Designing and Building Applications for Extreme Scale Systems CS598

William Gropp

www.cs.illinois.edu/~wgropp

Welcome!

- Who am I?
  - William (Bill) Gropp
  - Professor of Computer Science
  - One of the Creators of the MPI parallel programming system
    - We'll learn more about this in this course
  - Creator of the MPICH implementation of MPI
  - Creator of the PETSc library for scalable solution of PDEs on massively parallel processors
    - We'll talk about the ideas behind this library and similar scalable tools

What is this Course About?

- Making effective use of the most powerful parallel computer systems for problems in science and engineering
- Doing this requires paying attention to *every part* of the parallel system
- It also requires a scientific and rigorous approach to performance

Course Goals

- Understand the sources of performance in modern architectures
- How to establish performance expectations
  - ♦ *Not* precise performance *predictions*
- How to diagnose performance problems
- How to design algorithms and applications to maximize performance potential
- How to exploit parallelism at all levels

Course Organization: Lectures

- Lectures: Will primarily be prerecorded, available at least a few days before class
  - ♦ You will often be expected to view these before the scheduled class
    - We may review these during class time
  - ♦ Lectures may include questions for you to check your comprehension of the material.
  - ♦ Make sure you can apply the ideas before going on.
  - ♦ Lectures are typically 10-40minutes
    - So you will often be assigned two or more lectures before each class

Course Organization: In Class Time

- Will primarily be used for discussion, questions, and in-class assignments
- Weekly assignments, including "machine problems"
- Class is Wednesday and Fridays, 11-12:15, Siebel 1304

Other Logistics

- Course Online Presence
  - ♦ Moodle (start at https://bw-course.ncsa.illinois.edu)
  - ♦ Lectures (video and slide), assignments, discussion, etc. here
- TA
  - ♦ Xiao Chen xchen116@illinois.edu
  - ♦ Office hours TBA

Why This Organization?

- Evidence that conventional lectures are not the best way to instruct,
  - Particularly for a graduate class where backgrounds are varied
- Two other institutions are sharing this class:
  - University of North Dakota
  - University of Wyoming
- The Video lectures
  - Permit you to study the material, including followup and pause to check information, on your schedule
  - You should take advantage of the opportunity to checkout suggested readings and try your own experiments

Evaluation

- Homework every week
  - Includes machine problems as well as paper problems
- Project
  - Students will form small (2-3 person) teams to explore scalability and performance for a problem of interest
  - Project proposals due in March
  - Final report due at end of term

Sample Projects

- Model the performance of a halo exchange and improve a simple implementation
- Modify a benchmark to take chip and node topology into account and compare to modeled performance
- Take an application that you are working on, analyze the I/O performance, and study one approach to improve that performance

More on Projects

- Feel free to propose something on which you are working as the project
  - The best projects are the ones in which you have the most invested
  - The purpose of the project proposal is to ensure that the project is not too large and not too small

My Schedule

- As my students know, I have a busy travel schedule
  - I will lead as many of the class sessions as possible; shortening my trips where possible
  - I am happy to schedule additional time to meet with you individually or in groups – send me email
    - For our remote sites, we can arrange video sessions

Computer Resources

- Computer time will be made available to Illinois students on the Campus cluster (Taub) and for all students on Blue Waters
- Taub is a typical sized institutional cluster
  - Not an extreme scale system, but sufficient for many experiments
  - Some features of Taub make it a good platform to illustrate challenges ☺

Questions

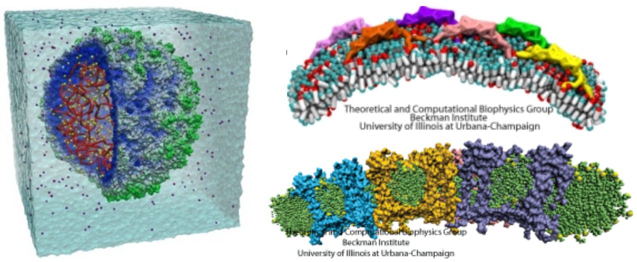
- Write down answers to these questions and turn them in
  - ♦ Do you know MPI (Message Passing Interface)? Do you consider yourself a beginner, intermediate, or expert?
  - ♦ Do you know OpenMP? Do you consider yourself a beginner, intermediate, or expert?
  - ♦ Which programming languages do you use? C? C++? Fortran? Python? Others?

Why Do We Need Extreme Scale Systems?

