Low-Level Parallel Programming

Project Information and Lab Assignment 2
Lab 2 Deadline: February 17 23:59

1 Deadlines, Demonstrations, and Submissions

Each lab has to be submitted by its corresponding deadline. During the following lab session, demonstrate your working solution to a lab assistant. Be prepared to answer questions.

2 Lab Instructions

The objective of this lab is to recognize vectorization possibilities within serial code, and to be aware of the possibilities and limitations for SIMD operations and CUDA kernels. If a part of the code isn't suitable for a particular parallelization, then keep it sequential and defend your decision during the demonstration and the report.

Note: This lab requires much more effort than the previous one. Make sure to start on time.

2.1 Building CUDA on the Windows labs

1. Edit Libpedsim → Properties → CUDA C/C++ →Device →Code Generation
   - Add "compute_20, sm_20"
   - Uncheck box "Inherit from parent or project defaults"
2. Edit Demo → Properties → Configuration Properties → Debugging → Environment
   - Add "$(CudaToolkitBinDir)" to your path
3. Edit Demo → Build Customizations
   - Check checkbox for the CUDA toolkit

Now you can create a new CUDA C++ file, add it to your project and build it.

2.2 Parallelization with SIMD

Make an additional parallelization of the code section you sped up during lab 1, using SIMD (vector) operations. Vectorization is quite a different beast, compared to multi-threading, often requiring a different mindset than the object-oriented one. Do not be afraid of doing extensive structural changes in the project for this lab. However, it is an requirement that there must still be a functioning sequential, Pthread, and OpenMP version[1] when you are done. Below are some helpful line of questions to get you started.

- Is this code as it is now suitable for vectorization?
- Are there any mathematical operations that could have been vectorized if the code structure would have supported it?
- What structural changes to the code do you want to do in order to expose the vectorization possibilities?

Do not be afraid of modifying given code and even refactoring out entire functions, if you want. The code will not bite back (unless you program it to). Additionally, do not be afraid of discussing your ideas with the lab assistants.

2.3 Parallelization with CUDA

Make a final parallelization of the same code section with a CUDA kernel. You may also choose to use OpenCL for this. However, make yourself aware of the differences between OpenCL and CUDA. Depending on your prior solution, it may be a good idea to use the vectorized code as a starting point.

2.4 Evaluation

Obtain the execution time for all versions (Serial, OpenMP, PThreads, CUDA, SIMD), using the three scenarios given for this lab (available at Studentportalen). Use the number of threads that you have identified as optimal for OpenMP and Pthreads during lab 1. Run each experiment several times and use the average value. Visualize the data with a bar plot. You can change the scenario file by passing it to the command line, i.e.:

./demo hugeScenario.xml –timing-mode

[1] It is ok to modify the lab 1 code to accomplish this if you need to, just make sure that you have done all the timing measurements for the first report before you do.

You may want to decrease the total number of ticks for **hugeScenario.xml**. If you decide to change the number of ticks (**main.cpp**), remember to take that into account when visualizing and comparing your results.

2.5 Questions

A. What kind of parallelism is exposed with your code refactorization?

B. Give a short summary of similarities and differences between SIMD and CUDA.

C. Which parallelization has been most beneficial in these two labs? Support your case using plotted timing measurements.

D. What kind of code transformations were required for this lab, and why were they needed?

E. Compare the effort required for each of the four parallelization techniques. Would it have made a difference if you had to build the program from scratch, instead of using given code?

3 Submission Instructions

Submit the code and a **short** report including all answers in **PDF** format to student-portalen.

1. **NEW:** Please add a short note on how much each of you has contributed to this assignment. If you feel that you have contributed equally, it's enough to write "We have worked on this assignment together".
2. Your code **has** to compile.
3. Document how to choose different versions (Serial, Pthreads, OpenMP, Vector, and CUDA implementation).
4. Include a specification of the machine you were running your experiments on.
5. State the question and task number.
6. Include all generated plots.
7. Put everything into a directory and create a zip file and name it team-n-lastname$_1$-lastname$_2$-lastname$_3$.zip, where n is your team number.
8. Upload the zip file on Studentportalen by the corresponding deadline.

4 Acknowledgment

This project includes software from the PEDSIM[2] simulator, a microscopic pedestrian crowd simulation system, that was adapted for the Low-Level Parallel Programming course 2015 at Uppsala University.

[2] http://pedsim.silmaril.org/