CS267
Class Project Suggestions
Spring 2014

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 early during spring break
  – Feedback from instructor over spring break (ongoing conversations)
  – Poster presentation (+ recording short video presentation) on Tuesday May 6 (class time, during RRR week)
  – Final report writeups due Monday May 12 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

How to Organize A Project Proposal (2/2)
• 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 compare to existing wheels to evaluate pros and cons
  – 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, 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
[www.cs.berkeley.edu/~demmel/cs267_Spr09/posters.html](http://www.cs.berkeley.edu/~demmel/cs267_Spr09/posters.html)

- Content based image recognition
  - “Find me other pictures of the person in this picture”
- Faster molecular dynamics, applied to Alzheimer’s Disease
- Better speech recognition through a faster “inference engine”
- Faster algorithms to tolerate errors in new genome sequencers
- Faster simulation of marine zooplankton population
- Sharing cell-phone bandwidth for faster transfers

## More Prior Projects

1. [High-Throughput, Accurate Image Contour Detection](#)
2. [CUDA-based rendering of 3D Minkowski Sums](#)
3. [Parallel Particle Filters](#)
4. [Scaling Content Based Image Retrieval Systems](#)
5. [Towards a parallel implementation of the Growing String Method](#)
6. [Optimization of the Poisson Operator in CHOMBO](#)
7. [Sparse-Matrix-Vector-Multiplication on GPUs](#)
8. [Parallel RI-MP2](#)

## 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](#)
Class Project Suggestions (1/7)

- Pick one (of many) functions from one of the 13 motifs
- Pick a target parallel platform
- Pick a "parallel programming framework," e.g., for dense linear algebra
  - LAPACK – all parallelism in BLAS
  - ScALAPACK – distributed memory using MPI
  - PLASMA – DAG scheduling on multicore
    - Parallel Linear Algebra for Scalable Multi-core Architectures
    - http://icl.cs.utk.edu/plasma/
  - MAGMA – DAG scheduling for heterogeneous platforms
    - Matrix Algebra on GPU and Multicore Architectures
    - http://icl.cs.utk.edu/magma/
  - Cloud
  - FLAME - http://z.cs.utexas.edu/wiki/flame.wiki/Flame
- Design, implement, measure, model and/or compare performance
  - Can be missing entirely on target platform
  - May exist, but with a different programming framework

Class Project Suggestions (2/7)

- Many new algorithmic ideas for sparse linear algebra
- Come to BEBOP meetings (W 12:00 – 1:30, 380 Soda)
- Experiment with SpMV on different architectures
  - Which optimizations are most effective?
- Try to speed up particular matrices of interest
  - Data mining, “bottom solver” from AMR
- Explore tuning space \([x, Ax, \ldots, A^k x]\) kernel
  - Different matrix representations (last slide)
  - New Krylov subspace methods, preconditioning
- Experiment with new frameworks (SPF)

Class Project Suggestions (3/7)

- Proposed by Sherry Li, LBL Staff Scientist
- “Feasibility of Communication-Avoiding Panel Factorization in Sparse LU”
- Based on SuperLU – widely used parallel sparse LU factorization routine
  - Bottleneck: factorization of “small” panel at each step
  - Project (1): instrument code to evaluate potential communication bottleneck, potential for speedup by using “Tall-Skinny LU” (TSLU)
  - Project (2): implement, insert TSLU
Class Project Suggestions (4/7)

• Proposed by Oded Schwartz, ParLab postdoc
• “Automatic parallelization of BFS/DFS algorithms using SEJITS”
• Motivated by common pattern in several optimal D&C algorithms for matrix multiplication
  – Traverse D&C tree by BFS until out of memory for replicating data, then switch to DFS
• Use SEJITS to parallelize Python code exploiting this pattern, apply to various algorithms

Class Project Suggestions (5/7)

• Proposed by Oded Schwartz, ParLab postdoc
• Variety of fast linear algebra algorithms to be paralleled
  – Yuster-Zwick algorithm for sparse matrix multiplication
  – Variety of fast algorithms beyond Strassen: which ones are fast? Parallelizable as well as Strassen?

Class Project Suggestions (6/7)

• Proposed by Derrick Coetzee, ParLab grad student
• “Communication and arithmetic optimal long integer arithmetic”
• Derrick found an implementation of the Shonhage-Strassen integer multiply algorithm that minimizes communication, and would like to collaborate on implementing, measuring it

Class Project Suggestions (7/7)

• “Minimizing the energy of a computation”
• Energy required to perform computation (on a handheld device or supercomputer) is becoming the bottleneck
• Communication (moving data) takes much more energy than arithmetic
• How well do our communication-avoiding algorithms minimize energy?