

Lecture 1: Introduction to Heterogeneous Programming

Tsung-Wei (TW) Huang

Department of Electrical and Computer Engineering

University of Utah, Salt Lake City, UT



Logistics

- **Instructor: Dr. Tsung-Wei Huang (TW)**
 - tsung-wei.huang@utah.edu
 - MEB 2124
- **Time: MoWe / 1:25 PM – 2:45 PM (excluding holidays)**
 - In-person room: WEB L110
 - Zoom: <https://utah.zoom.us/j/2468214418>
- **Webpage**
 - GitHub: <https://github.com/tsung-wei-huang/ece6960-heterogeneous-programming>
- **Office hour: by appointment**

Scoring

- **Total 100 points**
 - Four programming assignments of heterogeneous algorithms
 - Class presentation for important heterogeneous algorithms
- **Academic integrity**
 - You should always finish assignments by yourself!
 - I trust you but don't take it for granted (it's your career not mine...)
 - <https://regulations.utah.edu/academics/6-400.php>
 - In the past, I usually give everyone “A” as long as you did something
- **Textbook**
 - No need to buy any textbook
 - Slides will be available and are enough

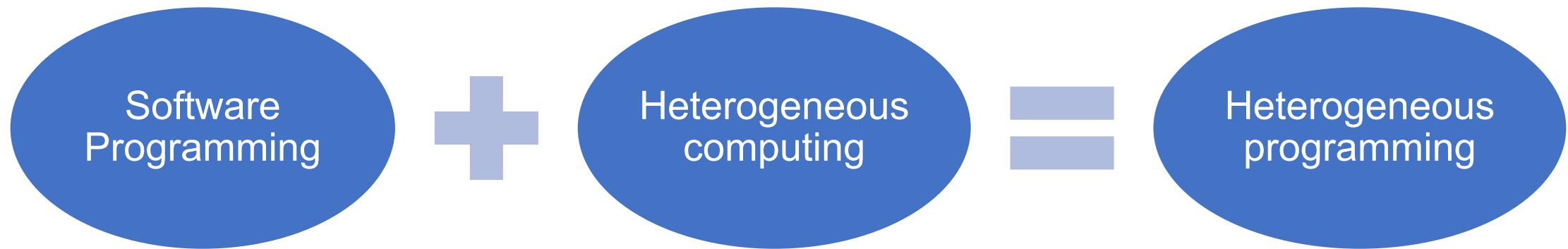
Class Philosophy

- Learn to program massively parallel processors to get
 - High performance
 - Functionality and maintainability
 - Scalability across future generations
 - **With a lot of hands-on programming experience**
- Technical subjects
 - Parallel programming basics
 - Principles and patterns of parallel algorithms
 - Programming API, tools and techniques
 - Processor architecture features and constraints
 - Killer apps

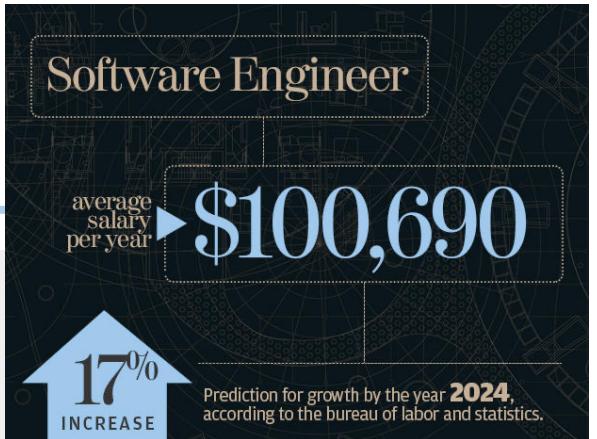
Programming Machines

- **Linux programming environment**
 - Utah CADE Linux machine: <https://www.cade.utah.edu/>
 - Your personal MacBook, Linux desktop, etc.
- **Evaluation environment**
 - Instructor's server: twhuang-server-01.ece.utah.edu
 - 80 CPU threads (hyperthreading)
 - 4 GPUs
 - We will assign everyone an account for programming assignments
- **Programming languages**
 - C and C++
 - Python (a little bit)

