EECS 594 – 006
Scientific Computing for Engineers

Web page for the course:

EECS 594 – 006
Wednesday’s 1:30 – 4:30

- *Scientific Computing for Engineers*
- *Spring 2009 – 3 credits*
  - Jack Dongarra
  - with help from:
    - George Bosilca
    - Heike Jagode
    - Jakub Kurzak
    - Stan Tomov

- *Class will meet in Room C233, Claxton Building*
To Get Hold of Us

- Email: dongarra@cs.utk.edu
  - Room: 413, Claxton
  - Phone: 974-8295
- Office hours:
  - Wednesday 11:00 - 1:00, or by appointment
- TA: Wesley Bland (w bland@cs.utk.edu)
- Rm 228, Claxton Complex, 974-6321
  - OH: Wednesday's 10am-12am or by request

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 solvers: both iterative for the solution of sparse problems of part II, and direct for dense matrix 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:

- 30% on weekly homework (the lowest homework grade will be dropped)
- 30% on a written report and presentation (20 pages circa.)
- 30% on a final exam (2 hours)
- 10% on class participation.

Homework

- Usually weekly
- Lowest grade will be dropped
- Must be turned in on time (no late homework)
- Don’t copy someone else’s homework.
- 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.

Using the SInRG Clusters

**Battlecat:**
- 8x Intel Core 2 Duo 2.13 GHz
  - 8 Nodes
  - 2 GB RAM
  - Gig-E

**Grig:**
- 128x Intel Xeon 3.2 GHz
  - 64 Nodes
  - 4GB RAM
  - Myrinet 2000

**Mordor:**
- AMD Opteron 8358 2.4GHz
  - 8 nodes (128 cores)x4x (16 cores total)
  - 32G RAM
  - GigE
  - 2x Myricom 10G PCI-E Cards

**Frodo:**
- 128x AMD Opteron 1.4 GHz
  - 64 Nodes
  - 4G RAM
  - Myrinet 2000

**Moria:**
- 96x AMD Opteron 265 (Dual Core) 1.8GHz Processors
  - 24 Nodes
  - 4G RAM
  - Silverstorm 10Gbps Infiniband NIC
  - Mellanox 20Gbps Infiniband NIC
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
- 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 (~20 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:

Other Sources

- Will use material from the internet (manuals, papers)
- Will use a variety of book sources; including
  - Ian Foster
    - Designing and Building Parallel Programs
  - Alices E Koniges
    - Industrial Strength Parallel Computing
  - Jack Dongarra, Iain Duff, Danny Sorensen, Henk van der Vorst
    - Numerical Linear Algebra for High Performance Computers
  - Ananth Gramma et al.
    - Introduction to Parallel Computing
  - Michael Quinn
    - Parallel Programming
  - David E. Culler & Jaswinder Pal Singh
    - Parallel Computer Architecture
  - George Almasi and Allan Gottlieb
    - Highly Parallel Computing
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 using PVM, MPI, HPF, and perhaps OpenMP.
Outline of the Course

1. January 7   Introduction to High Performance Computing
2. January 14  Message Passing
3. January 21 Parallel Programming Paradigms and Performance
4. January 28 Message Passing continued
5. February 4 MPI Dynamic Processing
6. February 11 HPC Architectures
7. February 18 Dense Linear Algebra
8. February 25 Projection and its importance in scientific computing
9. March 4    Dense Linear Algebra, part 2 and Cell processor
10. March 11  Discretization of PDEs and tools for Mesh generation;
            March 18- Spring Break
11. March 25  Sparse matrices
12. April 1   Iterative Methods in Linear Algebra
13. April 8   Iterative Methods in Linear Algebra, part 2
14. April 15  Floating Point Arithmetic, Memory Hierarchy & Cache
15. April 22  Performance Analysis Tools
16. April 29  Class Final reports

What you should get out of the course

In depth understanding of:

♦ When is parallel computing 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
Background

- C and/or Fortran programming
- Knowledge of parallel programming
- Some background in numerical computing.

Computer Accounts

- For much of the class computing you can use one of our set of computer clusters. More on this later
- If you have an account in the Department you have access to the TORC cluster: torc1 through torc8.
- Cluster of PC's:
I would like you to implement a version of the following mathematical operations:

- **vector norm:**
  \[ \|x\|_2 = \sqrt{x^T x} = \sum_{i=1}^{n} x_i \]

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

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

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 improve the performance. You should produce a software implementation for each and run some experiments on various systems, in particular not processors from different, e.g., gigahertz and cluster. I would like to see a report and analysis of your results, perhaps some plots of your performance data for a between any 30 and 100. Please verify and convince me that you are computing the correct results at each case. Let me know what computers you used and how you are getting the performance results as well.

Your TA, [Name], will have a set of these you can use to measure the execution time of your programs. See the course web page for details.

You can find out information on various processors at: