Stanford CS149, Fall 2019

PARALLEL COMPUTING

This page contains lecture slides and recommended readings for the Fall 2019 offering of CS149. Lecture videos are available via **SCPD**.

Lecture 1: Why Parallelism? Why Efficiency?

(motivations for parallel chip designs, challenges of parallelizing code)

Further Reading:

- **The Future of Microprocessors**. by K. Olukotun and L. Hammond, ACM Queue 2005
- **Power: A First-Class Architectural Design Constraint**. by Trevor Mudge IEEE Computer 2001

Lecture 2: A Modern Multi-Core Processor

(forms of parallelism: multicore, SIMD, threading + understanding latency and bandwidth)

Further Reading:

- **CPU DB: Recording Microprocessor History**. A. Danowitz, K. Kelley, J. Mao, J.P. Stevenson, M. Horowitz, ACM Queue 2005. (You can also take a peak at the **CPU DB** website)
- **The Compute Architecture of Intel Processor Graphics**. Intel Technical Report, 2015 (a very nice description of a modern throughput processor)
- **Intel's Haswell CPU Microarchitecture**. D. Kanter, 2013 (realworldtech.com article)
- **NVIDIA GV100 (Volta) Whitepaper**. NVIDIA Technical Report 2017

Lecture 3: Parallel Programming Abstractions

(ways of thinking about parallel programs, and their corresponding hardware implementations)

Further Reading: (some fun systems)

- **ISPC Programmer's Manual**
- **Thread Building Blocks**
- **MIT's StreaMIT Project**
- **Data Parallel Haskell**
- **Brook for GPUs: Stream Computing on Graphics Hardware**

Lecture 4: Parallel Programming Basics

(the thought process of parallelizing a program)

Lecture 5: Performance Optimization I: Work Distribution and Scheduling

(achieving good work distribution while minimizing overhead, scheduling Cilk programs with work stealing)

Further Reading:

- **CilkPlus documentation**
- **Scheduling Multithreaded Computations by Work Stealing**. by Blumofe and Leiserson, JACM 1999
- **Implementation of the Cilk 5 Multi-Threaded Language**. by Frigo et al. PLDI 1998
- **Intel Thread Building Blocks**

Lecture 6: Performance Optimization II: Locality, Communication, and Contention

(message passing, async vs. blocking sends/receives, pipelining, increasing arithmetic intensity, avoiding contention)

Further Reading:

- **Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures**
- **Intel V-Tune**
- **Intel Performance Counter Monitor**

Lecture 7: GPU Architecture and CUDA Programming

(CUDA programming abstractions, and how they are implemented on modern GPUs)

Further Reading:

- You may enjoy the free Udacity Course: **Intro to Parallel Programming Using CUDA**, by Luebke and Owens
- The **Thrust Library** is a useful collection library for CUDA.
- **Rise of the Graphics Processor**. D. Blythe (Proceedings of IEEE 2008) a nice overview of GPU history.
- **NVIDIA GeForce GTX 1080 Whitepaper**. NVIDIA Technical Report 2016
- **NVIDIA Tesla P100 Whitepaper**. NVIDIA Technical Report 2016
- **NVIDIA Tesla V100 Whitepaper**. NVIDIA Technical Report 2017
- **The Compute Architecture of Intel Processor Graphics**. Intel Technical Report, 2015 (a very nice description of a modern Intel integrated GPU)
- **Pascal Tuning Guide**. NVIDIA CUDA Documentation

Lecture 8: Data-Parallel Thinking

(map, reduce, fold, scan, gather/scatter. Parallel implementations of scan. Data-parallel algorithm design.)