COSC 594 –
Scientific Computing for Engineers

Web page for the course:
CS 594 –
Wednesday’s 1:30 – 4:30

- **Scientific Computing for Engineers**
- **Spring 2015 – 3 credits**
  - Jack Dongarra
  - with help from:
    - George Bosilca
    - Jakub Kurzak
    - Piotr Luszczek
    - Heike McCraw
    - Stan Tomov
- **Class will meet in Room C-233, Claxton Building**
To Get Hold of Us

- **Email:** dongarra@eecs.utk.edu
  - Room: 203, Claxton
  - Phone: 974-8295

- **Office hours:**
  - Wednesday 11:00 - 1:00, or by appointment

- **TA:**
- **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 assignment, 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.

◆ If you have an account in the Department you have access to the clusters.

◆ Cluster of PC’s:
Using the various computer systems from NICS

◆ UTK's Natilus, Darter, and Beacon

Darter
◆ 724 compute nodes
  ➢ Node: Two 2.6 GHz 64bit Intel 8-core XEON E5-2600 (Sandy Bridge)

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

Nautilus, an SGI Altix UV 1000 system

It has 1024 cores (Intel Nehalem EX processors), 4 terabytes of global shared memory, and 8 GPUs in a single system image. 2.0 GHz, and a peak performance of 8.2 Teraflops.
Build a Cluster

- Form subgroups
- Each subgroup will get a cluster to put together and experiment with.
  - Intel dual core based with a GPU from Nvidia and AMD
- Put software on and run experiments
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/](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 perhaps OpenMP.
Outline of the Course

1. January 7\textsuperscript{th}: Introduction to Class & High Performance Computing
2. January 14\textsuperscript{th}: Parallel programming paradigms and their performances
3. January 21\textsuperscript{st}: MPI
4. January 28\textsuperscript{th}: OpenMP & Hybrid programming
5. February 4\textsuperscript{th}: MPI
6. February 11\textsuperscript{th}: OpenSHM
7. February 18\textsuperscript{th}: Accelerators
8. February 25\textsuperscript{th}: Performance Modeling
9. March 4\textsuperscript{th}: Projection and its importance in scientific computing
10. March 11\textsuperscript{st}: Discretization of PDEs and Tools for the Parallel Solution
     March 18\textsuperscript{th} - Spring Break
11. March 25\textsuperscript{th}: Sparse Matrices and Optimized Parallel Implementations
12. April 1\textsuperscript{st}: No class?
13. April 9\textsuperscript{th}: Iterative Methods in Linear Algebra
14. April 15\textsuperscript{th}: Dense Linear Algebra
15. April 22\textsuperscript{nd}: Dense Linear Algebra
16. May 4\textsuperscript{th}: Class Final 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 7, 2015
Due: January 21, 2015
Simple Operations

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

- the 2-norm of a vector,
\[ \| x \|_2 = \sqrt{x^T x} = \sqrt{\sum_{i=1}^{n} x_i^2} \]

- matrix - vector multiplication,
\[ y = y + A^* x \]
\[ y_i = y_i + \sum_{j=1}^{n} A_{i,j}^* x_j \text{ for } i = 1, \ldots, n \]

- matrix multiplication
\[ C = C + A^* B \]
\[ C_{i,j} = C_{i,j} + \sum_{k=1}^{n} A_{i,k}^* B_{k,j} \text{ for } i, j = 1, \ldots, n \]

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 your 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:
\[ \| \text{your solution} - \text{known solution} \| \]

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.

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