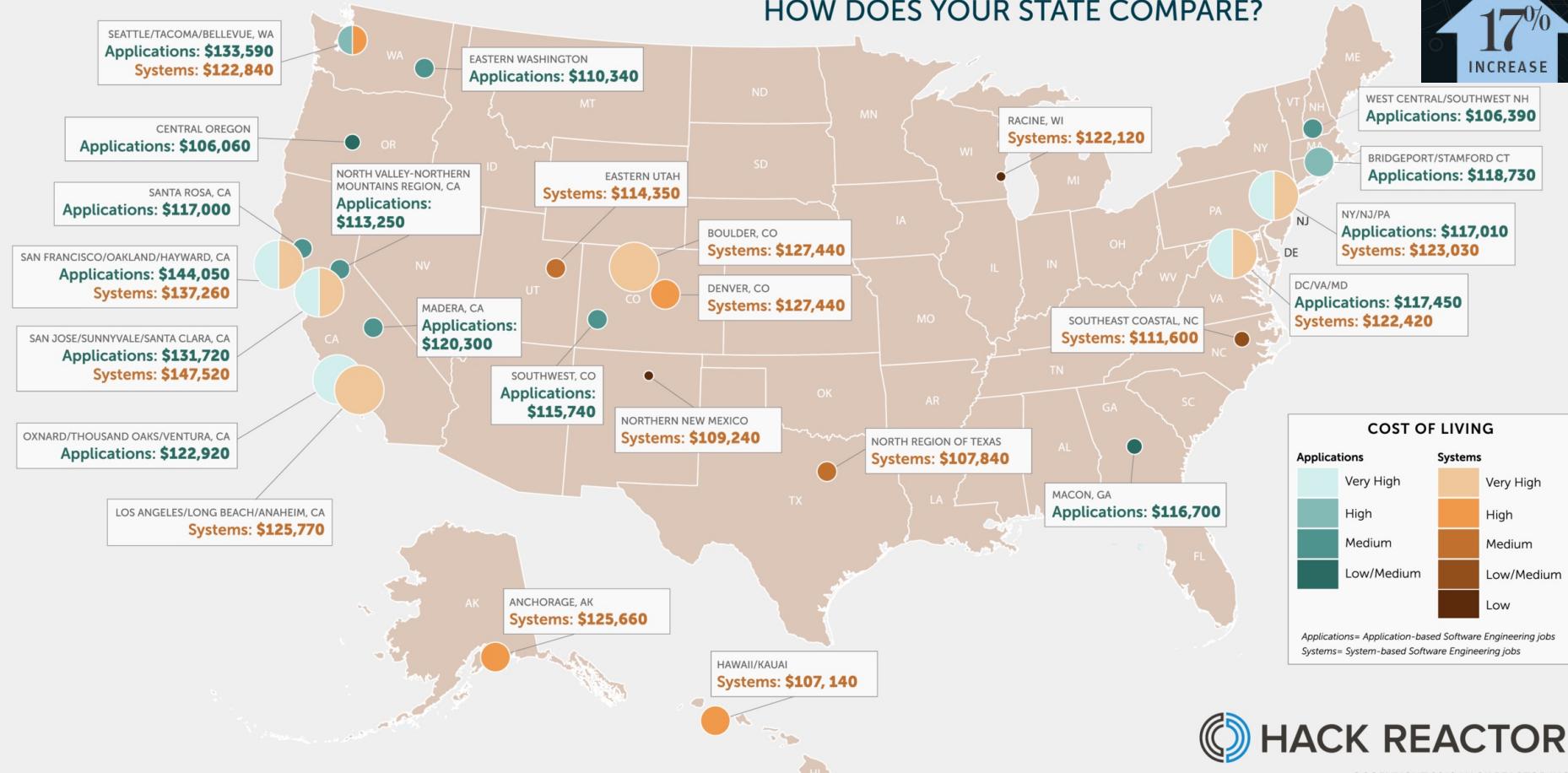
Heterogeneous Programming Concept



Why Programming...?



SOFTWARE ENGINEER SALARY REVIEW 2019: HOW DOES YOUR STATE COMPARE?



WHICH PROGRAMMING LANGUAGE TO LEARN FIRST?

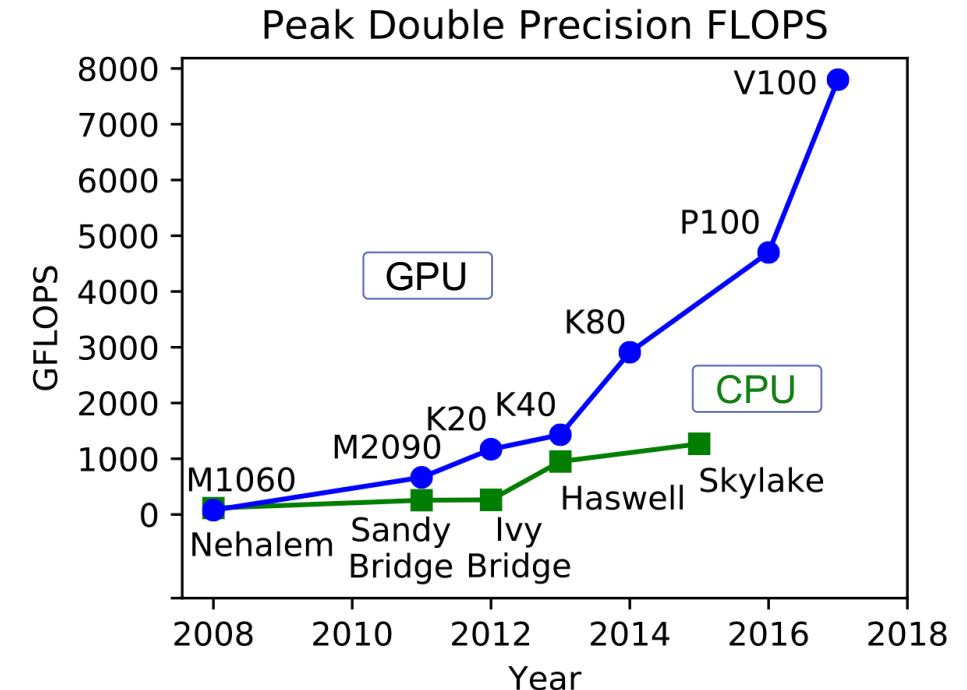
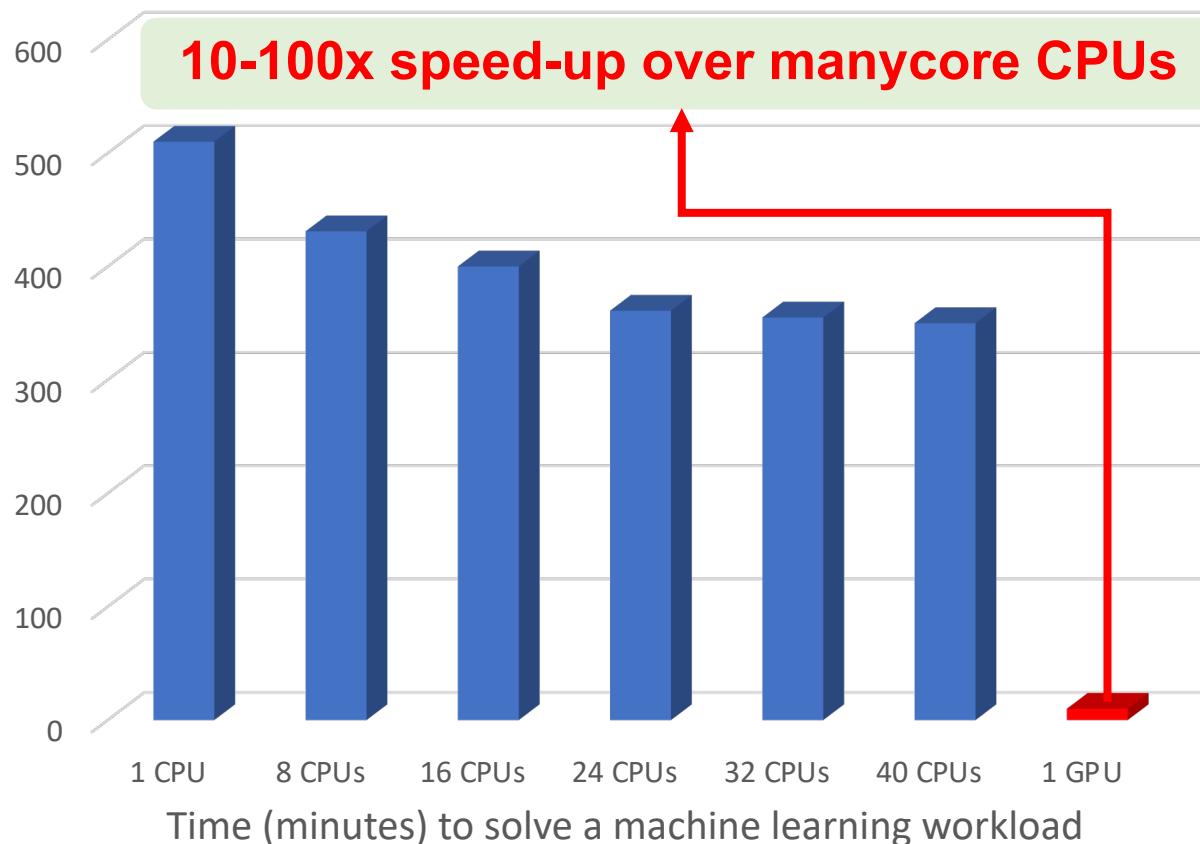


My 10-year Programming Perspective

- **In general, your logic thinking matters the most**
 - How to formulate a problem computationally?
 - How to solve a computational problem efficiently?
 - How to improve the solution quality programmatically?
- **However, as a CE/CS student, I highly recommend C++**
 - C/C++ let you know how performance works
 - Software design patterns
 - Computer hardware and memory hierarchies
 - Python let you know how productivity works
 - Java let you know how portability works

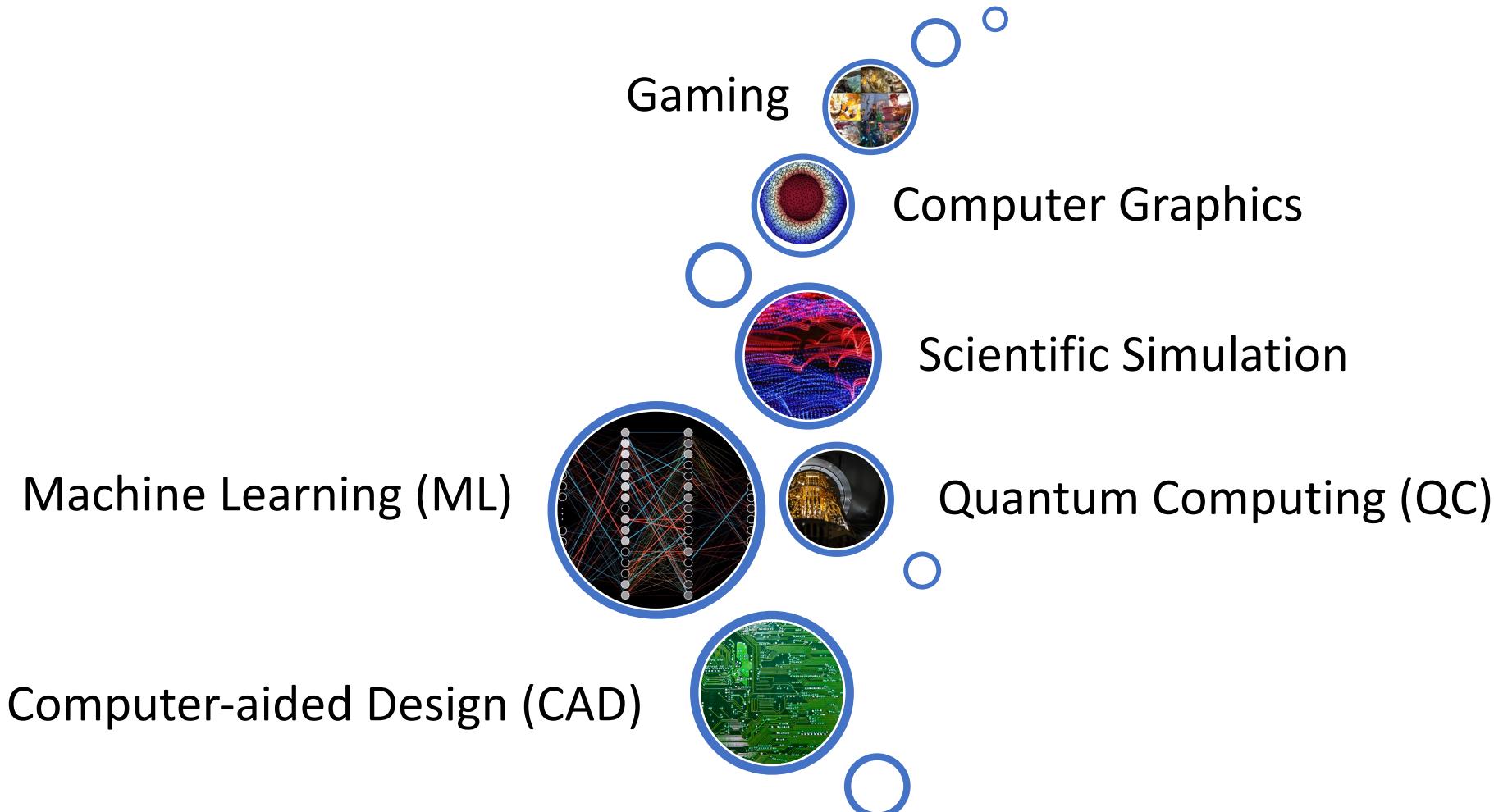
Why Heterogeneous Computing (HC)?

- Advances performance to a new level previously out of reach



Over **60x** speedup in neural network training since 2013

HC Enabled Vast Success in Computing



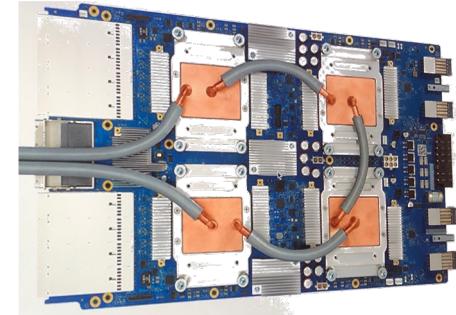
Increasing Heterogeneity in Computers ...



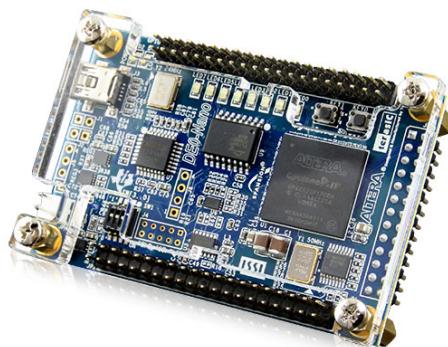
Central Processing Unit (CPU)



Graphics Processing Unit (GPU)



Tensor Processing Unit (TPU)



FPGA



Neuromorphic Devices



Quantum Accelerator

The future of computing is heterogeneous – DARPA ERI, DOE ASCR, NSF PPoSS, SRC Jump 2.0, etc.

Old Paradigm

We were able to understand, design, and manufacture what we can measure

- Physical instruments allowed us to see farther, capture more, communicate better, understand natural processes, control artificial processes...

New Paradigm

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...

Example: Old Paradigm



New Paradigm



Examples of Paradigm Shift

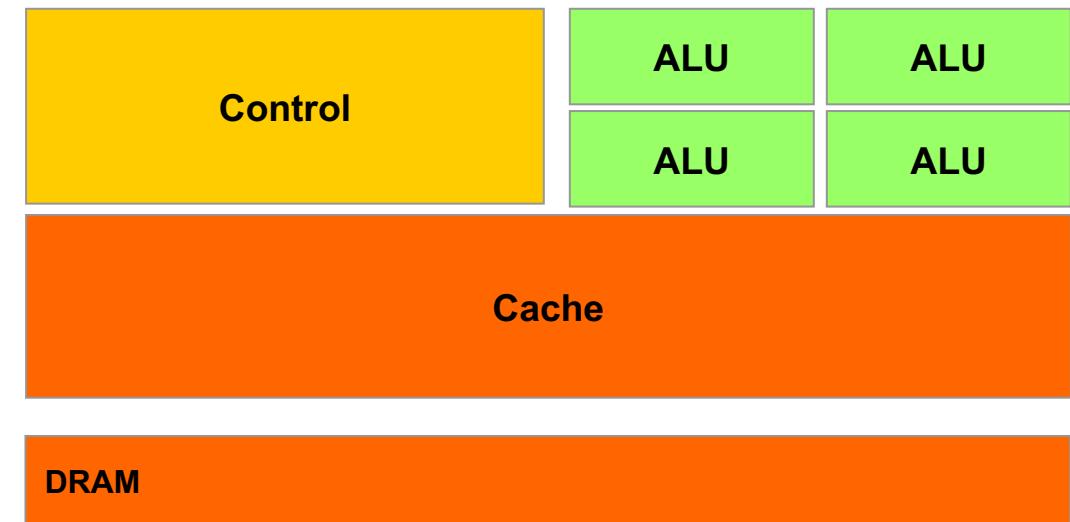
- Semiconductor Lithography → Computational correction
 - Electron Microscopy → Computational Microscopy
 - Film Photography → Deep-learning driven imaging
 - X-Rays → CT & MRI Scans with reconstruction
 - Land-line phones → Zoom video conferencing
 - Cars → Self-driving electric taxis
 - Print and Broadcast Ads → AI-assisted Digital Ad Placement
-
- **So, why is this happening now?**

Today's Computer Architecture

- Multiple cores with more moderate clock frequencies
 - Heavy use of threading, vector execution
 - Systems-on-a-chip with latency-oriented cores, throughput-oriented cores, and specialized cores
 - 3D packaging for more memory bandwidth
-
- **It's all because of performance need!**

