

Лабораторная работа № 2

Технологии распределенной обработки данных

Тема: Разработка распределенной системы обработки данных
Вариант № 3

Инв. № подл.	Подп. и дата	Инв. № дубл.	Взам. инв. №	Подп. и дата

Проверил:
Гай В. Е.
Выполнил:
Студент гр. 14-В-1
Кузнецова П.В.

1. Выполнение лабораторной работы

1.1. Требования к программе

Разработанный программный комплекс должен состоять из Сервера и Клиента.

Функции сервера: хранение удалённого объекта, предоставляющего доступ к заданиям для обработки и результату обработки. Предусмотреть на сервере возможность одновременного доступа к критической секции кода нескольких клиентов (с помощью оператора lock). Критическая секция кода - та, к которой гипотетически одновременно могут обратиться несколько клиентов.

Функции клиента (на сервере хранится список клиентов - эта функция уже предусмотрена исходным кодом библиотеки RemoteBase):

1) управляющие функции (выполняет только один клиент из всего множества клиентов, выполнение данной функции должно выполняться через вызов методов удалённого объекта (удалённый объект хранится на сервере)):

1.1) формирование и ведение списка заданий (под ведением понимается удаление уже обработанных и предоставление клиенту задания по запросу);

1.2) получение, объединение и вывод результатов вычислений (результаты вычислений должны выводиться в каждом клиенте, для этого необходимо проверять окончание обработки всех данных по таймеру; объединение результатов вычисления также можно реализовать с использованием таймера);

1.3) устанавливает флаг того, что управляющий клиент назначен, на сервере сохраняется идентификатор клиента;

2) вычислительные функции

2.1) запрос задания с сервера (клиент должен запросить задание только после того, как эти задания были сформированы);

2.2) обработка данных;

2.3) отправка результатов обработки на сервер.

Требования к системе:

1) предусмотреть возможность отключения одного из клиентов, получившего задание на обработку.

2) предусмотреть возможность отключения управляющего клиента (для этого можно хранить время последней операции на сервере).

Подп. и дата	Взам. инв. №	Инв. № дубл.	1.2) получение, объединение и вывод результатов вычислений (результаты вычислений должны выводиться в каждом клиенте, для этого необходимо проверять окончание обработки всех данных по таймеру; объединение результатов вычисления также можно реализовать с использованием таймера);								
			1.3) устанавливает флаг того, что управляющий клиент назначен, на сервере сохраняется идентификатор клиента;								
Подп. и дата	Взам. инв. №	Инв. № дубл.	2) вычислительные функции								
			2.1) запрос задания с сервера (клиент должен запросить задание только после того, как эти задания были сформированы);								
Подп. и дата	Взам. инв. №	Инв. № дубл.	2.2) обработка данных;								
			2.3) отправка результатов обработки на сервер.								
Подп. и дата	Взам. инв. №	Инв. № дубл.	Требования к системе:								
			1) предусмотреть возможность отключения одного из клиентов, получившего задание на обработку.								
Подп. и дата	Взам. инв. №	Инв. № дубл.	2) предусмотреть возможность отключения управляющего клиента (для этого можно хранить время последней операции на сервере).								
Подп. и дата	Взам. инв. №	Инв. № дубл.	Лабораторная работа № 2								
Подп. и дата	Взам. инв. №	Инв. № дубл.	Ли	Изм.	№ докум.	Подп.	Дата	Разработка распределенной системы обработки данных			
								Лит	Лист	Листов	
Инв. № подл.	Взам. инв. №	Инв. № дубл.	Разраб.							2	12
			Пров.								
Инв. № подл.	Взам. инв. №	Инв. № дубл.	Т. контр.								
			Н. контр.								
Инв. № подл.	Взам. инв. №	Инв. № дубл.	Утв.								
								14-B-1			

1.2. Вариант индивидуального задания на лабораторную работу

Разработайте систему распределенной обработки данных в соответствии с вариантом задания. Клиент отправляет данные для обработки на сервер. Сервер - обрабатывает их и возвращает результат. Разработать алгоритм поиска максимального и минимального значения массива а:

$$mn = \min_{i \in [1; N]} a_i, \quad mx = \max_{i \in [1; N]} a_i$$

1.3. Листинг программы

1.3.1. Клиент

```
using System;
using RBase;
using System.Runtime.Remoting.Channels.Tcp;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Services;
using System.Runtime.Remoting;

namespace Client
{
    class Program
    {
        private static Lib remoteObject = null;
        static void Main(string[] args)
        {
            TcpChannel clientChannel = new TcpChannel();
            ChannelServices.RegisterChannel(clientChannel, true);

            remoteObject = (Lib)Activator.GetObject(typeof(Lib),
                "tcp://localhost:2222/RemoteBase");

            if (remoteObject == null)
                return;

            bool isAdmin = false;
            int clientID = remoteObject.RegisterClient(out isAdmin);

            if (isAdmin)
            {
                Console.WriteLine("Connected as [ADMIN]");
            }
        }
    }
}
```

