ECE408/CS483/CSE408 Spring 2025

Applied Parallel Programming

Lecture 1: Introduction

Before We Get Started

- Welcome to the course!
- The course is taught in-person
  - Lectures are in-class; they are also recorded and posted on Mediaspace
  - Labs (aka MPs) & Projects are on your own out of class activities
  - Midterm exams are in-person
- Lecture slides will be posted on-line prior to the lecture on the course's Canvas page
  - https://canvas.illinois.edu/courses/53796

People

Instructor: **Prof. Volodymyr Kindratenko <kindrtnk@illinois.edu>**

TAs:	**Howie Liu (hanwenl4@illinois.edu)**
	Yifei Song (yifeis7@illinois.edu)
	Shengjie Ma (sm138@illinois.edu)
	Yue Yuan (yuey10@illinois.edu)
	Xiyue Zhu (xiyuez2@illinois.edu)
UCAs:	**Hrishi Shah (hrishis2@illinois.edu)**
	Colin Zeng (kelinz2@illinois.edu)
	Charles Tianwei Pei (ctpei2@illinois.edu)
	Vijay Beekay Daita (vijayd2@illinois.edu)

About me

- Ph.D. from University of Antwerp, Belgium, 1997
- At NCSA since 1997
  - Director of the Center for AI Innovation (CAII)
- Research: Computing Systems, HPC, Computational Accelerators (FPGAs, GPUs), ML/AI systems & applications

AC – first GPU HPC cluster built in 2008 (used to teach this course too)

- 32 S1070 GPUs

HAL – first AI-oriented cluster built in 2018

- 64 V100 GPUs

Course Goals

- Learn to program massively parallel processors and achieve
  - High performance
  - Functionality and maintainability
  - Scalability across future generations

- Technical subjects
  - Parallel programming basics
  - Principles and patterns of parallel algorithms
  - Programming API, tools and techniques
  - Processor architecture features and constraints
  - Killer apps (aka AI ☺)

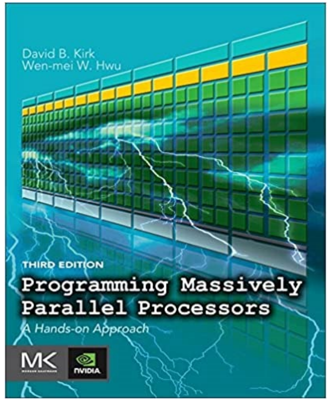
Web Resources

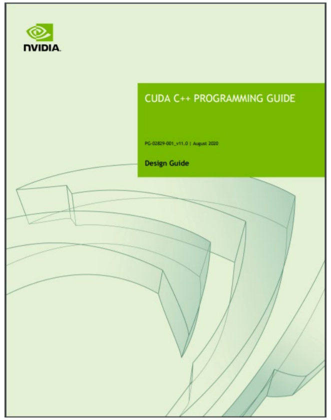
- Canvas (**https://canvas.illinois.edu/courses/53796**)
  - Syllabus, lecture slides, links to recordings, course files, etc.
  - Project reports
  - Grades
- GitHub
  - Lab and project assignments
- PrairieLearn
  - Lab quizzes
- Web board discussions in Campuswire
  - Channel for electronic announcements
  - Forum for Q&A – staff and your classmates can post answers

Text/Notes

1. D. Kirk and W. Hwu, “Programming Massively Parallel Processors – A Hands-on Approach,” Morgan Kaufman Publisher, 3rd edition, 2016, ISBN 978-0123814722

2. NVIDIA, *NVidia CUDA C Programming Guide*, (reference book) https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html





Tentative Schedule

- Week 1:
  - Tuesday: Lecture 1: Introduction
  - Thursday: Lecture 2: CUDA Intro
  - GitHub and computer access setup, Lab 0
- Week 2:
  - Tuesday: Lecture 3: Data Parallelism Model
  - Thursday: Lecture 4: CUDA Memory Model
  - Lab 1 due: VectorAdd example
- Week 3:
  - Tuesday: Lecture 5: CUDA Memory Model
  - Thursday: Lecture 6: Performance Considerations
  - Lab 2 due: Simple Matrix-Matrix multiplication

all *Labs (MPs)* and *Project Milestones* (PMs) are due on Fridays at 8:00pm US Central Time

Late Submission Policy

- Late Submission for Labs and Project
  - **All assignments are due by Friday at 8pm.**
  - However, recognizing potential issues such as Delta system maintenance, interviews, or illness, we provide a ***three-day grace period***.
  - You can submit an assignment by Monday at 8pm and still receive full credit without penalty. You do not need to notify us if you intend to use this default 3-day grace period.
  - However, to extend your grace period beyond 3 days, you must notify us by Friday 8pm via email to the TA(s) responsible for the assignment you are asking about. Extensions to the grace period are given out for extenuating circumstances at our discretion. Requests after this time will not be considered.
  - Please also note that the code in your GitHub repository at 8pm on Monday will be considered the final version, even if you had a correct version earlier. We will retrieve and grade the code at that time.

Grading

- Exams: 50%
  - Midterm 1: 25% -- ~ first 12 lectures
  - Midterm 2: 25% -- ~ the remaining lectures
- Labs (Machine Problems): 25%
  - Passing Datasets 90%
  - Correct answers to questions
  - Lowest graded lab will be dropped
  - Labs 0 and 1 are not counted in the final grade
- Project: 25%
  - Demo/Functionality/Coding Style: ~50%
  - Performance with full functionality: ~50%
  - Detailed rubric will be posted

Computer System for Assignments

- **Delta supercomputer** operated by NCSA will be used for working on labs (MPs) and projects
  - You will need to obtain an account on this system first
  - GitHub repo will include all the details about setting up the lab environment
- All Labs (MPs) and Project Milestones (PMs) are due on
  - Fridays at 8:00pm US Central Time
- Lab workflow:
  - #1: Get base code from GitHub → write your code & compile & run on Delta → submit final code for grading to GitHub
  - #2: Answer questions on Canvas <u>**AFTER**</u> #1 is completed
- **Lab 0 will have all the instructions to get started**

NCSA Delta Supercomputer

Delta is a dedicated, ACCESS-allocated resource designed by HPE and NCSA, delivering a highly capable GPU-focused compute environment for GPU and CPU workloads.

