

# Lecture 25 – Retrospective and Miscellaneous Topics

Prof. Brendan Kochunas

NERS/ENGR 570 - Methods and Practice of Scientific Computing (F22)



COLLEGE OF ENGINEERING  
NUCLEAR ENGINEERING & RADIOLOGICAL SCIENCES  
UNIVERSITY OF MICHIGAN

# Outline

- Retrospective
- Other Resources and Where to Go From Here
- Copyrights & Licensing

# Learning Objectives: By the end of Today's Lecture you should be able to

- (*Value*) appreciate the breadth of material needed to master scientific computing,
  - the method to the madness that is this course
  - and realize how much you learned
- (*Knowledge*) seek out further structured learning opportunities
- (*Skill*) use spack to manage HPC software



# Retrospective

A semester in review

# Course Objectives: Methods *and* Practice

- develop and run software in Linux,
- write code in multiple languages,
- use compilers and Makefiles,
- write their own linear solver
- compile and use third party libraries,
- work in software projects with other individuals,
- develop version controlled software,
- implement automated testing in a software project,
- increase the computational performance of their software
- write code that uses MPI and/or OpenMP parallelism,
- perform simulations on high-performance computing resources,
- debug programs more efficiently

# At the beginning of the semester, the stated Course Objectives...

- Enable students who complete the course to produce software in their research that can eventually grow into high quality software used in:
  - industry
  - national labs
  - the open source community
- What this means is, you'll learn:
  - Best practices in software engineering
  - How to optimize code
  - How to use HPC resources
  - What tools are available
  - Overall become better and more productive programmers and computational scientists

# With these Disclaimers

- Live programming is hard and computers are temperamental... please bear with us.
- Be prepared to learn (our expectations)
  - ...how to figure things out for yourself.
    - This is an invaluable skill as a researcher and computational scientist.
  - ...how to program in different languages.
    - You'll have assignments in C/C++ and Fortran.
    - None of your grade will be directly based on code written in MATLAB.
  - ...a lot about a lot of things
    - Several of the topics deserve their own semester long course, and its our job to condense this into a lecture (or 2) and give you an overview, but some depth as well.
    - By the end you'll each be a "Jill" or "Jack" of "all trades", but a "master of none"
- There is no silver bullet. This is not magic. You'll learn through failure, frustration, hard work.

# What happened (Prof. Kochunas's Perspective)

- Learned “the basics” linux, editors, programming languages, compilers
- Learned some *common applications* in Computational Science and Engineering and libraries that provide them
  - Linear algebra operations, solvers for linear systems, and third party libraries
- Learned how to apply software engineering “best practices” to work as a team and develop a “library”
  - OOD, OOP, Issue tracking, version control, reviews, testing, and lifecycles
- Learned about *computational performance*
- Learned about *parallel computing*
- Learned the tools
- Learned how to define the problem
- Learned how others have solved it
- Learned and Practiced how to solve it yourselves
- Learned how to make it fast
- Learned how to think about writing code to run on more than one processor



# Adapted on the fly...

- Some exercises did not work so well
- Prof. Kochunas experienced some technical difficulties (but hopefully not too much)
  - Turns out the bug was between the keyboard and the chair
- Revised schedule (repeatedly) to allow for more thorough understanding of a few topics, rather than speeding through
  - Extended assignment deadlines.
- Provided ample opportunity for extra-credit to explore some topics more deeply.

# Bottom Line

- Hope that you learned something
- Hope that it was fun
- Hope that you felt assignments reflected course material
- Hope that you felt I facilitated all these things.



# Other Resources

Continued Learning

# ACCESS (NSF): <https://access-ci.org/>

- Replaces longtime former program XSEDE.
- First point of contact is UM's Campus Champion: Brock Palen @ ARC-TS
- Resources
  - Expanse / Expanse GPU (San Diego Supercomputing Center)
  - Stampede2 (Texas Advanced Computing Center)
  - Brides 2 (Pittsburgh Supercomputing Center)
  - Delta (UIUC National Center for Supercomputing Applications)
  - Darwin (University of Delaware)
- Allocations: usually have least requirements, open to grad students

