CASE SCHOOL
OF ENGINEERING
CASE WESTERN RESERVE
UNIVERSITY

CSDS 451: Designing High Performant Systems for AI

Lecture 1

8/26/2025

Sanmukh Kuppannagari

sanmukh.kuppannagari@case.edu

https://sanmukh.research.st/

Case Western Reserve University

Outline

- Course Information
- Course Overview
- Introduction to Heterogeneous Computing
- Introduction to Key AI/ML Kernels

Outline

- Course Information
- Course Overview
- Introduction to Heterogeneous Computing
- Introduction to Key AI/ML Kernels

Introduction

Assistant Professor, Case – Fall 2022 - present
Senior Research Associate/Post Doc, USC – Fall 2018 to Summer 2022
PhD, Fall 2018, USC
B.Tech, CSE, 2011, IIT Guwahati

Research Focus: Accelerating AI Algorithms

Team

3 PhDs, 3 Masters, 2 undergraduates

https://sanmukh.research.st/people

CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

Research

- Deploy real-time, low-power, low-latency CNN models on Processing-in-Memory (PIM) architectures
- Scaling Transformer Models to Extremely Long Sequences by Exploiting Sparsity
- Scalable Training of Geometric Graph Neural Networks
- Scalable Training of Fourier Domain Models

CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

Basic Information

- Lecture: Tuesday, Thursday 4:00 – 5:15 pm, Olin 305

- Office Hours
  - Location: Olin 506
  - Times: TBD

Course Goals

- **Objective:** Provide a broad overview of the challenges and opportunities that exist in designing high performance AI systems
  - Key Focus: Algorithmic Innovations

- Course Audience
  - Data Scientists who want to look under the hood and speed up their models
  - Parallel Algorithms or Hardware Acceleration researchers

Course Pre-requisites

- Algorithms
  - Complexity Analysis
  - Problem solving
  - …
- Mathematical/Computer Science maturity

- Familiarity with Programming (preferably C/C++)
  - Ability to write programs given an algorithm
  - Ability to debug your programs or find answers online if you are stuck
  - In this world of Generative AI, programming language should not be a barrier

- Most importantly, ability to quickly learn new concepts
  - Most critical skills for a computer scientist given how fast our field evolves

Course Objectives

- Theory
  - Parallel Computing Basics
  - Algorithms to Accelerate Convolutional Neural Networks (CNN), Large Language Models (LLMs), and Distributed Training on a Cluster

- Practice
  - BLAS libraries for High Performant AI/ML kernel implementations
  - Heterogeneous Programming using AMD HIP
  - Pytorch Lightning for distributed training on a cluster

Grading Policy (1)

- Grading
  - Written Assignments (30%)
  - Programming Assignments (20%)
  - Mid-term exam (15%)
  - Final exam (15%)
  - Project (20%)

- Watch out for extra credit assignments or bonus points in your assignments

Grading Policy (2)

- Written Assignments (30%)
  - 3 assignments to test the theoretical concepts covered in the class
  - Fairly involved
- Programming Assignments (20%)
  - 2 assignments on programming frameworks covered in the class
- Due date typically within 2 week from the assigned date

Grading Policy (3)

- Mid-term (15%)
  - 9th October
  - In class
- Final (15%)
  - 25th November
  - In class
- Similar to assignments but in a timed manner
- Will be open books/slides
- No exam during the final exam day

Grading Policy (4)

- Project (20%)
  - Implement one or more AI model
  - Using one or more acceleration technique
  - Using one or more heterogeneous programming frameworks
- Groups of 2
- Source code and project report due during the finals week
- I will provide more details during the semester on expectations, topics, team formation and submission guidelines

Grading Policy – Late Assignments

- 10% reduction in achieved score per day with a total grace of 3 days.
  0 points after 3rd day

- Only exceptions for medical reasons

Pre-Requisite Knowledge Assignment

- Programming Assignment 0 has been uploaded
- Objectives:
  - Warm up your C/C++ skills
  - Illustrative of the complexity of the programming assignments in the course
- Counts as 2% extra credits

Reading Materials

- No textbook as the topics discussed are based on cutting edge research
- Course slides are intended to explain topics in sufficient detail
- List of papers will be provided for students to review

CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

Course Guidelines (1)

- Please start assignments early
- Please take advantage of the office hours
- Asking questions in class is encouraged
- Please check the course canvas regularly

Course Guidelines (2)

- Policy on using AI tools
- Use of AI tools encouraged for background research
  - Caveat: beware of hallucinations or misinformation
  - Could be a good tool to understand a topic covered in class
  - Could be a good tool to debug your programs if you are stuck
- Please refrain from directly asking the solutions for written or programming assignments
- Please keep me updated on your use of these tools. We as faculty would like to learn how students can best benefit from these tools while maintaining academic integrity

Course Slack Channel

- To Be Created
- Will put an announcement on canvas

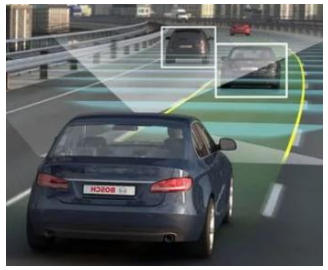
Outline

- Course Information
- Course Overview
- Introduction to Heterogeneous Computing
- Introduction to Key AI/ML Kernels

Why This Course? (1)

