socket to ObjectInputStream object
        ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());
        // Create ObjectOutputStream object
        ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());
        String operator;
        int a, b, c = 0;
        // Server keeps listening indefinitely until it receives 'exit'
        while (true) {
            // Create new Operation object to receive operands and operator from client
            Operation opr = new Operation();
            // Read new values from client side
            opr = (Operation) ois.readObject();
            a = opr.operand1;
            b = opr.operand2;
            operator = opr.operator;
            if (operator.equals("exit"))
                break;
            // Carry out calculation
            switch (operator) {
                case "add":
                    c = a + b;
                    break;
                case "subtract":
                    c = a - b;
                    break;
                case "multiply":
                    c = a * b;
                    break;
                case "divide":
                    c = a / b;
                    break;
            }
            // Send back result after calculating
            oos.writeObject("Answer is : " + c);
        }
        // Closing resources
        server.close();
        ois.close();
        oos.close();
    }
}
```

Source code (Client Side)

```
import java.io.*;
import java.net.*;
import java.util.Scanner;

class Operation implements Serializable {

    String operator;
    int operand1;
    int operand2;

    public Operation() {
    }

}

public class App implements Serializable {

    public static void main(String[] args)
        throws UnknownHostException, IOException, ClassNotFoundException, InterruptedException {

        // Open the socket connection
        Socket socket = new Socket("localhost", 5555);
        // Create ObjectOutputStream object
        ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());
        // Read from socket to ObjectInputStream object
        ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());
        // Creating scanner object to read values from terminal
        Scanner sc = new Scanner(System.in);
        int a, b;
        String c;
        // Keep running until 'exit' call from user
        while (true) {

            System.out.println("=====");

            // Declare new Operation object to send operands and operator to server
            Operation opr = new Operation();
            System.out.print("Enter first operand : ");
            a = sc.nextInt();
            System.out.print("Enter second operand : ");
            b = sc.nextInt();
            sc.nextLine();
            System.out.print("Enter operation (add,subtract,multiply,divide,exit) : ");
            c = sc.nextLine();
            if (c.equals("exit"))
                break;
            opr.operand1 = a;
            opr.operand2 = b;
            opr.operator = c;
            // Send Operation object to server side for calculation
            oos.writeObject(opr);

            // Receive response back from server after calculation
            System.out.println("\nServer response :\n" + ois.readObject());

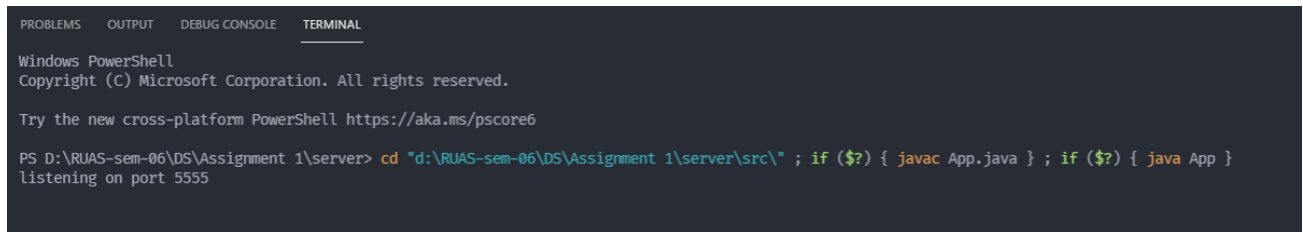
        }

        // Closing resources
        sc.close();
        socket.close();
        ois.close();
        oos.close();

    }

}
```

B1.3 Results and Analysis



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\RUAS-sem-06\DS\Assignment 1\server> cd "d:\RUAS-sem-06\DS\Assignment 1\server\src\" ; if ($?) { javac App.java } ; if ($?) { java App }
listening on port 5555
```

Figure 2 Output of server side



```
PS D:\RUAS-sem-06\DS\Assignment 1\client\src> cd "d:\RUAS-sem-06\DS\Assignment 1\client\src\" ; if ($?) { javac App.java } ; if ($?) { java App }
=====
Enter first operand : 5
Enter second operand : 22
Enter operation (add,subtract,multiply,divide,exit) : add

Server response :
Answer is : 27
=====
Enter first operand : 15
Enter second operand : 3
Enter operation (add,subtract,multiply,divide,exit) : divide

Server response :
Answer is : 5
=====
Enter first operand : 10
Enter second operand : 60
Enter operation (add,subtract,multiply,divide,exit) : multiply

Server response :
Answer is : 600
=====
Enter first operand : 100
Enter second operand : 30
Enter operation (add,subtract,multiply,divide,exit) : subtract

Server response :
Answer is : 70
=====
Enter first operand : 5
Enter second operand : 5
Enter operation (add,subtract,multiply,divide,exit) : exit
PS D:\RUAS-sem-06\DS\Assignment 1\client\src> []
```

Figure 3 Output of Client side

We first run the server before starting the client. The server sets up the socket and listens for any connection from a client. We then run the client and the connection is established between the server and the client.

The user then enters the values of the operands and which operation to carry out. This information is encapsulated in a single object and is sent to the server using an output stream.

The server receives the object from the client using an input stream and then deconstructs the object to get the respective operands and operation to carry out. Depending on the operator, the calculation is carried out and the answer is sent to the client using an output stream.

The client reads the response from the server using an input stream and then displays it.

If client sends an 'exit' call, the resources (Streams/Sockets) are closed.

- [1] Boulanger,J. Reerink; Deroussent,C.” *Preliminary Based Service Evaluation for Elderly People and Healthcare Professionals in Residential Home Care Units*”,Digital society,2008
- [2] Pradeep Ray;Weerakkody,G. “*Quality of service management in health care organizations: a casestudy*” computer-Based Medical systems,1999.Proceedings. 12th IEEE Symposium.
- [3] Jack S. Newman and John Kelly:”*A High Performance Web-Based Teleconsultation System for Island Telemedicine*”:Jurnal of Computing and Information Technology,2000