4-Way A40 GPU Compute Node Specs

Specification	**Value**
Number of nodes	100
GPU	NVIDIA A40
GPUs per node	4
GPU Memory (GB)	48 DDR6
CPU	AMD Milan
CPU sockets per node	1
Cores per socket	64
Cores per node	64
Clock rate (GHz)	~ 2.45
RAM (GB)	256

Getting Started with Delta - I

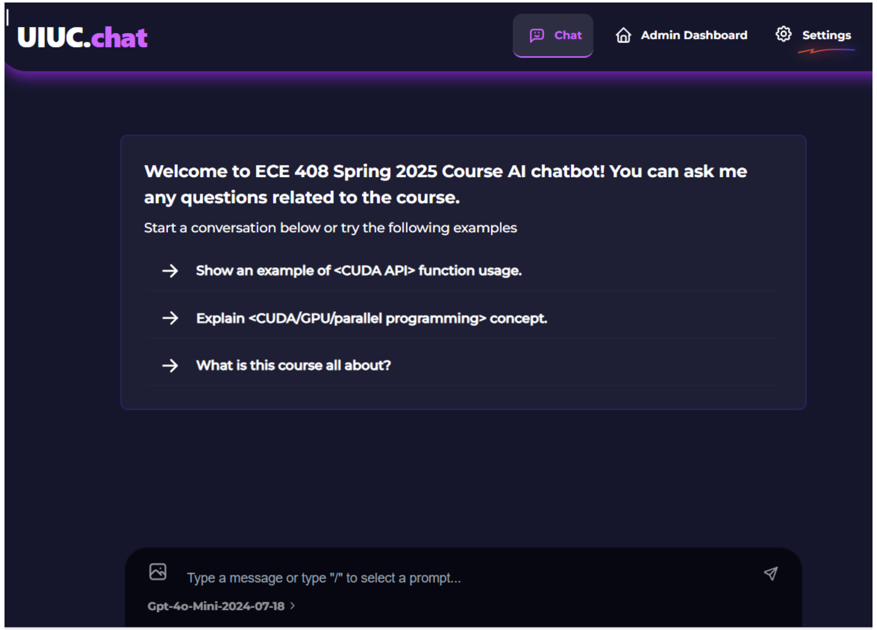
- Go to **https://identity.access-ci.org/new-user** and register yourself as a new user.
  - If you already have an ACCESS account (some of you might already have it due to a research project, etc.) do not register as new user, just use the same username.
  - When registering for new account, use "University of Illinois Urbana-Champaign" as your Identity Provider. This way, you can just use your UofI credentials.
- Once your account is created, note your ACCESS user ID and submit this user ID to the course staff via this form: **https://forms.illinois.edu/sec/1378190679**.
- Wait for an email from the **Delta team** confirming completion of your account registration. This may take anywhere between 24 to 48 hours.

Getting Started with Delta - II

- After you receive an email from the Delta team about your account, you should be able to login into Delta's head node `login.delta.ncsa.Illinois.edu`:
  - **`ssh yourusername@login.delta.ncsa.illinois.edu`**
  - DUO is needed for 2FA
- Read Delta's documentation at **https://docs.ncsa.illinois.edu/systems/delta/en/latest/**
- Clone course's GitHub repository and follow the instructions for editing, compiling, running and submitting your work. Important:
  - **Compile on the head node**
  - **Submit for execution to one of the compute nodes**

Course ChatBot

- https://uiuc.chat/ECE408SP25
- Developed by the CAII team at NCSA
- Uses GPT-4o-mini via OpenAI API or locally run Llama 3.1 70B
  - Context Stuffing and Mega-Prompting (with course materials)
- Please read the disclaimer posted on the course's website before using the chatbot!



Welcome to ECE 408 Spring 2025 Course AI chatbot! You can ask me any questions related to the course.

Start a conversation below or try the following examples

→ Show an example of <CUDA API> function usage.

→ Explain <CUDA/GPU/parallel programming> concept.

→ What is this course all about?

Type a message or type "/" to select a prompt...

Gpt-4o-Mini-2024-07-18 ›

Academic Honesty

- You are allowed and encouraged to discuss assignments with other students in the class. Getting verbal advice/help from people who've already taken the course is also fine.
- Any reference to assignments from previous terms or web postings is **unacceptable**.
- Any copying of non-trivial code is unacceptable
  - Non-trivial = more than a line or so
  - Copying includes reading someone else's code, including code produced by a chatbot, and then going off to write your own.
  - Those who have allowed copying will also be penalized

Academic Honesty (cont'd)

- Giving/receiving help on an exam is unacceptable.
- Deliberately sidestepping the lab requirements is unacceptable.
- Penalties for academic dishonesty:
  - Zero on the assignment/exam for the first occasion
  - Automatic failure of the course for repeat offenses

Permitted Use of Class ChatBot

- Asking about CUDA API, e.g.:
  - "What does cudaMemcopy do?" or "How do I call a CUDA kernel?"
- Asking about CUDA programming techniques, e.g.:
  - "How do I declare an array in shared memory?", or "How do I pass shared memory array size at the run-tine?"
- Asking about parallel programming concepts, performance optimization techniques, etc., e.g.:
  - "Explain how tiling helps with global memory bandwidth."
- Asking to help with compiler and run-time errors.
- Asking questions related to midterm exam preparation.
- Asking about general course details.

