# ECE 574 – Cluster Computing
# Lecture 1


Vince Weaver

https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu


12:30pm, Barrows 130

21 January 2025

# ECE574

- Distribute and go over syllabus
- Course website, all assignments and class notes will be posted here:

  https://web.eece.maine.edu/~vweaver/classes/ece574/ece574_2025s.pdf

# ECE574 Syllabus – Office Hours

- Office is 203 Barrows
- Office hours 2pm-3pm Tuesday/Thursday
- Feel free to e-mail for appointment too, or just stop by if my door is open

# ECE574 Syllabus – Textbook

- None

# ECE574 Syllabus – Computer Accounts

- Will be handing out computer accounts, please use them responsibly.

# ECE574 Syllabus – Programming

- Will involve a lot of coding, mostly C or C-like languages.

# ECE574 Grading – Homeworks

- Homeworks, 50% – roughly 10, lowest dropped.
- Generally will be due on Thursday by beginning of class.
- Will usually have at least a week to do them.
- Submission by e-mail, grades sent in response to that e-mail
- Will send out e-mail when assignment posted on website.

# ECE574 Grading – Exams

- Two Midterms, totaling 25%, around spring break and near end
- No final exam

# ECE574 Grading – Final Project

- Project, 20%
- HPC related project, open-ended
- Can program in any language
- Can work in groups
- Presentation during last week of classes
- Final writeup
- More details as we get closer.

# ECE574 Grading – Late Work

- Class participation: 5%
- Work will lose points for being late, but please turn in anyway, especially homeworks.
- Class notes will be posted on the website.

# ECE574 — Other

- Covid Policy – if really sick, please don't come to class if slightly sick, wear a mask if possible
- Code Help
  - If you have questions often the most efficient way is to send me your code to look at via e-mail
  - Sending me the actual code is best and will get better results than sending screenshots
- Standard UMaine boilerplate

# ECE574 — Academic Honesty

- I hate to dwell on this but it comes up more than you'd think
- **For coding assignments please only submit code you wrote yourself**
- Do not turn in as yours other people's code, either from this class, copied off the internet, or via AI
- Do not share your code with others in the class, even after the submission deadline. If you share your code and the person you shared with submits it too, **you both**

**will face consequences**

- The minimal consequence for this is a zero on the assignment. Note: a zero for cheating will not be dropped as part of the "lowest homework grade is dropped"

# ECE574 — Academic Honesty Part 2

- Note that this doesn't mean you can't get help. The following are allowed
  - You can always ask me (the professor) for help
  - You can always discuss and ask for high-level help from others about the assignment (just don't copy code)
  - You can even ask someone to look at your code to spot errors you might be missing (ideally do this without sending it to them)

# Cluster Computing

What is cluster computing?

Basically it's just a large number of computers working together.

So why is that interesting?

# Road to Parallelism – Serial

- Originally computers were serial, did one thing at a time (though could hide this via multi-tasking)
- Moore's Law meant performance would increase
- At some point limit hit for serial execution, so much effort went into trying to find parallelism in code automatically in hardware (pipelining, out-of-order execution, superscalar)
- We spend a whole semester on this in ECE571

# Road to Parallelism – SMP

- Moore's Law continued, but serial speedup trailed off (Memory Wall a big issue)
- Instead transistors used for multiple-cores in system.
- Suddenly parallelism was everyone's problem
- Even in embedded systems and phones

# Multicore Systems

- Multicore systems need special programming to run parallel code
- It's possible even then you'll want more processing power than a single multi-core system can provide
- In that case you'll need to spread the processing across multiple machines: a cluster

# Supercomputers/HPC

InsideHPC defines HPC: "High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business."

And a supercomputer is similarly vague, just a larger computer than the kind you'd normally use.

# Related terms

- Supercomputer – a large computer. No real definition, "I know one if I see one".
- High Performance Computing – using large computers
- Cluster – one way to make a large computer out of many small computers
- Grid computing – idea to make computing like the power grid, a utility you access to run your jobs. A large loosely coupled cluster.

- Cloud computing – more or less a newer and more buzzword-friendly term for grid computing. Often uses virtualization, not often used for high performance mostly because usually the network is not optimized in a cluster fashion.

- Datacenter – a large building with lots of network connected computers. Not a cluster unless they are working together. This gets complicated when things like google are involved.

- Mainframe – a type of large computer. Usually differentiated from a supercomputer because a mainframe concentrates on reliability and I/O performance rather than CPU performance.

- Distributed systems – a system that is made out of many subnodes that communicate by passing messages to work on large problems. Sort of the old CS term for cluster computing.

- Parallel computing

# What about AI?

- Systems used for AI overlap a lot with those for HPC
- Traditional HPC uses a lot of 64-bit floating point math
- AI instead operates on much smaller numbers, often FP16 or smaller
- While we won't cover much on AI in this class, a lot of what we will cover is relevant to low-level AI implementations

# What about Quantum Computing?

- Quantum has been in development for years and it's unclear if it will ever be fully practical
- There have been some breakthroughs recently
- In theory some problems currently done with HPC might be done faster with quantum computing
- I don't think all HPC workloads can necessarily be replace by quantum
- I did attend some talks at SC'24 on this topic

# Cluster Programming

- The easiest way is to just have single-thread programs but run multiple of them
- If instead you want to run one program but have it run faster across multiple cores you'll need to re-write it to be parallel
- You will need to do this to see a speed benefit on multi-core systems (cluster or otherwise)

# Parallel Programming

- It's hard
- It's really hard
- It's possibly orders of magnitude harder

# Parallel Programming

- Most systems, even embedded systems and cellphones are parallel these days
- Languages don't support it well
- Human brain has trouble grasping it
- It should probably be taught earlier than a 500 level grad course

# Can it be someone else's problem?

- Can code be automatically made parallel?
- Why not have compiler auto-parallelize code?
- Oh they try, how they try.
- Compilers and CPU hardware designers (OoO) wring out as much as they can
- Maybe AI will save the day?
- For now it has to all be done manually.

