Lab 2: Abstract Metrics

Instructor: Vivek Sarkar

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

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

Goals for this lab

- Understand abstract metrics
- Understand actual speedup metrics
- One HJlib API: `doWork`

ReciprocalArraySum Program Revisited

This week we will revisit the simple two-way parallel array sum program introduced last week and in the Demonstration Video for Topic 1.1. You will edit the `ReciprocalArraySum.java` program provided in your svn repository for this exercise (do not use the version from the video lecture).

- The goal of this exercise is to create an array of `N` random doubles, and compute the sum of their reciprocals in several ways, then comparing the benefits and disadvantages of each:
  - Sequentially in method seqArraySum().
  - In parallel using **two** asyncs in method parArraySum_2asyncs(). You have already written code almost exactly like this in Lab 1 last week! Remember to add the calls to doWork() as seen in the seqArraySum() method to keep track of abstract metrics.
  - In parallel using **four** asyncs in method parArraySum_4asyncs(). You are essentially creating a version of parArraySum_2asyncs that uses 4 asyncs instead of 2. Think about the following questions: How do you want to split up the work among the 4 tasks? Equally? Is this the best way?
  - Lastly, in parallel using **eight** asyncs in method parArraySum_8asyncs(). You are essentially creating a version of parArraySum_2asyncs that uses 8 asyncs instead of 2. Think about the following questions: Do you really want to have to manually create 8 asyncs manually? Is there a better way you could write this function? Remember that copying and pasting code is generally discouraged.
- Compile and run the program in IntelliJ to ensure that the program runs correctly without your changes. Follow the instructions for "Step 4: Your first project" in https://wiki.rice.edu/confluence/pages/viewpage.action?pageId=14433124. If you're not use IntelliJ, you can do this by running the `mvn clean compile exec:exec -Preciprocal` command as specified in the README file.

  Be sure you **run the Lab2Test** file, not the ReciprocalArraySum file.
- Compare the abstract metric results and the actual speedup metric results and be able to explain the discrepancies before leaving lab. Note that the actual speedups depend on the input array size, which is 10^6 for today's lab, as well as the characteristics of your laptop.

Demonstrating and submitting in your lab work

For each lab, you will need to demonstrate and submit your work before leaving, as follows:

- Show your work to an instructor or TA to get credit for this lab (as in COMP 215). Be prepared to explain the lab at a high level, as well as answer the following questions:
  - What is the trend in the abstract metrics as the number of asyncs increases?
  - What is the trend in the actual speedup metrics as the number of asyncs increases?
  - Is there a dependency between these two trends? If so, what is this dependency? What might be causing it? How might you go about resolving this slowdown problem?
- Check that all the work for today's lab is in the `lab_2` directory. If not, make a copy of any missing files/folders there.
- Submit all your changes in the `lab_2` directory using subversion as explained in Lab 1. Remember to explicitly add any new files (e.g. your report or any new Java files) you created into the subversion repository.