NOT Permitted Use of ChatBot

- Asking to generate code for an assignment, e.g.:
  - "Give me the CUDA kernel code for tiled matrix multiplication."
- Copy-paste lab quiz questions into the chatbot.
  - In many cases, the answer will not be correct anyway because the chatbot is not aware of all the assumptions.
  - Do not file regrade requests afterwards if you get a poor grade because the chatbot gave you an incorrect answer!
- Asking chatbot to fix bugs in your code by copy-pasting your entire code into the chatbot.

POST-DENNARD TECHNOLOGY PIVOT – PARALLELISM AND HETEROGENEITY

A major paradigm shift

- **In the 20th Century, we were able to understand, design, and manufacture what we can measure**
  - Physical instruments and computing systems allowed us to see farther, capture more, communicate better, understand natural processes, control artificial processes…

A major paradigm shift

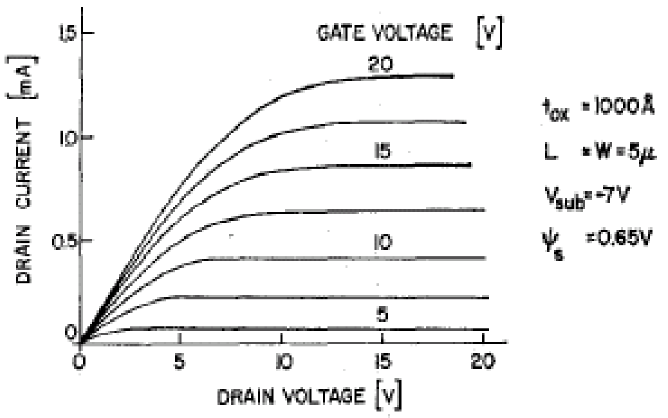
- **In the 21st Century, we are able to understand, design, and create what we can compute**
  – Computational models are allowing us to see even farther, going back and forth in time, learn better, test hypothesis that cannot be verified any other way, create safe artificial processes…

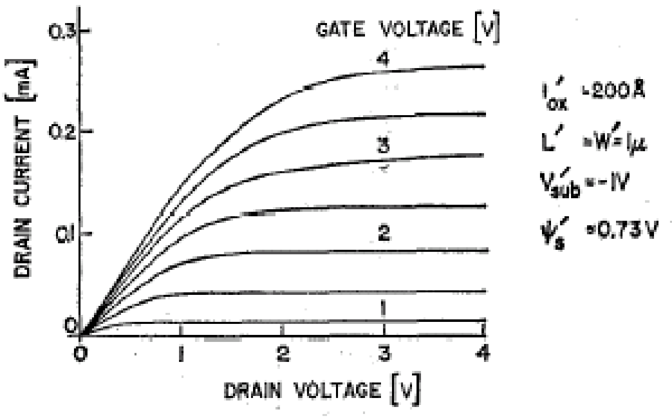
Examples of Paradigm Shift

- Conventional semiconductor Lithography → Computational correction
- Electron Microscopy → Computational Microscopy
- Film Photography → Deep-learning driven, computational imaging
- X-Rays → CT & MRI Scans with reconstruction
- Land-line phones → Zoom video conferencing
- Cars → Self-driving electric taxis
- Print and Broadcast Ads → AI-assisted Digital Ad Placement
- “Expert systems” → Multimodal Generative AI that can Plan and Execute tasks

Why is this happening now?

