



ECE 455

GPU Algorithm and System Design

Dr. Tsung-Wei (TW) Huang

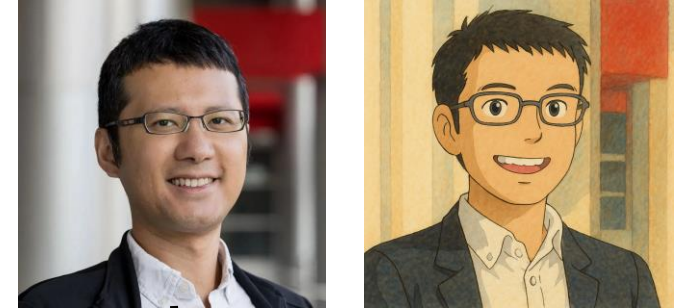
Department of Electrical and Computer Engineering

University of Wisconsin at Madison, Madison, WI



Instructor and TAs

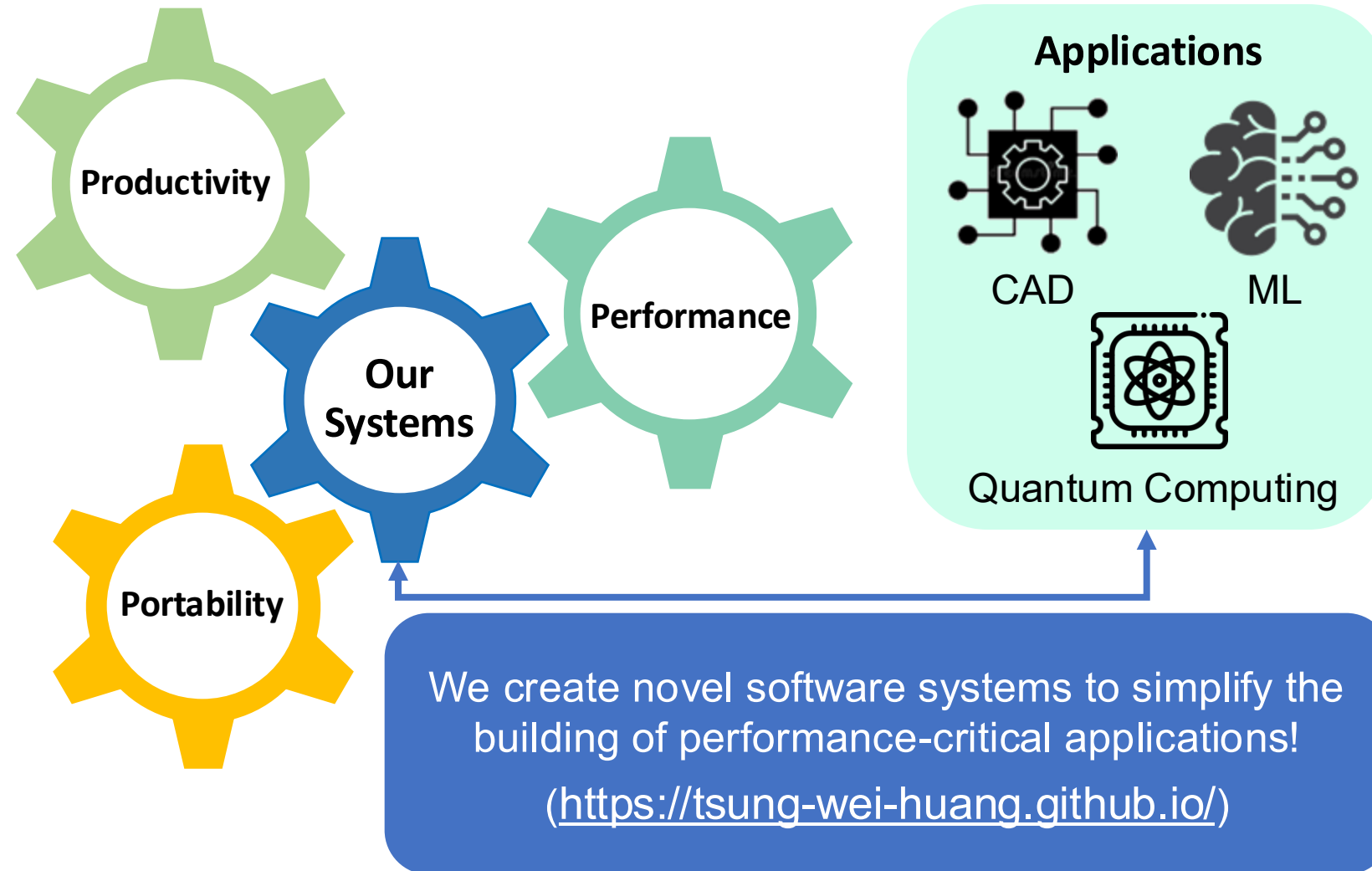
- **Dr. Tsung-Wei (TW) Huang, Dept. of ECE**
 - PhD, University of Illinois at Urbana-Champaign (UIUC)
 - BS/MS, Taiwan's National Cheng-Kung University
 - tsung-wei.huang@wisc.edu
 - Office hour: Available after class
 - Industrial experience: Siemens, IBM, and Citadel
- **TA: Randy Chang, PhD student, Dept. of ECE**
 - GPU-accelerated graph algorithms, EDA
 - cchang289@wisc.edu
 - Office hour: Thursday 12-2 PM at EH 1422
 - Industrial experience: Intel and Google