Инв. № подл	Подп. и дата	Инв. № дубл.	Взам. инв. №	Подп. и дата

Ли	Изм.	№ докум.	Подп.	Дат

```

const int SIZE = 200024;
int[] array = new int[SIZE];
int genMax, genMin;
Random rand = new Random();
for (int i = 0; i < SIZE; ++i)
array[i] = rand.Next();
Lib.F(array, out genMax, out genMin);
Console.WriteLine("Tested result: [max][min] [{0}][{1}]",
    genMax, genMin);
Console.WriteLine("Send task to server...");
remoteObject.UploadTaskToServer(clientID, array);
Console.WriteLine("Task sended");
Console.WriteLine("Press any key to disconnect admin client..");
Console.ReadKey();
Console.WriteLine("Disconnecting...");
remoteObject.UnregisterClient(clientID);
Console.WriteLine("Ended.");
return;
}
Console.WriteLine("Current client connected as processor client");
int[] inArray;
int max, min;
while (!remoteObject.isWorkFinished())
{
    Console.WriteLine();
    inArray = remoteObject.GetClientData(clientID);
    Lib.F(inArray, out max, out min);
    Console.WriteLine("Client found [max][min] value:
        [{0}][{1}]", max, min);
    remoteObject.ReturnClientData(clientID, max, min);
}
remoteObject.UnregisterClient(clientID);
Console.WriteLine("Client finished.");
}
}
}

```

1.3.2. Сервер

```

using System;
using RBase;
using System.Runtime.Remoting.Channels.Tcp;

```

Инв. № подл.	Подп. и дата				Лист										
Инв. № дубл.	Взам. инв. №				4										
Подп. и дата	Подп. и дата				Лист										
<table border="1"> <tr> <td>Ли</td> <td>Изм.</td> <td>№ докум.</td> <td>Подп.</td> <td>Дат</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </table>					Ли	Изм.	№ докум.	Подп.	Дат						Лабораторная работа № 2
Ли	Изм.	№ докум.	Подп.	Дат											

```

using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Services;
using System.Runtime.Remoting;

namespace NetRemotingServer
{
class Program
{
static void Main(string[] args)
{
Console.WriteLine("Server start...");
TcpChannel channel = new TcpChannel(2222);

ChannelServices.RegisterChannel(channel, true);
RemotingConfiguration.RegisterWellKnownServiceType(typeof(Lib),
"RemoteBase", WellKnownObjectMode.Singleton);
Console.WriteLine("Server started.");
Console.WriteLine("Press <ENTER> to shutdown server.");
Console.ReadLine();
}
}
}

```

1.3.3. База

```

//Lib
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace RBase
{
public class Lib : MarshalByRefObject
{
public static void F(int[] array, out int max, out int min)
{
int findMax = array[0];
int findMin = array[0];
for (int i = 0; i < array.GetLength(0); ++i)
{
if (findMax < array[i])
findMax = array[i];

```

Инв. № подл.	Подп. и дата	Инв. № дубл.	Взам. инв. №	Подп. и дата	<div style="text-align: right;">Лабораторная работа № 2</div>					Лист
										5
Ли	Изм.	№ докум.	Подп.	Дат						


```
public int RegisterClient(out bool isAdmin)
{
    lock ("connect")
    {
        globalID++;

        Client client = new Client(globalID);

        bool hasAdmin = false;
        for (int i = 0; i < clients.Count; ++i)
            hasAdmin |= clients[i].GetAdmin();
    }
}
```

```

if (!hasAdmin)
{
    client.SetAdmin(true);
    isAdmin = true;
}
else
    isAdmin = false;
    clients.Add(client);
    return globalID;
}

```

```
public void UnregisterClient(int clientID)
{
    lock ("disconnect")
    {
        for (int i = 0; i < clients.Count; ++i)
            if (clients[i].GetID() == clientID)
```

