Lab 1: Infrastructure Setup, Async-Finish Parallel Programming

Instructor: Vivek Sarkar

Course wiki : https://wiki.rice.edu/confluence/display/PARPROG/COMP322

Staff Email : comp322-staff@mailman.rice.edu

Importants tips and links

NOTE: It is recommended that you do the setup and execution for today's lab on your laptop computer instead of a lab computer, so that you can use your laptop for in-class activities as well. The instructions below are written for Mac OS and Linux computers, but should be easily adaptable to Windows with minor changes e.g., you may need to use \ instead of / in some commands.

Note that all commands below are CaSe-SeNsItIvE. For example, be sure to use "S14" instead of "s14".

edX site : https://edge.edx.org/courses/RiceX/COMP322/1T2014R

Piazza site : https://piazza.com/rice/spring2014/comp322/home

Java 8 Download : https://jdk8.java.net/download.html

IntelliJ IDEA : http://www.jetbrains.com/idea/download/

HJ-lib Jar File : http://www.cs.rice.edu/~vs3/hjlib/habanero-java-lib.jar

HJ-lib API Documentation : https://wiki.rice.edu/confluence/display/PARPROG/API+Documentation

HelloWorld Project : https://wiki.rice.edu/confluence/display/PARPROG/Download+and+Set+Up

1 edX Setup

We will use COMP 322's edX site, https://edge.edx.org/courses/RiceX/COMP322/1T2014R, for hosting videos, quizzes and module handouts. Please only register for COMP 322 on edX with your email address of the form, *your-netid*@rice.edu.

If you can access the above edX site, you are done and you can move on to the next section. If not, please send email to comp322-staff@mailman.rice.edu with a request to add your email address to the edX enrollment list for this class.

2 Piazza Setup

We will use COMP 322's Piazza site, https://piazza.com/rice/spring2014/comp322/home, for all discussions and Q&A. Please only register on this site with your email address of the form, *your-netid*@rice.edu.

If you can access the Q&A tab in this Piazza site, you are done and you can move on to the next section. If not, please send email to comp322-staff@mailman.rice.edu with a request to add your email address to the Piazza enrollment list for this class.

3 Subversion Setup

Each of you has a private repository for COMP 322 allocated in a "cloud" hosted by Rice's subversion (svn) server, svn.rice.edu. You can always examine the most recent contents of your svn repository by visiting `https://svn.rice.edu/r/comp322/turnin/S14/`*your-netid*. *It is possible that your svn account is not properly set up as yet. If you are unable to access the above URL, please send email to helpdesk@rice.edu cc'ing comp322-staff@mailman.rice.edu and requesting that they fix your access. After that, you can ignore this section for now (till you get access) and move on to the next section.*

The svn repository is empty to begin with, but will be populated with folders for homeworks and labs. We have a strict naming convention for these folders — "`hw_1`", "`hw_2`'', ... for homeworks and "`lab_1`", "`lab_2`", ... for labs. There are two ways in which you can turn in files to your subversion repository for lab and homework submissions:

1. You can use a Rice machine called CLEAR as a gateway to publish your data to svn. Rice IT has provided a script called "turnin" to enable you to submit content from a directory in your CLEAR account to your svn repository. Using this approach involves two steps — transferring your files to CLEAR, and transferring files from CLEAR to your svn repository. Both steps are explained below in detail.
2. (For advanced users) If you are familiar with subversion and have your own svn client on your local machine, you are welcome to use that instead.

You can follow the steps below to submit all your labs and homeworks using turnin on CLEAR. You should do your homework and lab work on a different system from CLEAR; it is important to not tie CLEAR down with long-running HJ computations. The instructions below include steps to copy files in a folder to CLEAR, and then to submit them. They are explained for a folder named *lab_1*, but you can use the same instructions for any homework or lab folder.