AI version of us

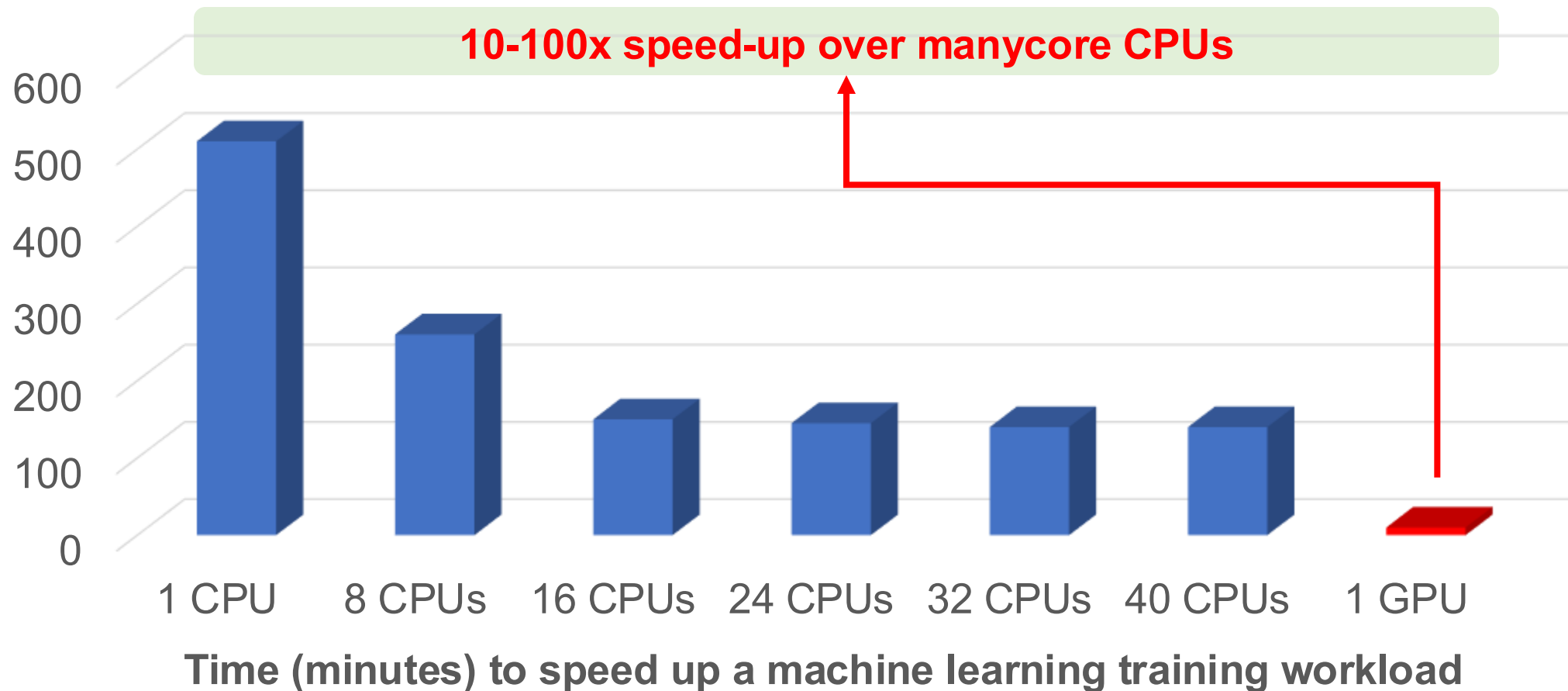


My Research: HPC + X

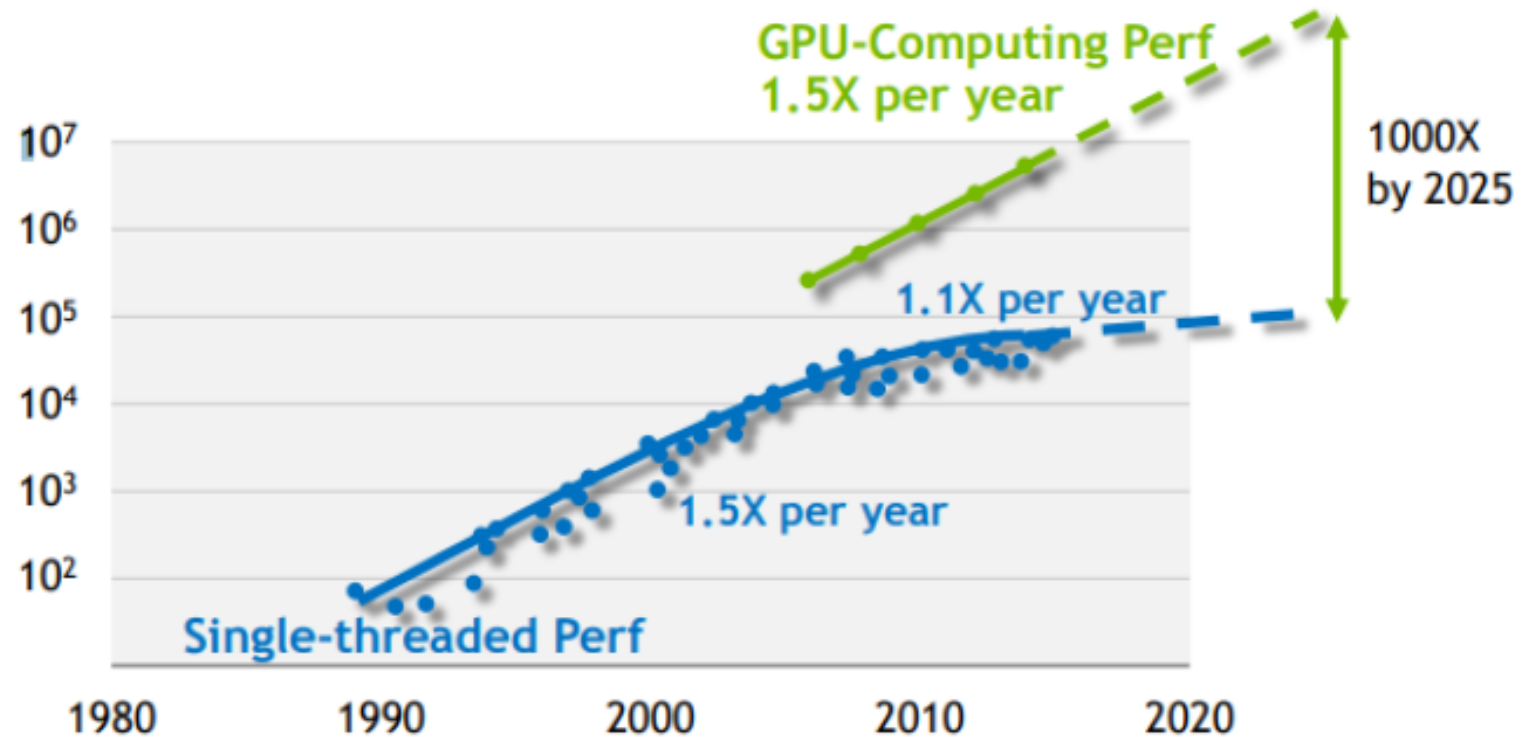
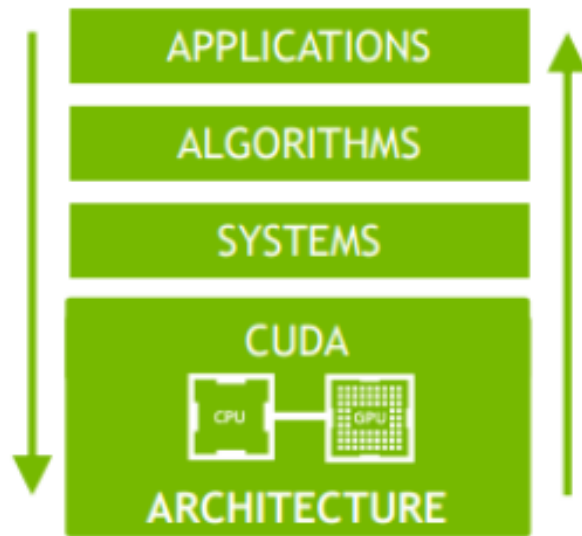


Why CPU- and GPU-parallel Computing?