Dennard Scaling of MOS Devices

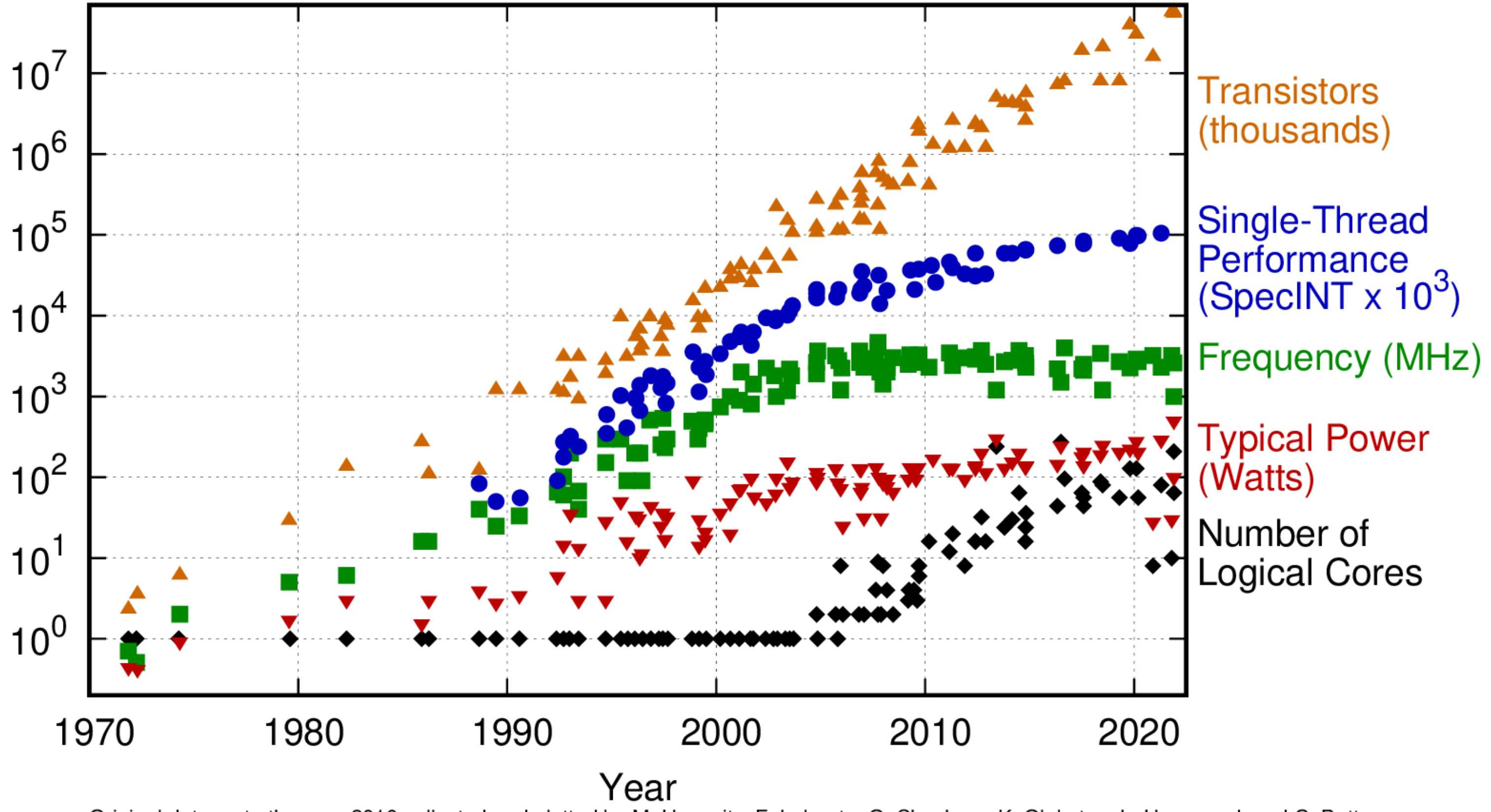




JSSC Oct **1974**, page 256

- In this ideal scaling, as L → α*L
  - V_{DD} → α*V_{DD}, C → α*C, I → α*I
  - Delay = CV_{DD}/I scales by α, so f → 1/α
  - Power for each transistor is CV^2*f and scales by α^2
    - keeping total power constant for same chip area