AI Increasingly Integral to Real-World

Autonomous driving



Surveillance

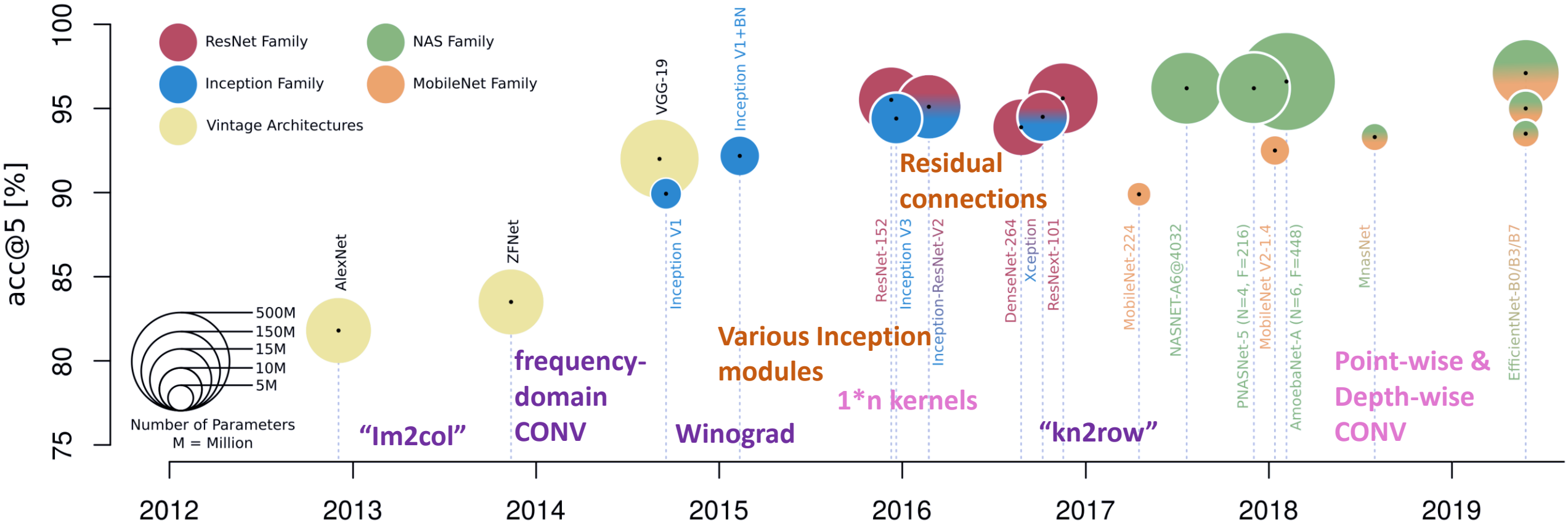
AI

Social Media

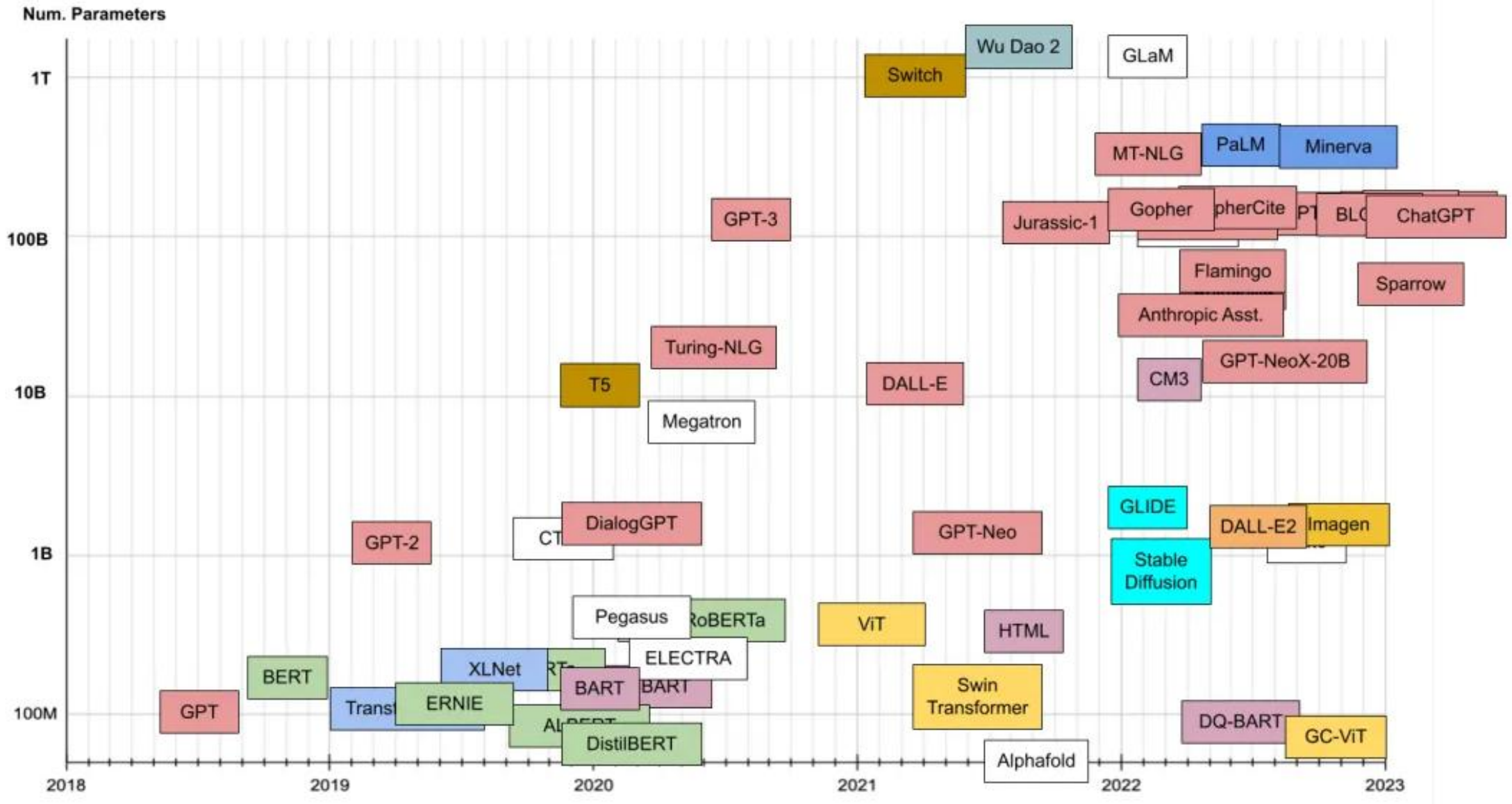
Power Grids

Why This Course? (2)

