CSCI 1675
Designing High Performance Network Systems

Fall 2025
Akshay Narayan
BioMed 291 10:30 - 11:50
https://cs.brown.edu/courses/csci1675/

About me: Akshay Narayan

- Second year at Brown
- Postdoc at Berkeley, PhD from MIT, Undergrad from Berkeley
- Focus throughout: systems and networking

Today’s class session

- What is this class about?
- Administrivia

Class Goals

- Understand performance for network systems
  - How do we measure performance?
  - How does system design affect performance?

What is the class not about?

- Non-network systems' performance (consider 1952Y)
- Learning about network protocols
- Interface between applications and hardware
- Building applications that use the network

These topics are addressed in this class's prerequisites, **CS 1680 / CS 1670 / CS 1380**

(We may cover some topics you have previously seen in greater depth)

~10 minute Discussion:

1. Why care about performance?
2. If system A has “better performance” than system B, what does that mean?

Meme 1: Specialization

Dave Patterson (UC Berkeley/Google)
(also Turing award winner):

- Scenario with 100M users speaking to phones 3 minutes per day: If only CPUs, double whole data center fleet!
- Goal: Custom Domain Specific Architecture (DSA) to reduce the Total Cost of Ownership (TCO) of DNN inference phase by 10X

One meta-strategy to improve performance: specialization

source: https://www.cs.ucla.edu/wp-content/uploads/cs/PATTERSON-10-Lessons-4-TPU-gens-CO2e-45-minutes.pdf

Meme 2: Premature Optimization

Don Knuth (Stanford), 1974 Turing Award lecture:

The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; **premature optimization is the root of all evil** (or at least most of it) in programming.

Trying to improve performance
might introduce bugs

Counterpoint: Costs of specialization

Can we optimize without specializing?
A plea for generality in research systems
Aurojit Panda (NYU), HotEthics Workshop 2024

- Building specialized hardware clusters takes a lot of resources, and they can only do one thing
- So, not many entities can afford to invest in specialized deployments
- Ultra-fast performance within a specific deployment, or decent performance across a wide range of deployments?

source: https://hotethics.github.io/papers/Panda_Hotethics24.pdf

Questions to keep in mind

- When is improving performance worth the costs?
  - What type of costs are they (societal? complexity?)?
  - How can we measure those costs?
- This class: It's important to understand performance, so we know not just **how** but **when** to improve it

Class Topics

- Module 1: Measuring performance
  - ~September
- Module 2: IO
  - ~Early October
- Module 3: Concurrency and Scheduling
  - ~Late October
- Module 4: Cloud Computing
  - ~November

Administrivia

Graduate Co-Instructor: Oğuzhan Çölkesen

- Research on parallelizing shell programs
- Will give ~one third of the lectures
- Office hours:
  - Thursdays, 12:00-14:00
  - Location TBD

HTA: Aryan Singh

- Returning HTA from spring offering
- Development work on projects 0 and 1
- Office hours:
  - Monday, Wednesday 17:00 - 18:00
  - Location TBD

UTA: Bokai Bi

- CS 2680 alumnus: research on eBPF
- Office hours:
  - Tuesday, Thursday 17:00 - 18:00
  - Location TBD

Enrollment

- Capacity won’t increase (you just met the entire staff).
- If you are going to drop, please do so sooner rather than later as a courtesy to people on the waitlist
- We will resolve the waitlist on a rolling basis
  - Last semester, everyone who wanted a spot got one
  - Waitlist form: https://forms.gle/a5GzEPi6Gw5BwgSk8
  - (I will publish these slides on the class website)

Class Components + Grading

- This class has two major components:
  - Reading + writing **homeworks (35%)**
  - Implementation + evaluation **projects (65%)**
    - Projects 0, 1, and 2: focus on network APIs
    - Project 3: focus on microservices

Reading + Writing Homework Assignments

- Goals: Get practice reading technical papers about content from lectures, and writing technical responses.
- We will read ~6 research papers across 4 homework assignments
- HW0 **out today**: https://cs.brown.edu/courses/csci1675/fall2025/hw/hw0.html
  - **Due 16 Sept. (the end of shopping period)**

Please don't use AI tools (LLMs) to write homework responses. The whole point is practicing your own technical writing skills.

