An Intro to and History of HPC Computing

Max Wyatt, e-Research

Introduction to High-Performance Computing

Part 1: Computing Foundations

Meet Your Instructors

- **Max Wyatt** - e-Research Operations Engineer
- **Liz Ing-Simmons** - e-Research Senior Research Software Engineer
- **James Graham** - Head of Software Engineering

Today's Journey

From your laptop to supercomputers

By the end of this workshop, you'll understand:

- What makes a computer tick
- When and why to use HPC
- How to work with HPC systems

A Brief History of Computing

From human calculators to digital machines

- **Before computers**: Slide rules, mechanical calculators, human “computers”
- **Katherine Johnson**: Calculated space flight trajectories by hand
- **1940s-1950s**: First electronic computers (room-sized!)
- **Today**: Computers everywhere, powering all modern research

What is a Computer?

A programmable machine that follows instructions

Modern computers excel at:

- Fast, accurate calculations
- Processing large amounts of data
- Running complex simulations
- Automating repetitive tasks

Computer Limitations

What computers still can’t do:

- Know what you *actually* wanted
- Work without clear instructions
- Understand context like humans do

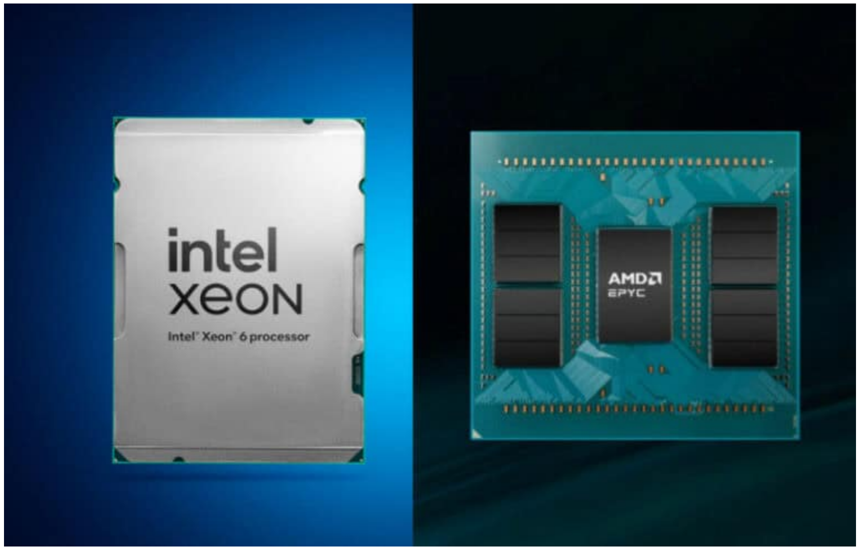
Even AI has these limitations - it needs training data and still makes assumptions

Part 2: Computer Components

Core Computer Components

The essential parts every computer needs:

- **CPU (Central Processing Unit)**: The “brain” - executes instructions
- **Memory (RAM)**: Short-term storage for active work (*volatile*)
- **Storage**: Long-term storage for files and programs (*persistent*)
- **Motherboard**: Connects everything together



Server-grade CPUs

Specialized Components

Additional parts for specific tasks:

- **GPU (Graphics Processing Unit)**: Parallel processing powerhouse
- **Network Interface**: Connects to other computers
- **Operating System**: Manages all the hardware and software

Activity: Share your laptop specs with a neighbor - what operating system, CPU, and RAM do you have?

CPU vs GPU - When to Use Each

CPU: Complex tasks, few at a time

- General-purpose computing
- Complex decision-making
- Sequential processing

GPU: Simple tasks, many at once

- Graphics rendering
- Machine learning training
- Parallel data processing

Activity: Look up whether your research workflows could benefit from GPU acceleration

Part 3: From Single Computers to HPC

What is High-Performance Computing?

When one computer isn't enough…

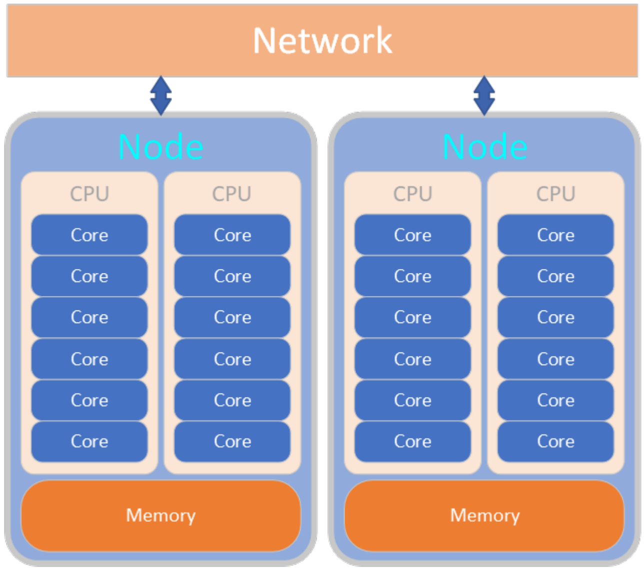
HPC = Many computers working together as one powerful system

- Handles problems too big for single computers
- Used in science, engineering, and research
- Connected by high-speed networks

HPC System Architecture

Four types of nodes (computers) in an HPC cluster:

- **Login nodes**: Where you connect and prepare work
- **Compute nodes**: Where the actual calculations happen
- **Storage nodes**: Where data is stored
- **Management nodes**: Keep everything running



HPC Node Diagram

Activity: Find examples of UK HPC systems (hint: King's has CREATE!)

Why Parallel Computing?