- Advances performance to a new level previously out of reach

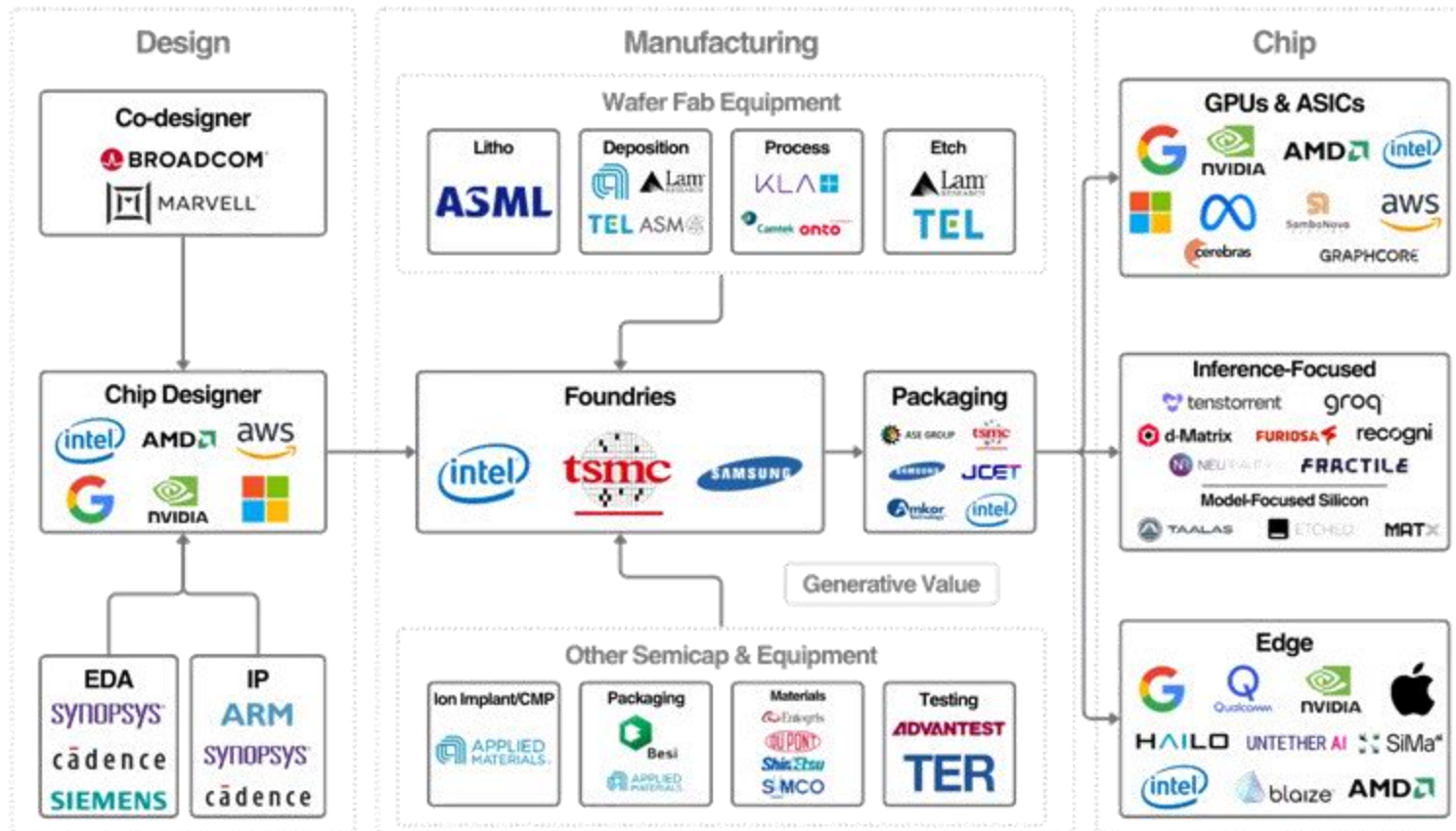


GPU Gives Rise to Accelerated Computing



GPU is of High Demand in Industry

AI Semiconductor Value Chain



*Not an exhaustive list of companies/segments. Companies may fall into multiple segments. Several inference-focused chips can be used for training.

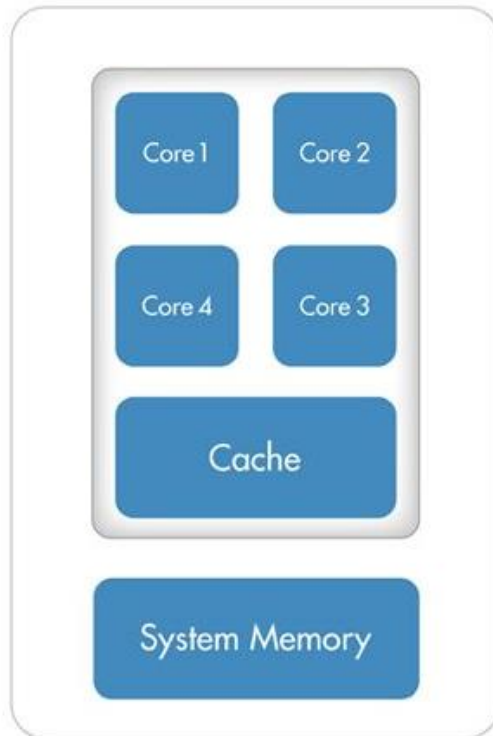
Examples of Salary Ranges:

- Entry-level: \$80,000 - \$120,000
- Mid-level: \$120,000 - \$180,000
- Senior-level: \$180,000 - \$250,000+
- M7 Companies: \$190,000+ starting salary for a Junior SWE, according to a Reddit thread
- Senior Software Engineer (Finance, Defense, etc.): Six figures or more
- Principal Engineer: Can make a million dollars a year or more.
- Nvidia: Entry-level (IC1) \$177K, IC2 \$210K, IC3 (Senior) \$289K

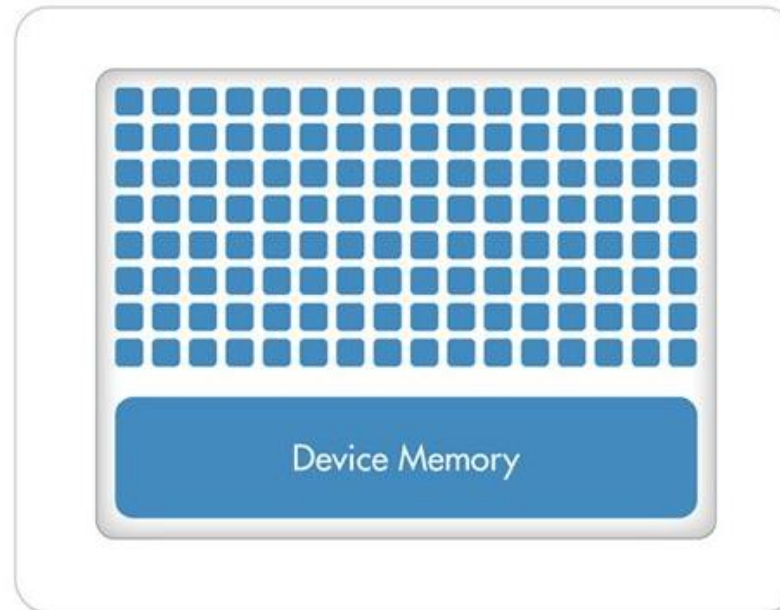
Goal of this Course

- **Know how to speed up your applications using parallelism**
 - We will focus on contemporary hardware (CPU + GPU)

CPU (Multiple Cores)



GPU (Hundreds of Cores)



Central Processing Unit (CPU)



Graphics Processing Unit (GPU)



When, Where, Who, How, ...

