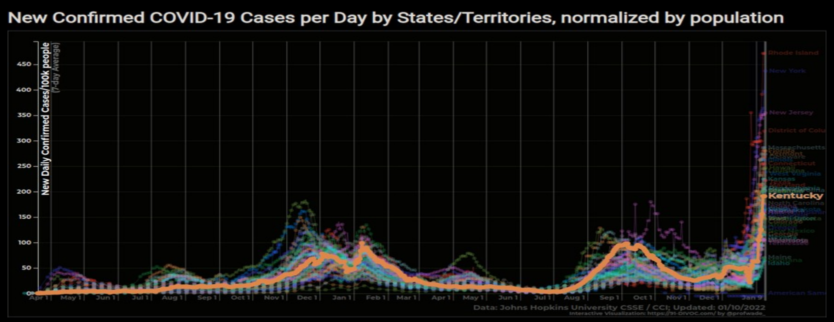
Introduction

EE699-001/EE599-001, Spring 2022

Hank Dietz

`http://aggregate.org/hankd/`

University of Kentucky



Class Meetings

- We are scheduled to meet in person, and that is the plan for the semester… *as I write this*
- **Masks must be worn for in-person meetings**
- Omicron is ***much more easily transmitted*** than Alpha or Delta strains
  - If you might have COVID19, get tested
  - If you are under quarantine, stay home
  - Quarantine is an excused absence, and we will help you keep up with recordings, etc.

Course Overview

- You know how to write a C/C++ program
- You understand basic computer architecture
- You probably haven't written much parallel code
- This course is about **parallel processing**
  - You will write parallel programs
  - How to **use architectural features**
  - It's really about **improving performance**

Changes for Spring 2022

- We're NOT going to start with parallel architecture, but with ideas about optimization
- Students had too weak programming skills
  - I'll be starting with sequential optimization
  - There will be more coding
  - There will be "flipped classroom" project work done virtually using Zoom
- Grads (EE699) do the same as undergrads (EE599), except projects are expanded

Textbook

- There is no textbook…
  - We'll use Canvas for course administration
  - Other materials will be at the course URL:
    `http://aggregate.org/GPUMC/`

Grading & Such

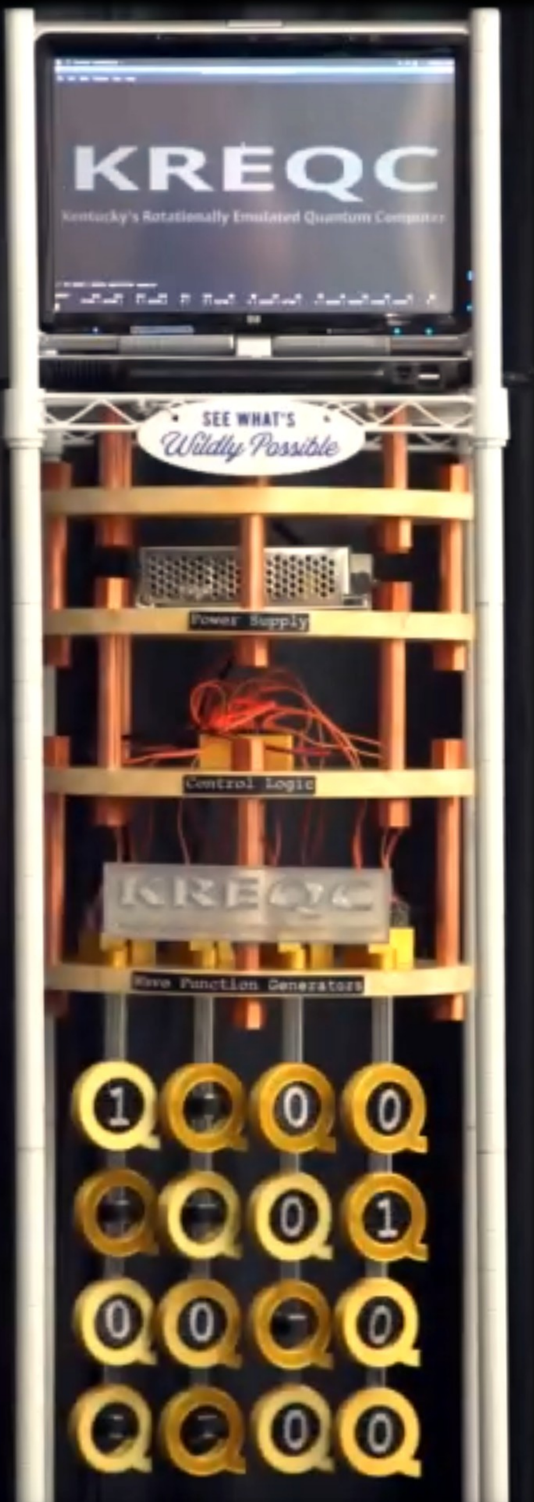
- Two exams:
  - Exam 0, ~15%: sequential & MIMD
  - Exam 1, ~25%: comprehensive, mostly SIMD
- Projects, ~60%:
  - All involve some coding in C/C++
  - May be some team projects
- Course grade limited to 1 letter above lower of exam or project average
- I try not to curve much; always in your favor