NOTE for Windows users: Skip **Step 1** below. Instead, install two pieces of software: putty and WinSCP. Putty allows you to connect to the clear servers, on which you can run programs and turn in assignments, while WinSCP will allow you to move files between clear and your computer.

1. To use putty, under the Host Name textbox, enter *netid@ssh.clear.rice.edu* and click open. Click yes, and when it prompts you for your password, enter your password and click enter (on the keyboard). Don't worry if you don't see the cursor moving. UNIX systems do not display password length as a safety mechanism.
2. To use WinSCP, enter *ssh.clear.rice.edu* for the hostname, your netid for the username, and your password in the password box. Click yes, and then continue.

For people new to UNIX systems, some common commands and usage tips will be posted on piazza soon, as well as some tips for using WinSCP
END NOTE for Windows users

1. **Step 1**: Transfer files from your local machine to CLEAR. This stage can be replaced by using a GUI tool such as WinSCP (www.winscp.net) to drag and drop files to CLEAR, if you prefer.
   (a) Go to the folder (in your machine) that contains all the files you need to submit. For now, you can create a new folder named `lab_1`, and a empty file named *DUMMY.txt* in that folder.
   (b) Zip the directory you want to submit.
   zip -r lab_1.zip lab_1
   (c) Use sftp to copy the zip file to CLEAR.
   sftp <your-netid>@ssh.clear.rice.edu

<your-password>
You should see the sftp prompt *'sftp>'* now.
mkdir comp322
The above command creates the comp322 directory. It may give an error message if the directory already exists, but that's fine. You can omit this step in the future once you know that the directory exists on CLEAR.
cd comp322
put lab_1.zip
You should see a confirmation that the zip file has been transferred.
quit

2. **Step 2**: Transfer files from CLEAR to the svn repository using turnin. To find out more about the turnin command type the following while logged on to CLEAR: *turnin -help*

   (a) Login to CLEAR
   ssh <your-netid>@ssh.clear.rice.edu
   <your-password>

   (b) Go to the comp322 directory
   cd comp322

   (c) Unzip the file
   unzip lab_1.zip

   (d) Delete the zip file (optional)
   rm lab_1.zip

   (e) Turnin the folder lab_1
   turnin comp322-S14:lab_1
   This should show all your files being added to the subversion. The first time you issue this command you will be asked if you wish to store your password unencrypted, twice.

   (f) Your submission is complete. You will need to repeat these steps at the end of today's lab to submit the work that you've done today.

NOTE: If you have problems with any homework or lab submission during the semester, just email your submission zip file to comp322-staff@mailman.rice.edu before the deadline.

4 Habanero Java library (HJ-lib) Installation and Setup

HJ-lib is a pure library implementation of the HJ constructs you have been learning in class. It relies on the use of Java 8 lambdas to simplify writing parallel programs. Being a library, the HJ constructs are exposed as method calls and we can write any valid Java code in Habanero programs.

You can follow the download and installation instructions from the wiki: `https://wiki.rice.edu/confluence/display/PARPROG/Download+and+Set+Up`. In summary, the instructions ask you to:

- Install Java 8 in your machine (the JDK version, not the JRE version) from the Oracle website.
- Install an IDE like IntelliJ IDEA[1]. An IDE is not strictly required since Java programs can also be written using a text editor and then compiled using the command-line. However, we strongly recommend using an IDE as it simplifies writing, compiling, running, and debugging your programs. A good IDE gives you error warnings, code completion and navigation, refactoring, syntax highlights, etc..
- Download the HJ-lib jar file from the url provided in the wiki.

[1]We cannot use DrJava for COMP 322 this semester because DrJava does not support Java 8 lambdas.

