Programming Parallel Computers

Aalto 2023

P: project

Please read the general instructions on this page first, and then check the individual tasks for more details. For each task you can download a zip file that contains the code templates you can use for development.

Task	Attempts	Expected	Points	Max	Rating	Rec.	Deadline for full points
P9a: project							
Please submit here the URL of your private Git repository.							
	0	–	–	5	★★+		2023-06-04 at 23:59:59

General instructions for this exercise

This is a small research project in which you are expected to explore both the theoretical and practical limitations of some real-world computer. You are free to use, for example, your own personal computer or some Linux computer that you can access remotely.

Overview

In this project you are expected to give (at least partial) answers to the following questions. You can focus either on the CPU, or the GPU, or both. If you only focus on the CPU, we would expect a very careful and precise analysis for full points (all the way to the level of individual machine-language instructions and execution ports).

(a) Theoretical limitations:

- Exactly what is the CPU/GPU that you have got in your own computer? What do you know about this specific CPU/GPU and its capabilities?
- What is the clock frequency at which it should run?
- How many floating-point operations should it be able to perform at best per second if you calculate it based on what you know about the capabilities of this CPU/GPU and its expected clock frequency? Please give detailed pointers to the documentation that you used to calculate these numbers.

(b) Practical limitations:

- Write a C++ program that performs as many arithmetic floating-point operations per second as possible using your CPU/GPU. The program does not need to do anything useful, but it has to take some input, then perform a very large number of arithmetic operations (that somehow depend on the input), and finally produce some output. You will need to calculate exactly how many arithmetic operations it will perform when you run it.
- Compile your C++ program with the right compiler options for your computer, and measure exactly how long it took to run your program. Please make sure your program does enough work so that it takes several seconds and you can get reliable time measurements. Please make sure you measure wallclock time, and you measure it correctly.
- While you are running your program, also gather as much useful information on what is happening in the computer (e.g. the actual clock frequency of the CPU, the number of instructions executed by the CPU, etc.).
- Calculate how many floating-point operations you managed to perform per second. We would expect this to be close to the numbers you got in part (a). If not, try to explain exactly why you got different results and why it is not possible to do much better in practice.
- If possible, also analyze the assembly code produced by the compiler to ensure that the compiler did exactly what you expected it to do.

(c) Comparison with your CP solutions:

- Measure also the performance of your fastest solutions to the CP exercises on the same computer. Again, calculate the number of floating-point operations that your code managed to perform per second.
- Compare the numbers with parts (a) and (b). Again, if there are differences, try to do your best to explain the differences. For example, if you suspect that you have got a memory bottleneck, try to demonstrate that if you modify your code so that memory lookups always hit L1 cache, your performance would be very close to the theoretical limitations.

What to submit and how?

You will need to set up a private Git repository in GitHub, and invite the GitHub user `suomela` as a collaborator to your private repository so that the lecturer can see what is in the repository.

Use this Git repository to organize your project work. Gather your notes, source material, C++ code, benchmark results, etc. in the repository.

In the root directory of your Git repository you will need to have also a written report. You can use any tools to write the report, but the end result has to be a single PDF file, with the file name `report.pdf`. The PDF file has to be committed and pushed to the Git repository.

In the PDF file, you should give your answers to the above research questions, for all three parts (a), (b), and (c). Please explain what you did and how, and give pointers to the C++ code and other material that is kept elsewhere in the same Git repository.

Once all is ready in the Git repository and it is ready for grading, simply submit here the link to your Git repository in GitHub. A URL that takes one to the front page of the repository is perfect, something like `https://github.com/USER/REPO/`.