Projects

- Goals: Learn evaluation techniques. Gain experience with modern network APIs and tools.
  - Projects 0, 1, 2 will progressively build and evaluate a fake-work app
    - Students found project 1 hard last semester, so we have split it into two milestones (7 Oct., 21 Oct.)
  - Project 3 is standalone
- Depending on enrollment, we might make some projects (not projects 0 or 1) team projects (teams of 2)
- All projects will be implemented in Rust

Projects

- Implementation:
  - Write a performance-oriented system feature
  - Mostly focused on IO (network systems)
- Evaluation:
  - Measure and interpret your system's performance
  - Write a **report** explaining your system's performance using instrumentation data
    - Run on course VMs to get consistent results, but your deliverable is really the report (submit on Gradescope)

Projects

- Different than projects in many other classes
- Focus on **understanding behavior** over implementation correctness
- You are responsible for testing your own implementation
- Grading format: “research meeting”
  - Meet with course staff to briefly present your report
  - Questions about the performance trends you observed
  - Answers using quantitative evidence from your report
- Regrade policy:
  - Fix any problems by the next grading meeting to regain points

Course Feedback about 1675 Projects

This course is more difficult than you expect. Having taken project heavy courses, I thought this course would take a lot less time.

Making sure your metrics meet the requirements can be difficult if you take the wrong approach, and the assignments will not tell you what approach to take.

The class's smallest focus is on the implementation, and the majority of it is on the data/graphs you produce, how you analyze them, and how you justify them.

Why Rust?

- Modern systems programming language
  - Great tooling and documentation, easy to work with
  - Compiler can detect memory safety bugs
  - If you’re familiar with C/C++, Rust is easier
  - If you’re familiar with Golang, its runtime bakes-in features that project 2 will explore implementing

Rust Lifetimes

Original C version (real bug from my code, 2022)

```
int make_match_rule(struct rte_flow_item *pattern_out, u16 dst_port) {
  struct rte_flow_item_udp udp_flow =
    { .hdr.dst_port = RTE_BE16(dst_port) };
  struct rte_flow_item patterns = {
    { .type = RTE_FLOW_ITEM_TYPE_UDP,
      .mask = &udp_dst_port_mask,
      .spec = &udp_flow, .last = NULL }};
  memcpy(pattern_out, &patterns, sizeof(struct rte_flow_item));
  return 0;
}
```

Bug: DPDK flow steering isn’t working.

Rust Lifetimes

Rust version

```
fn make_match_rule(pattern_out: &mut[rte_flow_item],
    dst_port: u16) -> i32 {
  let udp_flow = rte_flow_item_udp { dst_port };
  let patterns = [ rte_flow_item{ udp_flow: &udp_flow } ];
  pattern_out.copy_from_slice(&patterns);
  return 0;
}
```

Rust Lifetimes

```
error[E0597]: `udp_flow` does not live long enough
  --> src/lib.rs:11:45
   |
10 |   let udp_flow = rte_flow_item_udp { dst_port: 42 };
   |       -------- binding `udp_flow` declared here
11 |   let patterns = [ rte_flow_item{ udp_flow: &udp_flow } ];
   |                                             ^^^^^^^^^
   |                                             |
   |                                borrowed value does not live long enough
   |         this usage requires that `udp_flow` is borrowed for `'static`
...
14 | }
   | - `udp_flow` dropped here while still borrowed
```

Learning Rust

- Brown's Rust book: https://rust-book.cs.brown.edu/
  - Chapter 4, Lifetimes: https://rust-book.cs.brown.edu/ch04-00-understanding-ownership.html
- "Traditional" book, TRPL: https://doc.rust-lang.org/stable/book/
- Rust by Example: https://doc.rust-lang.org/stable/rust-by-example/
- std library documentation: https://doc.rust-lang.org/std/index.html

Action Items

- Fill out the override form if you are not registered
- Start working on HW 0
- Project 0 will be released soon
- Take some time to practice with Rust if needed