ECE 459

Programming for Performance

Winter 2024

Jeff Zarnett & **Huanyi Chen**

jzarnett@uwaterloo.ca, **huanyi.chen@uwaterloo.ca**

UNIVERSITY OF WATERLOO

About the Course

- Profiling computer systems; bottlenecks, Amdahl’s law.
- Concurrency: threads and locks.
- Techniques for programming multicore processors; cache consistency.
- Streaming architectures, vectorization, and SIMD (single instruction, multiple data hardware units).
- High-performance programming language

About the Course

- Prerequisites

- Remember what we learned about semaphores and mutexes
- Feel comfortable with programming, debugging, etc.
- Feel comfortable with learning a new language (Rust!)

General Information

- Over 400 students
- Two sections
- Pick your preference
- Jeff will do traditional-lecture
- I will do some flipped-classrooms

Jeff	05:30-06:50MW	RCH 101
Me	11:30-12:50MF	E7 5353

General Information

- Flipped classrooms means you learn course content and concepts outside the classroom, and you do exercises in class
- The in-class exercises are for practice only and not for marks
- But it is a good chance to reinforce your understanding

General Information

- The course has sufficient materials to assist self-learning.
- Lecture materials: https://github.com/jzarnett/ece459
- Video recordings: https://www.youtube.com/playlist?list=PLFCH6yhq9yAGTgG7r30clocD3-QUs9wPL
- The lecture materials are the source materials for video recordings and the content that we say in the class

UNIVERSITY OF WATERLOO

General Information

- Piazza will be used for questions, discussions, etc.
- You can also email or send messages to me on Teams but it’s always better to post on Piazza since other students might have the same questions

- LEARN will be used for sample exams, grades, tracking grace day usages, etc.

UNIVERSITY OF WATERLOO

Evaluation

- No midterm exam

Academic Integrity Exercise (A Quiz on LEARN)	1%
Assignments	64% (4 at 16% each)
Final exam	35%

Assignments (aka labs)

1. Manual parallelization of a computation using threads; and use of nonblocking I/O
2. Optimizing log file analytics
3. GPU programming with CUDA
4. Open-ended program improvement via profiling

Assignments should be done individually

Assignment hand-in will be done via **git** using the university provided https://git.uwaterloo.ca (GitLab) service

Assignments (aka labs)

- Lab Instructor. The LI is responsible for technical matters related to the assignments: git.uwaterloo.ca, git in general, administering the system where you need to run your labs.
  - **Richard Baverstock – rbaverst@uwaterloo.ca**

- Teaching Assistants:
  - **S M Taslim Uddin Raju – smturaju@uwaterloo.ca**
  - **Tanmayi Jandhyala – tjandhya@uwaterloo.ca**
  - **Thanushon Sivakaran – tsivakar@uwaterloo.ca**
  - **Vinayak Sharma – v236shar@uwaterloo.ca**

Final Exam

- At least 30% on the final exam to pass the course.
- Otherwise, all your assignment grades are cut in half, so your maximum grade for the course will be 43%.
- In-person, open-book, open-notes

Late policy

- Five grace days
- Grace days are counted in units of whole days
- We look at the last commit time in GitLab on the default branch to determine when your code was submitted.
- No credit for unused late days

Late policy

- Run out of grace days?
- Sixth? lowest assignment mark to be halved.
- Seventh? lowest two assignment marks to be halved.
- More? We’ll start converting marks to 0 and dropping the associated late days.
- (Or keep it simple, don’t run out of your grace days!)

Late policy

- Some suggestions from the computing education community

- Grace days aren’t excuses for procrastination
- It is found that students who use grace days tend to achieve lower grades than those who don’t
- It may be a trap!

Generative AI

We ask for two things if you choose to use them

1. Acknowledge if you used code given to you by an AI tool, just as you would if you found something on a website somewhere

2. Use the tool(s) to help you, not do it all for you

How to use Piazza

- ❌ Please no screenshot-only posts (since the content does not show up in search results)
- ✅ Please restrict a single thread to one question (or a closely-related set of questions).

