CS267
Applications of Parallel Computers

www.cs.berkeley.edu/~dемmель/cs267_Spr16/
(2016 class on which the XSEDE lectures are based)

https://sites.google.com/a/lbl.gov/cs267-spring-2017/
(2017 class)

Aydin Buluc, James Demmel and Kathy Yelick
aydin@eecs.berkeley.edu,
demmel@eecs.berkeley.edu,
yelick@berkeley.edu
Motivation and Outline of Course

• Why powerful computers must be parallel processors
  Including your laptops and handhelds

• Large Computational Science and Engineering (CSE) problems require powerful computers
  Commercial problems too

• Why writing (fast) parallel programs is hard
  But things are improving

• Structure of the course
Course summary and goals

• Goal: teach grads and advanced undergrads from diverse departments how to use parallel computers:
  - Efficiently – write programs that run fast
  - Productively – minimizing programming effort

• Basics: computer architectures and programming languages

• Beyond basics: common “patterns” used in all programs that need to run fast:
  - Linear algebra, graph algorithms, structured grids,…
  - How to compose these into larger programs

• Tools for debugging correctness, performance

• Guest lectures: climate modeling, astrophysics, …
Students completing class at Berkeley in 2014

- 54 enrolled (40 grad, 9 undergrad, 5 other)
- 28 CS or EECS students, rest from
  
  Applied Math  
  Applied Science & Technology  
  Astrophysics  
  BioPhysics  
  Business Administration  
  Chemical Engineering  
  Chemistry  

  Civil & Environmental Engineering  
  Math  
  Mechanical Engineering  
  Music  
  Nuclear Engineering  
  Physics

- 6 CS or EECS undergrads, 3 double
Students completing class at Berkeley in 2016

- 86 enrolled (>90% grad students)
- 31 (36%) CS, 22 (24%) EECS students, rest from:
  - Bioengineering (5)
  - Civil & Env (6)
  - Mech Eng (6)
  - Math (5)
  - Physics
  - Nuclear Engineering
  - Chemical Engineering
  - Political Science
  - …
Students completing class at Berkeley in 2017

- 74 finished
  - with >90% grad students, 58 (78%) are Ph.D
- 35 (47% some double counting due to double majors) CS,
- 11 (15%) EECS students, rest from:

  Mechanical Eng (7)
  ChemEng and Chem (5)
  Materials (2)
  Applied Math (2)
  Physics (2)
  Nuclear Engineering (2)
  Energy & Resources
  Earth & Planetary Sci.
  Civil & Environmental Eng.
  Applied Sci. & Tech.
CS267 Spring 2016 XSEDE Student Demographics

Degree Level:
- Master's Student: 26%
- Doctoral Student: 28%
- Junior/Senior: 44%
- Freshman/Sophomore: 2%

Major:
- Computer Science: 58%
- Physical Science: 23%
- Engineering: 10%
- Other: 9%
CS267 Spring 2016 XSEDE Student Demographics

Gender

- Male: 93%
- Female: 7%

Race/Ethnicity

- White: 69%
- Asian: 25%
- Black or AA: 2%
- Hispanic or Latino: 2%
- Other: 2%
Students from 9 institutions participated in Spring 2015

- Nebraska: 37
- South Florida: 22
- Houston Clear Lake: 16
- Oklahoma State: 13
- University of Montana: 10
- Tartu: 8
- Auburn Montgomery: 4
- New College: 4
- Fisk: 2

Total: N=116
## Which applications require parallelism?

Analyzed in detail in “Berkeley View” report

1. Finite State Mach.
2. Combinational
3. Graph Traversal
4. Structured Grid
5. Dense Matrix
6. Sparse Matrix
7. Spectral (FFT)
8. Dynamic Prog
9. N-Body
10. MapReduce
11. Backtrack/ B&B
12. Graphical Models
13. Unstructured Grid

[Chart showing parallelism requirements for different applications]

Analyzed in detail in “Berkeley View” report

www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html
What do commercial and CSE applications have in common?