- **Lecture and lab time:** 3:30PM-5:25PM Monday/Friday (excluding holidays)
 - Capstone students will meet together occasionally on Fridays 12:05-12:55PM at ME 1106
 - Schedule depends on the availability of our invited industrial speakers. Keep watching Canvas
- **Lecture and lab place:** ENGR HALL 2534
 - May use zoom if I am traveling or the weather is bad
 - Zoom link and passcode are available on the home page of class Canvas
- **Office:** EH 3423
- **E-mail:** tsung-wei.huang@wisc.edu
 - Please accept my apology for delayed responses since I am very busy
- **How people call me?** TW (short for Tsung-Wei)
 - PS: I am originally from Taiwan (TW)
- **Course managed under** [Canvas](#)
 - You should keep an eye on announcement, update, scores, assignments
 - You should all be added automatically to the Canvas page



Your Canvas Home Page Has All Information

Week	Topic	Due	Note
9/5 (Friday)	Introduction		
9/8 (Monday)	Why Parallel Computing?		
9/12 (Friday)	Slurm and Euler Tutorial ↓ Lab 1: Introduction to Slurm		
9/15 (Monday)	C++ Threads		
9/19 (Friday)	Lab 2: Parallel Programming using C++ Threads	Lab 1	Capstone Design Students Meeting together at 12:05-12:55 PM in ME 1106
9/22	OpenMP Basics		
9/26	Lab 3: Parallel Programming using OpenMP	Lab 2	Capstone Design Students Meeting together at 12:05-12:55 PM in ME 1106
9/29	Introduction to GPU and CUDA		

<https://canvas.wisc.edu/courses/479145/>



Class Organization

- **Monday class will be lecture-based**
 - Learn essential knowledge about parallel computing, both CPU and GPU
 - Discuss theoretical foundations, programming models, and performance considerations
- **Friday class will be lab-based**
 - Gain hands-on programming experience applying Monday's concepts
 - TA and I will go through hands-on programming exercise
 - You will need to submit your lab assignment by next lab time via Canvas assignment page
 - **You will need to bring your own laptop to the class to work on the lab assignment**
 - Submit code to the Euler campus cluster (with Nvidia GPU)
 - Evaluate the result of your code in a consistent environment
 - Labs will be completely software-driven, and so your laptop will be just fine
 - You will see how many amazing things can be achieved with your laptop 😊

Course-related Information

- **You can find all slides on Canvas**
 - Slides are good enough for this course
 - Typically, the slides will be made available after the lecture since we may update some information based on feedback (e.g., typos, clarifying points)
- **Textbook is not allowed but you can find some good books:**
 - R. Bryant and D. O'Hallaron, *Computer Systems: A Programmer's Perspective*, Prentice Hall, 3rd Edition, 2015. See [here](#).
 - CUDA C++ Programming Guide, version 12.2, see [here](#).
 - Wen-mei W. Hwu, David B. Kirk, Izzat El Hajj: *Programming Massively Parallel Processors: A Hands-on Approach*, Morgan Kaufmann, 2016 ([4th edition](#)).
 - Jason Sanders & Edward Kandrot: *CUDA by Example: An Introduction to General-Purpose GPU Programming*, Addison-Wesley Prof., 2010. See [here](#).
 - Peter Pacheco, Matthew Malensek: *An Introduction to Parallel Programming*, Morgan Kaufmann, 2nd ed., 2023. See [here](#).
 - T. Mattson, et al.: *Patterns for Parallel Programming*, Addison Wesley, 2004. See [here](#).



Scoring

- **7-8 lab assignments (until about 10/31): 40%**
 - One for learning the programming environment on the campus Euler cluster with SLURM
 - Two for learning CPU-parallel programming using C++ and OpenMP
 - Four or five for learning GPU-parallel programming using Nvidia CUDA
- **Weekly progress report (10/31 – capstone open house): 20%**
 - We will not have lectures and labs – this is the time for your project execution
 - We will still run classes where you “present” your progress to me, TA, and other students
- **Final Project (capstone open house demo): 40%**
 - You can choose to work on a problem in our provided list or any open topics that are related to your interest, research or others that can benefit from GPU
 - Encourage you to apply GPU to solve real-world problems
 - You must work as a team of at least 3 students
 - Note: Team size may be adjusted depending on the final number of registrations
 - Open house date will be announced later (typically the Tuesday of the last class week)
 - Around 12/8 (may change as we need to synchronize with other capstone students)



Grading

<u>Score</u>	<u>Grade</u>
92-100	A
86-91	AB
78-85	B
70-77	BC
60-69	C
50-59	D

- Grading will not be done on a curve
- Final score will be rounded to the nearest integer prior to having a letter assigned
 - Example:
 - 85.59 becomes AB
 - 85.27 becomes B
 - 91.5 becomes A
 - 91.4 becomes AB

LAB Assignment

- **Goal: prepare you with the necessary knowledge to program parallelism**
 - Both CPU- and GPU-parallel programming
- **A total of 7-8 lab assignment (until 10/31) due weekly**
 - For example, Friday's lab will be due next Friday
- **There will be **no exceptions** for late work**
 - Unless you have a proper reason (e.g., sick with medical evidence/proof)
 - You will get reminders from Canvas when the due is approaching
 - You will submit your assignments through the Canvas assignment page
- **Why are we strict about the due day?**
 - LAB assignment is VERY EASY – it's more like exercise rather than assignment
 - We will provide 3–5 programming exercise questions with solutions
 - You need to examine and practice those questions with your own laptop
- **Lab assignments will be built on top of each other layer by layer**
 - Finish one assignment is critically important for the next one and so
 - It becomes very difficult for you to catch up if you miss one layer



ChatGPT, Internet Source of Inspiration

- **You are allowed to use ChatGPT/Gemini on your assignments**
 - <https://chat.openai.com/>
 - <https://gemini.google.com/>
- **You are allowed to use any online sources of inspirations**
 - StackOverflow, GitHub repository
- **We do not hold it against you **at all** if you use these resources**
 - For me, being able to find the solution using available resources online is also very important for today's technology world
 - You are already graduate students (or adults), knowing how to be responsible for your own career
 - My personal opinion: if an instructor tells you not to use AI, then I will seriously consider whether that course should be on my schedule ...



Rule of Academic Integrity

- **You are encouraged to discuss assignments with others**
 - Collaboration is very important in today's software industry
 - 30% of your career success comes from technical capability
 - **70% of your career success comes from communication**
 - Are you able to communicate your ideas clearly and collaborate with other people?
 - This is why we will ask you to do a weekly progress presentation after 10/31
- **Copy/paste of trivial (boilerplate) code in your project**
 - OK to do from the internet
 - However, it is considered cheating
 - If you don't acknowledge source
 - If you get the code from somebody who is taking or took this course

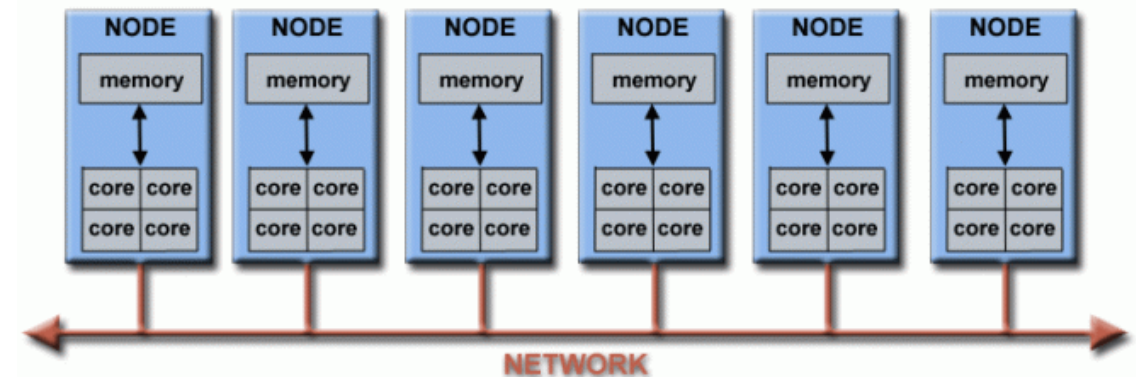
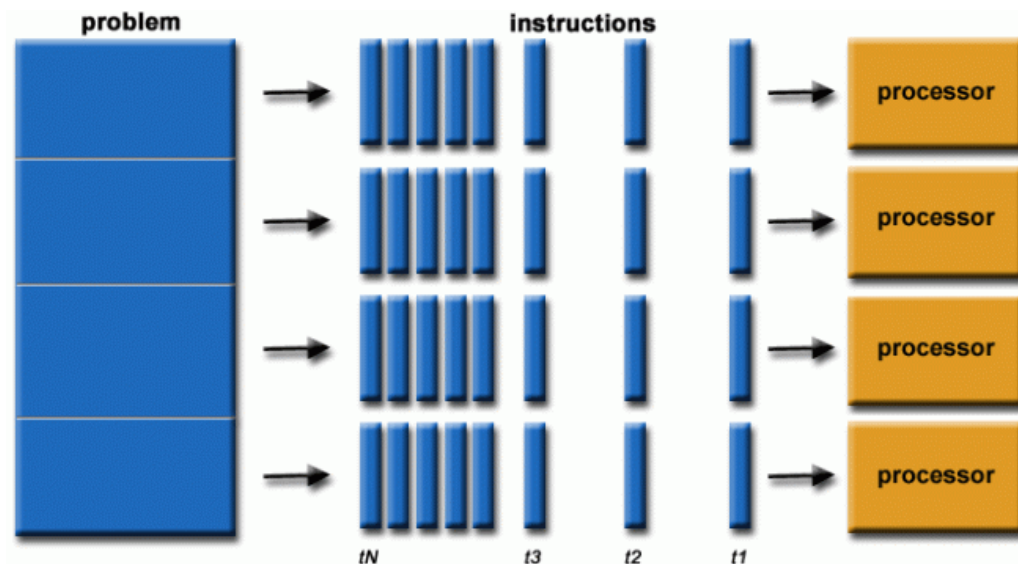


Consequences of Cheating

- **First time**
 - Everybody involved gets zero for that cheated assignment
 - Keep in mind each PA occupies 10% of your final grade
- **Second time**
 - The Dean of your college will be brought into the loop
- **Statistically, 2–4 cases of cheating each semester are identified**
- **Ultimately, it's your career not mine!**
 - Why are you paying so much tuition money and spending your time here for cheating?
 - Your honesty is gazillion times more important than the scores
 - Please, please, please... start forming teams and brainstorming your project idea earlier (you should at least have your team formed by the end of Sep)

Hitting the Ground and Running

- We will start with “why parallel computing is necessary” next week
 - Start with a warm-up lab to help with you learn the Euler cluster
 - How to submit a job to Euler cluster using the industry-grade SLURM?
 - How to request CPU/GPU resources from a shared cluster using SLURM?
 - Depending on when you get enrolled into this class, you may be granted access to Euler cluster late – the system automatically sync up every night



Final Project

- **Goal: Apply CPU-GPU parallelism to solve real-world problems**
- **You MUST form a team of at least 3 students**
 - Collaboration + communication take **70%** of your career success!!!
- **Project execution will roughly begin on early Oct**
 - When you have knowledge about parallel programming
- **Two stages:**
 - Stage #1: proposal due on 10/17 – 20% of your final project score
 - We will provide you with some ideas
 - You're more encouraged to choose a problem that fits your research interest
 - Try to tackle a meaningful problem
 - People buy “why” you do not “what” you do
 - Stage #2: present your work at capstone design open house
 - Poster + power point presentation + laptop demo

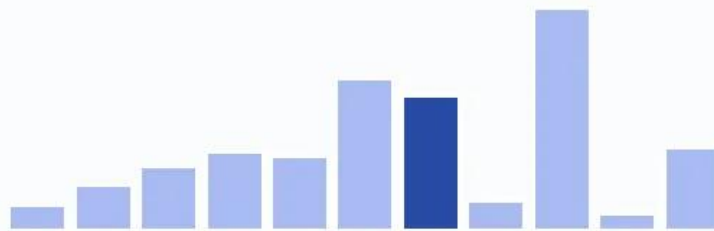


Pre-requisites: C and/or C++ Programming

- This is a project-driven and collaboration-driven class
- **Familiarity with C or C++ is expected**
 - You will probably be fine if you know Java or another C-like language
 - We do not require you to be an expert C or C++ program
 - But basic language understanding (syntax, idiom) is required
 - Being able to wrestle on your own with a compiler error
 - Being open to reading something on stackoverflow and trying it
 - You have heard of a debugger and are ready to give it a shot
 - You have heard of a profiler and don't mind trying it
 - We (TA + me) will NOT debug the code for you – fixing bugs is the most effective way to master a programming language
- **What if I do not have any C or C++ experience?**
 - Plenty of resources available online to study C++ (e.g., ChatGPT)
 - Take a look at the [slide deck of ME 459](#)

Why Programming...?

