分布式计算 第一次作业

张俊华 16030199025

1. 题目要求

- 题目1：将基于UDP协议的Client-Server通信程序示例的服务器端程序改造成多线程版。
- 题目2：将基于TCP协议的Client-Server通信程序示例的服务器端程序改造成线程池版。

2. 具体实现

题目1

- 首先定义 UDPServerThread 类，用于多线程并发响应客户端连接

  在其构造函数内，需要传入 UDP Socket，以及收到的数据包 `packet` ，因为客户端的地址和端口，需要通过 packet 来获得

```
UDPServerThread(DatagramSocket socket, DatagramPacket packet){
        this.socket = socket;
        this.packet = packet;
}
```

- 之后，定义 UDPServer 类，设定监听端口，在循环中每收到一个 UDP 数据包，就新建一个线程，来完成响应

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

/**
 * UDP 服务端主程序
 * @author 张俊华 16030199025
 */
public class UDPServer{

    public static void main(String args[]){
        // 设定监听端口
        int serverPort = 6789;

        //建立 UDPSocket
        try (DatagramSocket aSocket = new DatagramSocket(serverPort)) {
            byte[] buffer = new byte[1000];
            while (true) {
                // 读取客户端请求
                DatagramPacket request = new DatagramPacket(buffer,
buffer.length);
                aSocket.receive(request);
                // 建立新线程
                UDPServerThread thread = new UDPServerThread(aSocket, request);
                thread.run();
            }
        } catch (SocketException e) {
            System.out.println("Socket: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO: " + e.getMessage());
        }
    }
}
```

题目2

- `TCPServerThread` 类

  要将 TCP协议的Client-Server通信程序的服务器端改造成线程池，首先就需要将服务器端程序中的处理请求和构建发送响应的部分，构建新的线程来处理

  因此，需要构建 `TCPServerThread` 类，继承自 `Thread` 类，实现多线程并发：

```
public class TCPServerThread extends Thread {

    private Socket socket = null;

    public TCPServerThread(Socket socket) {
        this.socket = socket;
    }

    public void run(){
    ...
    }
}
```

  向其构造函数传入参数 socket，是其需要响应的会话 Socket，在 `run` 函数中，从 socket 中读取数据，并将其原样通过 socket 管道回传给客户端。

- TCPServerWithThreadPool

  与一客户一线程服务器一样，Server 类首先需要创建一个ServerSocket实例。用于监听端口，响应 TCP Socket 连接。

```
int serverPort = 6790;
        try {
            ServerSocket listenSocket=new ServerSocket(serverPort);
        } catch (IOException e) {
            e.printStackTrace();
        }
```

  创建一个线程池，用于来避免持续地创建新线程，限制最大线程数量。

```
ThreadPoolExecutor executor = new ThreadPoolExecutor(5, 30, 20,
TimeUnit.MILLISECONDS,
                new ArrayBlockingQueue<Runnable>(10));
```

  之后，在循环中不断监听 ServerSocket，每获取到一个新的客户端，就将服务socket交给线程池进行处理

```
while(true){
    socket=listenSocket.accept();
    count++;
    System.out.println("The total number of clients is " + count + ".");
    executor.submit(new TCPServerThread(socket));
}
```

3. 运行结果

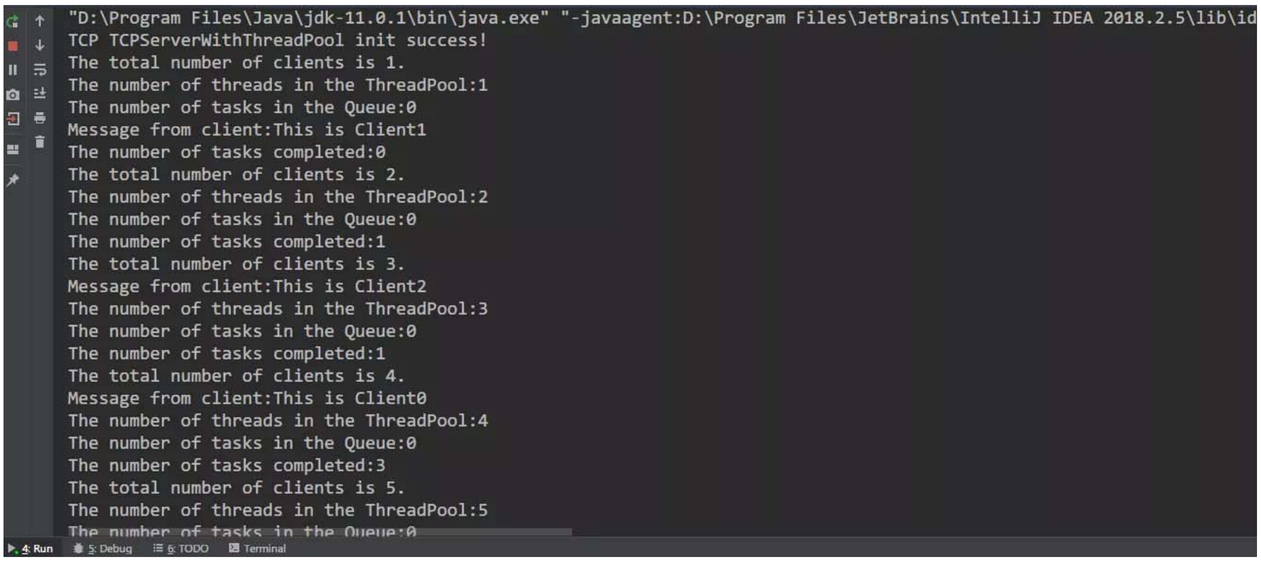
为了验证编写的服务端程序对并发请求的处理能力，分别为 TCP 和 UDP 客户端编写了并发连接测试程序，创建大量线程，同时发送请求：