- The syllabus is on GitHub

- Check it out if you have further questions

Performance!

ECE459: Programming for Performance

UNIVERSITY OF WATERLOO

What is the meaning of "Performance"?

What is Performance?

“Can you draw me a picture where one is programming on stage and audiences are watching,” image generated by OpenAI’s DALL·E, Jan 2, 2024.

UNIVERSITY OF WATERLOO

What is Performance?

- Think about where you use the word “performance”
  - A student gets a full mark in the final exam
  - An AI can recognize the faces in photos
  - A search engine can handle million requests per second
  - …

What is Performance?

- Probably most of the time, it means

- Correctness
- Efficiency

Performance

- Correctness matters in many cases, e.g., accuracy in machine learning
- But we care more about efficiency in the course

- Note that it does not mean correctness is not important
- It just means we assume correctness is guaranteed at the first place

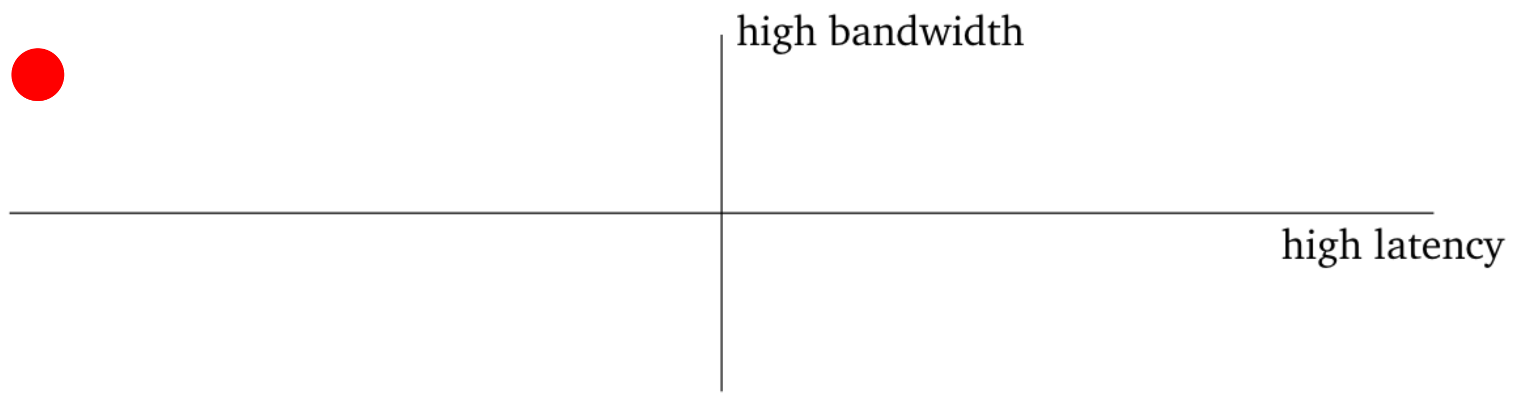
Performance

- Efficiency can be measured by two metrics

- Items per unit time
- Time per item

Performance

- Items per unit time also means bandwidth (or throughput)
- Time per item also means latency
- (What is the best point here?)



Improving performance

- With that said, now we can talk about how to improve the performance
- There are typically two approaches to improve performance

- Improving (reducing) latency
- Do more work at a time (increasing throughput)

Reducing latency

- Do less work
  - avoid (re)calculating intermediate results

```python
# Fibonacci numbers
def f(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    return f(n-1) + f(n-2)
```

Reducing latency

- Do less work
  - avoid (re)calculating intermediate results

- String-searching problem
- Searching a word (W) in a document (S)

```
S:ABC ABCDABAB CDABCDABDE
W:               ABCDABD
```