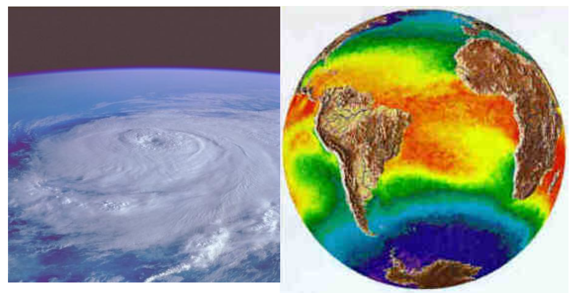
- Many problems cannot be solved exactly
  - ♦ Even the three-body problem can only be solved for a few very special cases
  - ♦ Apparently symmetric situations may not be
    - E.g., evidence supernovae depend on fully 3D, non radially (or axially with rotating star) symmetric solution

Extreme Scale Computing Applies To A Broad Range Of Science And Engineering Disciplines

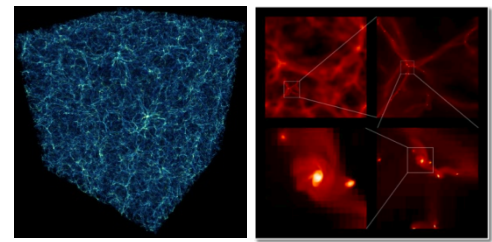
Molecular



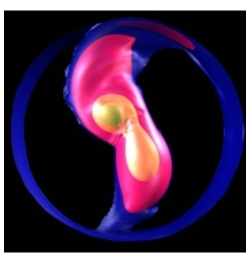
Weather & Climate



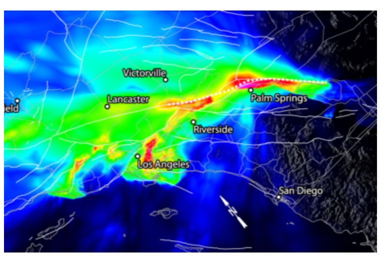
Astrophysics



Astronomy

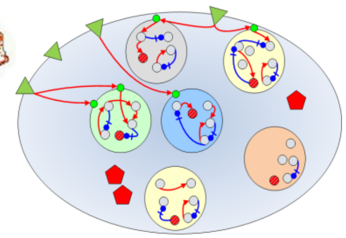


Earth

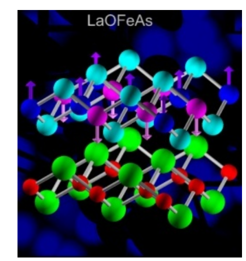


Health

Life Science



Materials



What are the Limits of Extreme Scale Systems?

- PDE Simulations:
  - 10^{10} nodes common (about 2k cubed)
  - Often unstructured mesh
- Molecular dynamics and n-body problems
  - 10^8 – 10^{10} particles common
- Discrete event simulations
  - 10^9 and greater possible

Tentative Course Outline

1. Introduction
2. Extreme scale systems, Simple performance models
3. Benchmarks; performance models with sparse matrix-vector multiply
4. Cache memory; transpose
5. Memory hierarchy; blocking; Cache oblivious models
6. Processor execution

Tentative Course Outline

7. Vectors; Amdahl's law and n1/2
8. Moore's law and Dennard scaling; Multicore and Manycore
9. Threads and programming models
10. OpenMP basics; Loop parallelism
11. Task parallelism; Locks
12. Memory consistency; performance hazards
13. Distributed memory architecture

Tentative Course Outline

14. MPI Basics and performance models
15. Strategies for parallelism
16. MPI nonblocking and asynchronous communication; Progress
17. LogGP performance models
18. MPI Topology and interconnects

Tentative Course Outline

19. MPI Collectives and performance models
20. I/O and Parallel I/O
21. MPI I/O Basics
22. Different I/O organizations
23. MPI RMA and performance models
24. RMA put/get/accumulate operations; Atomic RMW

Tentative Course Outline

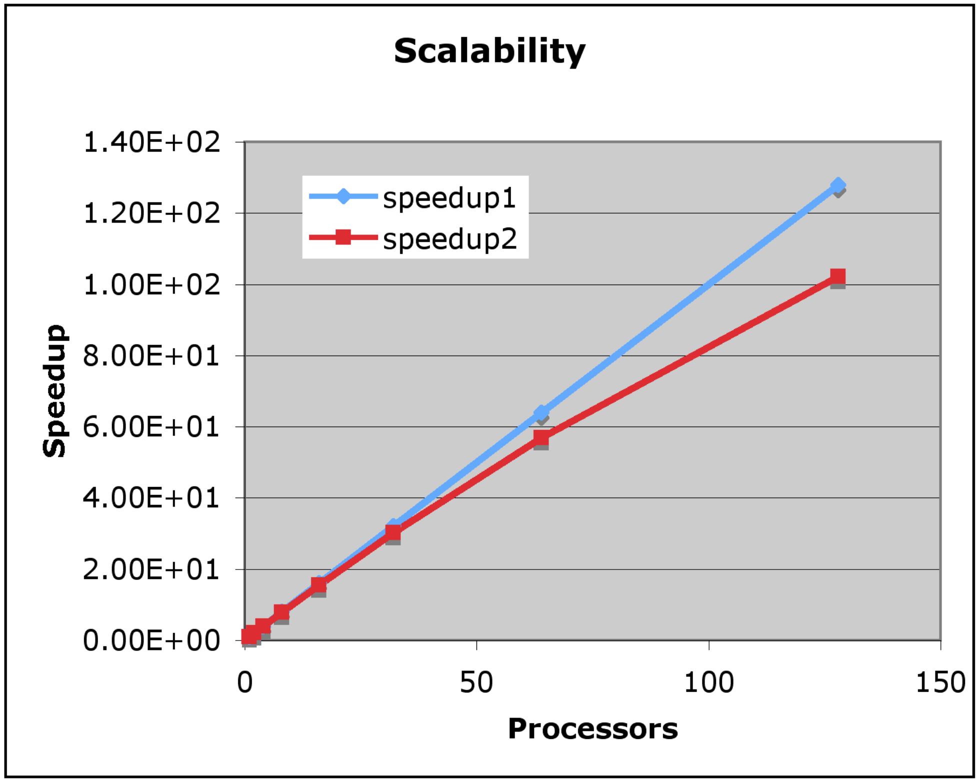
25. More on scaling; isoefficiency
26. Checkpointing basics
27. In memory checkpointing; fault models; performance models