```
import java.util.ArrayList;
```

```java

/**
 * TCPClientTest TCP 客户端并发连接测试程序
 * @author 张俊华 16030199025
 */
public class TCPClientTest {

    public static void main(String args[]){
        ArrayList<TCPClientThread> ClientArray  =  new ArrayList<>();
        for(int i = 0; i < 100; i++){
            String msg = "This is Client" +i;
            ClientArray.add(new TCPClientThread(i,msg));
            System.out.println(i);
        }

        for(int i = 0; i < 100; i++){
            ClientArray.get(i).start();
        }
    }
}

```

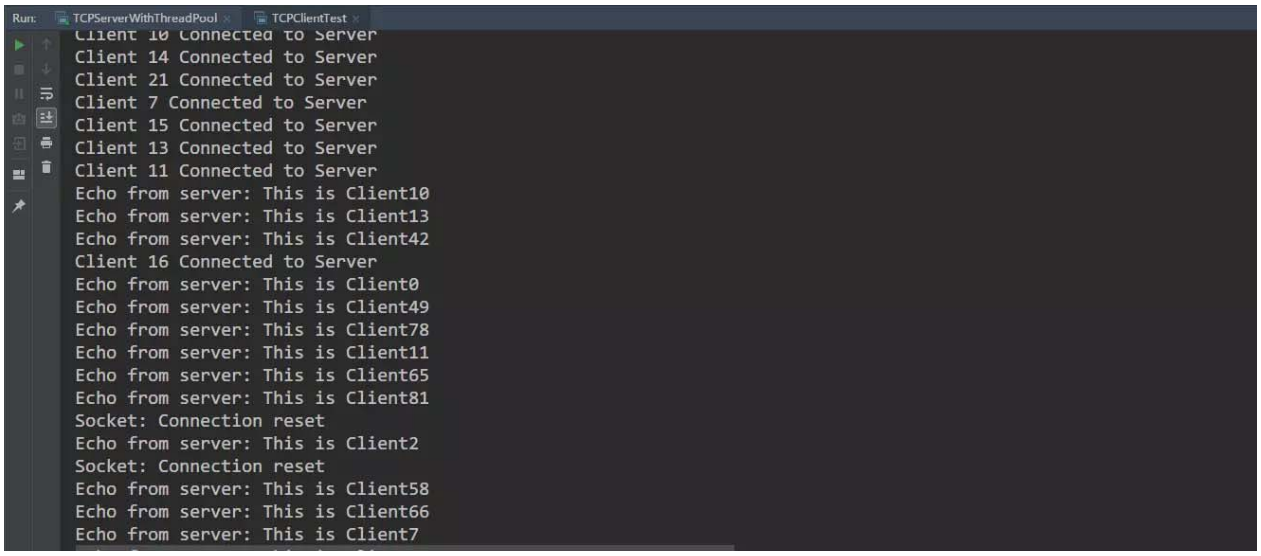
测试结果如下

- TCP 服务端



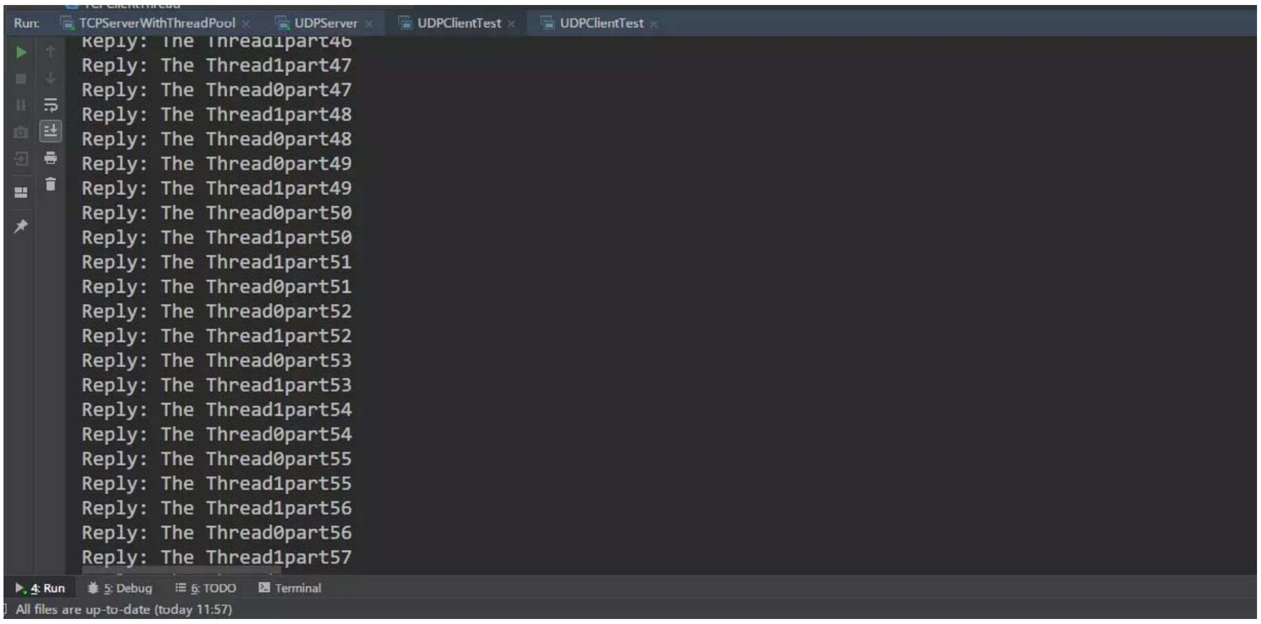
```
"D:\Program Files\Java\jdk-11.0.1\bin\java.exe" "-javaagent:D:\Program Files\JetBrains\IntelliJ IDEA 2018.2.5\lib\id
TCP TCPServerWithThreadPool init success!
The total number of clients is 1.
The number of threads in the ThreadPool:1
The number of tasks in the Queue:0
Message from client:This is Client1
The number of tasks completed:0
The total number of clients is 2.
The number of threads in the ThreadPool:2
The number of tasks in the Queue:0
The number of tasks completed:1
The total number of clients is 3.
Message from client:This is Client2
The number of threads in the ThreadPool:3
The number of tasks in the Queue:0
The number of tasks completed:1
The total number of clients is 4.
Message from client:This is Client0
The number of threads in the ThreadPool:4
The number of tasks in the Queue:0
The number of tasks completed:3