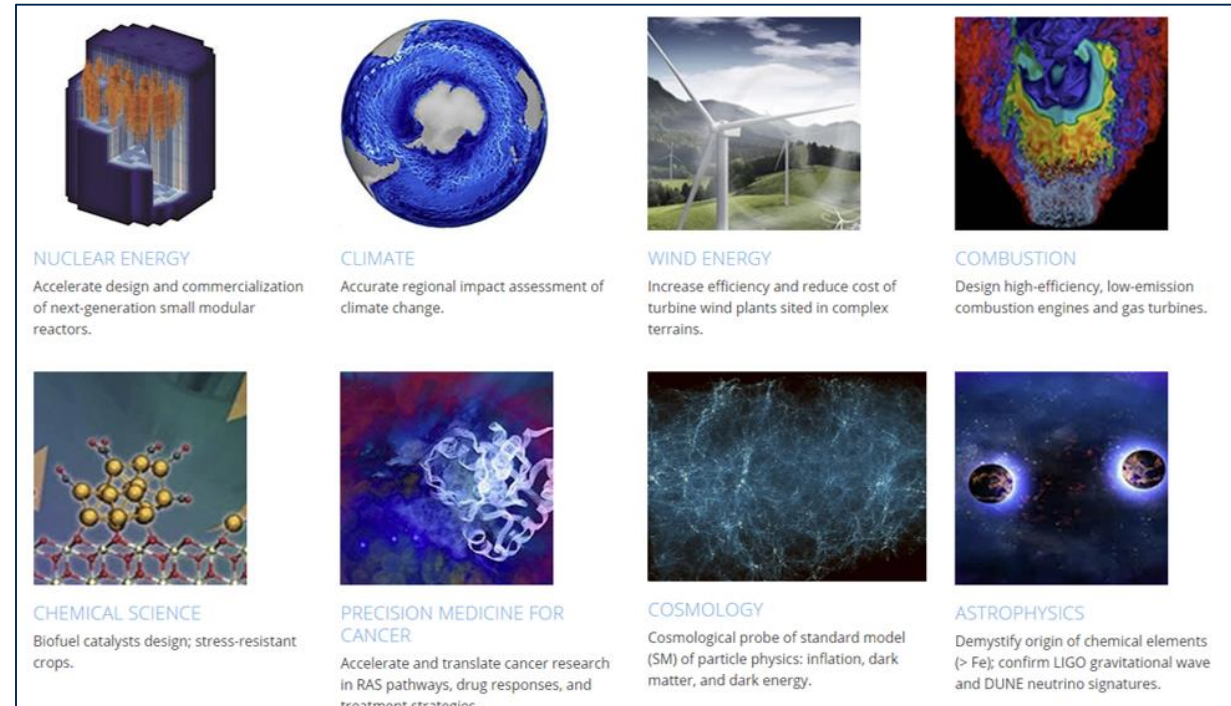
Allocation	Credit Threshold
<u>Explore ACCESS</u>	400,000
<u>Discover ACCESS</u>	1,500,000
<u>Accelerate ACCESS</u>	3,000,000
<u>Maximize ACCESS</u>	Not awarded in credits.

# Exascale Computing Project: <https://www.exascaleproject.org/>



(In its final year—expect a new program in a year or 2)

- Sponsored by US DOE Office of Science and National Nuclear Security Administration
- Focus Areas
  - Application Development
  - Software Technology
  - Hardware & Integration
- Resources
  - Training and software ecosystems



# IDEAs Project: <https://ideas-productivity.org/>

- Mission:
  - “The IDEAS Project has specific use case requirements, an ambitious effort to improve compatibility and usability of important DOE libraries, improve the practices, processes, and tools for scientific software development, and improve community knowledge.”
- Focus Areas:
  - eXtreme Scale Development Kit (xSDK)
    - Portable, TriBITS based build environment for common software stack
  - HowTo
    - Guides for Software Engineering in Scientific Software
  - Outreach
    - Developing and executing training for DOE/ASCR computing facilities
- Now a part of ECP
- Tutorials/Webinars: Best Practices for HPC Software Developers



# Better Scientific Software ([bssw.io](https://bssw.io))

- Intro to CSE and HPC
- Numerous Webinars (These are very good)
  - Practical Introduction to Debugging
  - An Introduction to Software Licensing
  - Best Practices for HPC Software Developers Webinar Series
- Largely aggregates material they have sponsored

OLCF: <https://www.olcf.ornl.gov/>

- Resources

- Frontier – Coming soon aiming for first Exaflop machine (AMD CPUs & GPUs)
- Summit – 2<sup>nd</sup> fastest computer in the world
  - IBM Power 9+ 27,648 NVIDIA Volta GPUs have to have code running on GPUs
- Andes – Data Pre/Post Processing and Visualization cluster
  - 8192 cores, 8 GB/core, fast connections to lustre file system

- Allocations

- Director's Discretionary – small award (< 1M core hours), available all the time, like a “start-up” award, up to 1 yr
- ALCC – medium scale (~10M core hours), available once per year, for some dev but mostly production
- INCITE – very large award (> 100M core hours), available once per year, production only. For grand challenge type problems.