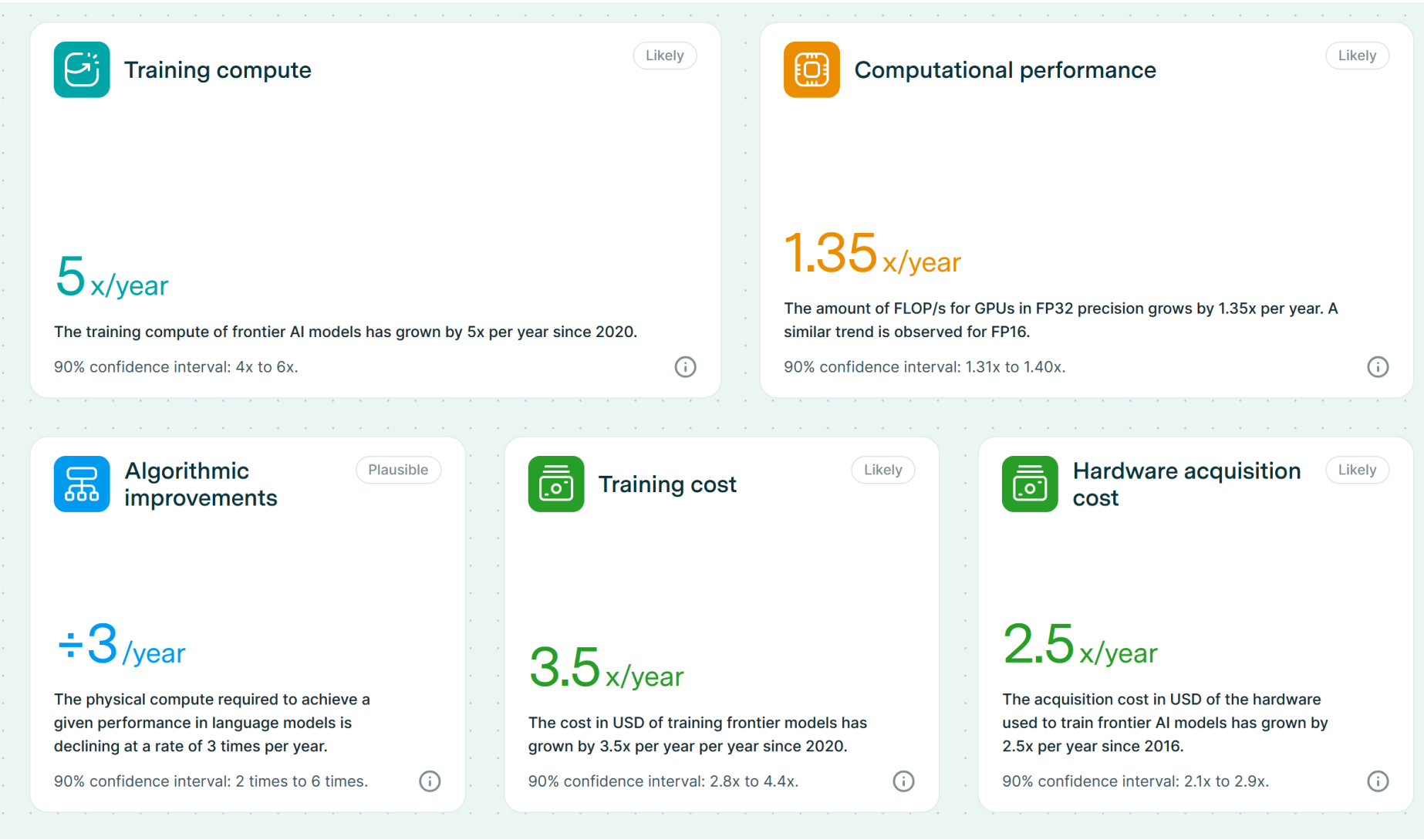
- Models are increasing in size and complexity



Why This Course? (3)



Why This Course? (4)

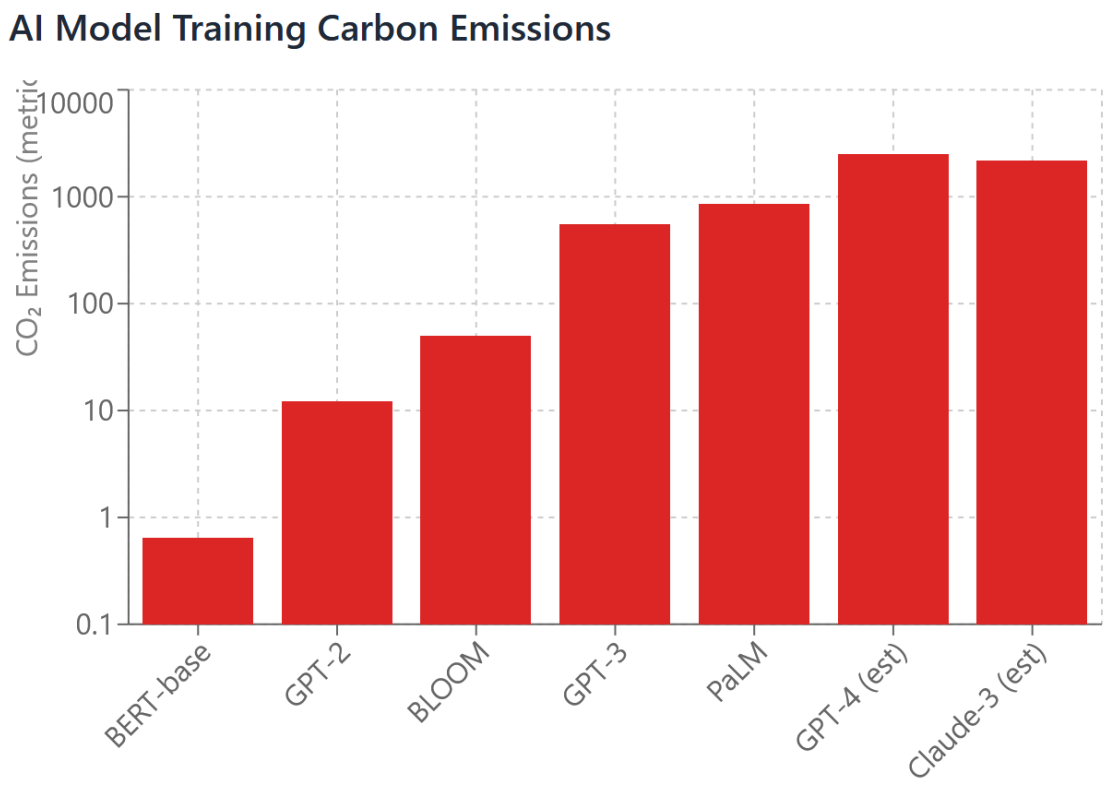


- Gap between training compute requirements and available computation performance keeps on increasing
- Source: https://epoch.ai/trends

Why This Course? (5)

- https://claude.ai/public/artifacts/c4d13724-3130-4d23-89b6-63b08514b299

- (Not sure how much CO2 got emitted when I created this interactive graph in Claude)



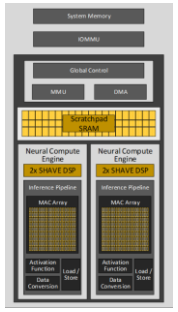
Key Insight: Training emissions have grown exponentially with model size. GPT-3 training produced equivalent emissions to **626,000 pounds of CO_2**- more than 100 times the annual footprint of an average person.

Why This Course? (6)

- Rise of Domain Specific Architectures
  - End of Moore's Law: Power becomes key constraint
  - Limitations and inefficiencies in exploiting instruction level parallelism: reaching Amdalh's law's limits
  - Shift in Application focus: individual devices, IoT, data-oriented, AI/ML…

CPU

+



NPU

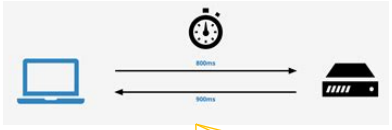
GPU

TPU

Figure from Presentation: The End of Moore's Law, CPUs (as we know them), and the Rise of Domain Specific Architectures, John Hennessy, September 2019

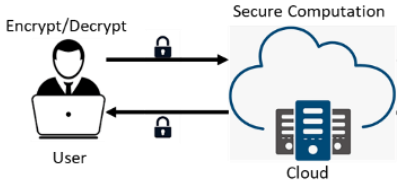
Why This Course? (7)