Лист
7

```

        clients.RemoveAt(i);
    }
}

public void UploadTaskToServer(int clientID , int[] array)
{
    lock ("admin_upload_task")
    {
        Client client = GetClientByID(clientID);
        if ((client == null) || (!client.GetAdmin()))
        {
            Console.WriteLine("[ERROR] Only admin can give task's to server.");
            return;
        }
        if ((processedParts.Count != 0) && (!isWorkFinished()))
        {
            Console.WriteLine("Server not fully completed previous work.Aborting");
            return;
        }
        Console.WriteLine();
        Console.WriteLine("Admin [client ID: {0}] upload task to server...",

            clientID);
        Console.WriteLine("Receiving task...");
        serverArray = new int[array.GetLength(0)];
        for (int i = 0; i < array.GetLength(0); ++i)
            serverArray[i] = array[i];
        for (int i = 10; i > 2; i--)
        {
            if (array.GetLength(0) % i == 0)
            {
                separatedBy = i;
                break;
            }
        }
        for (int i = 0; i < array.GetLength(0); i += separatedBy)
            processedParts.Add(i);
        sendArray = new int[separatedBy];
        resultMax = serverArray[0];
        resultMin = serverArray[0];
        Console.WriteLine("Task received.");
        Console.WriteLine("Received array len: {0}\nDivided by: {1}",
            array.GetLength(0), separatedBy);
    }
}

```

Инв. № подл.	Подп. и дата	Инв. № дубл.	Взам. инв. №	Подп. и дата	
Инв. № подл.	Подп. и дата	Инв. № дубл.	Взам. инв. №	Подп. и дата	
Ли	Изм.	№ докум.	Подп.	Дат	Лабораторная работа № 2
					Лист 8


```

    }
}
public int[] GetClientData(int clientID)
{
    lock ("client_data_out")
    {
        Client client = GetClientByID(clientID);
        int j = processedParts[0];
        processedParts.RemoveAt(0);
        client.SetStatus(Client.ClientStatus.BUSY);
        client.SetMeta(j);
        for (int i = 0; i < separatedBy; ++i)
            sendArray[i] = serverArray[j + i];
        return sendArray;
    }
}
public void ReturnClientData(int clientID, int max, int min)
{
    lock ("client_data_in")
    {
        Client client = GetClientByID(clientID);
        if (max > resultMax)
            resultMax = max;
        if (min < resultMin)
            resultMin = min;
        client.SetStatus(Client.ClientStatus.FREE);
    }
    // Checkinh end
    bool flag = true;
    for (int i = 0; i < clients.Count; ++i)
        flag &= (clients[i].GetStatus() == Client.ClientStatus.FREE);
    // Server ended
    if (flag && isWorkFinished())
    {
        Console.WriteLine("Server solved task");
        Console.WriteLine("Array max: {0}\nArray min: {1}",
            resultMax, resultMin);
    }
}
public bool isWorkFinished()
{
    lock ("finished")

```

Инв. № подл.	Подп. и дата	Инв. № дубл.	Взам. инв. №	Подп. и дата	
Ли	Изм.	№ докум.	Подп.	Дат	Лист
					9

```

    {
        return (processedParts.Count == 0);
    }
}
}

//Client

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace RBase
{
    public class Client
    {
        public enum ClientStatus { BUSY, FREE }
        private int id = 0;
        private ClientStatus status = ClientStatus.FREE;
        private bool hadAdminRights = false;
        private DateTime lastInteraction = DateTime.Now;
        private int meta = 0;

        public Client(int clientID)
        {
            id = clientID;
            status = ClientStatus.FREE;
            hadAdminRights = false;
        }
        public void SetAdmin(bool admin)
        {
            hadAdminRights = admin;
        }
        public bool GetAdmin()
        {
            return hadAdminRights;
        }
        public int GetID()
        {
            return id;
        }
    }
}

```

Подп. и дата

Взам. инв. №

Инв. № дубл.

Подп. и дата

Инв. № подл.

Лист

10

Лабораторная работа № 2

Ли

Изм.

№ докум.

Подп.

