Class Project
Proposal Due Nov 12, 2023 at 11:59pm

1 Objective

The purpose of the project is to apply data-parallelism and CUDA features to a more substantial piece of code than in the lab assignments. This could take many forms, including:

- Thorough performance analysis and improvement of existing GPU code or architectures
- Implement a GPU version of some existing computationally-intensive CPU code
- Reproduce some existing GPU research work
- Do novel GPU research
- And more!

At the end of the day, the goal is to demonstrate a thorough command of GPU programming and data-parallel concepts in an open-ended problem.

2 Project Outline

Broadly, a successful application-parallelization project might take the following steps.

Broad Outline	**Concrete Example**
Choose an application	Dense Matrix-Matrix Multiply
Determine what part of the application is taking the majority of the time	
Determine one or more data-parallel approaches to solving the problem	Assign one output to each thread in a gather-style approach
Create multiple implementations of the approach	One naïve version, one version with shared memory tiling, one version with global memory tiling
Measure the performance and execution characteristics of the implementations for various parameters	Record memory transfer time, kernel time, utilization, FLOPS, etc.
Relate results to course concepts	Performance may be impacted by utilization, shared-memory accesses, memory coalescing, and control divergence

This approach would be modified according to the exact goals of the project. For example, many GPU research projects have specific tasks and evaluation methods, and those would be the relevant targets for implementations.

2.1 Evaluation Criteria

Your project will be evaluated based on the following critetria:

- Demonstration of functionality and performance (50%): Produce correct results with good speedup relative to base code and/or existing implementation
- Final presentation (20%): Quality of oral delivery (in English)
- Report (20%): Clear, well-written, and organized report with in-depth analysis of the problem, the approach, and the results
- Code quality (10%): well-documented and organized code

3 Project Milestones

3.1 Project proposal: Nov 12 Sunday, 11:59pm

Short project proposal, as a PDF file via PLMS submission (2-page max). This proposal should include:

- a short description of the algorithm to be parallelized in CUDA
- a list of references
- a short summary of your parallelization and optimization strategies and/or how you want to improve on existing versions
- a break-down of the work into a short list of tasks with tentative completion dates

3.2 Presentation: Dec 14 Thursday, in class

You will give a 5-minute talk about your project during our online class hours. The talk should include brief introduction (including motivation and background) for your work, key design ideas and implementation details, experimental results, and conclusion. The slides should be made in English, while the talk can be given in either English or Korean.

3.3 Final report: the last day of the semester

Your final project report should be submitted to me via email in PDF form. It should be maximum 7 pages (10pt font, single-column article format), not including References and Appendix.

1. The problem statement and motivation (1/2 page)
2. Brief summary of existing work in the literature, with citations (1/2 page)
3. Overview of your parallel algorithm design (2 page):
   (a) Detailed design and implementation description with pseudo-code
   (b) Short examples that show how your code works in different scenarios. Choose the example carefully to highlight the major technical capabilities and limitation.
4. Evaluation (2-4 pages):
   (a) Evaluation methodology (system configurations, input dataset, compilation/runtime parameters, etc.)
   (b) Experimental results: Compare the performance of your algorithm with a baseline sequential version and/or other parallel versions
5. References
6. Appendix: a description of where to find your source code, executable, and input programs, and how to run the executable for example inputs