Performance
Latency, throughput, …



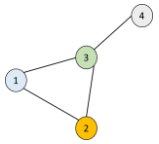
…

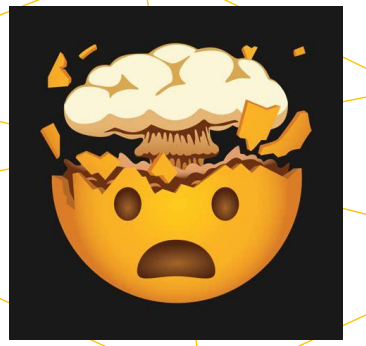
Privacy
Security Level, Scheme, …



…

Data
image, graph, …

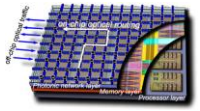




Systems Person

Power
Datacenter, edge, …

Programming
General Purpose, vector, spatial, …



Technology
Optical, ASICs, Reconfigurable

Why This Course? (8)

Data Scientist

PyTorch

TensorFlow

Why This Course? (9)

- Building accurate AI/ML models is not sufficient
- You should be able to deploy them efficiently in real world for them to be impactful
  - Time is $$$
- Deep learning frameworks are optimized for widely used use case scenarios
  - What if your problem does not fit into this?

Why This Course? (10)

- Hypothetical Scenarios (inspired by real world anecdotes)

- You are a data scientist at a security company that performs intrusion detection using images. The client wants the model to be hosted on their premises on a low cost server.

- You are a research scientist at a biomedical imaging company. A new dataset you want to explore has 10x the number of pixels and cannot be simply handled using your current GPU server.

Why This Course? (11)

- Hypothetical Scenarios (inspired by real world anecdotes)

- You are a data scientist at a security company that performs intrusion detection u to be hosted on their premi

Will you ask your client or company to upgrade server?

- You are a research scientist at a biomedical imaging company. A new dataset you want to explore has 10x the number of pixels and cannot be simply handled using your current GPU server.

Why This Course? (12)

- Hypothetical Scenarios (inspired by real world anecdotes)

- You are a data scientist at a security company that performs intrusion detection u to be hosted on their premi

Will you ask your client or company to find a systems/parallel computing engineer?

- You are a research scientist at a biomedical imaging company. A new dataset you want to explore has 10x the number of pixels and cannot be simply handled using your current GPU server.

Topics

- Parallel Computing Basics
  - Learn how to model three popular computing platforms – processor-memory architecture (GPUs), Systolic arrays (TPU), cluster
- Heterogeneous Computing Framework
  - Learn how to program CPU+accelerator platforms
- AI/ML Acceleration – Apply the above two concepts to accelerate AI models
  - Convolutional Neural Network Acceleration
  - Transformer Model Acceleration
  - Distributed Training of Neural Networks

Outline

- Course Information
- Course Overview
- Introduction to Heterogeneous Computing
- Introduction to Key AI/ML Kernels

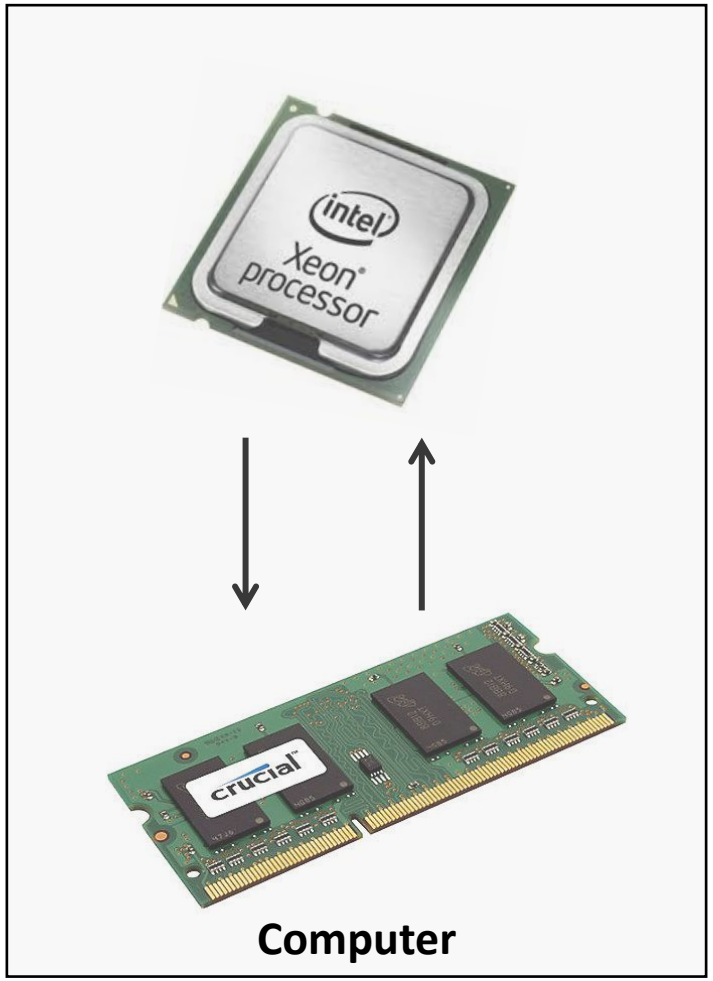
Acknowledgements

- Slides on Heterogeneous Computing are borrowed from the TOUCH program
  - Faculty Training Workshop for Teaching Heterogeneous Computing

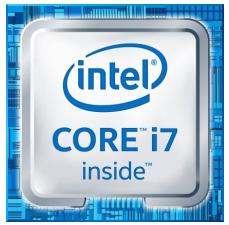
- Heterogeneous Computing Resources: https://touch.cs.txstate.edu/training_ws22_resources.php

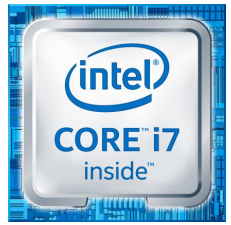
Von Neumann Architecture

What's a Computer?



Evolution of Parallelism in Processors





+

Single Core Computing

- Pipelining
- Out of order execution
- …

Multi-Cores/ Multi-Processor Computing

- Multi-threading
- Parallel For
- Parallel Tasks
- Tasks to core mapping
- …

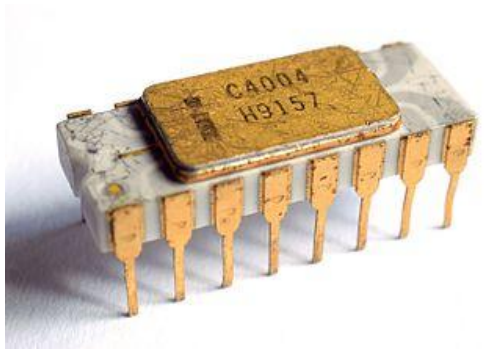
Heterogeneous Computing

- Parallel For
- Parallel Tasks
- Capability based Task to Device Mapping
- …

Hidden from Programmer

Explicit Specifications from Programmer

Single Core Computing



Intel 4004

Intel Pentium

Single Core Computing

code

```
int main() {
      int x, y, result;
      x = 17;
      y = 13;
      result = x + y;
      return result;
}
```