The power of working together

- **Problem**: Some tasks are too big or slow for one computer
- **Solution**: Split work into smaller pieces
- **Benefit**: Multiple processors work simultaneously
- **Result**: Faster completion, bigger problems solved

What Benefits from Parallel Computing?

Perfect for parallel processing:

- Large dataset analysis
- Scientific simulations
- Machine learning training
- Image/video processing

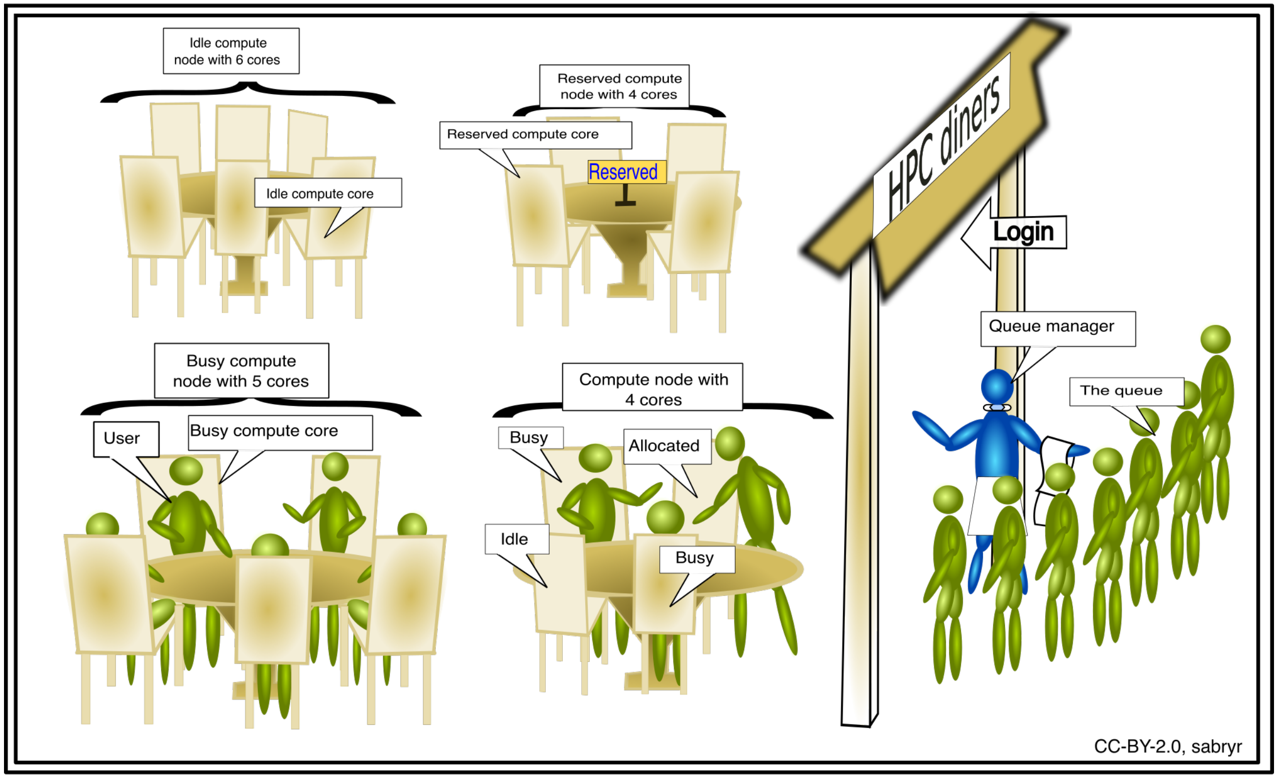
Activity: Discuss with your table - what workflows might you run in parallel?

Part 4: Working with HPC Systems

The Job Scheduler - Your HPC Manager

Think of it like a restaurant queue manager

- **You**: Submit a “job” (your computational task)
- **Scheduler**: Manages the queue and assigns resources
- **System**: Runs your job when resources are available



Restaurant Queue Manager

We’ll use Slurm scheduler in this workshop

Jobs and Job Scripts

A job = A unit of work for the HPC system

A job script = Instructions telling the system:

- What program to run
- How much memory/CPU needed
- How long it will take
- Where to save results

Think of it as a recipe for your computation

Software Management

Three ways to manage software on HPC:

Modules: Load/unload software packages

- `module spider python` - search for software

Containers: Portable, complete environments

- Everything packaged together

Virtual Environments: Isolated Python setups

- Keep project dependencies separate

Activity: Think about what software packages you would like to use on the HPC. Which of these management approaches would you expect to use to manage them?

HPC Best Practices & Etiquette

Be a good HPC citizen:

- **Plan ahead**: Estimate resources needed (CPU, memory, time)
- **Test small**: Use small datasets first
- **Monitor jobs**: Check progress with `squeue`, `sacct`
- **Clean up**: Remove unnecessary files
- **Share fairly**: Don’t monopolize resources
- **Ask for help**: Contact support when stuck

Part 5: Wrap-up

Key Terms to Remember

Essential HPC vocabulary:

- **Node**: Individual computer in the cluster
- **Core**: Processing unit within a CPU
- **Job**: Your computational task
- **Queue**: Waiting line for jobs
- **Scheduler**: Resource manager (like Slurm)
- **Cluster**: Group of connected computers

What's Next?

Your HPC journey continues with:

- Hands-on CREATE system access
- Writing your first job scripts
- Running real computational tasks
- Exploring software modules

Resources:

- CREATE documentation
- Institutional HPC support
- Community forums and guides

References & Acknowledgments

The material in this course was inspired by / based on the following resources

- CREATE documentation
- EPCC Introduction to High-Performance Computing
- A previous iteration of this course developed at Maudsley BRC