UNIVERSITY OF WATERLOO

Reducing latency

- Do less work
  - avoid (re)calculating intermediate results

- Simple solution
  - Two `For` loop

```
S:ABC ABCDABAB CDABCDABDE
W:ABCDABD
W: ABCDABD
W:  ABCDABD
…
```

Reducing latency

- Do less work
  - avoid (re)calculating intermediate results

- Simple solution
  - len(W) = k, len(S) = n
  - Worst-case performance is O(k·n)
- Much unnecessary work
  - First three letters are all different

S:ABC ABCDABAB CDABCDABDE

W:ABCDABD

~~**W**: ABCDABD~~

~~**W**: ABCDABD~~

UNIVERSITY OF WATERLOO

Reducing latency

- Do less work
  - avoid (re)calculating intermediate results

- Producing text output to a log file or to a console screen is expensive
- Only log necessary information
- Bunch of studies are working on how to achieve that

Reducing latency

- Do less work
  - avoid (re)calculating intermediate results
  - computing results to only the accuracy that you need in the final output

- Do you always need exact numbers from a large database?
- Or you probably just want numbers with a certain level of accuracy
- Approximate Query Processing (AQP)

Reducing latency

- Do less work
  - avoid (re)calculating intermediate results
  - computing results to only the accuracy that you need in the final output

- One idea for AQP
- Data are coming from the real world
- They follow an unknown distribution
- You estimate the distribution and return **approximated aggregates** efficiently

Reducing latency

- Do less work
  - avoid (re)calculating intermediate results
  - computing results to only the accuracy that you need in the final output

- It can be very accurate if the underlying model is precise

- E.g.,
- physics laws, arithmetic sequence

Reducing latency

- Do less work
- Be prepared

- Pre-generated reports
  - Request transcript
- Pre-aggregation of Online Analytical Processing (OLAP) data, etc.
  - What’s the annual profit of last year?

Reducing latency

- Do less work
- Be prepared
- Be smarter

- Better algorithms
  - From bubble sort to quicksort, etc.
- Compiler optimization to get smaller constant factor
- Aware of cache and data locality/density issues

Reducing latency

- Do less work
- Be prepared
- Be smarter

- Usually done in libraries
- Commonly, you just pick what libraries you want to use, instead of writing it on your own
- Improve once and benefit all

UNIVERSITY OF WATERLOO

Reducing latency

- Do less work
- Be prepared
- Be smarter
- Improve the hardware

- If your computer is too slow, buy a new one …
- Hard disk drive (HDD) => Solid-State Drives (SSD)
- Larger memory
- Better CPU
- Better GPU, etc.