Motif/Dwarf: Common Computational Methods
(\textit{Red Hot} → \textit{Blue Cool})

<table>
<thead>
<tr>
<th></th>
<th>Embed</th>
<th>SPEC</th>
<th>DB</th>
<th>Games</th>
<th>ML</th>
<th>HPC</th>
<th>Health</th>
<th>Image</th>
<th>Speech</th>
<th>Music</th>
<th>Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Finite State Mach.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Combinational</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Graph Traversal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Structured Grid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Dense Matrix</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Sparse Matrix</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Spectral (FFT)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Dynamic Prog</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>N-Body</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>MapReduce</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Backtrack/ B&amp;B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Graphical Models</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Unstructured Grid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Detailed List of Lectures #1 (2016 list)

<table>
<thead>
<tr>
<th>Lecture 1: Introduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lecture 2: Single Processor Machines: Memory Hierarchies and Processor Features</td>
</tr>
<tr>
<td>Lecture 3: Parallel Machines and Programming Models</td>
</tr>
<tr>
<td>Lectures 4 and 5: Sources of Parallelism and Locality in Simulation</td>
</tr>
<tr>
<td>Lecture 6a: Shared Memory Programming: Threads and OpenMP</td>
</tr>
<tr>
<td>Lecture 6b: Tricks with Trees</td>
</tr>
<tr>
<td>Lecture 7: Distributed Memory Machines and Programming</td>
</tr>
<tr>
<td>Lecture 8: UPC and UPC++: Partitioned Global Address Space Languages</td>
</tr>
<tr>
<td>Lecture 9: Cloud Computing and Big Data Processing (Shivaram Vankataraman),</td>
</tr>
<tr>
<td>Lecture 10: NERSC, Cori, Knights Landing, and Other Matters (Jack Deslippe)</td>
</tr>
<tr>
<td>Lecture 11: An Introduction to CUDA/OpenCL and Graphics Processors (GPUs), (Forrest Iandola)</td>
</tr>
<tr>
<td>Lecture 12 and 13: Dense Linear Algebra</td>
</tr>
<tr>
<td>Lecture 14: Graph Partitioning</td>
</tr>
<tr>
<td>Lecture 15: Automatic Performance Tuning and Sparse-Matrix-Vector-Multiplication</td>
</tr>
<tr>
<td>Lecture 16a: Class Project Suggestions,</td>
</tr>
<tr>
<td>Lecture 16b: Structured Grids</td>
</tr>
</tbody>
</table>
Lecture 17: Parallel Graph Algorithms
------------------ Spring Break -----------------
Lecture 18: Architecting Parallel Software with Patterns (Kurt Keutzer)
Lecture 19: Fast Fourier Transform
Lecture 20: Climate Modeling (Michael Wehner)
Lecture 21: Scientific Software Ecosystems (Mike Heroux)
Lecture 23: Dynamic Load Balancing
Lecture 24: Accelerated Materials Design through High-throughput First-Principles Calculations and Data Mining (Kristin Persson)
Lecture 25: Hierarchical Methods for the N-Body Problem
Lecture 26: Communication Lower Bounds and Optimal Algorithms
Lecture 27: Big Bang, Big Data, Big Iron: High Performance Computing and the Cosmic Microwave Background Data Analysis (Julian Borrill)
Lecture 28: Big Data and Exascale: A Tale of Two Ecosystems
May 5: Project Poster Presentations
Quiz Information

- Simple questions after each 20-30 minute video ensuring students followed speaker and understood concepts

- Around 20 questions per lecture depending on length

- All questions multiple choice with 4 options and students allowed 3 tries (can be varied via moodle options)

- Most questions had about 10% wrong answer with 2-3 questions per lecture having over 25% wrong answer
HW0 – Describing a Parallel Application

• A short description of self and parallel application that student finds interesting

• Should include a short bio, research interests, goals for the class then a description of parallel application

• Helps group students later for class projects

• Examples of the 2009-2015 descriptions can be found at this link (at the bottom of the page):

  • https://people.eecs.berkeley.edu/~mme/cs267-2016/hw0/index.html
Programming Assignments

Assignment 1: Optimize Matrix Multiplication.
Tests ability to minimize communication, maximize data locality, and exploit parallelism **within a single core**
Shows the difference between highly-optimized code and naïve code, the performance gap, and resources wasted by not using single cores to their full potential

Assignment 2: Parallelize Particle Simulation
Tests ability to write scalable and **efficient shared/distributed-memory** codes.
Teaches recognizing the important of starting from an efficient serial implementation before parallelization

Assignment 3: Parallelize Graph Algorithms for de Novo Genome Assembly
Introduces students to **Partitioned Global Address Space** (PGAS) languages as well as codes with **irregular** data access patterns.
Programming Assignments

- Each assignment has ‘autograder’ code given to students so they can run tests and see potential grade they will receive.

- Submission of files is done to the moodle website for each assignment with instructions for archiving files or submitting particular compiler directives.

