Lab 6: Barriers, Phasers

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:

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`

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

Linux Tutorial visit `http://www.rcsg.rice.edu/tutorials/`

As indicated earlier in a lecture, adding this call in your program before the call to `initializeHabanero()` will increase the limit on blocked threads:
System.setProperty(HjSystemProperty.maxThreads.propertyKey(), "200");

On SUGAR, JDK8 is already available at `/users/COMP322/jdk1.8.0` and HJ-Lib is already installed at `/users/COMP322/habanero-java-lib.jar`. Run the following command to setup the JDK8 path.

```
source /users/COMP322/hjLibSetup.txt
```

When you log on to Sugar, you will be connected to a *login node* along with many other users. To request a dedicated *compute node*, you should use the following command from a SUGAR login node:

```
qsub -q commons -I -V -l nodes=1:ppn=8,walltime=00:30:00
```

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

IMPORTANT: please refer to the tutorial on Linux and SUGAR, before staring this lab. Also, if you and others experience long waiting times with the "qsub" command, please ask the TAs to announce to everyone that they should use ppn=4 instead of ppn=8 in their qsub command (to request 4 cores instead of 8 cores).

1 One-Dimensional Iterative Averaging with phasers

1.1 Basic Phasers

1. Download the `OneDimAveraging.java` program from the course wiki by going to the wiki
2. The code in `OneDimAveraging.java` performs the iterative averaging computation discussed in the lectures. This code performs a sequential version of the computation in method `runSeq()` and a parallel chunked `for-finish-forasync–for` version in method `runForAllChunked`.
3. *Your assignment is to create a more efficient phased version of* `runForAllChunked()` *by using phasers with a barrier (*`next`*) operation instead.* Write this code in the space provided in `runForAllPhaser()`.
4. The input arguments for the main method in this program are as follows:
   (a) `tasks` = number of chunks to be used for chunked parallelism. The default value for `tasks` is `Runtime.getNumOfWorkers()`, which is the number of workers w specified with the "`-places` $1:w$" option (default is $w = 8$ on SUGAR).
   (b) `n` = problem size. Iterative averaging is performed on a one-dimensional array of size `(n+2)` with elements 0 and `n+1` initialized to 0 and 1 respectively. The final value expected for each element i is $i/(n+1)$. The default value for n is 200,000.
   (c) `iterations` = number of iterations needed for convergence. The default value is 20,000. This default was set for expediency. For this synthetic problem, you typically many more iterations to guarantee convergence.
   (d) `rounds` = number of repetitions for the entire computation. As discussed earlier, these repetitions are needed for timing accuracy. The default value is 3. For 3 repetitions, a reasonable approach is to just report the minimum time observed.
5. You should run your program on SUGAR, to evaluate the parallelization. As before, you can compile the program as follows:

   ```
   javac -cp /users/COMP322/habanero-java-lib.jar OneDimAveraging.java
   ```

   To run the program using 8 cores, use the following command on a *compute node*:

   ```
   java -cp /users/COMP322/habanero-java-lib.jar:.  -Dhj.numWorkers=8 OneDimAveraging
   ```

6. Record in `lab_6_written.txt` the best sequential and phaser-parallel times observed for the default inputs (using 8 cores), and then compute their ratio as the speedup. Compare your results for runForAllPhaser() with the results that you obtained in class for runForAllChunked().

1.2 Chunked Phasers

1. Now, modify runForAllPhaser to use not only phasers, but to have the various components being run as chunks (essentially, combine phasers and chunking)
2. You should run your program on SUGAR, to evaluate the parallelization. As before, you can compile the program as follows:

   ```
   javac -cp /users/COMP322/habanero-java-lib.jar OneDimAveraging.java
   ```

   To run the program using 8 cores, use the following command on a *compute node*:

   ```
   java -cp /users/COMP322/habanero-java-lib.jar:.  -Dhj.numWorkers=8 OneDimAveraging
   ```

3. Record in `lab_6_written.txt` the best time for this version of chunked phasers for the default inputs. Compare your results with Compare your results for the chunked and unchunked versions

2 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_6` 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_6` directory to your turnin directory as explained in the first handout: *turnin comp322-S14:lab_6*. 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_6.zip` *file to comp322-staff@mailman.rice.edu.*