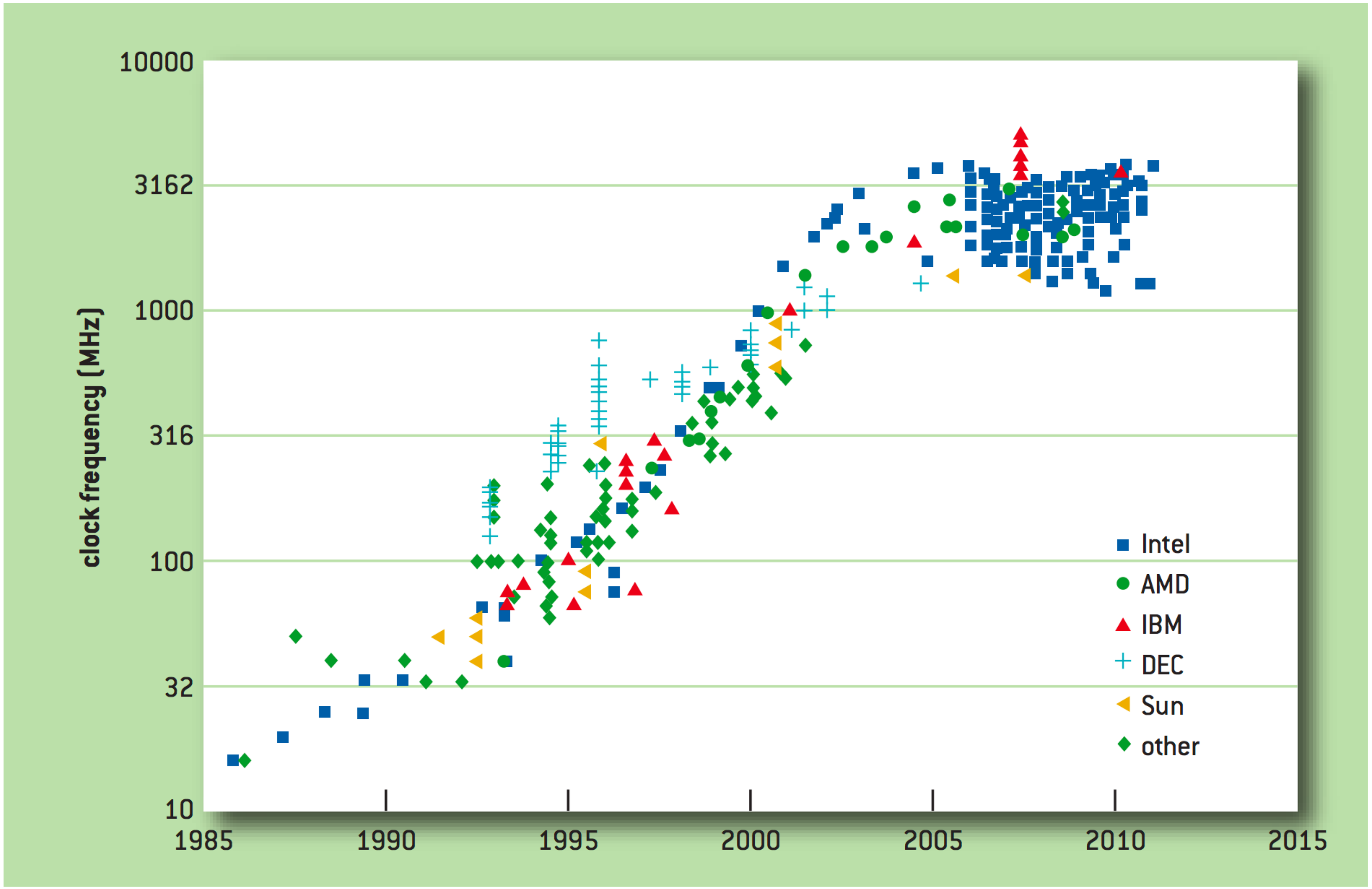
Reducing latency

- Do less work
- Be prepared
- Be smarter
- Improve the hardware
- Using assembly code

- Compiler transforms your high-level code to assembly code (kind of like text representation of machine code)
- Compilers may be not very smart

Improving throughput

- While you may frequently hear about multi-threading, parallel-processing, multi-core CPUs, distributed systems, etc.
- They were not popular phrases 20 years ago.
- Why do we want to look at them these days?



CPU clock speed (frequency) over time [DKM+12]

