Lab 7: Isolated Statements, Atomic Variables

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/`

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

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
qsub -I -V -l nodes=1:ppn=8,advres=classroom

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

IMPORTANT: Please refer to the tutorial on Linux and SUGAR from Lab 5, as needed. Also, if you edit files on a PC or laptop, be sure to transfer them to SUGAR before you compile and execute them (otherwise you may compile and execute a stale/old version on SUGAR).

As in past labs, create a text file named `lab_7_written.txt` *in the* `lab_7` *directory, and enter your timings and observations there.*

1 Parallelization using Isolated Statements

A parallelization strategy for the spanning tree algorithm was introduced this week in Lecture 19, along with an introduction to isolated statements. Recall the following constraints on isolated statements — an isolated statement may not contain any HJ statement that can perform a blocking operation e.g., finish, future get(), and phaser next/wait. In addition, a current limitation in the HJ implementation is that it does not support return statements within isolated.

Your task is to perform the following for the `spanning_tree_seq.java` program provided for the lab. *As always, please use a SUGAR compute node (not the login node) for all performance evaluations:*

1. Compile the sequential `spanning_tree_seq.java` program:
   javac spanning_tree_seq.java

2. Execute the program with a small problem size using two command line arguments, 1000 (number of nodes in graph) and 10 (number of neighbors):
   ```
   java spanning_tree_seq 1000 10
   ```

3. Parallelize this program by adding async, finish, and isolated constructs as described in Lecture 19. Call the parallelized version `spanning_tree_isolated.java`

4. Compile the parallel `spanning_tree_isolated.java` program:
   ```
   javac -cp /users/COMP322/habanero-java-lib.jar spanning_tree_isolated.java
   ```

5. Execute the program with 1 and 8 workers with a large problem size using two command line arguments, 100,000 (number of nodes in graph) and 100 (number of neighbors)[1]:
   ```
   java -cp /users/COMP322/habanero-java-lib.jar:. -Dhj.numWorkers=1 spanning_tree_isolated 100000 100
   java -cp /users/COMP322/habanero-java-lib.jar:. -Dhj.numWorkers=8 spanning_tree_isolated 100000 100
   ```

6. Record the best of 5 execution times reported for `spanning_tree_isolated.java` (1 and 8 workers) in `lab_7_written.txt`. What speedup do you see?

2 Parallelization using Object-based Isolation

Object-based isolation was also introduced in Lecture 19, with the form

```
isolated(obj1, obj2, ..., () -> <body>)
```

where $obj1, obj2, \ldots$ is a list of object references. Your task in this section is create a `spanning_tree_object_isolated.java` program that replaces `isolated` in your `spanning_tree_isolated.java` version by an equivalent object-based isolated construct. Compile and execute your program `spanning_tree_object_isolated.java` program by repeating the steps from the previous section. Record the resulting performance in `lab_7_written.txt`.

3 Turning in your lab work

1. Check that all the work for today's lab is in the `lab_7` 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_7` directory to your turnin directory as explained in the first handout: *turnin comp322-S14:lab_7.* 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_7.zip` *file to comp322-staff@mailman.rice.edu.*

[1] The sequential version will likely encounter a stack overflow for the large problem size, but the parallel version should run to completion successfully on 1 worker.