Software Engineer Salary in the United States

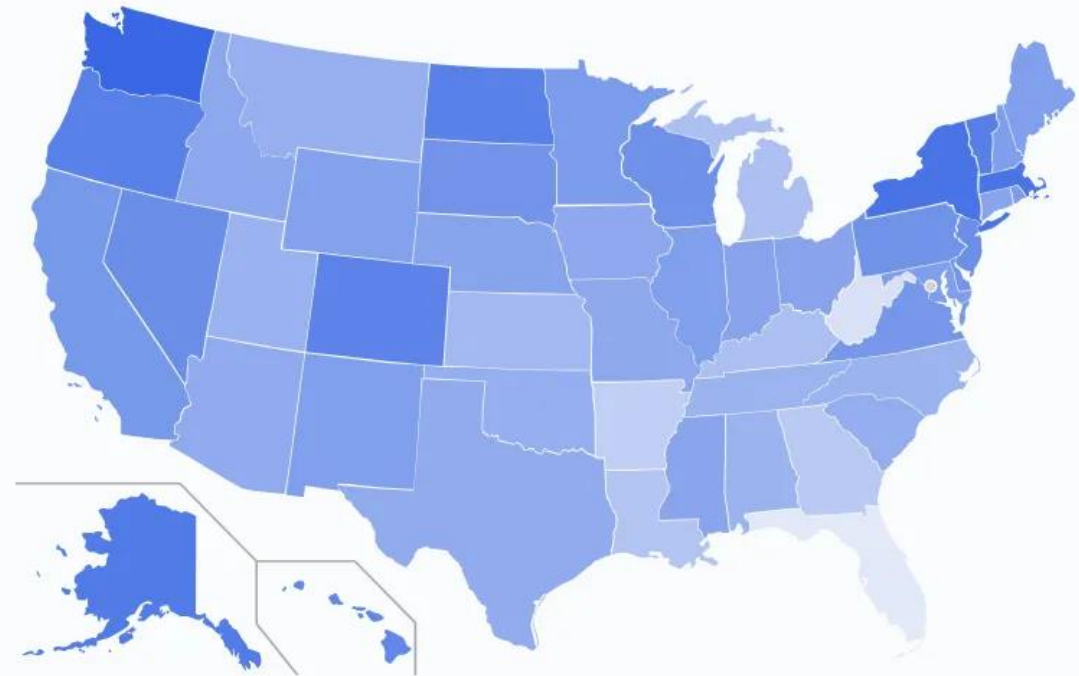


National Average

\$147,524 /year

\$71 /hour

	Annual Salary	Monthly Pay	Weekly Pay	Hourly Wage
Top Earners	\$205,000	\$17,083	\$3,942	\$98
75th Percentile	\$173,000	\$14,416	\$3,326	\$83
Average	\$147,524	\$12,293	\$2,837	\$71
25th Percentile	\$120,000	\$10,000	\$2,307	\$58



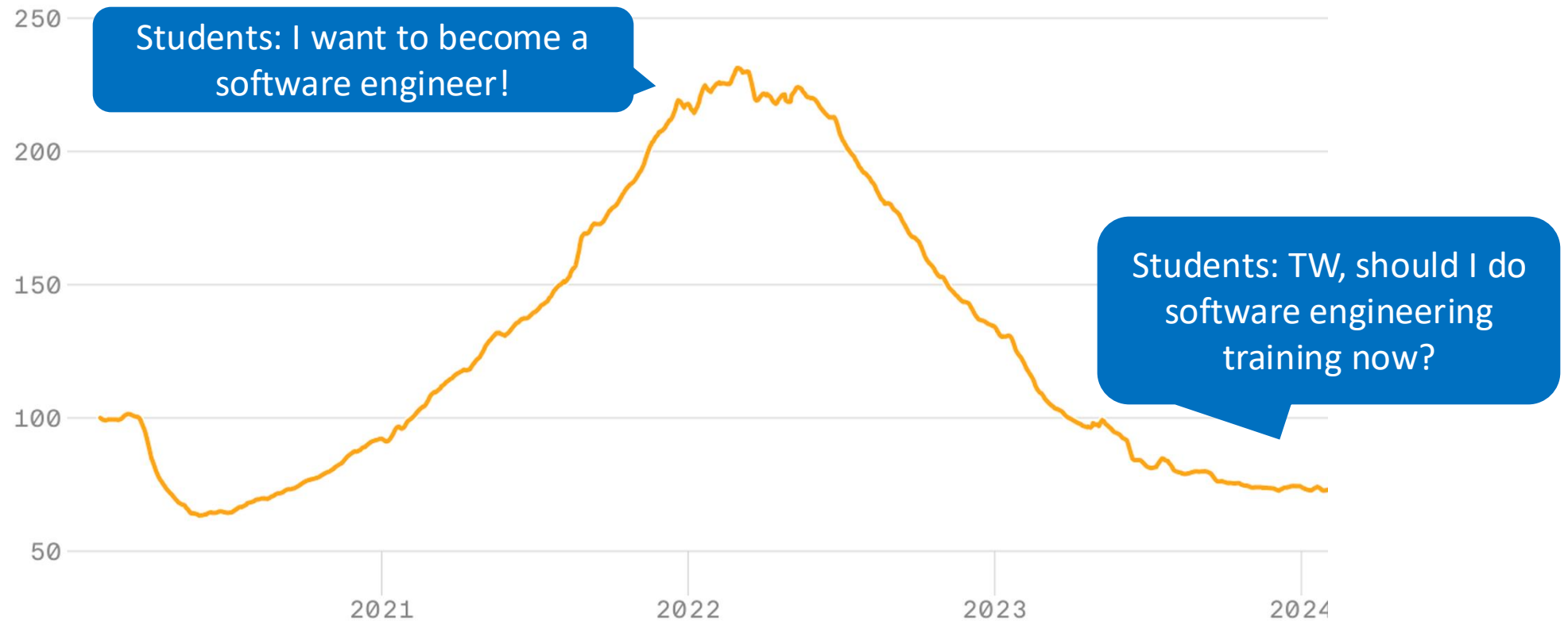
\$63,500 Lowest Highest \$205,500

ZipRecruiter salary estimates, histograms, trends and comparisons are derived from both employer job postings and third party data sources.

Your Probably Have also Seen this ...

Job postings for software developers on Indeed

Indexed to 100 as of Feb. 1, 2020; Daily, Feb. 1, 2020, to July 12, 2024



Students: I want to become a software engineer!