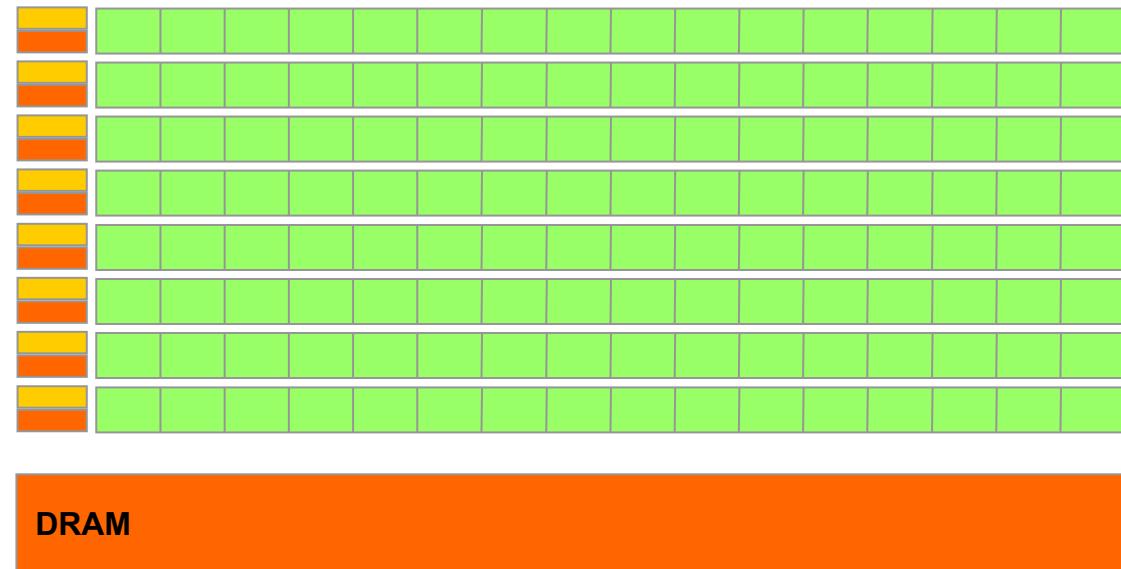
CPU: Latency Oriented Design

- 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



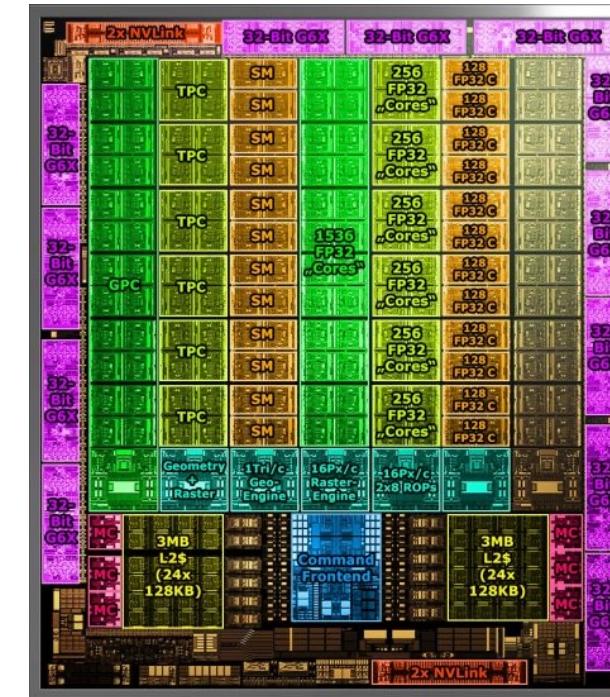
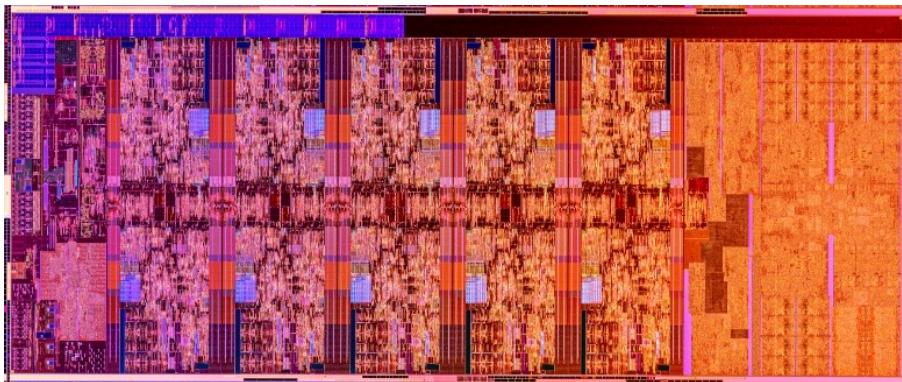
GPU: Throughput Oriented Design

- 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



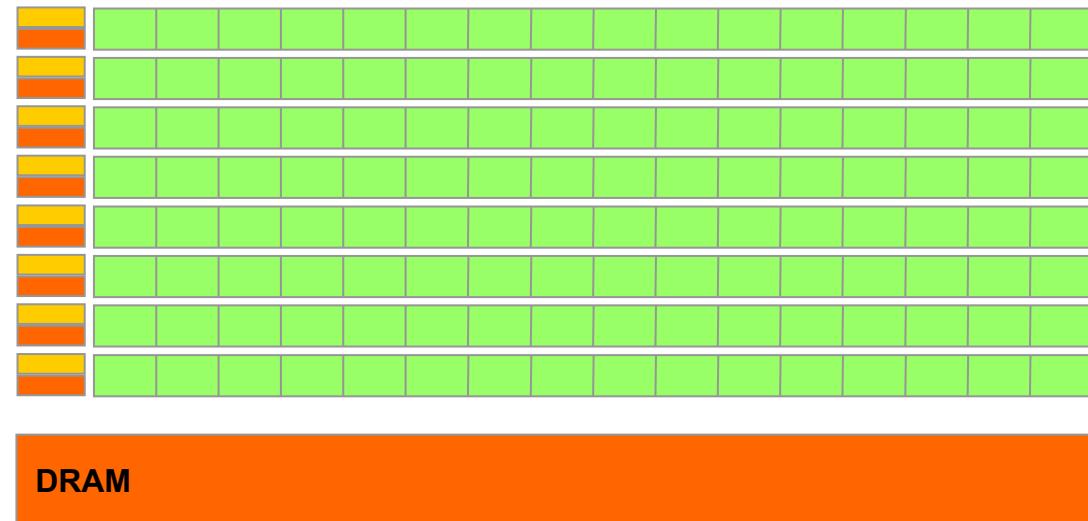
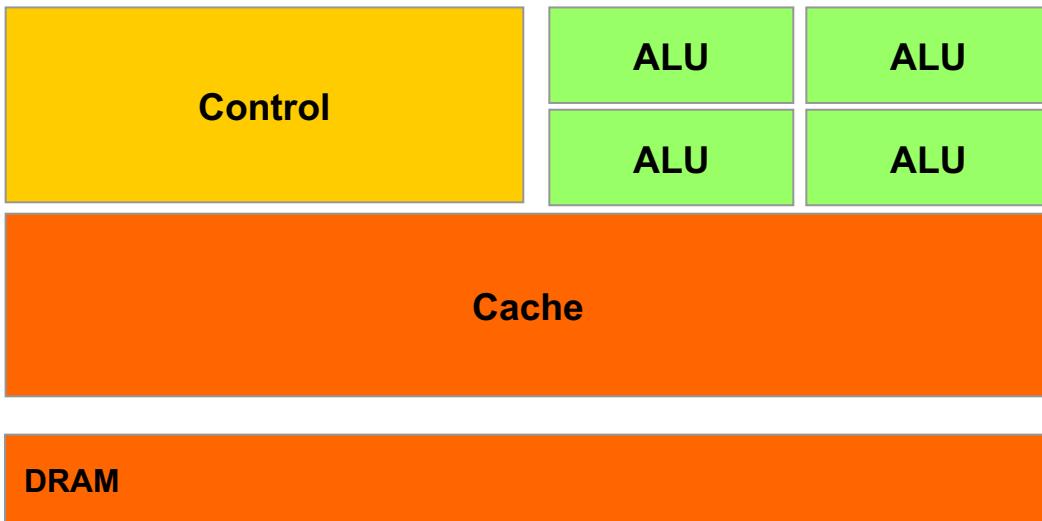
CPU vs GPU

- 10th Gen Intel Core
 - 10 cores in silicon
 - 14 nm process
- NVIDIA A100
 - 3456 CUDA cores
 - 7 nm process



Today's Approaches

- 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 Computing Applications

Financial
Analysis

Scientific
Simulation

Engineering
Simulation

Data
Analytics

Digital Audio
Processing

Digital Video
Processing

Computer
Vision

Machine
Learning

Biomedical
Informatics

Statistical
Modeling

Ray Tracing
Rendering

Interactive
Physics

Medical
Imaging

Design
Automation

Numerical
Methods

...