ALCF: <https://www.alcf.anl.gov/>

- Resources

- Aurora – Coming Soon (Intel CPU & GPU)
- Polaris – Smaller Aurora for testing & porting (AMD CPU & NVIDIA A100 GPU)
- Theta & ThetaGPU – (Cray/Intel/NVIDIA)
  - 3,240 nodes w/ 64 core processors (207,360 core). Uses second generation Xeon Phi and NVIDIA A100's

- Allocations

- Director's Discretionary – small award (< 1M core hours), available all the time, like a “start-up” award, up to 1 yr
- ALCC – medium scale (~10M core hours), available once per year, for some dev but mostly production
- INCITE – very large award (> 100M core hours), available once per year, production only. For grand challenge type problems.
- Early Science Program (ESP) – Very small, access to new machines during installation and prior to production

NERSC: <https://www.nersc.gov>

- Resources

- Perlmutter – Just came on-line (Cray “Shasta” with AMD Milan and NVIDIA A100s)
- Cori – 2<sup>nd</sup> General Xeon Phi’s (Cray XC40)
  - 63,616 conventional cores, 4 GB/core
  - 9,304 Xeon Phi compute nodes

- Allocations

- DOE Production – CSGF, SBIR, SciDAC, DOE supported
- ALCC - medium scale (~10M core hours), available once per year, for some dev but mostly production
- Director’s award – appears to be an internal award
- Education & Startup – small award (< 50,000 cpu hours), ongoing, 18 month award

# Additional Training

- All of these computing centers do education and outreach
  - Continually holding webinars and training sessions, past sessions are available
  - Some come with certifications you can put on your CV
  - **Highly recommend** you utilize these resources to gain more depth on particular topics covered in this course.
    - <https://www.olcf.ornl.gov/training>
    - <https://www.alcf.anl.gov/training>
    - <http://www.nersc.gov/users/training/>
- Summer Schools
  - ATPESC 2023 (Chicago area-dates TBD) – <https://extremecomputingtraining.anl.gov/>
    - 2-week annual summer school, premier training, must apply, all expenses paid, held in Illinois
    - Videos from past years are online!
    - very intense and most comprehensive
  - IHPCSS (July 9 – 14 2023, USA TBD) - <http://www.ihpcss.org/>
    - 1-week annual summer school, also prestigious, must apply, a lot of expenses (all?) are covered, rotates around the world
    - Applications are open in December

IHPCSS – 2011, South Lake Tahoe



ATPESC – 2013





# Spack

<https://spack.readthedocs.io/en/latest/>



# Copyright & Licensing



# Copyrights and Licensing

- First resource: UM's Office of Tech Transfer <http://techtransfer.umich.edu>
- Copyright
  - *Copyright is a form of protection provided by the laws of the United States to the authors of "original works of authorship." This includes literary, dramatic, musical, artistic, and certain other intellectual works as well as computer software. This protection is available to both published and unpublished works. The Copyright Act generally gives the owner of copyright the exclusive right to conduct and authorize various acts, including reproduction, public performance and making derivative works.*
- Licensing
  - *License Agreements grant third parties the rights under U-M patents or copyrights. University license agreements typically stipulate that the licensee should diligently seek to bring the intellectual property into commercial use for the public good, provide a reasonable return for U-M, and specify limitations to U-M's liability.*






# Getting Information on types of Licenses


- Too Long Didn't Read (TLDR) Legal: <https://tldrlegal.com/>
- Licenses define a lot of ground rules for distribution, modifications, commercial use, liability, trademark, and ability to modify.
  - Important to understand how these license can work together if your software uses other software.
- Common Open Source Licenses:
  - MIT License
  - BSD
  - FreeBSD
  - GPL
  - LGPL
  - Apache
  - Creative Commons



# MIT License

- Developed to encourage faculty to “walk their inventions out the front door, rather than sneak them out the back door”.
- A short, permissive software license. Basically, you can do whatever you want as long as you include the original copyright and license notice in any copy of the software/source. There are many variations of this license in use.