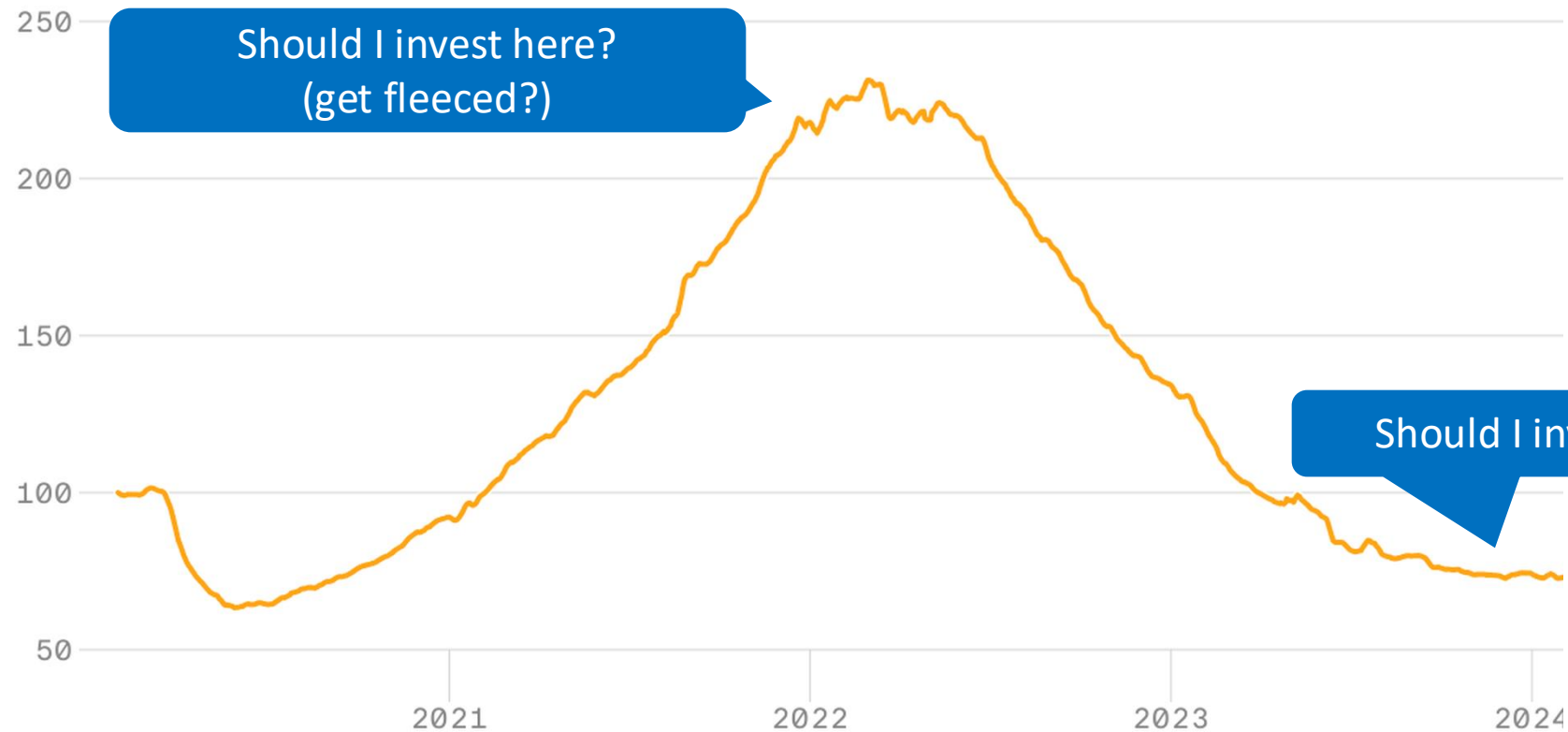
Students: TW, should I do software engineering training now?

My Interpretation

~~Job postings for software developers on Indeed~~

Indexed to 100 as of Feb. 1, 2020; Daily, Feb. 1, 2020, to July 12, 2024

Stock Price of a HPC Software Engineer






















We will Speak C++ Mostly in this Class

- **This is a one-semester journey in the pursuit of speed**
 - C++ allows you to program very closely to the hardware
 - Architecture, cache, performance optimization, machine code, etc.
- **C++ programming is what's being used for low-level stuff**
 - Because it's fast
 - The Linux and Windows kernels are both written in C
 - Python is written in C
- **CUDA, OpenMP, MPI: they cater to C and C++ programmer**
 - A low-level language like C is not friendly but the sky is the limit: you can do as you wish in your C program

Most Popular and Important Languages

Discussed in ECE 455 →

Discussed in ECE 455 →

Aug 2023	Aug 2022	Change	Programming Language		Ratings	Change
1	1			Python	13.33%	-2.30%
2	2			C	11.41%	-3.35%
3	4	^		C++	10.63%	+0.49%
4	3	v		Java	10.33%	-2.14%
5	5			C#	7.04%	+1.64%
6	8	^		JavaScript	3.29%	+0.89%
7	6	v		Visual Basic	2.63%	-2.26%
8	9	^		SQL	1.53%	-0.14%
9	7	v		Assembly language	1.34%	-1.41%
10	10			PHP	1.27%	-0.09%
11	21	^^		Scratch	1.22%	+0.63%
12	15	^		Go	1.16%	+0.20%
13	17	^^		MATLAB	1.05%	+0.17%
14	18	^^		Fortran	1.03%	+0.24%
15	31	^^		COBOL	0.96%	+0.59%
16	16			R	0.92%	+0.01%
17	19	^		Ruby	0.91%	+0.18%

[<https://www.tiobe.com/tiobe-index>]→

My 10-year Programming Perspective

- **People often ask me “TW, X language is cool, you should use it!?”**
 - Ex: Rust is so cool! You should stop using C++ but Rust!
- **In general, your logic thinking (aka *algorithm*) 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 college student, I highly recommend C++**
 - C++ takes time to master, better to start learning it earlier
 - 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

C++



```
#include <iostream>

int main()
{
    std::cout << "Hello world!";
}
```

Python



```
print('Hello, world!')
```



Quiz: Big Mod Problem

- **Computational problem description**
 - Input: a, b, c ($3 < a, c < 50000000$, $1 < b < 2147483647$)
 - Output: $x = a^b \pmod{c}$
- **Example:** $a = 2, b = 7, c = 13$
 - $x = 2^7 \% 13 = 128 \% 13 = 11$ (since $128 = 9 * 13 + 11$)

Quiz: Big Mod Problem

- **Computational problem description**
 - Input: a, b, c ($3 < a, c < 50000000, 1 < b < 2147483647$)
 - Output: $x = a^b \pmod{c}$
- **Example:** $a = 2, b = 7, c = 13$
 - $x = 2^7 \% 13 = 128 \% 13 = 11$ (since $128 = 9 * 13 + 11$)
- **A simple algorithm**

```
for(i=1, x=1; i<=b; i++)  
    x = (x*a)%c;  
std::cout << "a^b % c = " << x << std::endl;
```


Quiz: Big Mod Problem

- **Computational problem description**
 - Input: a, b, c ($3 < a, c < 50000000, 1 < b < 2147483647$)
 - Output: $x = a^b \pmod{c}$
- **Example:** $a = 2, b = 7, c = 13$
 - $x = 2^7 \% 13 = 128 \% 13 = 11$ (since $128 = 9 * 13 + 11$)
- **A simple algorithm**

```
for(i=1, x=1; i<=b; i++)  
    x = (x*a)%c;  
std::cout << "a^b % c = " << x << std::endl;
```
- **Is the solution smart enough?**
 - We need “b” iterations – very slow for large b

Refined Solution (Recursive)

- Input: $a=2$, $b=16$, c ; Output: $x = a^b \pmod{c}$
- **Simple solution needs a total of $b=16$ iterations**
- Let's refine the iteration count recursively