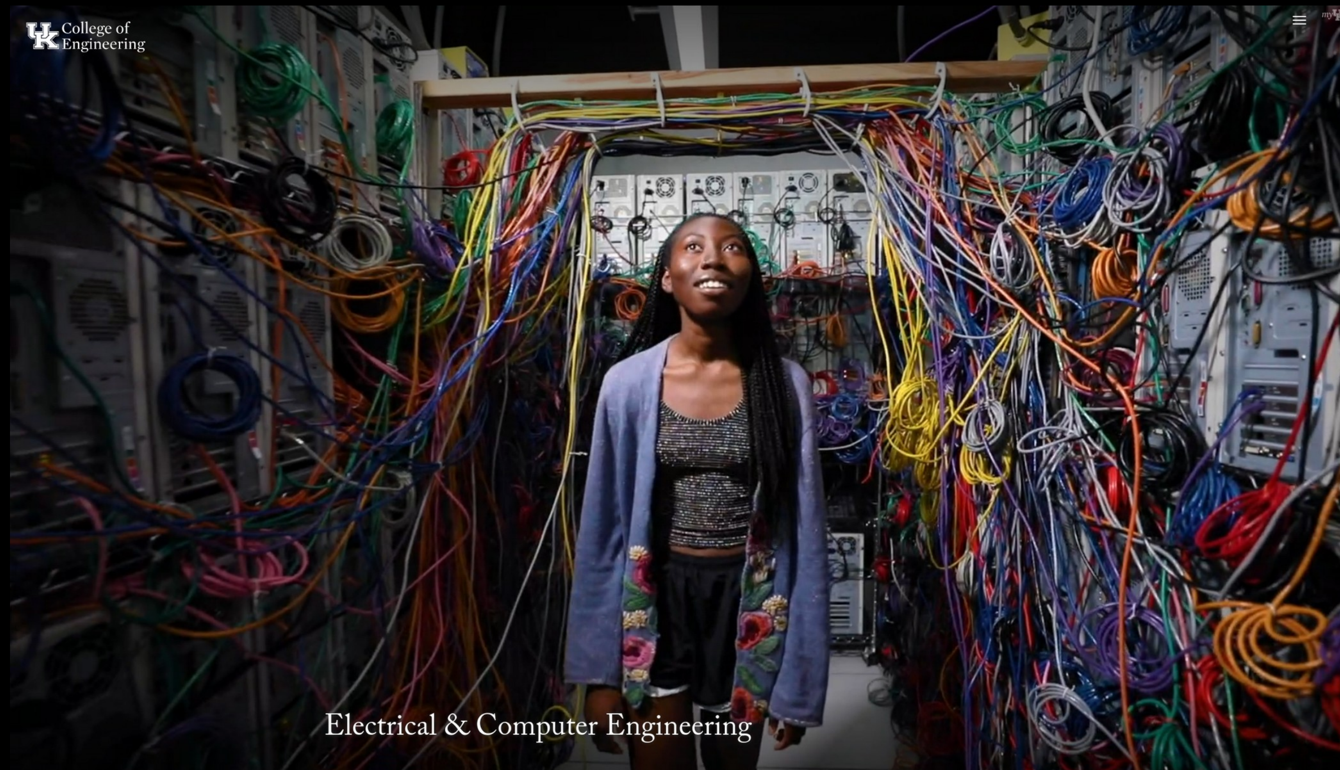
Course Content

Topic	Weeks	Exam
Introduction; concept of performance-critical architectural features, parallel processing, attached processors, heterogeneity, set-up for projects	1	0, 1
Optimizing code (sequential C/C++ code) — Performance analysis tools — Tuned libraries, magic algorithms — Software tools: specializers, superoptimizers, genetic programming (GP)	2	0, 1
Introduction to MIMD parallel architecture	1	0, 1
Shared memory programming & multi-core processors (Pthreads, **OpenMP**)	2	0, 1
Distributed memory programming & clusters (**MPI**)	2	0, 1
Review for Exam 0: The C/C++ Exam		
Introduction to SIMD parallel architecture	1	1
SWAR: SIMD Within A Register and vectors	1	1
GPUs: Graphics Processing Units (**OpenGL**)	1	1
GPGPU: General-Purpose GPU coding (**CUDA**, OpenCL)	2	1
All together	2	1
Review for Exam 1 (the final): The And SIMD Too Exam		

Me (and why I'm biased)

- Hank Dietz, ECE Professor and James F. Hardymon Chair in Networking
- Office: 203 Marksbury
- Research in parallel compilers & architectures:
  - Built 1st Linux PC cluster supercomputer
  - Antlr, AFNs, SWAR, FNNs, MOG, ...
  - Various awards & world records for best price/performance in supercomputing
- Lab: 108/108A Marksbury – I have **TOYS**!


KREQC
Kentucky's Rotationally Emulated Quantum Computer
SEE WHAT'S
Wildly Possible
Power Supply
Control Logic
KREQC
Wave Function Generators


College of Engineering
Electrical & Computer Engineering


UK ECE

Why Do I Care About Computer Performance?

- I don’t really care about supercomputers… I care about making computations cheap