5 HelloWorld program

The first exercise is to familiarize yourself with the kind of code you will see and be expected to write in your assignments. The HelloWorldError.java program serves two main purposes:

1. Introducing Java 8 Lambda Expressions: Lambda Expressions are one of the most interesting changes in Java 8. An important idea with Lamba expressions is the ability to create a function and pass it just like any other parameter. A lambda expression represents an anonymous function. It comprises of a set of parameters, a lambda operator (`->`) and a function body. In Java, the lambda expressions are represented as objects, and so they must be bound to a particular object type known as a functional interface.
2. Introducing the Habanero Java library (HJ-lib) methods. The starter set for HJ-lib consists of four method calls:
   - `initializeHabanero()` must be called to initialize the HJ execution environment before invoking any other HJ-lib method.
   - `finalizeHabanero()` must be called to exit from the HJ execution environment. Once this routine is called, no HJ parallel construct may be called. Calls to `initializeHabanero()` and `finalizeHabanero()` must occur in pairs, and should not be nested.
   - `async` contains the API for executing a Java 8 lambda asynchronously. For example,

     ```
     async(() -> {S1; ...});
     ```

     spawns a new child task to execute statements S1, ... asynchronously.
   - `finish` contains the API for executing a Java 8 lambda in a finish scope. For example,

     ```
     finish(() -> {S1; ...});
     ```

     executes statements S1, ..., but waits until all (transitively) spawned `asyncs` in the statements' scope have terminated.

Go to `https://wiki.rice.edu/confluence/display/PARPROG/COMP322` and scroll to the bottom of the page (lab materials section) to download the HelloWorldError.java program. Get it running by creating a project of your own in IntelliJ.Attempting to run this program should result in errors such as:

```
cannot find symbol variable ss
```

Fix the error by replacing "ss" by "s" and running the program again. Ask your instructor / TA in the lab if you run into any issues.

6 ReciprocalArraySum Program

We will now work with the simple two-way parallel array sum program introduced in the Demonstration Video for Topic 1.1. This program has a slight modification — it computes the sum of reciprocals of elements in an array of doubles rather than the direct sum. You will need to use the turnin command to submit your work for this and other labs in the course.

- Download the ReciprocalArraySum.java program from the Code Examples at `https://wiki.rice.edu/confluence/display/PARPROG/COMP322` for Lab 1 in the course web page into the `lab_1` directory.
- The goal of this exercise is to create an array of `N` random int's, and compute the sum of their reciprocals in two ways:

1. Sequentially in method seqArraySum()
2. In parallel using two tasks in (the currently sequential) method parArraySum() with two loops in lines 58 and 62 for lower and upper halves of the array.

The profitability of the parallelism depends on the size of the array and the overhead of async creation. Your assignment is to use two-way parallelism in method parArraySum() to obtain a smaller execution time than seqArraySum().

- Compile and run the program in IntelliJ to ensure that the program runs correctly without your changes.
- Edit the current version to add two-way parallelism to method parArraySum().
- Experiment with different sizes (specified as an integer `N`) *e.g.,* 10^3, 10^4, 10^5, 10^6, 10^7. You can choose to change the value of `DEFAULT_N` or pass command-line arguments in IntelliJ. NOTE: You may get an OutOfMemoryError when experimenting with large values of `N`. If you get an OutOfMemoryError try smaller values of `N`. You will learn how to address that issue in a future lab.
- What speedup (ratio of sequential to parallel time) do you see for different values of `N`? Enter the speedups in a file named `lab_1_written.txt` in the `lab_1` directory.

7 Turning in your lab work

For each lab, you will need to turn in your work before leaving, as follows.

1. Check that all the work for today's lab is in the `lab_1` directory. If not, make a copy of any missing files/folders there. It's fine if you include more rather than fewer files — don't worry about cleaning up intermediate/temporary files.
2. Before you leave, create a zip file of your work by changing to the parent directory for `lab_1/` and issuing the following command, "`zip -r lab_1.zip lab_1`".
3. Use the turn-in script to submit the contents of the `lab_1.zip` file as a new `lab_1` directory in your turnin directory as explained in Section 3.

NOTE: If the turnin command does not work for you, please email your `lab_1.zip` *file to comp322-staff@mailman.rice.edu.*