Heterogeneous Programming Workflow

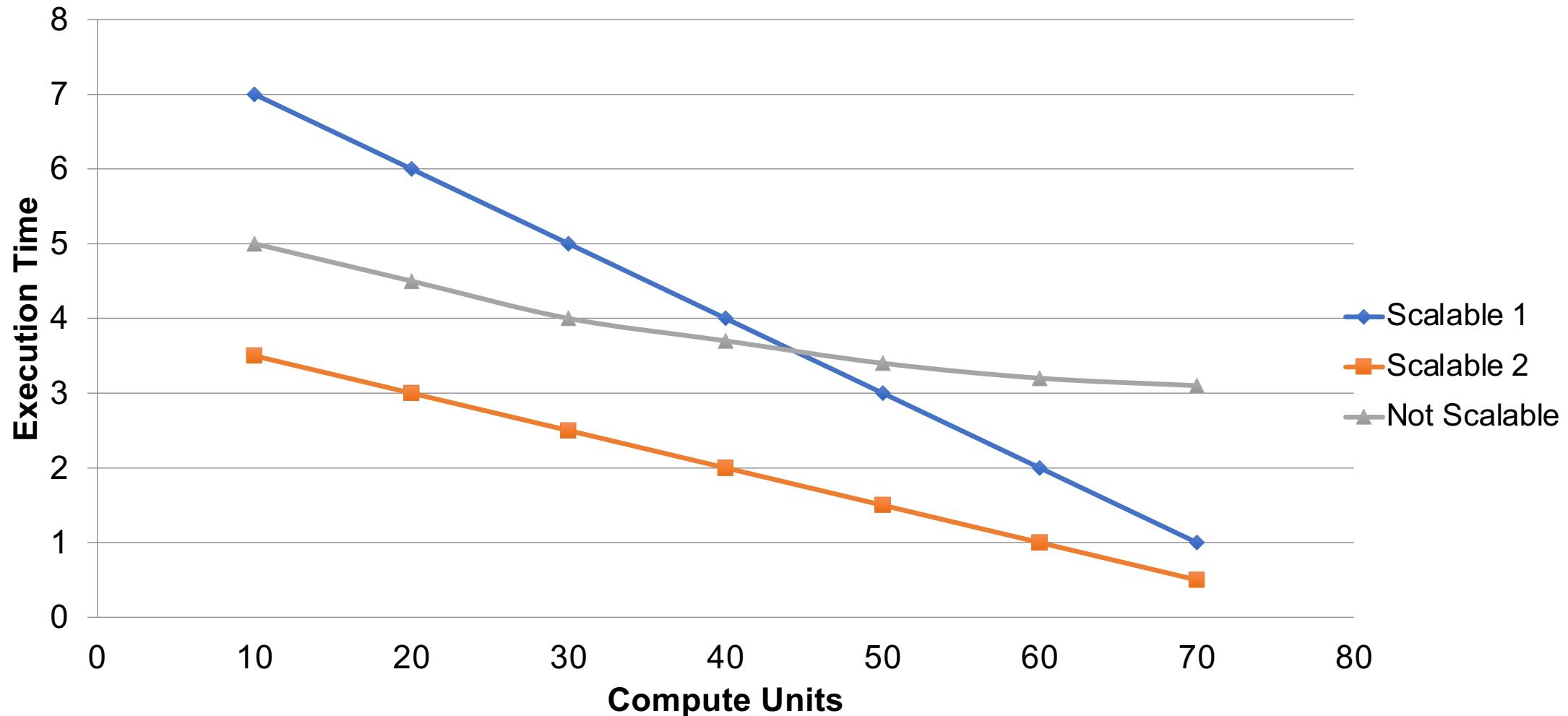
- Identify compute intensive parts of an application
- Adopt/create scalable algorithms
- Optimize data arrangements to maximize locality
- Performance tuning
- Pay attention to code **portability, scalability, and maintainability**