- Computers are tools:

  “A workman is known by his tools.”

  Anyone can buy lumber & wood working tools, but that doesn’t make them able to build nice furniture… it’s largely about knowing how to use the tools and how to make special tooling

Performance-Critical Architectural Features

- My PhD work called these **ECF**s (Efficiency Critical Features): architectural features that, if used poorly, destroy performance
- Examples:
  - Conditional control flow
  - The memory hierarchy
  - **Parallel processing support!!!**

A Memory Hierarchy Example

- Loop nest traversal order vs. data layout
  - Matching increases spatial locality
  - Mismatch causes cache & TLB misses

E.g., if `a[0][0]` is next to `a[0][1]`:

```
for (i=0; i<N; ++i)
  for (j=0; j<M; ++j) a[i][j] = 0;
for (j=0; j<M; ++j)
  for (i=0; i<N; ++i) a[i][j] = 0;
```

Parallel Processing

- Done faster working on parts simultaneously
  - Scalable speed-up
  - Generally not automatically used…
    well, ILP (Instruction-Level Parallelism) is,
    but there's so much more
- Scalable parallel hardware:
  - MIMD: each processing element has a pc
  - SIMD, vector, GPU: one pc shared by all

Parallel Processing

- Homogeneous vs. heterogeneous
  - Multiple cores are homogeneous
  - Multi-core processor + GPU isn't
- Attached processors (e.g., GPUs)
  - Literally computers hanging off a host
  - There is cost associated with interactions

How We’ll Start

- We’ll start with optimizing sequential code
  - Then MIMD
  - Then SIMD (SWAR, vector, GPU)
  - Then all together
- Fill-out the ungraded quiz on Canvas so we can get started interactively coding...