Microprocessor Trends

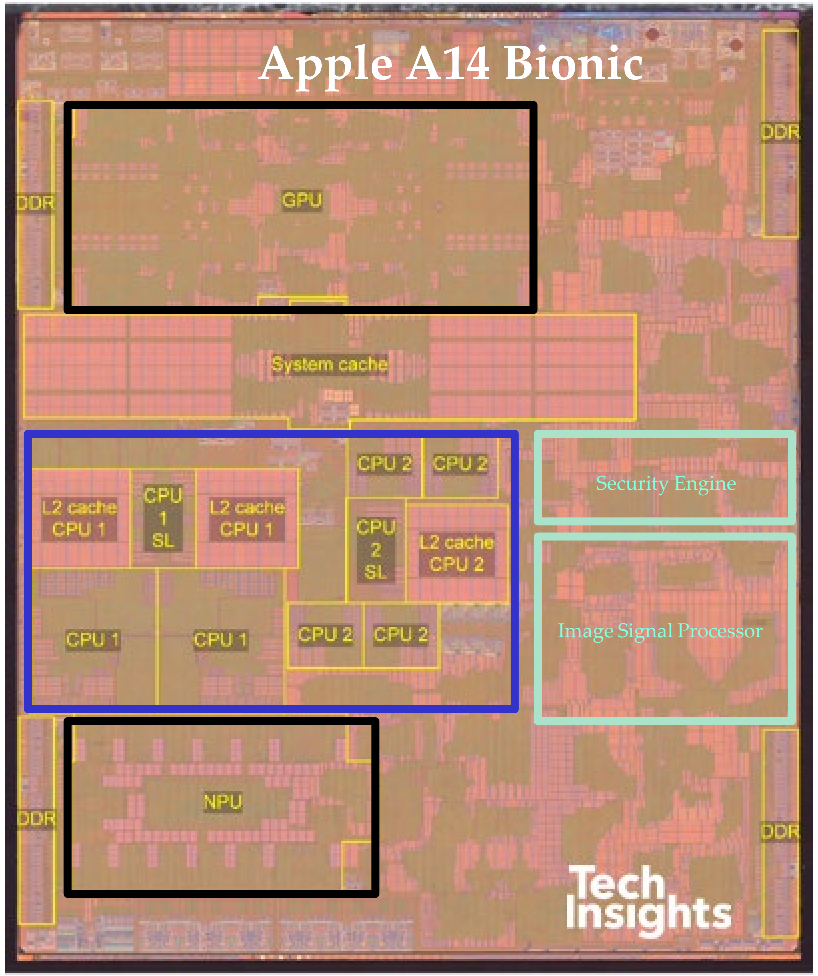


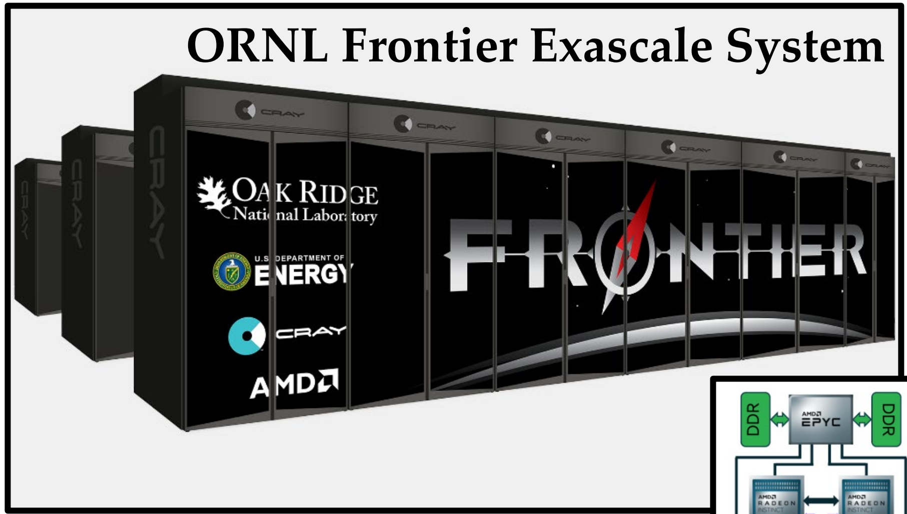
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

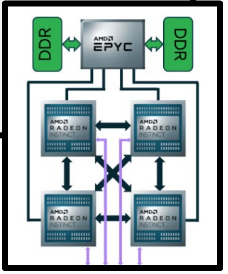
https://github.com/karlrupp/microprocessor-trend-data

Post-Dennard Pivoting

- Multiple cores with more moderate clock frequencies
- Heavy use of vector execution
- Employ both latency-oriented and throughput-oriented cores
- 3D packaging for more memory bandwidth



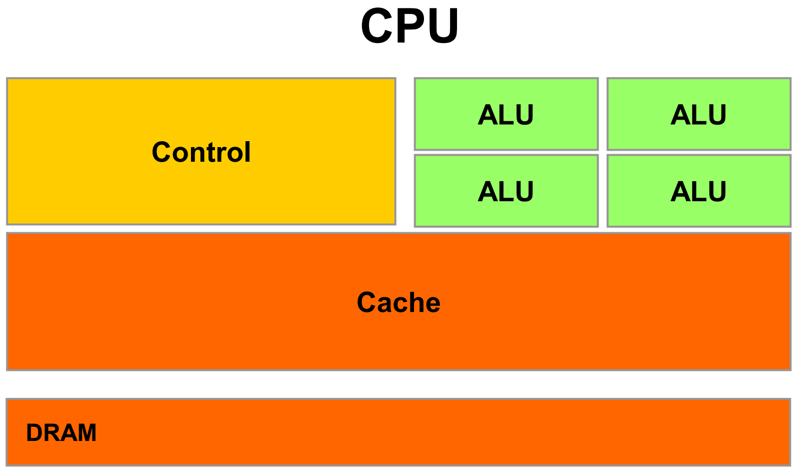

ORNL Frontier Exascale System
Oak Ridge National Laboratory
U.S. Department of Energy
CRAY
AMD
FRONTIER



From the smallest to the largest, computing today involves accelerators architectures.

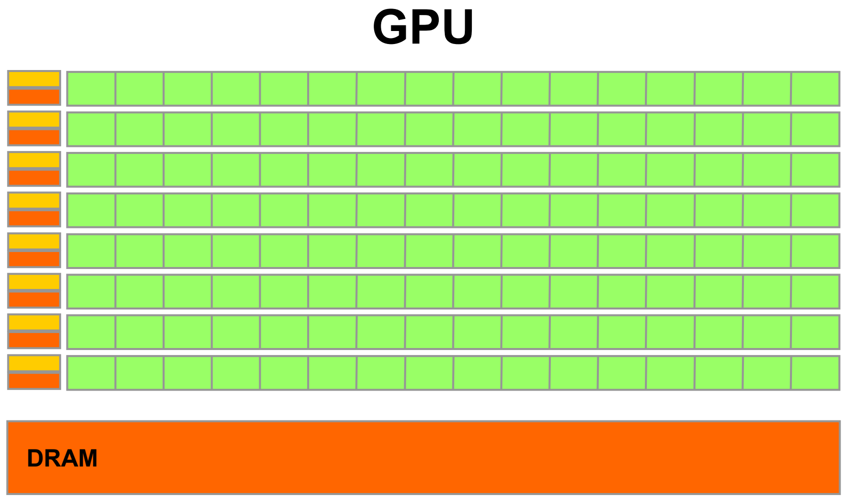
CPUs: Latency Oriented Design

- High clock frequency
- Large caches
  - Convert long latency memory accesses to short latency cache accesses
- Sophisticated control
  - Branch prediction for reduced branch latency
  - Data forwarding for reduced data latency
- Powerful ALU
  - Reduced operation latency



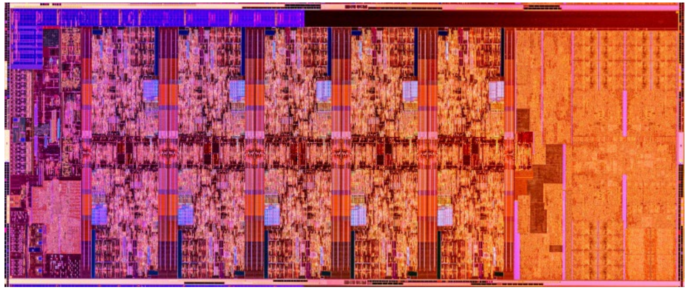
GPUs: Throughput Oriented Design

- Moderate clock frequency
- Small caches
  - To boost memory throughput
- Simple control
  - No branch prediction
  - No data forwarding
- Energy efficient ALUs
  - Many, long latency but heavily pipelined for high throughput
- Require massive number of threads to tolerate latencies