Дат

```

    }
    public void SetStatus(ClientStatus clientStatus)
    {
        status = clientStatus;
        lastInteraction = DateTime.Now;
    }
    public DateTime GetTimeSinceLastInteraction()
    {
        return lastInteraction;
    }
    public ClientStatus GetStatus()
    {
        return status;
    }
    public int GetMeta()
    {
        return meta;
    }
    public void SetMeta(int clientMeta)
    {
        meta = clientMeta;
    }
}
}

```

1.4. Скриншоты

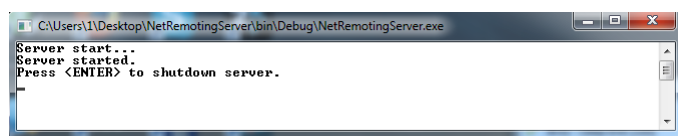


Рисунок 1.1. Server при включении

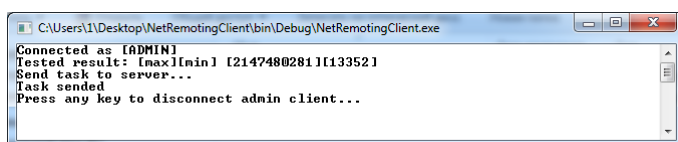
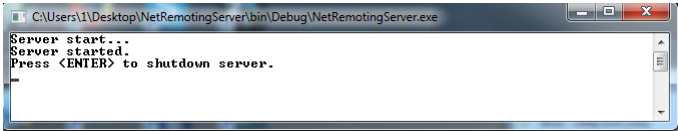


Рисунок 1.2. Первый подключенный клиент

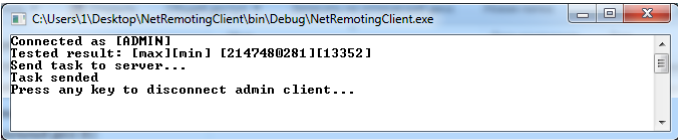
Инв. № подл	Подп. и дата	Инв. № дубл.	Взам. инв. №	Подп. и дата	<div>}</div>						
Ли	Изм.	№ докум.	Подп.	Дат		Лабораторная работа № 2					Лист
											11

1.4. Скриншоты



Server start...
Server started.
Press <ENTER> to shutdown server.

Рисунок 1.1. Server при включении



Connected as [ADMIN]
Tested result: [max][min] [2147480281][13352]
Send task to server...
Task send
Press any key to disconnect admin client...

Рисунок 1.2. Первый подключенный клиент

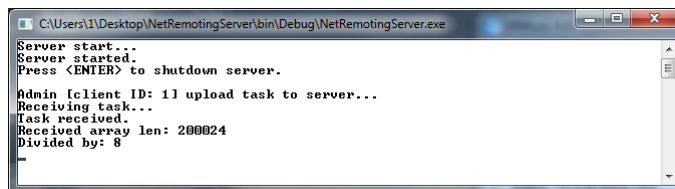


Рисунок 1.3. Server при подключении первого клиента

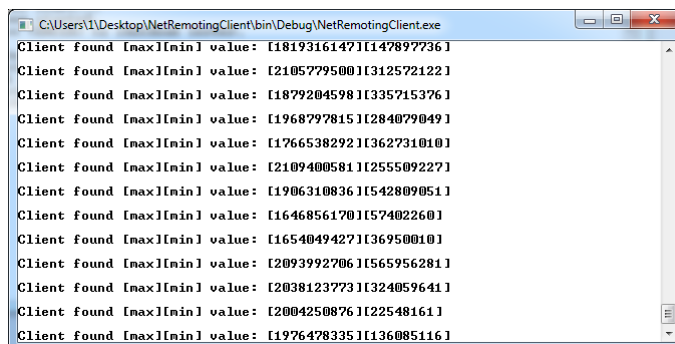


Рисунок 1.4. Последующие подключенные клиенты

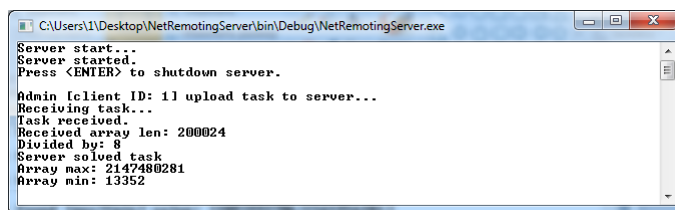


Рисунок 1.5. Server по завершении обработки всего массива

Выводы

Таким образом, в ходе данной лабораторной работы был разработан программный комплекс, состоящий из сервера и клиента. Этот программный комплекс реализует алгоритм нахождения максимального и минимального значений массива А.

Инв. № подл.	Подп. и дата	Инв. № дубл.	Взам. инв. №	Подп. и дата						Лист 12
Ли	Изм.	№ докум.	Подп.	Дат	Лабораторная работа № 2					