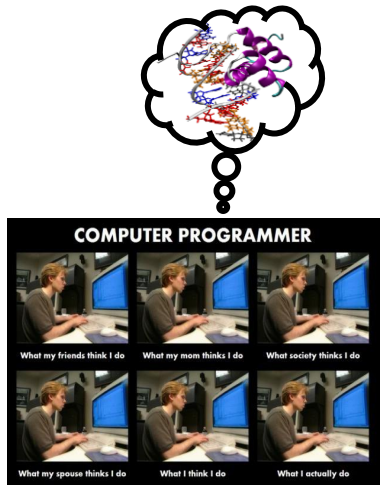
compile

```
            .text
.globl _main
_main:
LFB2:
            pushq       %rbp
LCFI0:
            movq        %rsp, %rbp
LCFI1:
            movl        $17, -4(%rbp)
            movl        $13, -8(%rbp)
            movl        -8(%rbp), %eax
            addl        -4(%rbp), %eax
            movl        %eax, -12(%rbp)
            movl        -12(%rbp), %eax
            leave
            ret
```

execute

....1100101010100...

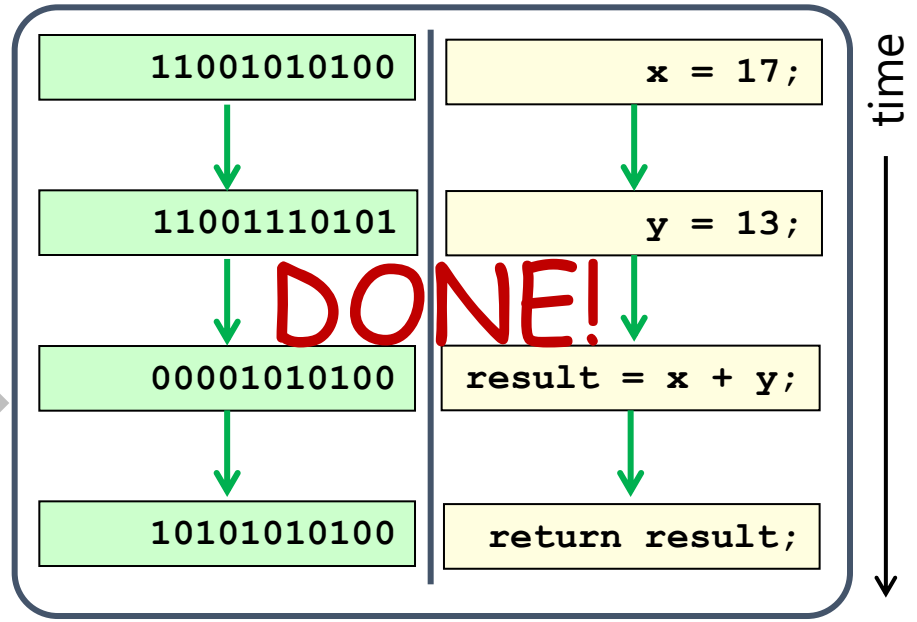
Program Execution



code

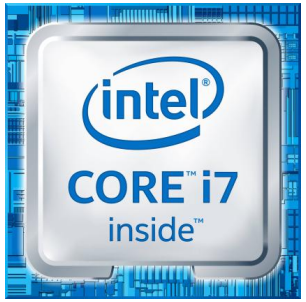
```
int main() {
        int x, y, result;
        x = 17;
        y = 13;
        result = x + y;
        return result;
}
```

compile



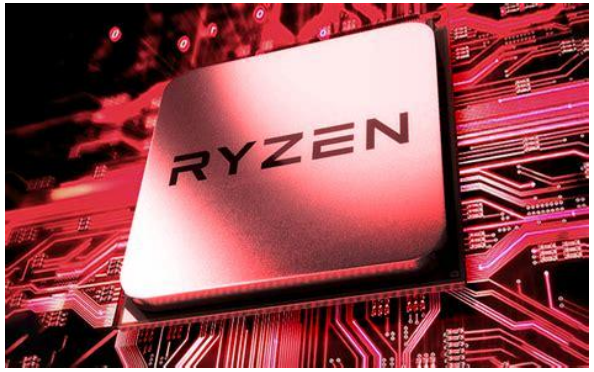
Instruction execution follows program order

*Processor executes one instruction at a time**



Multi-Core/Multi-Processor Computing

Intel i7



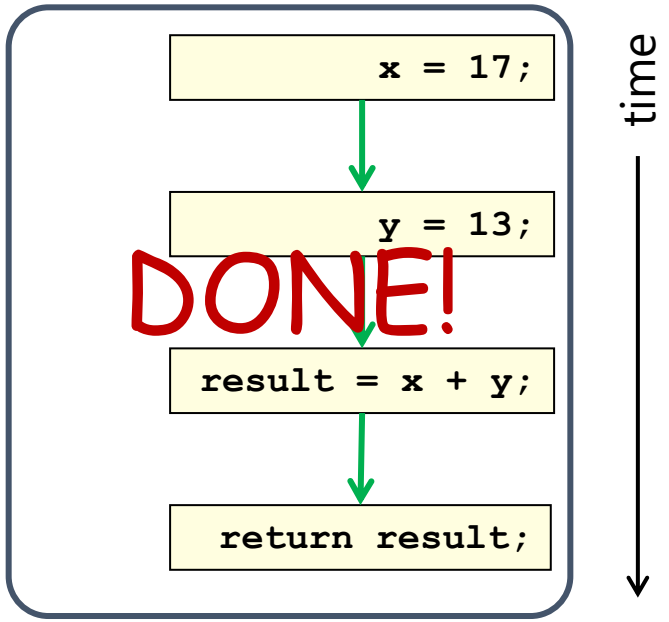
AMD Ryzen

Parallel Program Execution

code

```
int main() {
        int x, y, result;
        x = 17;
        y = 13;
        result = x + y;
        return result;
}
```