The total number of clients is 5.
The number of threads in the ThreadPool:5
The number of tasks in the Queue:0
```

- TCP 客户端



```
Client 10 Connected to Server
Client 14 Connected to Server
Client 21 Connected to Server
Client 7 Connected to Server
Client 15 Connected to Server
Client 13 Connected to Server
Client 11 Connected to Server
Echo from server: This is Client10
Echo from server: This is Client13
Echo from server: This is Client42
Client 16 Connected to Server
Echo from server: This is Client0
Echo from server: This is Client49
Echo from server: This is Client78
Echo from server: This is Client11
Echo from server: This is Client65
Echo from server: This is Client81
Socket: Connection reset
Echo from server: This is Client2
Socket: Connection reset
Echo from server: This is Client58
Echo from server: This is Client66
Echo from server: This is Client7
```

- UDP 客户端:



```
Reply: The Thread1part46
Reply: The Thread1part47
Reply: The Thread0part47
Reply: The Thread1part48
Reply: The Thread0part48
Reply: The Thread0part49
Reply: The Thread1part49
Reply: The Thread0part50
Reply: The Thread1part50
Reply: The Thread1part51
Reply: The Thread0part51
Reply: The Thread0part52
Reply: The Thread1part52
Reply: The Thread0part53
Reply: The Thread1part53
Reply: The Thread1part54
Reply: The Thread0part54
Reply: The Thread0part55
Reply: The Thread1part55
Reply: The Thread1part56
Reply: The Thread0part56
Reply: The Thread1part57
```

All files are up-to-date (today 11:57)