COSC 594 – 001
3 credit hours

Scientific Computing for Engineers

Web page for the course:
CS 594

Wednesday’s 1:30 – 4:30

- Scientific Computing for Engineers
- Spring 2017 - 3 credits
  - Jack Dongarra
  - with help from:
    - Hartwig Anzt
    - George Bosilca
    - Mark Gates
    - Jakub Kurzak
    - Piotr Luszczek
    - Heike Jagode
    - Stan Tomov

- Class will meet in Room C-233, Claxton Building
To Get Hold of Us

- **Email**: dongarra@icl.utk.edu
  - Room: 203, Claxton
  - Phone: 974-8295
- **Office hours**:
  - Wednesday 11:00 - 1:00, or by appointment
- **TA**: Stephen Richard,
  srichmond2486@gmail.com
- **TA’s Office**: Claxton
Four Major Aspects Of The Course:

1. Start with current trends in high-end computing systems and environments, and continue with a practical short description on parallel programming with MPI, OpenMP, and pthreads. Put together a cluster and experiment.

2. Deal with numerical linear algebra solvers: both direct dense methods and direct and iterative methods for the solution of sparse problems. Algorithmic and practical implementation aspects will be covered.

3. Illustrate the modeling of problems from physics and engineering in terms of partial differential equations (PDEs), and their numerical discretization using finite difference, finite element, and spectral approximation.

4. Various software tools will be surveyed and used. This will include PETSc, Sca/LAPACK, MATLAB, and some tools and techniques for scientific debugging and performance analysis.
Grades Based on:

- 40% on weekly assignments (the lowest grade will be dropped)
- 40% on a written report and presentation (20 pages circa.)
- 20% on a final exam (2 hours) & on class participation.
Homework

- Usually weekly
- Lowest grade will be dropped
- Must be turned in on time (no late assignments)
- Don’t copy someone else’s work.
- Sometimes problems, sometimes programming assignments, sometimes requiring running a program to find the solution.
Homework (continued)

- We expect an analysis and detailed discussion of the results of your efforts.
  - The program itself is not very interesting.
- Programming in C or Fortran.
- Will go over the assignments the week they are due.
- See class web page weekly for details.
Computer Accounts

- For much of the class computing you can use one of our set of computer clusters or systems you have access to.
- If you have an account in the Department you have access to the clusters.
Using the various computer systems from NICS

- **UTK’s Beacon**

  - **Beacon**
    - 48 compute nodes
    - Node: 2 8-core Intel Xeon E5-2670 processors & 4 Intel Xeon Phi™ coprocessors

  ![Beacon Banner](image)
Project

- Topic of general interest to the course.
- The idea is to read three or four papers from the literature (references will be provided)
- Implement the application on the cluster you build
- Synthesize them in terms of a report (~10-15 pages)
- Present your report to class (~30 mins)
- New ideas and extensions are welcome, as well as implementation prototype if needed.
Remarks

◆ Hope for very interactive course
◆ Willing to accept suggestions for changes in content and/or form
Final Exam

- In class
- Will cover the material presented in the course
- ~2 hours
Material

- **Book:**
  

- For each lecture a set of slides will be made available in pdf or html.

- Other reading material will be made available electronically if possible.

- The web site for the course is:
  
Important Place for Software

- Netlib - software repository
  ➢ Go to http://www.netlib.org/
What will we be doing?

- **Learning about:**
  - High-Performance Computing.
  - Parallel Computing
  - Performance Analysis
  - Computational techniques
  - Tools to aid parallel computing.
  - Developing programs in C or Fortran using MPI and OpenMP.
Outline of the Course

1. January 11th: Introduction to Class & High Performance Computing
2. January 18th: Parallel programming paradigms and their performances
3. January 25th: OpenMP
4. February 1st: Architectures and POSIX threads
5. February 8th: MPI Projection and its importance in scientific computing
6. February 15th: MPI Part 2
7. February 22nd: Accelerators
8. March 1st: Performance Modeling
9. March 8th: Discretization of PDEs
10. March 15th: Spring Break
11. March 22rd: Sparse Matrices and Optimized Parallel Implementations
12. March 29th: GPU Computing
13. April 5th: Iterative Methods in Linear Algebra
15. April 19th: Deep Learning
16. April 26th: Dense Linear Algebra
17. May ?th: Final and reports
What you should get out of the course

In depth understanding of:
◆ Why parallel computing is useful.
◆ Understanding of parallel computing hardware options.
◆ Overview of programming models (software) and tools.
◆ Some important parallel applications and the algorithms
◆ Performance analysis and tuning techniques.
Background

- C and/or Fortran programming
- Knowledge of parallel programming
- Some background in numerical computing.
CS 594 – Scientific Computing for Engineers
Assignment #1
January 11th, 2017
Due: January 25th, 2017
Simple Operations

I would like you to implement a version of the following mathematical operations:

- the 2-norm of a vector,

\[ \| x \|_2 = \sqrt{x^T x} = \sqrt{\sum_{i=1}^{n} x_i \cdot x_j} \]

- matrix - vector multiplication,

\[ y = y + A \cdot x \]
\[ y_j = y_j + \sum_{j=1}^{n} A_{ij} \cdot x_j, \text{for } i = 1, ..., n \]

- matrix multiplication

\[ C = C + A \cdot B \]
\[ C_{ij} = C_{ij} + \sum_{k=1}^{n} A_{ik} \cdot B_{kj}, \text{for } i, j = 1, ..., n \]

The point of this assignment is not to write software, but to look at the performance for each of your implementations and try to explain why you are getting the performance you see and what you could do to increase the performance. You should produce a software implementation for each and run some experiments on various computers with different processors. I would like to see a report and analysis of your results; perhaps some plots of your performance data for \( n \) between say 10 and 1000. Please verify and convince me that you are computing the correct results in each case. Let me know what computers you used and how you are getting the performance results as well.

You can find out information on various processors at:
http://www.cpu-world.com/CPUs/index.html


Notes on Assignment #1

The programming that is part of this homework is not the important aspect of the assignment; it is the analysis of what your program is doing. In particular I’m interested in the following, first you should convince me that your program is doing the right thing, that is you should verify that you are computing the “correct” solution. You can do this by using data that when used with you program produces a known result or solution. (Don’t use zeros or ones in your data as that may give an incorrect timing behavior.) Or you can compare your results with a routine from a standard numerical library (that you assume is correct) and compute a “residual”, say something like:

\[ \| \text{your solution – known solution} \| \]

I would like you to analyze your timing results by graphing the rate of execution (ops/sec) as you vary the size of the problem.

I would also like an analysis of the rate you achieve to the theoretical peak performance rate of the processor. You should also describe why the performance you are achieving is so different than the peak.
- Also Intel has a power meter, see:
  https://software.intel.com/en-us/articles/intel-power-gadget-20