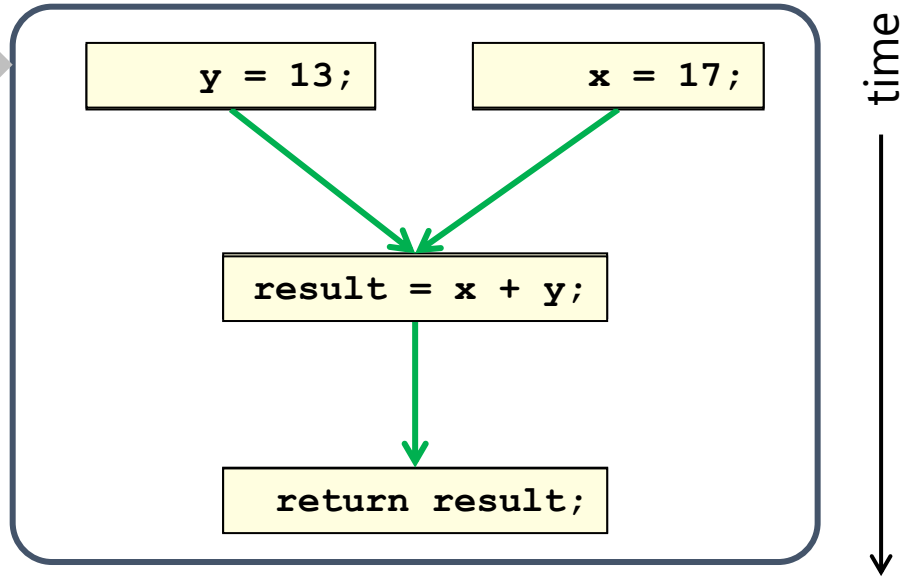
compile



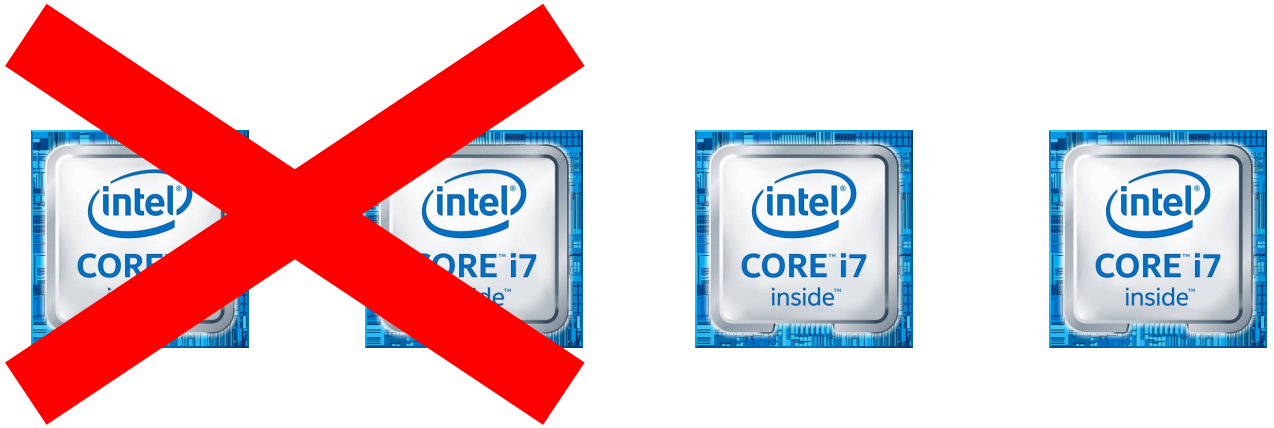
The two assignment statements
x = 17; and y = 13; will execute in parallel

Parallel Program Execution


```
int main() {
      int x, y, result;
      x = 17;
      y = 13;
      result = x + y;
      return result;
}
```

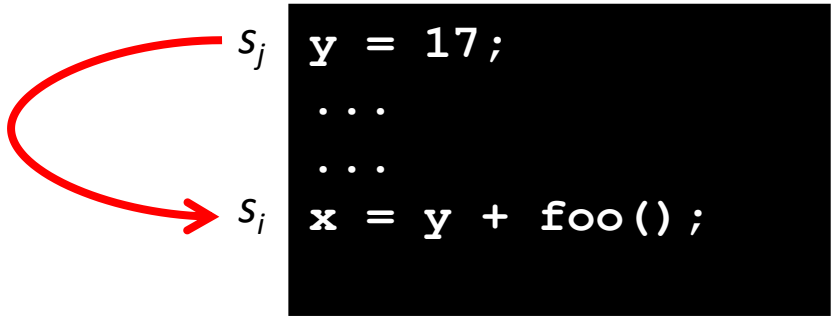


Cannot arbitrarily assign instructions to processors



Dependencies in Parallel Code

- If statement s_i needs the value produced by statement s_j then, s_i is said to be dependent on s_j



- All dependencies in the program must be preserved
- This means that if s_i is dependent on s_j then we need to ensure that s_j completes execution before s_i

responsibility lies with programmer (and software)

Heterogeneous Computing

Heterogeneity

- CPU + XPU
  - XPU = GPU, FPGA, NPU, custom accelerator
- CPU
  - Runs OS; performs general bookkeeping
  - Offloads Task to XPU
- XPU
  - Processors specialized to perform a particular type of task
  - Trades off generalizability with performance

