**CSCI 2251  
Lab 4 – Concurrent Processing via Network**

**Lab 4 is an extension of Lab 3, which has four objectives:**

1. Integrate the previously developed concurrent processing into a client/server-based system.
2. Implement the socket for data-stream communication.
3. Establish and develop your communication protocol.
4. Apply simple graphical event-driven user interface to the client/server-based system.

**Description (Computation, Same Lab 3)**

Given two integer matrices A and B, you are requested to compose a program to perform matrix addition (A + B). Both matrices have N rows and M columns; N > 1, M > 1; You need to divide both (A and B) into four equal (or close to equal) size of submatrices (A0,0, A0,1, A1,0, A1,1 and B0,0, B0,1, B1.0, B1.1) and each submatrix has dimension close to (N/2) x (M/2). You need to create four Java threads each thread performs a subset of addition on one pair of the submatrices. Example, thread 0 performs addition on A0,0 and B0,0, thread 1 performs addition on A0,1 and B0,1, . . . etc. The final result should store in the matrix C of size N by M.

You also need to construct another thread (process), GUI implementation, to interact with the user via a simple GUI. The interaction includes: 1) get user input, the name of the matrix file. 2) Output the computation result with adequate formatting. 3) Print the error messages. 4) End the program upon user request, such as push the END button.

Note: you need to manage the case of odd number of rows and/or odd number of column matrices. Try to divide them into “almost” equivalence size of submatrices.

**Requirements**

1. Client – a GUI with text area for user input (file name) and a text area to display the result. After receiving the file name from the user interface, the client opens and reads the file (the format of the file is described below) for the two matrices A and B.

Send both matrices to the Server (define your own communication protocol, including the operation – in this program adding the two matrices), wait for the result from the server, and display the result on the display area with N rows and M columns.

1. Server – Starting and waiting for client connection. According to the communication protocol you defined, receive both matrices A and B, and the operation (adding the two matrices) from client, divide both matrices into four submatrices and generate four threads (processes) as described above (you are encouraged to re-use your Lab3 objects) to perform the requested addition.

After completing the computation, send the result to the client. Then wait for another request from the client. Note: the server should run forever, waiting for clients to request connections.

Note: a good GUI code reference for this assignment is the example in Chapter 28, Networking

**File Format**

1. the first line has two numbers, N and M (N rows, M columns), the size of both matrices A and B
2. the next N lines each has M elements for one of the rows of A
3. the next N lines each has M elements for one of the rows of B

Example:

4 6

2 3 1 2 5 1

3 1 2 2 2 4

1 2 3 2 7 2

3 6 1 5 1 3

6 5 4 1 4 3

3 3 2 2 1 1

7 5 4 3 2 5

2 1 8 4 8 4

For the above example, the result of the computation on the console should be

8 8 5 3 9 4

6 4 4 4 3 5

8 7 7 5 9 7

5 7 9 9 9 7

**Instructions**

* You must use NxM two-dimension arrays for your implementation
* You must divide the two-dimension array into the form such as: . Same for *B* and *C*.
* After the computation, the result needs to be stored in the matrix C.
* Your program should work any size matrix. One of the goals is to minimize the resource usage, such as memory and processor cycles. Explain how you accomplish this goal in your document.
* All matrices, *A*, *B*, and *C* need to be in a shared memory to eliminate memory copy.

Server and Client are located on different machines, or logically separate from each other if a single machine is used. When making connection, the client needs to specify where the server is (the server IP address, localhost is one of the choices). There are no prior requirements on how to design your GUI, however you should provide a reasonable format to display the N by M result with clear content. You need to name your java source file Server.java for the server and Client.java for the client.

**Note:**

* Program header – Comments at the top with short description of your program, your name, e-mail, date and course title. This is required for all your java program files.
* We expect programming assignments to be implemented using Java 1.8. Your code will be tested on the machines, Windows or Linux, installed the same Java 1.8 version (Java compiler and Java Virtual Machine). Make sure your code runs on at least one of those machines.
* Submit your program to Brightspace.
* No late, refer to the syllabus for the late policy.

I will test your program as following example after download and unzipping of your files:

javac Server.java

javac Client.java

on Server Machine

java Server

on Client machine

java Client <Server IP>

or

java Client

Client needs to prompt the user for the input file name, open and read the file. Send the two NxM matrices to the server first and then sends the command (operation, such as ADD, SUB, MUL, . . ., for this Lab, the only operation command is ADD) to the Server. After the Server receives the matrices and the operation command, it performs the computation, then sends the results back to the Client. Client then displays the results on the Client’s GUI.

Note: Both Client and Server may terminate the connection by issuing the TERMINATE command. Please refer to the textbook examples Fig 28.3, 28.4, 28.5, and 28.6 for the communication protocol.