Parallel scientific computation and visualization

Prabhu Ramachandran

Department of Aerospace Engineering

IIT Bombay

Scientific Workflow Cartoon

Scientific Workflow: big picture

- Computational modeling of physical system
  - Software/numerical representation
- Typical situation
  1. Reproduce older numerical method
  2. New method! (Exciting!)
- Requires iteration
  - Rapid prototyping
  - Correctness/accuracy
  - Decent performance
  - Visualization

Scientific Workflow: phase 1

- Human time is important
  - Quick for you
  - Easy implementation
  - Scientific computation
- Scientific visualization can be complex

Scientific Workflow: phase 2

- Computer time is important
- Parameter sweeps
- Production runs
- Parallelization is important

In this course

- Use Python for this
- Better use of numpy/vectorization
- General tools from Python
    - Command line arguments
    - Better quality code
    - Automation
    - Debugging/profiling
- HPC: Optimization fundamentals
- Parallelization strategies
    - Powerful parallel algorithms
    - compyle
    - mpi4py
- Scientific data visualization
    - Mayavi/VTK/ParaView

Why not *favorite language*?

- Can it be done efficiently with this language?
- Ideas are relevant to other languages too
- We will use Python

What about ME 766?

- The emphasis is different
- High-level
- Generic scientific computing / visualization
- Parallel algorithms

Why me?

- Been doing this for > 20 years
- Developed some of the tools we discuss

Operational overview

- Course details
  - Pre-reqs
  - Target audience
  - Objectives
- Grading structure
- TAs
- Textbooks
- Ground rules

Pre-requisites

- Must be comfortable programming with Python
- Good programming skills
- Exposure to numerical methods
- Ideally work with "computational methods"
- CANNOT pickup Python during the course

Target audience

1. Primary focus is for engineering (non-CSE) and science students.
2. PhD students/research scholars
3. DD/MTech students
4. Advanced UGs

- CSE students are **welcome** but emphasis on application

Objectives

At the end of the course students should be able to:

- Understand and implement vectorized code
- Visualize and animate their data using modern libraries
- Perform parallel computation using multi-core CPUs or GPGPUs.
- Understand how to use MPI for distributed computation
- Make their computational workflows reproducible through automation.
- Build simple user interfaces

Topics

- Introduction to scientific computing with high-level languages
- Efficient use of arrays for vectorized computation
- Using modern tools for improved performance
- Basic debugging and profiling
- Basic serial computing and optimizations
- High-level parallel computing on multi-core CPUs and GPUs:
  - Programming using parallel primitives: map, reduce, and scans.
  - Introduction to other high-level parallel computation packages.
- Brief introduction to distributed computing with MPI.
- Three dimensional plotting of scientific data
  - Using VTK and Mayavi for 3D visualization
  - Introduction to other popular visualization tools like ParaView
- Building simple UIs for scientific computing
- Automation and reproducibility of scientific computing workflows
- Case study of how these can be put together to build a modern, HPC package, and perform reproducible research.

Course specifics

- Timings: Tuesday and Friday: 5:30 pm to 7pm
- No office hours

- Lectures live at AE seminar hall
- Assignment every week
- Moodle for announcements, assignments, discussion: https://moodle.iitb.ac.in

Grading structure

- Rough but overall
- 40 marks: assignments
- 15 marks: end-sem.
- 45 marks for project

TAs

- Navaneet 204010012@iitb.ac.in
- K T Prajwal Prathiksh prajwal_prathiksh@iitb.ac.in

References

There are not many textbooks:

1. Introduction to High Performance Computing for Scientists and Engineers, Georg Hager and Gerhard Wellein, CRC Press, 2010.
2. Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition, https://vtk.org/vtk-textbook/
3. Prefix Sums and Their Applications. Blelloch, Guy E, Sythesis of parallel algorithms, Morgan Kaufmann Publishers Inc., 1990, http://www.cs.cmu.edu/~guyb/papers/Ble93.pdf
4. What Every Programmer Should Know About Memory, Ulrich Drepper, 2007. https://www.akkadia.org/drepper/cpumemory.pdf
5. Multicore and GPU Programming An Integrated Approach, Gerassimos Barlas, Morgan Kaufmann, 2015.

Expectations and rules

- No copying!
- Please sign the honor code
- Please do the assignments by yourself
- Submit them on time. Late penalty applied.
- Any copying will give you full negative marks.
- Do not upload assignments on github/gitlab etc.

Hardware

- Reasonably configured graphics card for the visualization
- Access to some GPU and any multi-core CPU
- Own hardware
- Google colab
- Google cloud/AWS

Summary

- Learn many powerful high-level tools
- Scientific computing
- Visualization
- Parallel programming
- Automation
- Simple UIs
- General techniques and approaches