Lab 3: Futures

Instructor: Vivek Sarkar

Resource Summary

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

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

Clear Login: ssh *your-netid*@ssh.clear.rice.edu and then login with your password

Important tips and links

As indicated earlier in a lecture, adding this call to the beginning of ArraySum2 will increase the limit on blocked threads:
System.setProperty(HjSystemProperty.maxThreads.propertyKey(), "200");

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 Example Program for Futures: ArraySum2

1. Update your `habanero-java-lib.jar` file! (`http://www.cs.rice.edu/~vs3/hjlib/habanero-java-lib.jar`). You need the latest version of the library for the programs provided today to work.
2. Download the `ArraySum2.java` file from the Code Examples column for Lab 3 in the course web page, `https://wiki.rice.edu/confluence/display/PARPROG/COMP322`.
3. Compile and run this java program.
4. Notice the following statistics printed at the end of program execution for the default array size of 128:
   (a) "`TOTAL NUMBER OF OPS DEFINED BY CALLS TO doWork()`" the total ($WORK$) in the computation in units implicitly defined by calls to doWork()

(b) "`CRITICAL PATH LENGTH OF OPS DEFINED BY CALLS TO hj.lang.perf.doWork()`", the critical path length (*CPL*) of the computation in units implicitly defined by calls to perf.doWork()

(c) "`IDEAL PARALLELISM = WORK/CPL`", the *ideal parallelism* in the computation

5. You can repeat the run for a different array size by clicking open the *Run* menu at the top and then choose *Edit Configurations*. In the popped out window, enter the size in *Program arguments* and click *OK*. Now run the program again.

   What WORK, CPL and IDEAL PARALLELISM values do you you see for different array sizes? Enter these values in a file named `lab_3_written.txt` in the `lab_3` directory. for array sizes that range across all powers of 2 up to 128 — 1, 2, 4, 8, 16, 32, 64, 128.

2 Array Sum Revisited with Variable Execution Times: ArraySum4

1. Download the `ArraySum4.java` file from the Code Examples column for Lab 3 in the course web page, `https://wiki.rice.edu/confluence/display/PARPROG/COMP322`.

2. The main difference compared to `ArraySum2.java` is that the call to `doWork()` in `ArraySum4.java` estimates the cost of an add as the number of significant bits in both operands. Thus, the cost depends on the values being added.

3. Again, enter WORK, CPL and IDEAL PARALLELISM values in `lab_3_written.txt` for array sizes that range across all powers of 2 up to 128 — 1, 2, 4, 8, 16, 32, 64, 128. While it is reasonable to see higher WORK and CPL values for ArraySum4 than ArraySum2, comment on how the IDEAL PARALLELISM for ArraySum4 compares with that of ArraySum2.

3 Binary Trees using Futures

1. Download the `BinaryTrees.java` file from the Code Examples column for Lab 3 in the course web page, `https://wiki.rice.edu/confluence/display/PARPROG/COMP322`.

2. Compile and run the program and record the WORK and CPL values. (They will be the same since this is a sequential program.)

3. Now modify the program by replacing "`final private TreeNode left;`" by "`final private HjFuture<TreeNode> left;'`" in line 76 and "`final private TreeNode right;`" by "`final private HjFuture<TreeNode> right;'`" in line 81. After this modification, you will need to replace references to left and right by using future expressions.

4. After you get your modified program to work, again record the WORK and CPL values, and comment in `lab_3_written.txt` if you've achieved an increase in parallelism, and, if so, why.

4 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_3` 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. Use the turn-in script to submit the `lab_3` directory to your turnin directory as explained in the first handout: *turnin comp322-S14:lab_3*. Note that you should *not* turn in a zip file.

   NOTE: Turnin should work for everyone now. If the turnin command does not work for you, please talk to a TA. As a last resort, you can create and email a `lab_3.zip` *file to comp322-staff@mailman.rice.edu.*