Advanced Computing For Engineering and Scientific Software At Scale

Jacob Merson

December 6, 2024

Course Logistics

- Class Schedule: Mondays and Thursday, 2:00 PM - 4PM
- Office Hours: TBD (after class?), or by appointment
- Course Website: LMS
- Communication: Zulip channel for discussions and announcements

`https://computing-at-scale.zulipchat.com/join/edstxivbgnmyzobuljlriyoy/`

About My Research

- Develop tools for multiscale and multiphysics simulations.
- Target high performance computing.
- My lab has run simulations on some of the largest supercomputers in the world! Currently we have allocations on Frontier, Perlmutter, Aurora, and Empire AI.
- Applications of my lab's software include:
  - Multiphysics coupling of fusion reactor simulations
  - Multiscale modeling of biological materials
  - Multiscale modeling of microelectronic devices

Why are we here?

- Writing and designing research software is difficult, requires many skills that are not typically emphasized in engineering and science classes.
- Our goal is to develop these skills and learn how to write software that is:
  - Correct
  - Maintainable
  - Efficient
  - Can operate on large heterogeneous computing systems.
- What's your superpower?
  - Trust that your research is correct.
  - Capability to make impact through software.

What you do?

This course is a project based course and significant engagement is expected.

- Write, document, and test code
- Read and review code
- Utilize standard tooling
- Discuss and present about state of the art software development practices

Evaluation

- 10% Class Participartion (in-class exercises, discussions, quizzes, attendance)
- 50 % Homework
  - 30% implementation, documentation, testing, tooling
  - 20% code review (performing and responding/updating code)
- 40% Final Project

In-Class

- In-class exercies and hands-on activities.
- Discussions about software design, testing, and tooling.
- Short quizzes to check understanding.

Homeworks

- Implementation (IMPL): You will write code to perform a specific task, including documentation and testing. There is no “right” way to do it. But, you will be expected to explain your choices in comments in-class and during code reviews.
- Code Review (CR): After each assignment, you will review two classmates implementations and provide feedback through github pull requests. You will also be expected to respond to the feedback and update your code, documentation, tests as needed.
- Tooling (TOOL): Tooling assignments will focus on standard tools that are used in C++ and HPC software development.

Term Project

- Contribute to a piece of research software.
- Coordinate with your advisor (or Prof. Merson) to identify a project.
- You will be expected to utilize the tools and techniques learned in class.
- Can work together on a large project, but individual contributions must be clear.
- 1 page proposal due 1/27.
- 1-2 page progress report due 3/10.
- Final report due on the last day of classes.
- Presentation during finals week.

Note

- Aim for the course content to be useful to your research.
- We may decide to adjust the schedule, assignments, or topics to better meet the needs of the class.
- Let's learn a bit about eachother.

Class Discussion

- Who are you? Name, Major, Year, Research Advisor
- What are you currently doing for research?
- What do you hope to get out of this class?
- What software challenges have you faced in your research?

Class Discussion (cont.)

Rate your level of experience on a scale of 1-5 (1 being no experience, 5 being expert):

1. Version control
2. Testing
3. Continuous integration/continuous deployment
4. High performance computing
5. GPU programming
6. Distributed computing
7. Containers
8. Cloud computing
9. Documentation
10. Licensing
11. Code review
12. Profiling
13. Debugging
14. Optimization
15. C++
16. C/Fortran
17. Python, Matlab, Julia, etc.

Useful Books

- B. Stroustrup, A tour of c++, 2nd edition. Boston, MA: Addison-Wesley, 2018.
- D. Vandevoorde, N. M. Josuttis, and D. Gregor, C++ templates: the complete guide, Second edition. Boston: Addison-Wesley, 2018.
- C. Scott, Professional CMake: A Practical Guide.
- M. Reddy, API design for C++. Burlington, MA: Morgan Kaufmann, 2011.
- S. Meyers, Effective modern C++: 42 specific ways to improve your use of C++11 and C++14, First edition. Beijing [?]; Sebastopol, CA: O'Reilly Media, 2014.

Other Resources

- cppcon talks https://www.youtube.com/playlist?list=PLHTh1InhhwT4TJaHBVWzvBOYhp27UO7mI
- https://en.cppreference.com/w/
- https://godbolt.org/
- https://cppinsights.io/
- https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines
- C++ Weekly https://www.youtube.com/playlist?list=PLs3KjaCtOwSZ2tbuV1hx8Xz-rFZTan2J1
- C++ Standard https://isocpp.org/std/the-standard https://eel.is/c++draft/
- CSCI-4320, Parallel Computing Course taught by Prof. Carothers in spring semester.

Syllabus

- Review Syllabus / Course Policy / Schedule

Hands On

Operating Systems

In this class we will make use of unix based systems. In particular, we will focus on linux as it is the most commonly used operating system for high performance computing.

To get started, we need to install some tools.

- a linux / unix based operating system
- a terminal emulator
- a text editor (vim, vscode, etc.)
- a version control system (git)
- CMake
- a C++ compiler (gcc, clang)
- a debugger (gdb, lldb)
- a documentation generator (doxygen)
- a build system (make, ninja)

Step 1: Install Linux

Note: Ubuntu 24.04 is suggested, but any linux distribution will work.

- Windows—WSL2: `https://learn.microsoft.com/en-us/windows/wsl/install`
- Mac—Homebrew: `https://brew.sh/`
- I'm going to demonstrate using a Digital Ocean droplet.

Generating a SSH Key

- > ssh-keygen -t ed25519 -C ”your_email@example.com”

Digital Ocean

- Ubuntu 22.04
- Mount a volume
- .bashrc file
- Basic Linux commands
  - ls, **cd**, mkdir, touch, cp, mv, rm, **cat**, less, grep, find
  - How to get help: (-h, –help, man)

Step 2: install base compilers

```
> sudo apt update
> sudo apt-get install git gcc g++ gfortran gdb
```

Step 3: Spack

```
> git clone -c feature.manyFiles=true --depth=2 https://
  github.com/spack/spack.git -b releases/v0.23
> source spack/share/spack/setup-env.sh
> spack compiler find
> spack mirror add v0.23.0 https://binaries.spack.io/v0
  .23.0
> spack buildcache keys --install --trust
> spack config add modules:default:enable:[tcl]
> spack install cmake doxygen lmod
> echo 'source␣~/spack/share/spack/setup-env.sh' >> ~/.
  bashrc
> echo 'source␣$(spack␣location␣-i␣lmod)/lmod/lmod/init/
  bash' >> ~/.bashrc
```