Lab 11: Loop-level Parallelism

Instructors: Zoran Budimlić, Mack Joyner

Course Wiki: `http://comp322.rice.edu`

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

Goals for this lab

- Experimentation with Loop-Level Parallelism and Chunking in Image Convolution
- Performance Evaluation of Loop-Level Parallelism on NOTS

Downloads

As with previous labs, the provided template project is accessible through your private GitHub repo at: `https://classroom.github.com/a/fuV9bNiV`

For instructions on checking out this repo through IntelliJ or through the command-line, please see the Lab 1 handout. The below instructions will assume that you have already checked out the lab11 folder, and that you have imported it as a Maven Project in IntelliJ.

1 Image Kernels

The code provided in `Lab11.java` offers a sequential implementation of a convolution kernel which applies a transformation to an image. Your task for this lab is to make this sequential implementation parallel and measure its real world speedup on the NOTS cluster.

You should start by gaining an understanding of the application you will be parallelizing today. This website provides an excellent introduction to convolutional image kernels.

Next, try running the provided tests using the provided sequential code. These tests read the image at `src/main/resources/kobe.jpg`, apply two different transformations to it, and write the output of those transformations to `src/main/resources/kobe.1.jpg` and `src/main/resources/kobe.2.jpg`. The first transformation highlights edges in the image. The second transformation brightens the image. After running the provided tests you should be able to open `kobe.jpg`, `kobe.1.jpg`, and `kobe.2.jpg` next to each other and clearly see the results of the applied transformation.

You are now ready for the first part of the lab assignment, which is to *efficiently* parallelize the kernel in `Lab11.convolve`, by using the `forall` or `forallChunked` constructs in HJlib. (You are welcome to implement chunking by hand instead of using `forallChunked`.) It may help you to know that the bounds for the four loops are determined by the following values in the test case: inputWidth = 3280, inputHeight = 4928, and kernelWidth = kernelHeight = 3.

In order to do this, you will have to take a look into the sequential computation of the provided code. Which loops (i, j, kw, kh) are data-parallel and can be parallelized?

Once you have efficiently parallelized this kernel on your laptop, you should submit it to NOTS manually using the process detailed below (i.e. by transferring your changed Lab 11 to the cluster using scp/sftp/GIT and then submitting using `sbatch`).

2 Demonstrating and submitting your lab work

For this lab, you will need to demonstrate and submit your work before Wednesday, April 13th at 4:30pm as follows.

1. Show your work to an instructor or TA to get credit for this lab. They will want to see your files submitted to GitHub in your web browser and the passing unit tests on NOTS.
2. Check that all the work for today's lab is in your `lab11` directory by opening `https://classroom.github.com/a/fuV9bNiV` in your web browser and checking that your changes have appeared.