CPU

+

FPGA

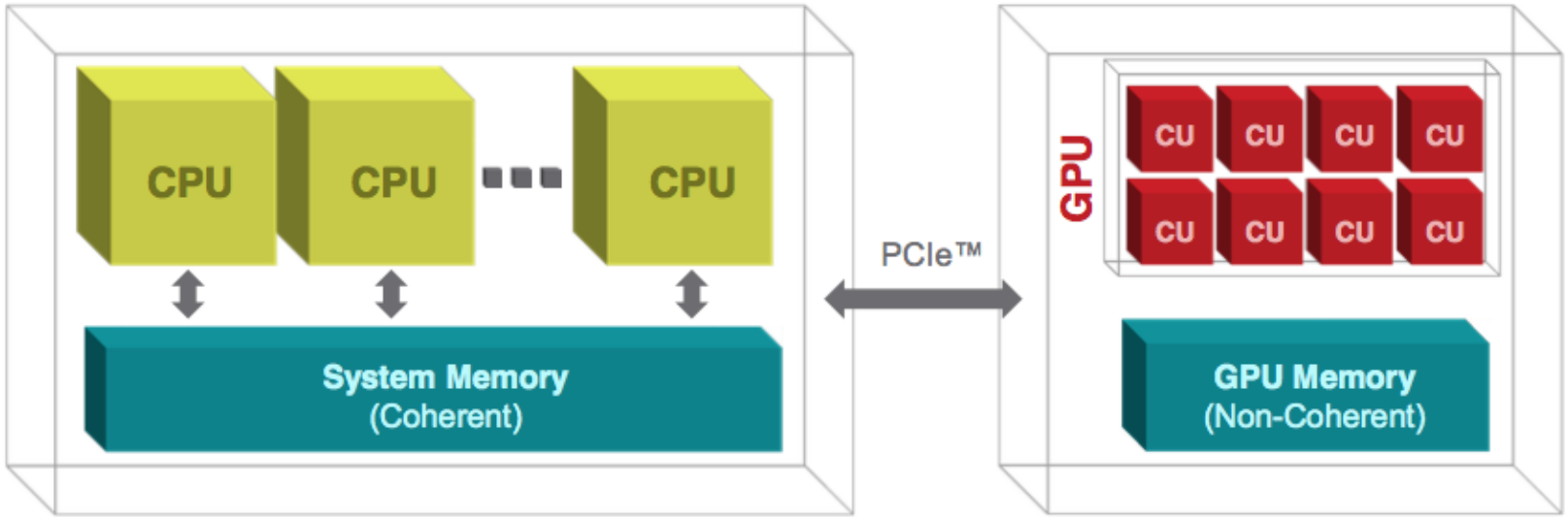
GPU

TPU

XPU

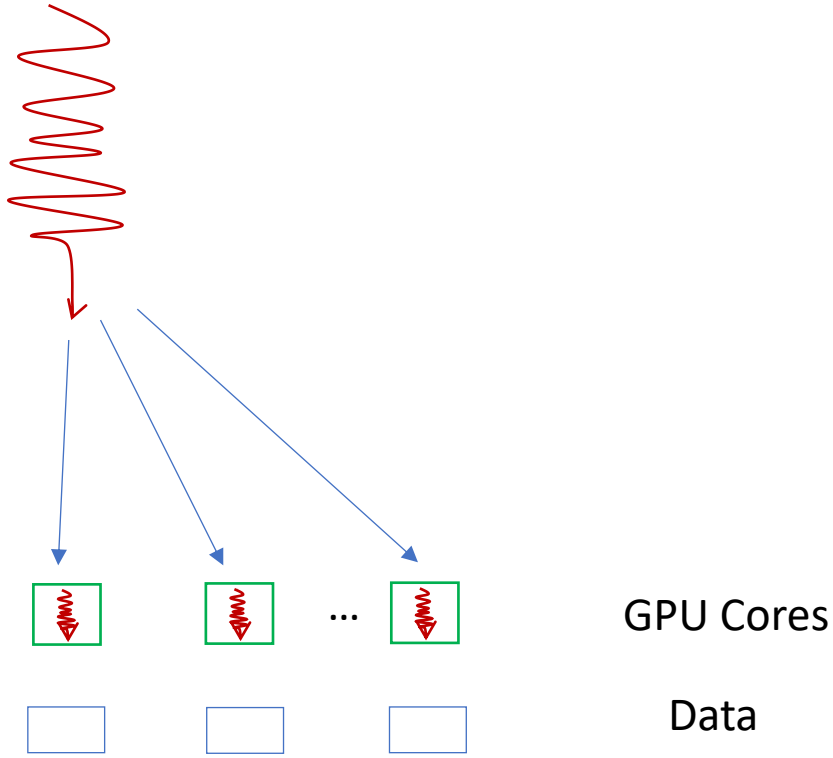
For more info on XPU:
https://www.intel.com/content/www/us/en/architecture-and-technology/xpu.html

CPU + GPU Heterogeneity



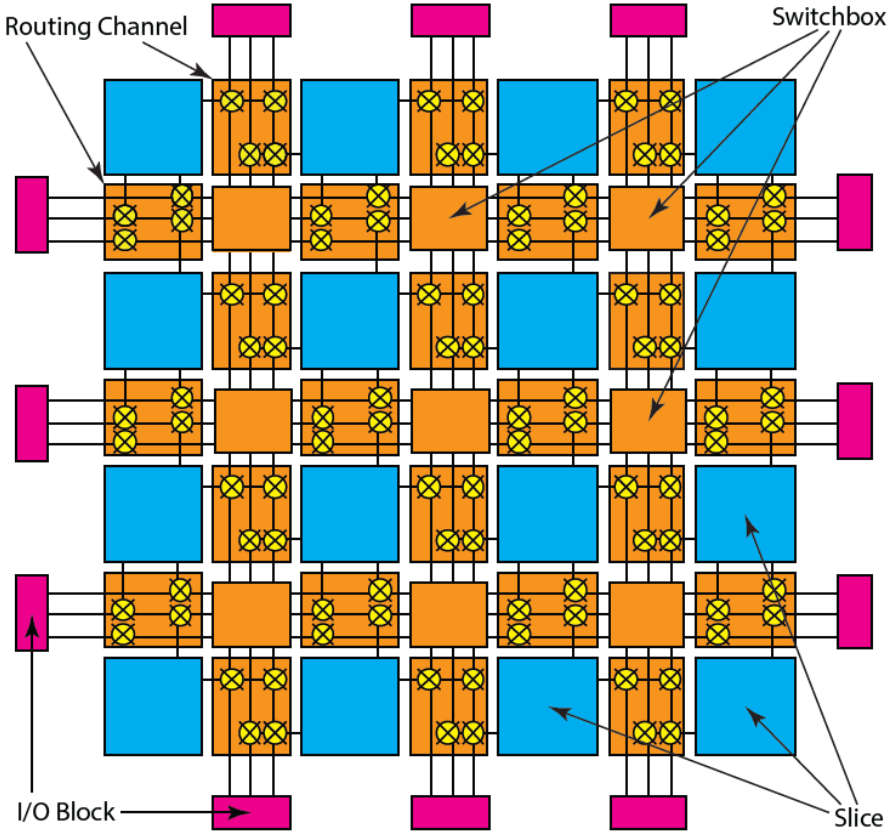
CPU + GPU Heterogeneity

- CPU
  - Runs the main program
- GPU
  - Runs the “kernel” program
  - Specializes in Data Parallel Programming
- Excels at
  - Dense, parallel, structured, computations
  - Image based; Dense Matrix based
- Limited for
  - Sparse irregular workloads
  - Graph based; …



CPU+FPGA Heterogeneity

- CPU
  - Runs the main program
- FPGA
  - Specializes in “User Programmed” Task
- Excels at
  - Dense parallel structured computations
  - Sparse Irregular computations
- Caveats
  - FPGA computation speeds 10x slower than GPUs



Crash Course in FPGAs (1)

- Field Programmable Gate Array
  - Programmable Logic Blocks +
  - On-Chip Memory +
  - Programmable Interconnects

- Logic + Interconnect can be programmed to represent ANY program/circuit/algorithm

Crash Course in FPGAs (2)

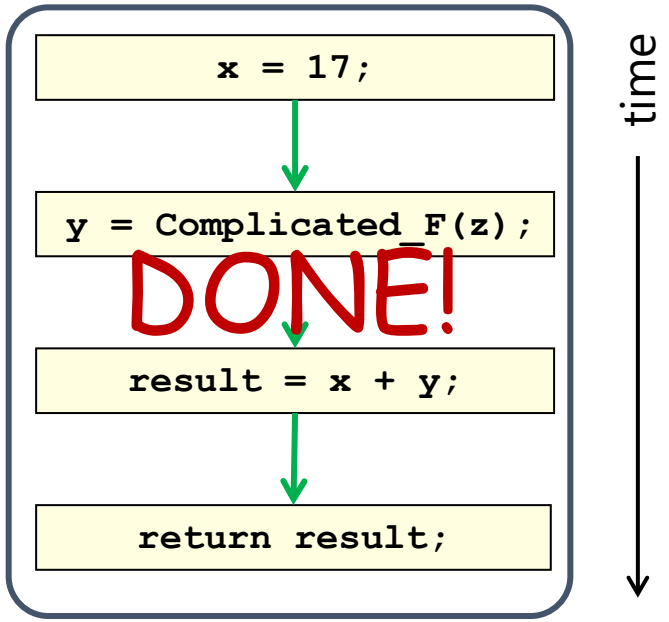
- Why FPGAs?
  - “Programmable” Hardware
  - Processors/Microcontroller
    - Inherently constrained by Instruction Set
    - Limited parallelism
  - FPGAs
    - No software, no instruction set constraints, a digital circuit that implements the exact functionality
    - High fine grained parallelism