# Parallel Programming Languages

- C?
- Fortran?
- C++?
- What about things like Javascript, Python?
  Python widely used but actual parallel stuff in C libraries
  (global lock means until recently Python couldn't scale)
- What about things like Go or Rust or Zig?

# HPC Workloads

- Linear Algebra
- Modeling (weather, chemistry, biology, nuclear, aerospace)
- Business (high-speed trading)
- Simulation
- 3d rendering (movie studios, games)
- Data crunching (geology exploring)
- AI? Bitcoin Mining?

# A note on Models

- "It's tough to make predictions, especially about the future." — Yogi Berra
- "All models are wrong, but some are useful" — George E. P. Box
- Wasted lots of time on computer architecture simulations
- GIGO

# A little history

- 1960s Seymour Cray (interesting guy) CDC6600 (parallel) and 7600 (pipelined) and Cray 1 (vector) 1976.
- 1970s-1980s – clustered minicomputers, DEC, IBM, Tandem
- 1990s SGI = NUMA
- 1994 Beowulf clusters
  - Beowulf was a 1994 NASA project to build a gigaflop for less than $50k.
  - Managed a 16-node 486 cluster for $40k.

- (comparison – raspberry pi 2 gives you a gigaflop for \$35)
- D.J. Becker and T. Sterling and D Savarese and J.E. Dorband and U.A. Ranawak and C.V. Packer "BEOWULF: A parallel workstation for scientific computation"
- Usually cheap headless off-the-shelf computers connected by Ethernet. Commodity cluster.

# Commodity Cluster Notes

- Commodity Cluster – a large number of off-the-shelf machines hooked up via cheap off-the-shelf network (usually Ethernet)

- As opposed to a custom cluster that will be custom hardware and custom fast network.

# Are all supercomputers clusters?

- Back in early 2000s was much debate
- *High Performance Computing: Crays, Clusters, and Centers. What Next?* by Gordon Bell (DEC/MS) and Jim Gray (MS), 2001. They propose that everything was converging to all clusters, all the time for all supercomputer needs.
- In a fairly strongly worded rebuttal, Dongarra, Sterling, Simon and Strohmaier ( *High performance computing: Clusters, constellations, MPPs, and future directions*

(2003)) said that no, the world is more complex than that and commodity clusters are not the end word.

- Currently (2025) the top list is mostly commodity clusters
- Although maybe you would not consider a $50,000 GPU to be commodity

# Supercomputer Features?

- From Dongarra et al.
- Is it clustered? meaning identical grouped together
  c for commodity, m monolithic
- How is the namespace organized?
  distributed means local variables on one node not visible on other
  d distributed, s shared, c cache coherent
- How is the parallelism?

t for multithreading, v for vector, c communicating sequential

s systolic, w for VLIW, h for producer/consumer, p parallel process

- How does it handle latency/locality?

  c caches, v vectors, t multithreaded, m for processor in memory.

  f for prefetching, etc

# Top500 List

# Top500 Historical Categories

Definitions from Dongarra et al.

Cluster – system where inter-node communication is 10-1000 times slower than local memory access. (bell and gray)

- single-cpu = just one serial CPU
- simd = single instruction multiple data
- SMP = symmetric multiprocessing, more than one processor
- MPP = massive parallel programming, one machine with

lots and lots of processors (see SGI)

- cluster = number of nodes outnumbers number of CPUs per node (dominant programming style distributed, MPI)
- constellation = each node has more CPUs than total nodes (dominant programming style would be multithread, OpenMP)

# Top500 Historical Makeup

- In 1994, 20% single-cpu, 50% SMP, 24% MPP,7% SIMD
- In 1997, 0% single-cpu, 43% SMP, 0.5% cluster, 54% MPP, 0.5% SIMD
- in 2002, 40% Constellations, 18% cluster, 40% MPP
- in 2015, 90% cluster, 12% MPP
- in 2025: ?

# Software Concerns

- One OS image or multiple?
- single/smp/mpp = one image
- cluster = each node has own copy of operating system
- Shared memory (just write to memory and all cores can see it, hard work in hardware) vs Message Passing (have to explicitly send data before others see it, hard work in software)

# Heterogeneous Computing

- Unequal nodes
- ARM big/little (Intel Power/Efficiency)
- clusters with GPUs
- xeon-phi (almost like a cluster on a PCIe card)

# Example Top500 Listing

- From the November 2002 list, entry #332
- Location: Orono, ME
- Proc Arch: x86
- Proc Type: Pentium III, 1GHz
- Total cores: 416
- RMax/RPeak: 225/416 GFLOPS
- Power: ???
- Accelerators: None

# UMaine Supercomputer Details

- Located at Target Tech Center (Orono Business Park)
- 208 desktop PIIIs, 100Mb eth admin, 1G Myrinet
- Originally single socket. With that they got #501 on list (briefly on before getting kicked off)
- Populated rest of the sockets, made the list

# Homework 1

- A brief homework for next class!
- Look up the info on the Top500 super computer I assigned to you
- E-mail your results, but also bring the info to class for discussion
- Also watch the Top500 BoF if you have the chance