Parallelism

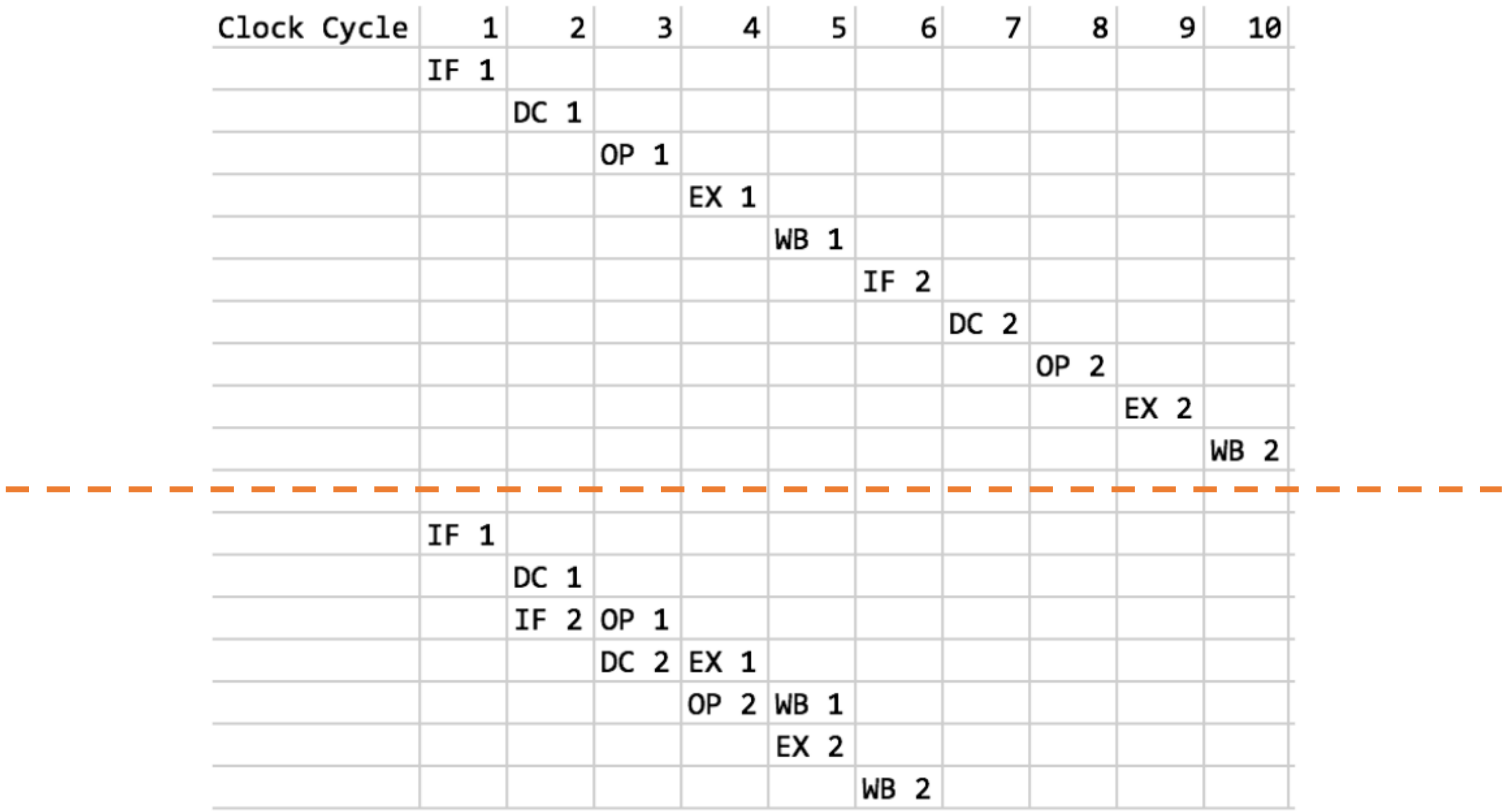
- In general, parallelism improves bandwidth, but not latency.
  - For example, a web server can handle multiple requests simultaneously, but each request still needs the same time to handle.

- However, parallelism also complicates your life

Parallelism

- Different problems are amenable to different sorts of parallelization
- Depending on how you look at the problem, you may discover different approaches to parallelize things
  - Algorithm itself?
  - CPU Pipelining?
  - Single Instruction Multiple Data (SIMD)?
  - etc.

