COMP 6321: Distributed Systems Design

Lab 9 Instructions

MPJ and RMI

July 12, 2022

1 General Information

Lab Date: Tuesday, July 12th, 2022.

Your lab instructor will provide you with instructions on how to do this lab activity.

2 Overview

In this lab we will address the issues that may be caused by out-of-the-process-group communication that may affect MPI, especially when an external API may be implemented. For instance, a solution to the assignment three would be to implement and host a client API using RMI in Java. Since RMI interceptors runs under a different process group, no communication with MPI nodes would be allowed. As a result, no MPI send / receive would be allowed to any nodes. This will causes issues, if any internal calls to some methods is invoking an MPI send / receive. To resolve the issue, the MPI node needs to host a worker process (within the MPI process group) and listens to incoming requests, and invokes the calls by proxy.

In this lab, a implement a simple API in RMI by which some messages are sent to other MPI nodes. We use MPJExpress for the implementation, however, same logic may be applied to other variants. In our example, the RMIClient sends mathematical equations to the RMI

server. When server receives the equation, it will pass it to one of the worker nodes by proxy. The worker node receives and solves the equation, and sends the result back to requesting node (master node in our example), by which the result is returned back to the client.

2.1 What is RMI

The RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

2.2 What is MPI

The Message Passing Interface (MPI) is an Application Program Interface that defines a model of parallel computing where each parallel process has its own local memory, and data must be explicitly shared by passing messages between processes. Using MPI allows programs to scale beyond the processors and shared memory of a single compute server, to the distributed memory and processors of multiple compute servers combined together.

2.3 What is MPJ Express

MPJ Express is an open source Java message passing library that allows application developers to write and execute parallel applications for multicore processors and compute clusters clouds. MPJ is an implementation of Open-MPI specification in Java. While in MPJ, the processes are implemented in Java, MPI alo allows using native interface, by which an native library is used for message passing (see section 4).

3 Instructions

Clone the template from the following repo: `https://github.com/COMP6231/T09`
See files: `Problem.java`, `RMIServer.java`, and `MPJAndRMIDemo.java`

1. `Problem.java` - is used to generate and solve the mathematical equations.

2. `RMIServer.java` - a helper class to host RMI objects. It also provides a messaging proxy.

3. `MPJAndRMIDemo.java` - the main file; Both RMIClient and MPJ classes are implemented in here.

Step 1 - Running the Code

Compile and run the code using following instructions. Run the code and report errors. Fix the errors by using the proxy class.

Execution

To execute the code from a CLI (or CMD on windows)

1. Linux or MacOS
   (a) `export MPJ_HOME=path-to-folder-mpj-v0_44`
   (b) `javac -cp $MPJ_HOME/lib/mpj.jar *.java`
   (c) `$MPJ_HOME/bin/mpjrun.sh -np 2 MPJAndRMIDemo`
2. Windows (run CMD as admin)
   (a) `SET MPJ_HOME=path-to-folder-mpj-v0_44`
   (b) `javac -cp %MPJ_HOME%\lib\mpj.jar *.java`
   (c) `$MPJ_HOME\bin\mpjrun.bat -np 2 MPJAndRMIDemo`

Step 2 - Completing the Tasks

Complete the TODOs in `MPJAndRMIDemo.java` file. The process is as follows: every client send mathematical equations to the RMI listener on the master node; the master node communicates with the child nodes to solve that equation in MPJ. the equations are to be solved using `Problem.java` class. the result is then passed back to the master node.

4 After the Lab

The following section is optional / Take-Home exercise. For any help, you may go to a POD session.

Step 3 (Optional) - Running on Native MPI

You may want to run the code on a native MPI. To do so, visit `http://mpjexpress.org/readme.html` and follow instructions.
Note that running the code on native interface should not affect the design.
To complete this step, you need to have the following software installed on your machine: **cmake**, **gcc**, and **jdk**[1].

```
sudo apt install default-jdk-headless
sudo apt-get install cmake
sudo apt-get install gcc
```

Using ubuntu is recommended.

References

1. A Guide to RMI: `https://docs.oracle.com/javase/tutorial/rmi`
2. A Guide to MPJ Express: `http://mpjexpress.org/`
3. MPJ Express - README: `http://mpjexpress.org/readme.html`

[1] See `export JAVA_HOME` and using `find_package(JNI_REQUIRED)` in cmake.