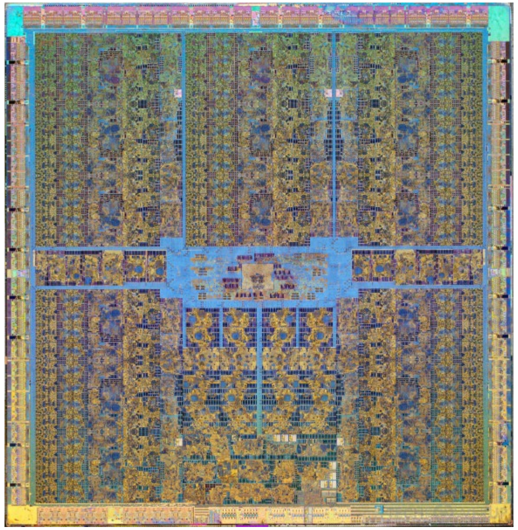


CPU vs GPU

- 10th Gen Intel Core processor
  - 10 cores silicon
  - 14 nm process



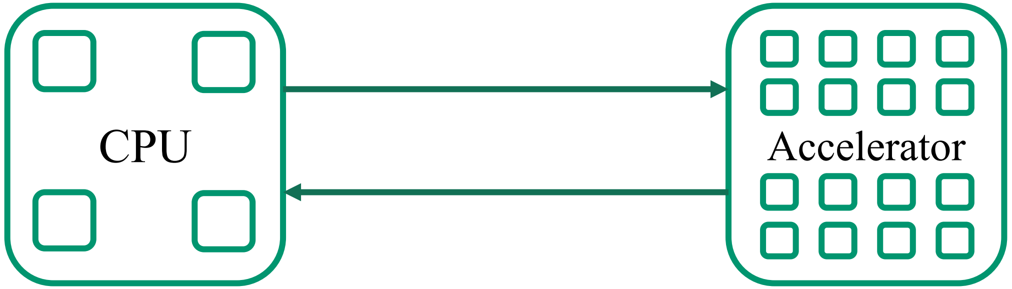
- NVIDIA GK110
  - 2,880 CUDA cores
  - 28 nm process



Winning Strategies Use Both CPU & GPU

- CPUs for sequential parts where latency hurts
  - CPUs can be 10+X faster than GPUs for sequential code

- GPUs for parallel parts where throughput wins
  - GPUs can be 10+X faster than CPUs for parallel code



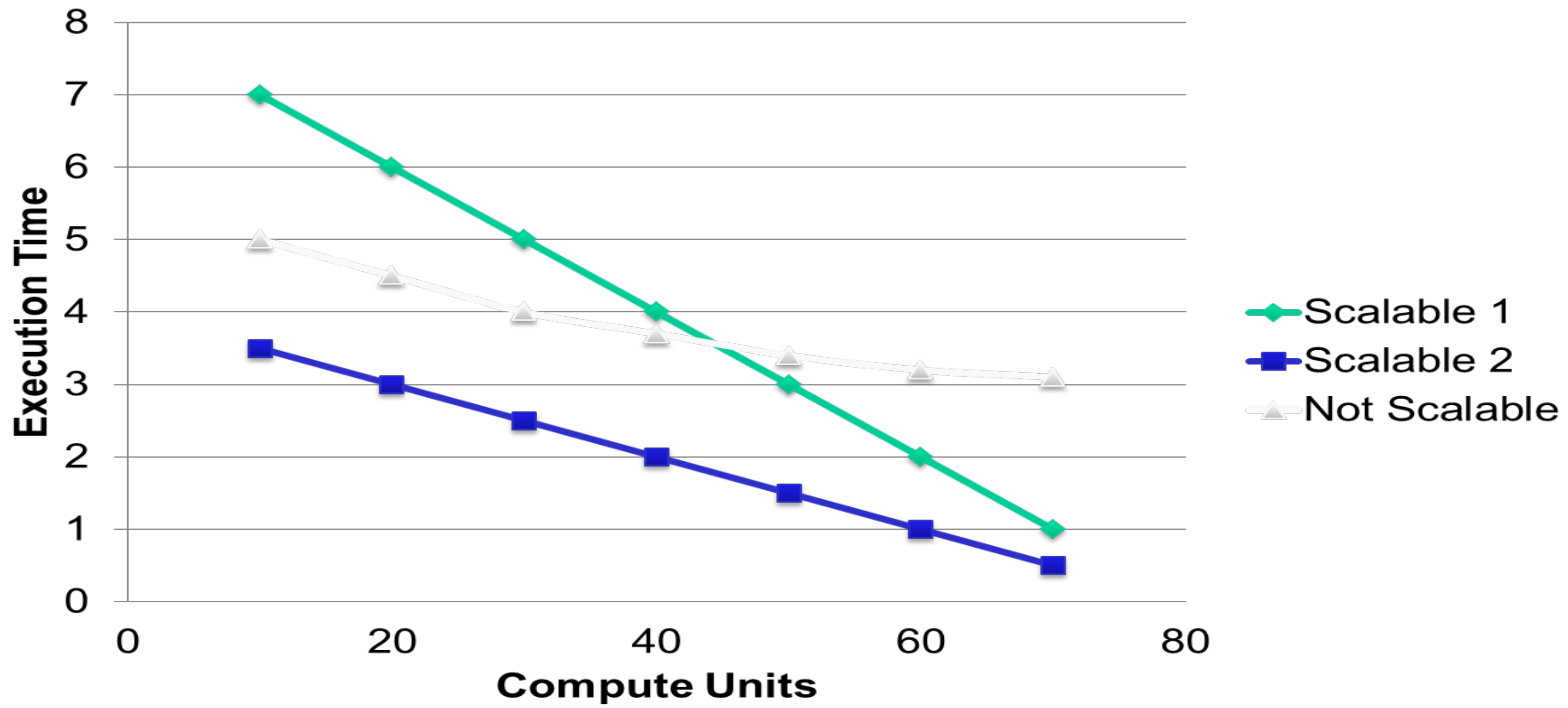
Heterogeneous Parallel Computing Applications

