

Lab 9: Java Threads

Instructor: Vivek Sarkar

1 Setup on SUGAR

As before, run the following command on SUGAR to setup the environment for executing HJ and Java programs:

```
source /users/COMP322/hjsetup.txt
```

To request a dedicated *compute node*, you should use the following command (as usual) from a SUGAR login node:

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

When successful, it will give you a command shell on a dedicated 8-core compute node for your use for 30 minutes at a time. Your home directory is the same on both the login and compute nodes.

2 Convert to Java Threads - N-Queens

1. Download the `nqueens.hj` program from lab 4 that uses `finish` and `async` constructs along with `AtomicInteger` calls.
2. Convert it to a pure Java program by using Java threads instead of `finish/async`, using the concepts introduced in Lecture 27. (The `AtomicInteger` calls can stay unchanged.) For simplicity, you can include joins within each call to `nqueens_kernel()`. This is correct, but more restrictive than the `finish/async` structure for the given code. But it simplifies parallelization using Java threads.
3. Compile and run the program as follows to solve the N-Queens problem on a 14×14 board. Try different depth sizes.

```
javac nqueens.java
java nqueens 14
```
4. Compare the performance of your Java program with the given HJ program. (It is recommended that you use separate directories for compiling the Java and HJ versions so as to avoid any possible interference among classfiles generated for both versions.)

3 Convert to Java threads - Spanning Tree

1. Download the `spanning_tree_atomic.hj` solution from lab 7 that uses `finish` and `async` constructs along with `AtomicReference` calls.
2. Convert it to a pure Java program by using Java threads instead of `finish/async`, using the concepts introduced in Lecture 27. (The `AtomicReference` calls can stay unchanged.) For simplicity, you can include joins within each call to `compute()`. This is correct, but more restrictive than the `finish/async` structure for the given code. But it simplifies parallelization using Java threads.
3. Compile and run the programs as follows with the default input size.

```
javac spanning_tree_atomics.java
java spanning_tree_atomic
```
4. Compare the performance of your Java program with the given HJ program. (It is recommended that you use separate directories for compiling the Java and HJ versions so as to avoid any possible interference among classfiles generated for both versions.) You can feel free to add cutoff threshold values for this program as was done for N-Queens.