- With help from TAs, local instructors will download files and run submissions and update scores on moodle after runs return.
  - For the 2nd and 3rd assignment this usually takes 30+ minutes for scaling studies to finish.
Square matrix multiplication
Simplest code very easy but inefficient

\[
\begin{align*}
&\text{for (i=0;i<n;i++)} \\
&\quad \text{for (j=0;j<n;j++)} \\
&\quad \quad \text{for (k=0;j<n;k++)} \\
&\quad \quad \quad C(i,j) = C(i,j) + A(i,k) \times B(k,j);
\end{align*}
\]

\[n^3 + O(n^2)\] reads/writes altogether

Give students naïve blocked code
HW1 – Tuning Matrix Multiply

- Memory access and caching
- SIMD code and optimization
- Using Library codes where available
HW1 – Tuning Matrix Multiply

• Possible modifications, depending on audience
• Divide assignment into stages, such as
  • Week 1: multilevel cache blocking
  • Week 2: copy, transpose optimizations
  • Week 3: SIMD and final tuning of all parameters
HW2 – Parallel Particle Simulation

- Simplified particle simulation

- Far field forces null outside interaction radius (makes simple O(n) algorithm possible)

- Give O(n^2) algorithm for serial, OpenMP, MPI and CUDA (2nd part)
HW2 – Parallel Particle Simulation

- Introduction to OpenMP, MPI and CUDA (2\textsuperscript{nd} part) calls
- Domain decomposition and minimizing communication
- Locks for bins and avoiding synchronization overheads
Possible modifications, depending on audience
Divide assignment into stages, such as
  - Week 1: O(n) code
  - Week 2: MPI and OpenMP code
  - Week 3: CUDA code
Given a large number of overlapping DNA fragments (sequences of the letters A, G, C and T), sort them so that together they represent the original complete strand of DNA.

Simplified version of real genome sequence assembly code developed by former CS267 TA Evangelos Georganas, which has been used to sequence wheat genome on 18K processors.

Benefits from one-sided communication model of UPC/UPC++ (irregular remote memory accesses are required).
Class project suggestions

• Many kinds of projects
  - Reflects broad scope of field and of students, from many departments

• Need to do one or more of design / program / measure some parallel application / kernel / software tool / hardware

• Can work alone or in teams
  - HW0 posted to help identify possible teammates based on interest

• What you need to do
  - Project proposal by midnight Saturday Mar 24
  - Feedback from instructor over spring break (ongoing conversations)
  - Poster presentation (+ recording short video presentation) on Thursday May 3 (class time, during RRR week)
  - Final report writeups due Monday May 7 at midnight
How to Organize A Project Proposal (1/2)

• Parallelizing/comparing implementations of an Application
• Parallelizing/comparing implementations of a Kernel
• Building /evaluating a parallel software tool
• Evaluating parallel hardware
• What is the list of tasks you will try?
  - Sorted from low-hanging fruit to harder

• What existing tools you will use, compare to?
  - Don’t reinvent wheels, or at least compare to existing wheels to see if yours is better
  - For applications, consider using frameworks like Chombo or PETSC
  - For applications, identify computational and structural patterns you plan to use

• What are your success metrics
  - Get application X up on Edison or Stampede, solve problem Y
  - Get motif Z to run W times faster on GPU
  - Collect data V to evaluate/compare approaches
A few sample CS267 Class Projects

all posters and video presentations at https://people.eecs.berkeley.edu/~dемmel/cs267_Spr16/

• Parallel Finite Element Analysis
• Parallel Stable K-means
• Models of Flooding for Digital Elevation Models
• Multivariate Time series analysis engine
• Coarse-Grained Molecular Dynamics
• Parallel Genetic Minimization Algorithm
• Scalable Structure Learning
• Blocked Time Analysis for Machine Learning

3/22/12
More Prior Projects

1. High-Throughput, Accurate Image Contour Detection
2. Parallel Particle Filters
3. Scaling Content Based Image Retrieval Systems
4. Towards a parallel implementation of the Growing String Method
5. Content based image recognition
6. Faster molecular dynamics, applied to Alzheimer’s Disease
7. Speed recognition through a faster “inference engine”
8. Faster algorithms to tolerate errors in new genome sequencers
9. Faster simulation of marine zooplankton population
10. Parallel multipole-based Poisson-Boltzmann solver
Still more prior projects

1. Parallel Groebner Basis Computation using GASNet
2. Accelerating Mesoscale Molecular Simulation using CUDA and MPI
3. Modeling and simulation of red blood cell light scattering
4. NURBS Evaluation and Rendering
5. Performance Variability in Hadoop's Map Reduce
6. Utilizing Multiple Virtual Machines in Legacy Desktop Applications
7. How Useful are Performance Counters, Really? Profiling Chombo Finite Methods Solver and Parsec Fluids Codes on Nehalem and SiCortex
8. Energy Efficiency of MapReduce
9. Symmetric Eigenvalue Problem: Reduction to Tridiagonal
10. Parallel POPCycle Implementation