Parallelism



Clock Cycle	1	2	3	4	5	6	7	8	9	10
	IF 1									
		DC 1								
			OP 1							
				EX 1						
					WB 1					
						IF 2				
							DC 2			
								OP 2		
									EX 2	
										WB 2
	IF 1									
		DC 1								
		IF 2	OP 1							
			DC 2	EX 1						
				OP 2	WB 1					
					EX 2					
						WB 2				

CPU pipelining

Parallelism

- Single Instruction Multiple Data (SIMD)

UNIVERSITY OF WATERLOO

Parallelism

- Real parallelism needs hardware supports
- Which can execute multiple instruction streams simultaneously
- Multicore processors, SMP (symmetric multiprocessor) systems, or a cluster of machines
- Or we can put some work on to graphics processing units (GPUs)

Difficulties with using parallelism

- Coordination overhead
  - Sometimes it is easier to do a project when it's just you rather than being you and a team
- Sequential part dominates the program (Amdahl's Law)
  - Trivial improvement from parallelizing other parts
- Ordering
  - Some events A are guaranteed to happen before other events B, but many events X and Y can occur in either the order XY or Y X
  - Data races
  - Deadlock

Laws of Performant Software

- Programming language << Programmers awareness of performance
- Small details matter. A missing `free()` can cause big problems.
- There is a very high correlation between performance degradation and unbounded use of resources
- More hardware will not save you from poorly-written code

Don’t guess; measure

- It is important to profile you code
- Intuition seems to be often wrong here
- Run your program with realistic workloads
- See http://computers-are-fast.github.io

Rust

ECE459: Programming for Performance

Why Rust?

- Remember the time when you struggled with memory leaks and race conditions?
- A lot of the problems we frequently encounter are the kind that can be found by Valgrind, such as memory errors or race conditions.
- However, Valgrind is a runtime checker

Why Rust?

- A design goal of Rust is to avoid issues with memory allocation and race conditions
- Compile-time checking
- Will gradually replace C/C++ to some extent
  - Linux 6.1: Rust to hit mainline kernel
  - https://www.theregister.com/2022/10/05/rust_kernel_pull_request_pulled/