Simple Parallelism

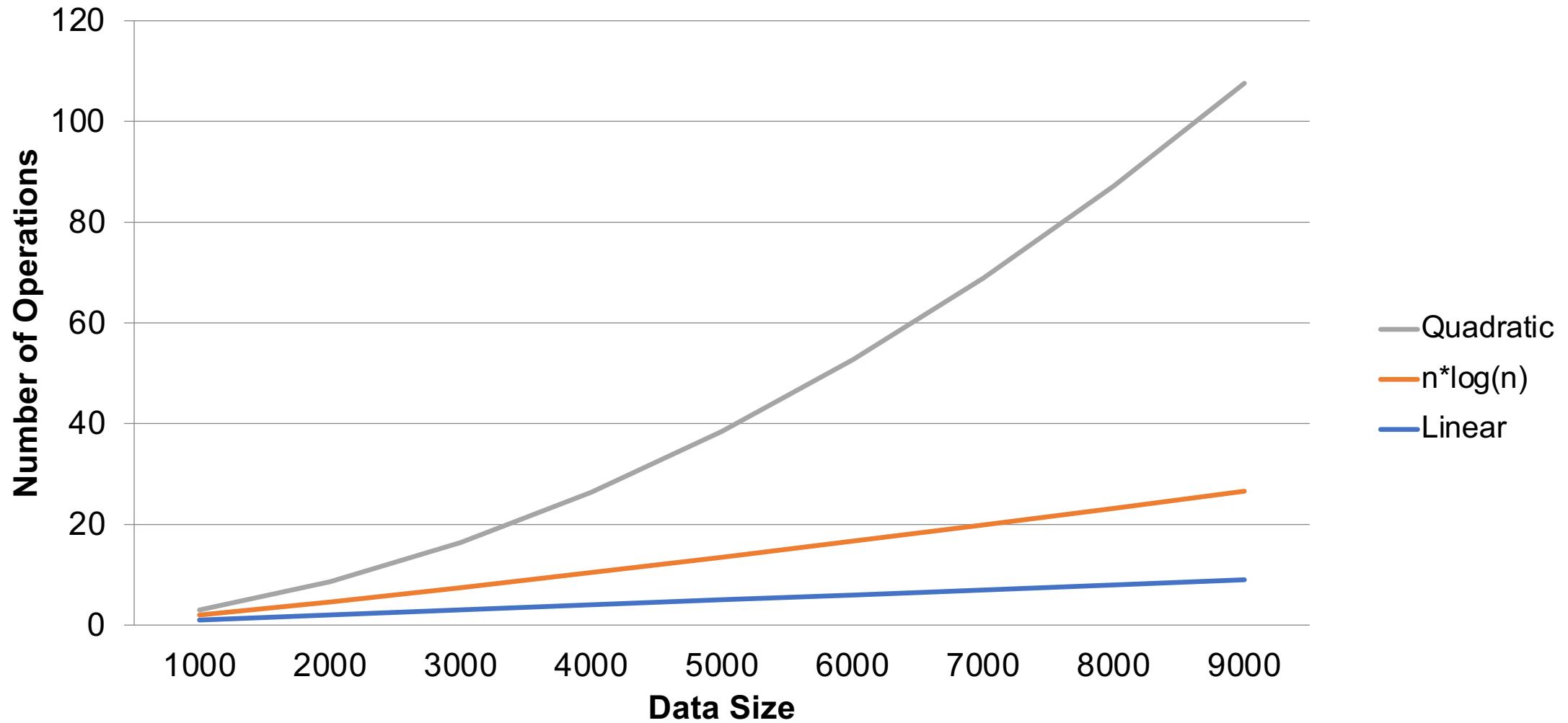
```
for (i = 0, i < n, i++) {  
    C[i] = A[i] + B[i];  
}
```

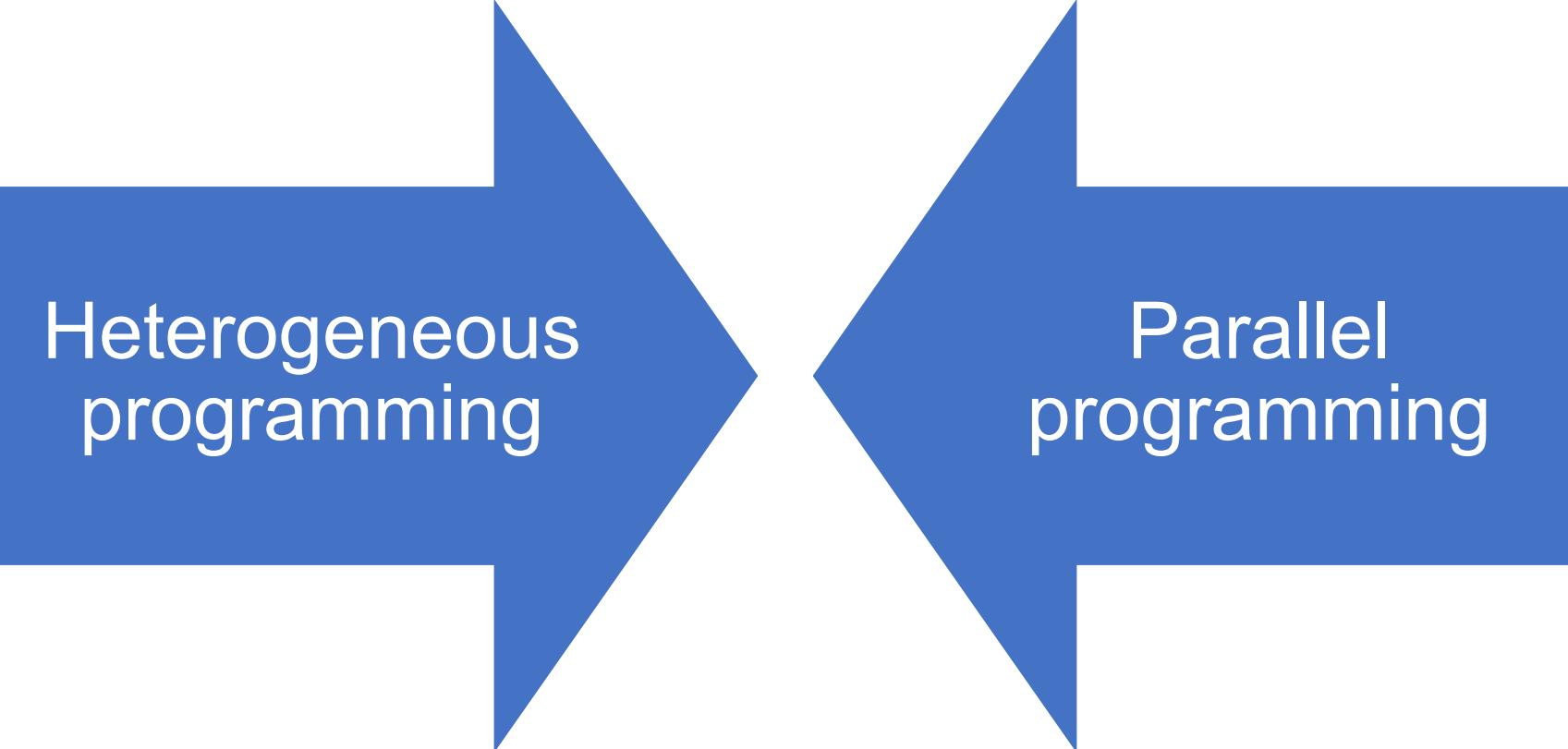
- Loop iterations are independent of each other (caveat: what about *i*?)
- We want to express this loop in a form where the parallelism can be converted into independent operations.
- This semester we will learn how to parallelize it ...