Financial Analysis	Scientific Simulation	Engineering Simulation	Data Intensive Analytics	Medical Imaging
Digital Audio Processing	Digital Video Processing	Computer Vision	Machine Learning	Electronic Design Automation
Biomedical Informatics	Statistical Modeling	Ray Tracing Rendering	Interactive Physics	Numerical Methods

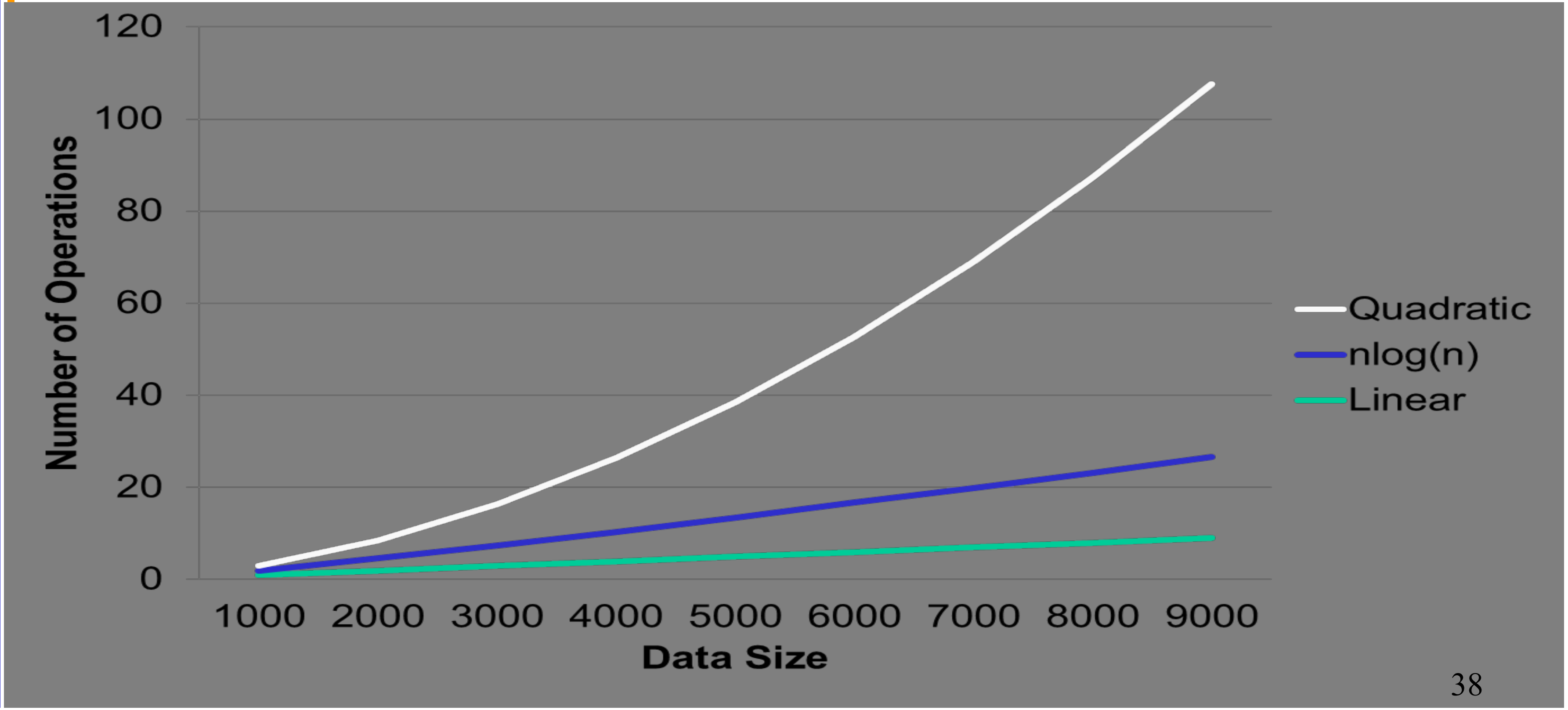
Parallel Programming Workflow

- Identify compute intensive parts of an application
- Adopt/create scalable algorithms
- Optimize data arrangements to maximize locality
- Performance Tuning
- Pay attention to code **portability**, **scalability**, and **maintainability**

Parallelism Scalability



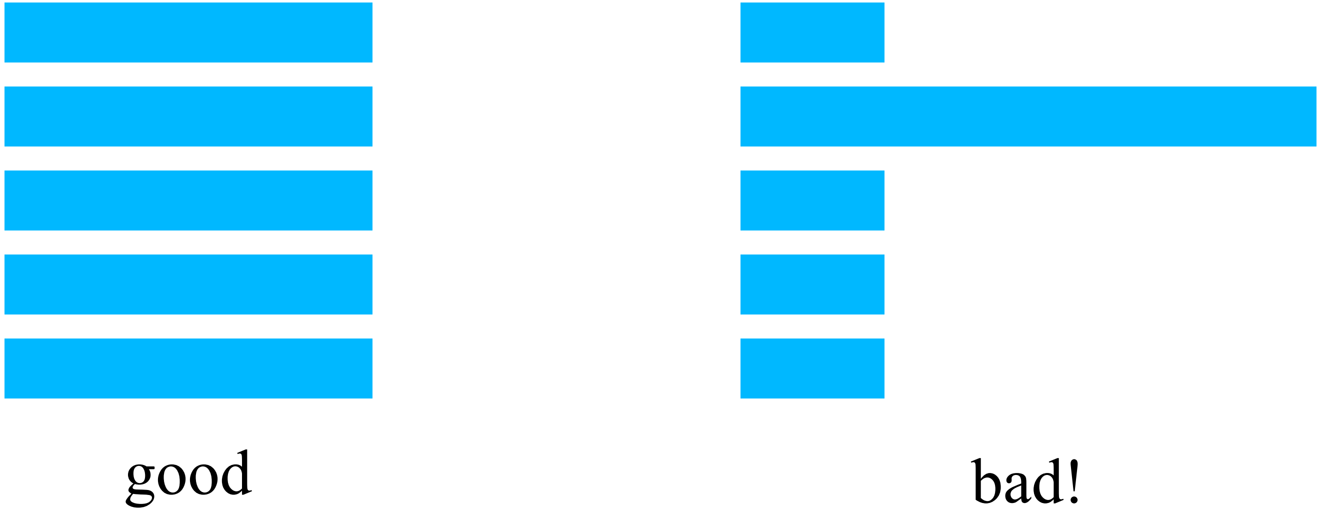
Algorithm Complexity and Data Scalability