Common Theme for Course

- Use as simple as possible models of computation (an *execution model*) to gain insight into performance issues
  - ♦ Use both to diagnose issues and to design for performance
- One quick example that we'll revisit – comparing scalable algorithms

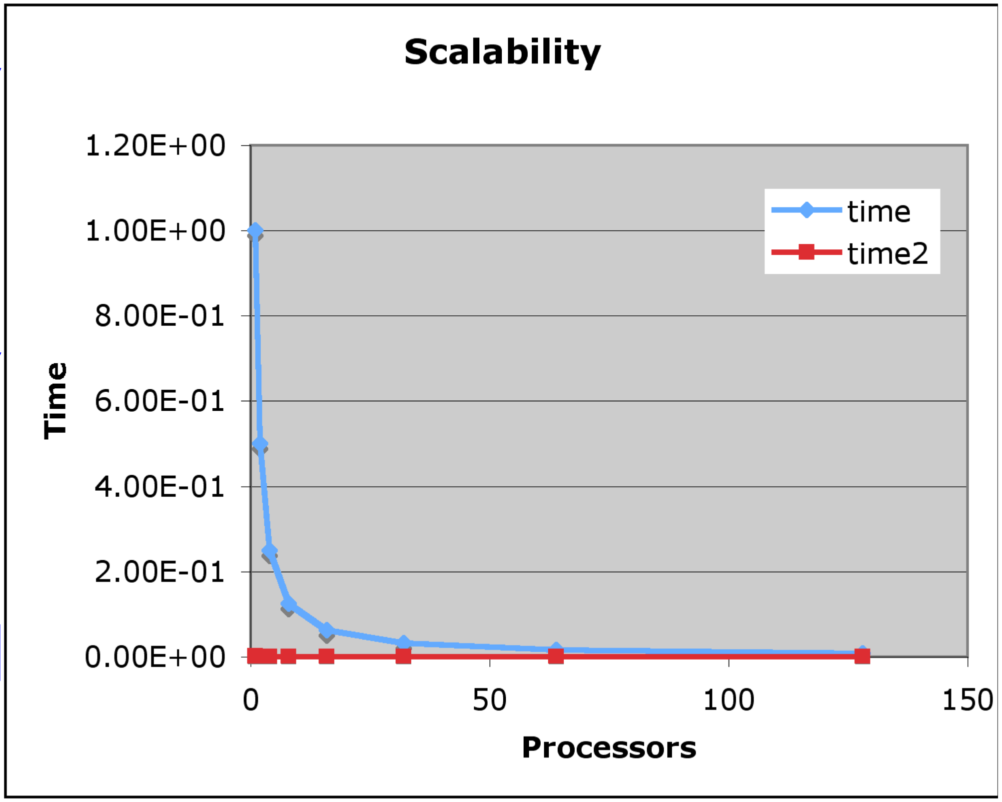
Comparing Scalability

- Two algorithms run in parallel
- Perfect speedup is # of processors
- Algorithm 1 is (nearly) perfect
- Algorithm 2 is fading
- Which is best?



Scalability and Time

- Algorithm 1 has very poor uniprocessor performance
- Algorithm 2 has very good uniprocessor performance
- Not a (too) contrived case – such examples have appeared in papers and proposals



PARALLEL@ILLINOIS

Questions

- Why are you taking the class? Do you have a specific application in mind?
- What is the largest system you've run on?
- What is the longest job you've run (in terms of elapsed time from start to finish)?

How Do We Know if there is a Performance Problem?

- My application scales well!
  - So what!
    - Is it efficient?
    - Making the scalar code more efficient *decreases* scalability
  - How can we *know?*
  - To what do we compare?
- In this class, we will develop techniques to answer this question and to guide in the development of high performance applications