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

Coursera Login: visit `http://rice.coursera.org` and log in via Shibboleth

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

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

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.hj` 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.hj` program:
   hjc spanning_tree_seq.hj
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):
   hj -places 1:1 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.hj`
4. Compile the parallel `spanning_tree_isolated.hj` program:
   hjc spanning_tree_isolated.hj
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):
   hj -places 1:1 spanning_tree_isolated 100000 100
   hj -places 1:8 spanning_tree_isolated 100000 100
6. Record the best of 5 execution times reported for `spanning_tree_isolated.hj` (1 and 8 workers) in `lab_7_written.txt`. What speedup do you see?

2 Parallelization using Atomic Variables

Lecture 19 also introduced Java atomic variables. As discussed in the lecture, the operations that can be performed on atomic variables are limited to what is supported in the API, whereas isolated statements can be used to convert any general computation into critical sections.

Your task in this section is create a `spanning_tree_atomic.hj` program that replaces `isolated` in your `spanning_tree_isolated.hj` version by equivalent functionality using AtomicReference objects. In addition to the lecture slides, you can find a summary of AtomicReference operations at `http://docs.oracle.com/javase/6/docs/api/java/util/concurrent/atomic/AtomicReference.html`.

Compile and execute your program `spanning_tree_atomic.hj` 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. *NOTE: there is no quiz for Lab 7*
2. 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.
3. Before you leave, create a zip file of your work by changing to the parent directory for `lab_7/` and issuing the following command, "`zip -r lab_7.zip lab_7`".
4. Use the turn-in script to submit the contents of the `lab_7.zip` file as a new `lab_7` directory in your turnin repository as explained in Lab 1. You can always examine the most recent contents of your svn repository by visiting `https://svn.rice.edu/r/comp322/turnin/S13/`*your-netid*.