Parallelism Scalability



Algorithm Complexity





Heterogeneous
programming

Parallel
programming

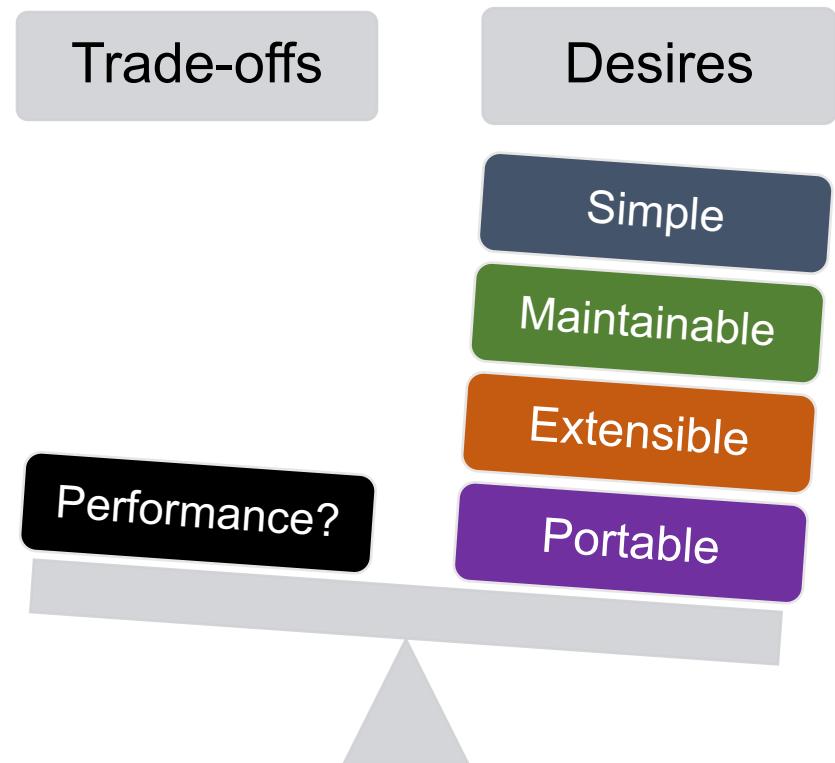
Parallel Programming is a “Big” Challenge

- You need to deal with A LOT OF technical details

- Parallelism abstraction (software + hardware)
- Concurrency control
- Task and data race avoidance
- Dependency constraints
- Scheduling efficiencies (load balancing)
- Performance portability
- ...

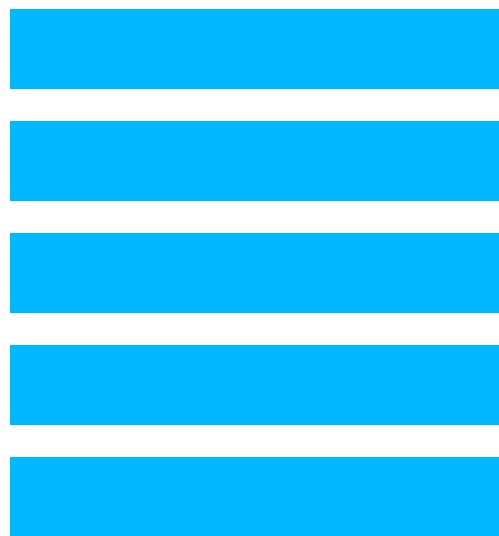
- And, don't forget about trade-offs

- Desires vs Performance

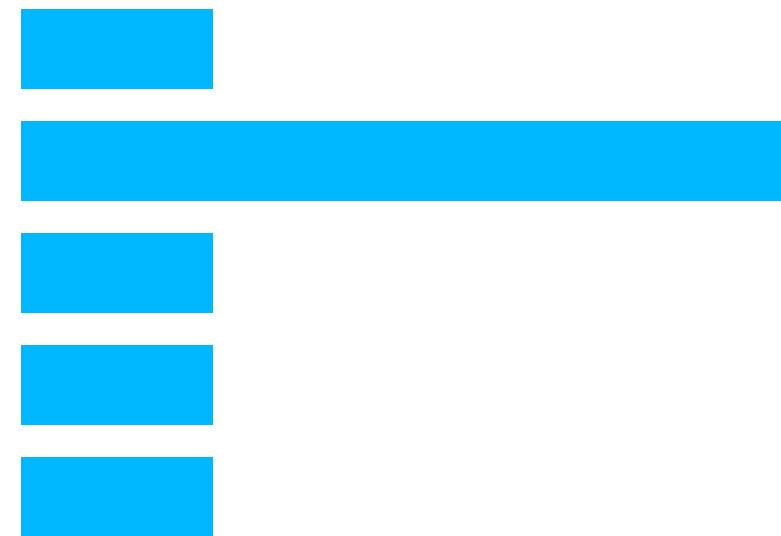


Load Balance

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



good



bad!

Global Memory Bandwidth

- Ideal



- Reality



Data Conflicts Cause Serialization

- Massively parallel execution cannot afford serialization



- Contentions in accessing critical data causes serialization

Memory Optimization

```
std::string* ptr = nullptr;
int data = 0;

void producer()
{
    std::string* p = new std::string("Hello");
    data = 42;
    ptr = p;
}

void consumer()
{
    std::string* p2;
    while (!(p2 = ptr));
    assert(*p2 == "Hello");
    assert(data == 42);
}

int main()
{
    producer();
    consumer();
}
```

```
std::string* ptr = nullptr;
int data = 0;

void producer()
{
    std::string* p = new std::string("Hello");
    data = 42;
    ptr = p;
}

void consumer()
{
    std::string* p2;
    while (!(p2 = ptr));
    assert(*p2 == "Hello");
    assert(data == 42);
}

int main()
{
    std::thread t1(producer);
    std::thread t2(consumer);
    t1.join();
    t2.join();
}
```

Summary

- We have discussed class logistics
- We have discussed heterogeneous programming
- We have discussed application shift
- We have discussed modern computer architectures
- We have discussed scalability
- We have discussed three key challenges
 - Memory bandwidth
 - Data conflicts
 - Memory optimization