

TITLE: Applied High-Performance GPU Computing (GPUCompute)

Instructor: Colin Reinhardt, *Affiliate Assistant Professor, UW ECE*

<https://faculty.washington.edu/colinrei>

Summary:

Massively-parallel graphics processing unit (GPU) computations are an essential part of the meteoric rise and dominance of the modern machine-learning era. Without these fantastically complex and powerful devices and the corresponding parallel-programming ISAs, APIs, and paradigms the revolution of AI/ML could not have happened.

In this course we will begin a 3-quarter adventure of diving deeply into the hardware (HW) and software (SW) architectures of the latest generations of GPUs.

We will focus on NVIDIA GPUs and the CUDA SW infrastructure, but will also consider alternatives such as OpenCL/ROCm/HIP and the AMD/Intel GPUs.

We will progressively develop our understanding of the GPU device architecture and SW techniques to design and implement optimal programs by looking at the core computations and algorithms underlying and enabling state-of-the-art (SoTA) machine learning network training and inference. Along the journey we'll learn to design, implement, analyze the detailed HW/SW execution and performance, isolate and debug, and integrate programs which make optimal use of the massive parallelism and specialized HW units inside modern GPUs, including multi-level memory architectures, tensor cores, ray tracing cores, asynchronous memory units, distributed multi-GPU/node computations, and more.

- Learn the real-world skills and undocumented “black art” profiling and debugging techniques, tools and theory you will need to develop optimal parallelized GPU algorithms and software.
- Study underlying SoTA GPU processor microarchitectures & memory subsystems that govern performance, SIMD/SIMT/SPMD, including recent NVIDIA Hopper and Blackwell architecture additions such as 4th/5th-generation tensor cores and the tensor memory accelerator (TMA) unit, 3rd/4th-generation ray-tracing (RT) cores.
- Learn and practice GPU kernel design and coding, debugging, profiling, performance analysis, host- and device-side memory and cache usage analysis and optimization.
- Learn and use the current CUDA platform (13.x) and NVIDIA GPUs.
- Case studies focused on SoTA GPU-centric machine learning algorithms.

Schedule:

- **Week 1** : Course Intro & logistics, overview of parallel GPU computing and NVIDIA CUDA platform. Parallelism concepts.
- **Week 2** : CUDA multidimensional grids. CUDA device code. GPU HW architecture 101.
 - Homework-1
- **Week 3** : kernel languages. Profiling and HW Arch 201 and iterative optimization. Analysis of algorithms. Roofline model.
 - Homework-2
- **Week 4** : GPU HW Arch 201: memory hierarchy. Parallel performance theory. GEMM kernels. PTX/SASS.
 - Homework-3
- **Week 5** : Introduction to parallel patterns. Advanced kernel language features and kernel performance. Kernel profiling.
 - Homework-4
- **Week 6** : 2D/3D Convolution and stencil patterns. Tensor Cores.
- **Week 7** : GPU Numerical precision. Parallel histogram. atomic operations.
 - Final Project : *Proposal*
- **Week 8** : Parallel reduction pattern. FlashAttention
 - Final Project : *Design Specification*
- **Week 9** : Parallel prefix sum/scan patterns. FlashAttention improvements.
- **Week 10** : Key parallel algorithms in Machine Learning Workflows
 - Final Project : *Final Report and Code*

Course Overview:

There will be 10 lectures given via Zoom (one per week) and supplemented with additional video recordings and reading assignments. Hand-on programming exercises will be part of the weekly lecture. Additionally you can expect 4 homework assignments which will consist of (a) readings, (b) coding (c) theoretical/mathematical problems.

The remainder of the quarter will be focused on a final project which will be comprised of a fairly significant GPU code design and implementation on a topic approved by the course instructor, utilizing and applying tools and techniques learned in the class.

Grading:

- Homeworks (4) 55% : (11 / 12 / 12 / 20 pts)
- Final Project 45%

Late Homework Policy

Lose 1 pt if it's late, and another pt each additional 24hrs late until 5 days (at -6 pts). Then you lose 2 more at 10 days late and 1 more per additional 5 days until you hit zero pts.

If you have a personal emergency or a doctor's note (or some other legitimate extenuating circumstance) you will be granted an extension.

Course Materials:

Primary textbook:

Hwu, Kirk, El Hajj. *Programming Massively Parallel Processors*, 4th Ed. Morgan Kaufmann, 2022 (or 5th Ed if it releases in time...)

- Available through UW Libraries as online ebook.
- Get the O'Reilly App and create account using @uw.edu student SSO

Additional course reading materials and a list of supplementary reading materials will be posted on the course website. Nearly all materials will be available in online electronic formats, either freely available public literature or through UW Library (www.lib.washington.edu)

Course Prerequisites

- Proficiency programming with C and/or C++ programming and using integrated software development environments (Visual Studio, Eclipse) for building (compiling, linking) and visual/interactive debugging of programs.
- Comfort with applied matrix analysis and linear algebra and numerical analysis, eigensystems, eigenvalue problems and solutions, particularly the basic vector and matrix operations.
- Familiarity with vector calculus and partial differential equations (PDEs); physical foundations and formulation of PDEs, Maxwell's Eqns, the wave equation, and the dispersion equation.

Course Policies

You may collaborate and discuss homework assignments and project design and implementation with your fellow classmates, professor, TA and others. However, the work you submit must be your own, and you must write your own code(s). *Directly copying code without explicit attribution and/or plagiarizing is not allowed.*

AI Usage Policy

- You are allowed (and even encouraged) to use AI coding assistant tools, such as ChatGPT, CoPilot, Claude, Codex, Gemini, etc., for your homework programming assignments.
- Code obtained from AI-assisted coding should be manually enhanced with inline comments and descriptions of important variables, sections, algorithms, and other programmatic details.
- **However, it is still strictly prohibited to directly copy code from the Internet or from someone else.** Such actions will result in a score of zero and a report to the university.