Can	
▶ Commercial Use	
▶ Modify	
▶ Distribute	
▶ Sublicense	
▶ Private Use	

Cannot	
▶ Hold Liable	

Must	
▶ Include Copyright	
▶ Include License	

# MIT License: Full Text





Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:


The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

# Berkeley Software Distribution (BSD)

- Developed at UC Berkeley for Unix operating system
  - Several versions: 1 clause, 2 clause (FreeBSD), 3 clause
- FreeBSD (Most common)
  - The BSD 2-clause license allows you almost unlimited freedom with the software so long as you include the BSD copyright notice in it (found in Fulltext). Many other licenses are influenced by this one.





Can	
▶ Commercial Use	
▶ Modify	
▶ Distribute	
▶ Place Warranty	

Cannot	
▶ Hold Liable	

Must	
▶ Include Copyright	
▶ Include License	

# GNU General Public License (GPL)

- Developed as part GNU project and free software foundation.
- Several versions: v1.0, v2.0, v3.0
- GPL v2.0 (most common)
  - You may copy, distribute and modify the software as long as you track changes/dates in source files. Any modifications to or software including (via compiler) GPL-licensed code must also be made available under the GPL along with build & install instructions.














Can	
▶ Commercial Use	
▶ Modify	
▶ Distribute	
▶ Place Warranty	

Cannot	
▶ Sublicense	
▶ Hold Liable	

Must	
▶ Include Original	
▶ Disclose Source	
▶ Include Copyright	
▶ State Changes	
▶ Include License	

# Apache License

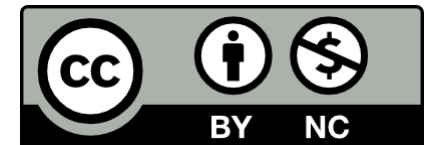
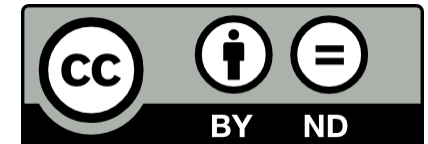
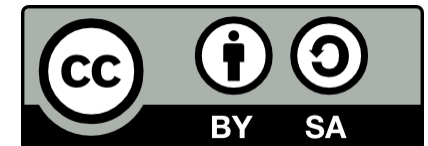
- Developed as a part of Apache Webserver software
  - Several version: 1.0, 2.0
- You can do what you like with the software, as long as you include the required notices. This permissive license contains a patent license from the contributors of the code.

Can	Cannot	Must
<ul style="list-style-type: none"><li>▶ Commercial Use </li><li>▶ Modify </li><li>▶ Distribute </li><li>▶ Sublicense </li><li>▶ Private Use </li><li>▶ Use Patent Claims </li><li>▶ Place Warranty </li></ul>	<ul style="list-style-type: none"><li>▶ Hold Liable </li><li>▶ Use Trademark </li></ul>	<ul style="list-style-type: none"><li>▶ Include Copyright </li><li>▶ Include License </li><li>▶ State Changes </li><li>▶ Include Notice </li></ul>

# Creative Commons

<https://creativecommons.org/about/cclicenses/>

- Essentially 4 Modifiers to basic license/copyright
- Attribution (BY) -- Must Credit Authors
- Share-Alike (SA) -- Any use of the material by a third party must also be licensed under same terms
  - Has a disadvantage in that third parties may see this as “toxic” and not want to use it because they do not want to share they’re material
- No Derivatives (ND) -- Cannot be changed or modified must be in its complete form\*
  - This modifier is generally disliked for being too restrictive. For example, a translation of the original is considered a derivative.
- Non-commercial use (NC) -- Cannot be used by a third party to make money, sell services, design commercial products, etc.
- Licenses are free to use
- Non-exclusive
  - e.g. you could later license software under different terms for commercial use
- Prof. Kochunas’s personal preference would be CC-BY-NC 4.0
  - With some valid instances of CC-BY-ND and CC-BY-ND-NC



# Contributor License Agreement

- Important if you intend to have people outside your institution make contributions.
  - Some examples: OpenMPI, Trilinos (based on OpenMPI)
  - Set's rules for ownership of IP from contributors
- <https://www.open-mpi.org/community/contribute/open-mpi-individual-contributor-agreement.pdf>
- Seem to be losing popularity. Most permissive licenses include clauses for contribution (e.g. Apache v2.0)

# Licenses & Contribution Templates

- Github now lets you initialize a project with a license
- Licensing resource curated by Github: <https://choosealicense.com>
- Contributing guidelines
  - <https://help.github.com/articles/setting-guidelines-for-repository-contributors/>