Exercise Sheet 3

Introduction to OpenCL Programming

Lecture *Parallel Computing Systems*, Winter semester 2025/2026

Prof. Dr.-Ing. Mladen Berekovic
A discussion forum for the exercise can be found at: `https://moodle.uni-luebeck.de`

Requirements

1. Complete the following two problems in separate Notebook files.
2. In each Notebook, place all task answers together in the final IPython cell.
3. Always begin by running the template / your design with a small input size to confirm the notebook executes successfully from start to finish, before attempting performance improvements.
4. Provide a correctness check comparing the sequential and parallelized versions.
5. Before conducting a full benchmark, first test your design with the largest input size.
6. Read the requirements carefully, answer directly, and avoid unnecessary overengineering.
7. Google Colab is recommended to save time on environment setup. However, if you already have, or wish to build, a local self-hosted Jupyter server with GPU access, you are free to use it.

Vector Addition

Vector addition is a fundamental problem where two arrays of numbers are added element-wise to produce a third array. It is often used as a simple benchmark because it demonstrates parallelism clearly and serves as a building block for more complex computations.
OpenCL provides a framework to run such tasks on GPUs, which excel at handling large numbers of parallel operations, making vector addition an ideal example to showcase GPU acceleration.
A vector addition IPython Notebook template file is given, developed on a local self-hosted Jupyter server equipped with an NVIDIA GPU featuring 84 compute units. At the end of the template, a benchmark plot illustrates that increasing the workload processed in parallel enhances the speed-up metric (Figure 1), though this improvement quickly approaches a limitation. Moreover, because the input size is very large and OpenCL does not permit precise control over the number of compute units utilized, all available units were engaged despite efforts to adjust global and local work sizes to achieve the intended configuration.

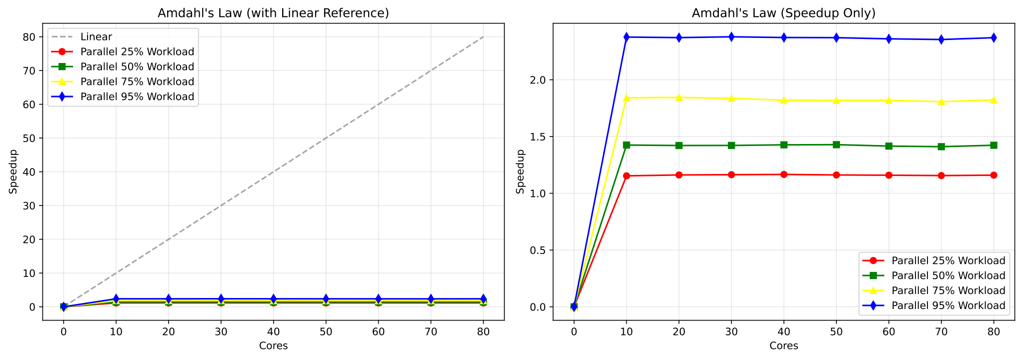

Figure 1: Speed-up of Vector Addition Illustrated by Amdahl's Law

Tasks:

1. Run the template file on Google Colab, but exclude the variation of compute units from the benchmark suite.
2. Make a rough estimation of actions should be taken to improve the design (e.g., adjust the suitable values for Global Work Size and Local Work Size). Update the source code accordingly, re-run the benchmark, and then observe and discuss your results. Hint: Don't aim for a perfect match between your estimation and the actual outcome - we want to see your learning process from planning to the moment of truth.
3. Increase the input size to stress the GPU's VRAM capacity, and include a screenshot showing system hardware utilization.

Matrix Multiplication

Based on the given template, work on Matrix Multiplication problem.

Tasks:

1. Develop a Matrix Multiplication problem in a separate Notebook.
2. Make a rough estimation of actions should be taken to improve the design. Update the source code accordingly, re-run the benchmark, and then observe and discuss your results.
3. Plot an Amdahl's Law speed-up figure based on the results obtained from running the benchmark suite.
4. Include a screenshot showing system hardware utilization when the GPU's VRAM capacity is being stressed.