Massive Parallelism - Regularity

Load Balance

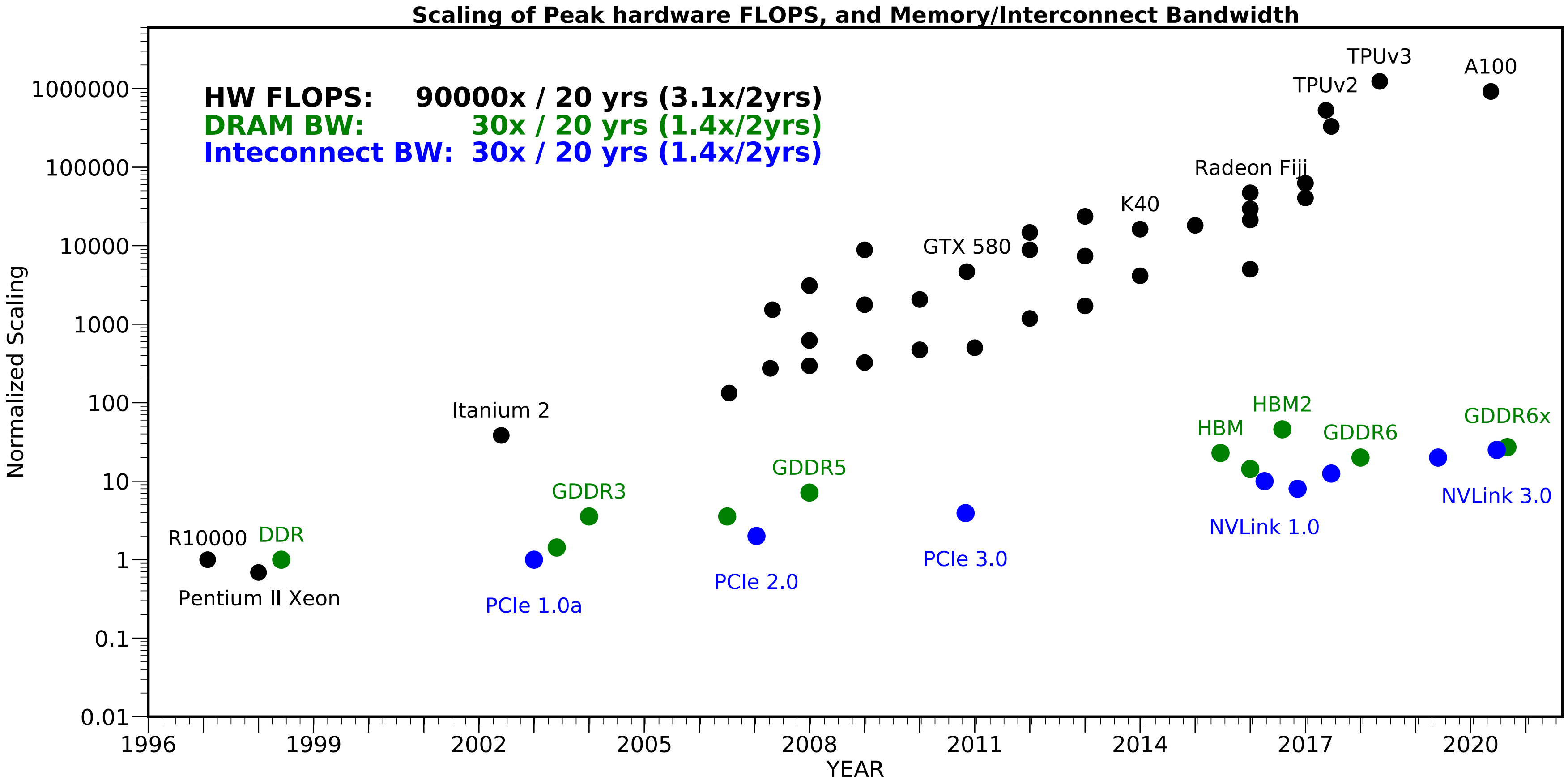
- The total amount of time to complete a parallel job is limited by the thread that takes the longest to finish



Global Memory Bandwidth

Ideal

Reality



Credit: Amir Gholami, et al, RiseLab Blog, March 2021

Conflicting Data Accesses Cause Serialization and Delays

- Massively parallel execution cannot afford serialization

- Contentions in accessing critical data causes serialization

What is the stake?

- Scalable and portable software lasts through many hardware generations

Scalable algorithms and libraries can be the best legacy we can leave behind from this era

ANY MORE QUESTIONS?