$2^{16} = (2^8)^2$	total 1 calculation
$2^8 = (2^4)^2$	total 1 calculation
$2^4 = (2^2)^2$	total 1 calculation
$2^2 = (2^1)^2$	total 1 calculation
$2^1 = (2^0) * 2$	total 1 calculation

Refined Solution (Recursive) (cont'd)

```
#define SQUARE(x) (x*x)
int bigmod(int a, int b, int c) {
    if(b==0) return 1;
    else if(b%2==0)
        return SQUARE(bigmod(a,b/2,c)%c)%c;
    else return ((a%c)*bigmod(a,b-1,c))%c;
}
int main() {
    int a,b,c;
    while(std::cin >> a >> b >> c) std::cout<<bigmod(a,b,c)<<endl;
    return 0;
}
```

$2^{16} = (2^8)^2$	total 1 calculation
$2^8 = (2^4)^2$	total 1 calculation
$2^4 = (2^2)^2$	total 1 calculation
$2^2 = (2^1)^2$	total 1 calculation
$2^1 = (2^0)$	total 1 calculation

Refined Solution (Iterative)

- **Quiz:** rewrite the recursive bigmod to an iterative version (no recursion)

```
int bigmod(int a, int b, int c) {  
    int result = 1;  
    while(b > 0) {  
        if(b & 1) {  
            result = (result * a) % c;  
        }  
        b = b >> 1;  
        a = a*a%c;  
    }  
    return result;  
}
```

Runtime Comparison

- <https://www.onlinegdb.com/>

Runtime (with -O2) – Enabled Optimization			
(a, b, c)	Naive	Refined (recursive)	Refined (iterative)
(2, 1000000, 3)	13720 us	24 us	22 us
(2, 10000000, 3)	136148 us	9 us	4 us
(2, 100000000, 3)	1320049 us	8 us	4 us

Code Analysis

- <https://godbolt.org/z/6zT4cK3Wz>

	Naive	Refined (recursive)	Refined (iterative)
Lines of C++ Code	7	11	11
Lines of Assembly (-O0)	55	99	59
Lines of Assembly (-O2)	27	57	32



Magic behind the Refined Solution

In fact, there is a name for the refined method:

Divide and Conquer (D&C) algorithm

(used to solve >50% large and complex computational problems, e.g., Google Map, sorting, etc.)

You can use ANY languages to implement this solution – logic thinking (algorithm) matters the most

Back to Course... Hardware Aspects

- **The course designed to leverage a dedicated CPU/GPU cluster**
 - Called Euler :
 - pronounced: /'ɔɪlər/ (IPA) or *OY-lər* (Phonetic) or "**oiler**" (approximate)
- **Students use their CAE accounts to login to Euler for:**
 - CPU-parallel computing
 - GPU-parallel computing
 - Euler login requires 2-factor
 - More info: <https://kb.wisc.edu/cae/page.php?id=138004>
- **Please open your CAE account ASAP**
 - <https://newuser.my.cae.wisc.edu/account>



More on Euler

- **Make sure you can login to Euler no later than next Monday**
 - The first lab (next Friday) needs you to access to Euler
 - Again, if you get registered for the class late, you may not have access until you get in
 - This is totally ok, and we will accommodate that into the grading
 - So, don't worry :)
 - First assignment is out and due in two weeks!
- **Using Euler is clunky (since it's done through Slurm)**
 - **Use your home machine or CAE machine as much as possible**
 - Move to Euler only at the end, to verify your work
 - Euler is the platform used to do your lab assignments & final project
 - We need to ensure a consistent environment to evaluation your code
- **Let's see... how many of your have Nvidia GPUs?**
 - If you have Nvidia GPU and CUDA installed on your machine, you can do most of the work locally – but you still need to learn how to use Slurm on Euler (it's the industry standard)



Course Emphasis

- **Multiple choices when it comes to implementing parallelism**
 - PThreads, Intel's TBB, OpenMP, MPI, Cilk, C++, CUDA, Taskflow, etc.
- **We will focus both CPU- and GPU-parallel programming**
 - C++ standard for low-level thread programming
 - OpenMP standard, aimed both at fine and coarse-grain parallelism
 - CUDA running on Nvidia Graphics Processing Unit (GPU) cards, for data-parallel computing
- **The class is designed with “applied science” in mind**
 - This is a project-driven course, not a theoretical course
 - Slant: a more applied approach to learning about computing
 - You learn parallel computing and apply the techniques to real-world problems
 - This is not designed for theoretical learning
 - Basic idea: understand how computing takes place to leverage it better
 - That being said, materials covered will be useful for other departments too



Course Objectives

- **Recognize applications that can draw on advanced computing**
- **Know how to speed up your code using parallelism**
 - Mostly done through parallel computing, at multiple levels
- **Gain basic skills that will help you map these applications onto a parallel computing hardware/software stack**
 - Write code, build, link, run, debug, profile
- **Help you write software from a viewpoint of parallelism**
 - How to orchestrate multiple cores to do many different jobs?
 - How to avoid multiple cores from racing the same data?
 - How to scale up an algorithm using task- and data-parallel computing techniques?

How we'll go about it ...

- **Before talking about how to write software to run on a piece of hardware, we'll learn something about that hardware asset**
 - Why? Because ultimately your software runs on your hardware
 - One theme of this course “know your hardware before you write your software”
- **We don't focus on low-level hardware details**
 - Dedicated courses already available
 - ECE 353: Introduction to Microprocessor Systems
 - ECE 354: Machine Organization and Programming
 - ECE 552: Introduction to Computer Architecture
 - ECE 556: Design Automation for Digital Circuits
 - ECE 756: Computer-aided Design for VLSI
 - ECE 752: Advanced Computer Architecture I
 - ...

At the Beginning of the Road

- HPC is very much in flux, high rate of change
- There will be issues that I don't know and/or don't understand
 - I always end up learning something new, in most cases with your help
- There might be questions that I don't have an answer for
 - I'll try to follow up on these and get back with you
 - Or you can teach me!
- Most of my statements should be qualified by the preamble “*On most systems...*”, or “*By and large...*”
 - There is no one-fit-all solution in today's highly diverse HPC ecosystem
 - No single programming solution is optimal for all, but the general concept (e.g., cache, memory hierarchy) by and large is true for most system



How I See this Course?

- **Ultimately, this is not a difficult course but a busy one**
 - Weekly lab assignment + weekly project progress presentation
 - Although completing lab assignment is very easy, understanding the knowledge behind will take a while!
 - Preparing your open-house presentation will take a lot of time
 - Poster, machine setup, report, etc.
 - Start thinking about some project ideas the earlier the better!
- **TA is a good resource whom you can learn a lot from**
- **There will be a lot of “Googling” for the class**
 - Dealing w/ situations when you’ll have to address pesky hurdles on your own
- **There will be a lot of collaboration among you during project design**
 - Don’t forget 70% of you career success come from communication skills!