We will not discuss in the class but if you are interested to do projects on FPGA as opposed to GPUs, contact me directly

Heterogeneous Computing

Heterogeneity

```
int dostuff(int z) {
      int x, y, result;
      x = 17;
      y = Complicated_F(z);
      result = x + y;
      return result;
}
```



Offload Complicated Function to XPU

Example Frameworks (1)

- Pytorch/Tensorflow: CPU+GPU
  - python based ML framework
- Deep Graph Library: CPU+GPU
  - Python based graph learning framework
- OpenMP: CPU + XPU (currently XPU = GPU)
  - Pragma based programming for heterogeneous computing

Example Frameworks (2)

- OneAPI: CPU+XPU (XPU = GPU or FPGA)
  - C++ based device agnostic programming
- CUDA/HIP: CPU+GPU
    - C/C++ based programming for GPU
- High Level Synthesis (HLS): CPU + FPGA
  - Pragma based programming for FPGA
  - Typically, used in conjunction with OpenCL

Pytorch

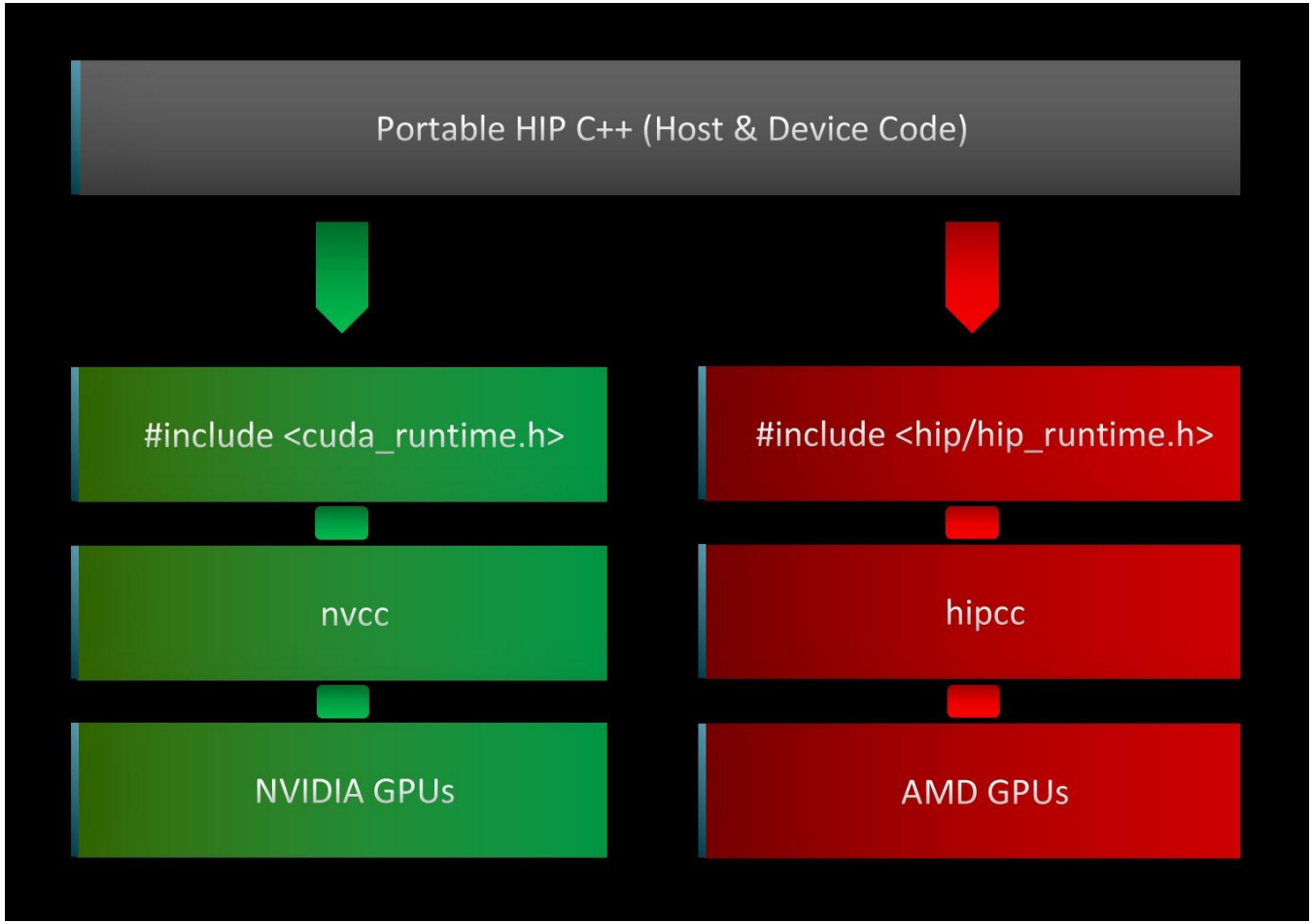
- Open Source ML Framework

- Explore
  - Basics of ML Model development
  - Distributed ML (later in the course)

- Pytorch: https://pytorch.org/

Heterogeneous-compute Interface for Portability (HIP)

- C++ based runtime API and kernel language
- Enables developers to design platform-independent GPU programs that can run on both AMD and NVIDIA GPUs
- Syntax very similar to CUDA



Our Focus in this Course

- Target Platform – CPU+GPU Heterogeneous Platforms

- Programming Frameworks
  - C++ - CPU-only
  - HIP – CPU+GPU
  - Pytorch – CPU+GPU

Outline

- Course Information
- Course Overview
- Introduction to Heterogeneous Computing
- Introduction to Key AI/ML Kernels

Key Computational Kernels in Machine Learning

- Majority of Computations in Machine Learning occur using a small set of kernels (operations that are a bit more complex than addition/multiplication)

- Vector – Vector Operations
- Matrix – Vector, Matrix – Matrix Operations
- Sparse Matrix Operations
- Hashing
- Message Passing on Graphs

Ungraded HW Assignment

- Use your favorite generative AI tool to learn about each of the kernels
- We will introduce them in this class as and when needed

Next Class

- 8/28 Lecture 2 – Processor-Memory Architecture – Modeling, Analysis, Challenges, Opportunities for Optimization

CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

Thank You

- Questions?

- Email: sanmukh.kuppannagari@case.edu