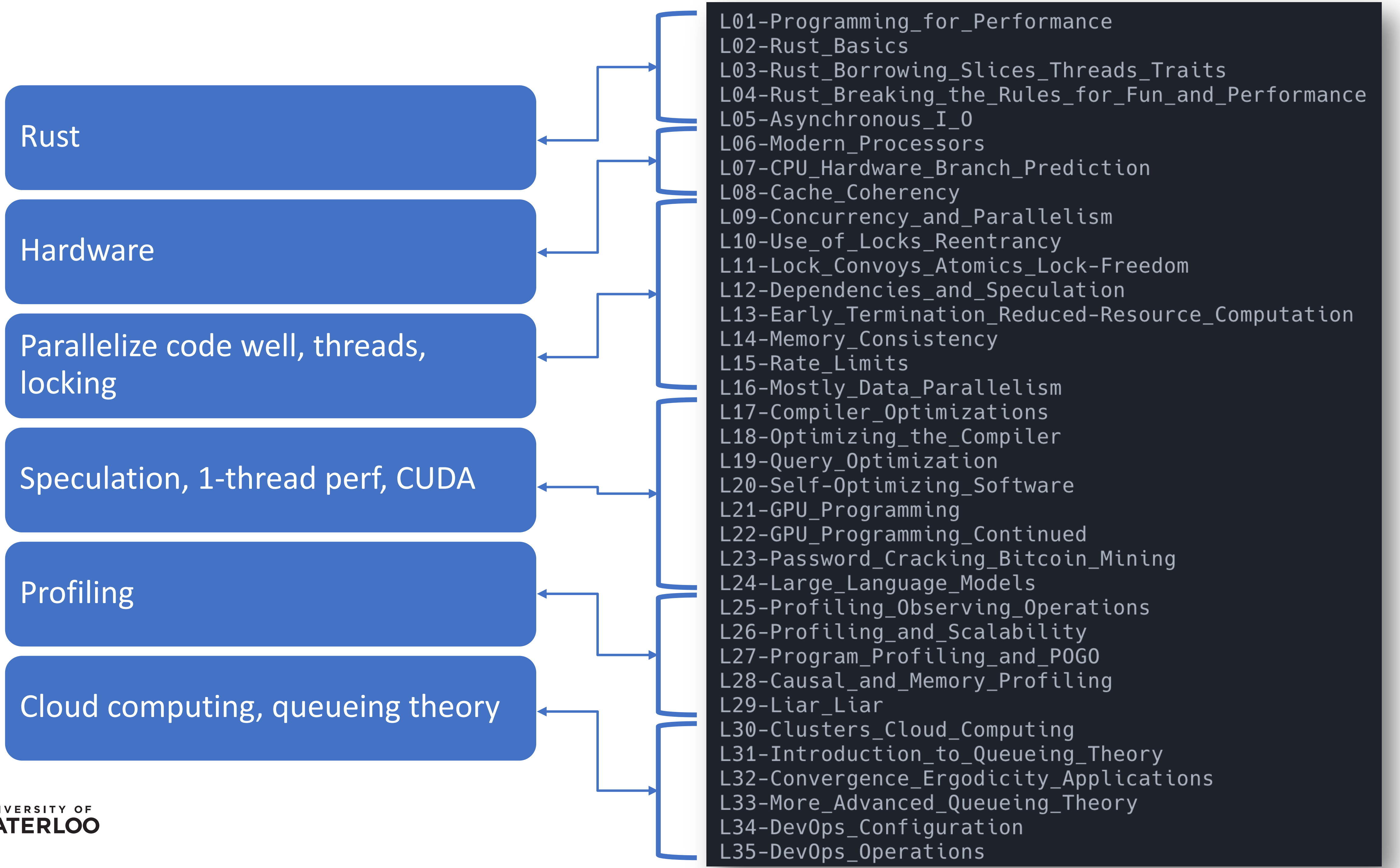
Trade-off in using Rust

- In particular, you will find some things harder to code in Rust than in C/C++
- You put more time in compiler-time debugging instead of in runtime
- However, they are also more likely to be correct (remember we care about correctness and assume it at the first place when we talk about performance!)

Trade-off i

- It may seem lik developers have put a lot o hat try to help.
- As always, plea

https://infosec.exchange/@AstraKernel/109636334567212340


Rust
Hardware
Parallelize code well, threads, locking
Speculation, 1-thread perf, CUDA
Profiling
Cloud computing, queueing theory

```
L01-Programming_for_Performance
L02-Rust_Basics
L03-Rust_Borrowing_Slices_Threads_Traits
L04-Rust_Breaking_the_Rules_for_Fun_and_Performance
L05-Asynchronous_I_O
L06-Modern_Processors
L07-CPU_Hardware_Branch_Prediction
L08-Cache_Coherency
L09-Concurrency_and_Parallelism
L10-Use_of_Locks_Reentrancy
L11-Lock_Convoys_Atomics_Lock-Freedom
L12-Dependencies_and_Speculation
L13-Early_Termination_Reduced-Resource_Computation
L14-Memory_Consistency
L15-Rate_Limits
L16-Mostly_Data_Parallelism
L17-Compiler_Optimizations
L18-Optimizing_the_Compiler
L19-Query_Optimization
L20-Self-Optimizing_Software
L21-GPU_Programming
L22-GPU_Programming_Continued
L23-Password_Cracking_Bitcoin_Mining
L24-Large_Language_Models
L25-Profiling_Observing_Operations
L26-Profiling_and_Scalability
L27-Program_Profiling_and_POGO
L28-Causal_and_Memory_Profiling
L29-Liar_Liar
L30-Clusters_Cloud_Computing
L31-Introduction_to_Queueing_Theory
L32-Convergence_Ergodicity_Applications
L33-More_Advanced_Queueing_Theory
L34-DevOps_Configuration
L35-DevOps_Operations
```

Prepare for the next lecture

- **Setup GitLab if you haven’t done so (!)**
- Read the lecture 2 notes
- Read relevant chapters of the Rust book (https://doc.rust-lang.org/stable/book/)
- Setup Rust development environment
- Bring your laptops and we will do some in-class exercises