Center for Scalable Application Development Software (CScADS):

Automatic Performance Tuning Workshop

http://cscads.rice.edu/

Katherine Yelick
LBNL and UC Berkeley
CScADS Goals

- Conduct research leading to software tools and systems to help applications scale to the petascale and beyond
  - Focus on DOE systems at ORNL, ANL and NERSC
  - Focus on systems composed of multicore processors
- Catalyze activities in the computer science community
  - Focus on interactions with systems vendors, application developers, and library designers
  - Includes a series of workshops
- Foster development of new software through support of common software infrastructures and standards
  - E.g., Open64, ROSE (with LLNL), Telescoping Languages, D System, Dyninst, HPCToolkit
CScADS Participants

- Funded by DOE SciDAC program
- Rice University (Lead Institution)
  - John Mellor-Crummey (Lead PI), and Keith Cooper
- Argonne National Laboratory
  - Peter Beckman (Site Director), William Gropp, and Ewing Lusk
- University of California, Berkeley
  - Katherine Yelick
- University of Tennessee, Knoxville
  - Jack Dongarra
- University of Wisconsin, Madison
  - Barton Miller
Performance Engineering Research Institute (PERI)

• Related SciDAC-funded project
• Goal: Performance engineering:
  – Systems are more complicated
  – Applications are more complicated
  • Multi-disciplinary and multi-scale
• PERI approach:
  – Modeling: performance prediction
  – Application engagement: assist in performance engineering
  – Automatic performance tuning: tools to improve performance

IBM BlueGene at LLNL
Cray X3 at ORNL
Beam3D accelerator modeling
Participating Institutions

Lead PI: Bob Lucas

Institutions:
- Argonne National Laboratory
- Lawrence Berkeley National Laboratory
- Lawrence Livermore National Laboratory
- Oak Ridge National Laboratory
- Rice University
- University of California at San Diego
- University of Maryland
- University of North Carolina
- University of Southern California
- University of Tennessee

CScADS

Autotuning '07
History of Autotuning: Need for Search

- Autotuning is the use of search to select from a set of possible code versions.
- Automatic compared to the hand-tuning historically used for libraries (BLAS, FFTs, etc.).
- Example: tile size selection in dense matrix-matrix multiply.
- Several papers on automated tiling (3 nested loops → 6 nested loops).
- Several papers on tile size selection, but not definitive.
History of Autotuning: Use of Recursion

Recursion in Matrix Multiply

$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} A*E + B*G & A*F + B*H \\ C*E + D*G & C*F + D*H \end{bmatrix}$

Recursion in FFTs

- Traditional implementations were non-recursive
- But the recursive formulation gives better locality
History of Autotuning: Portability through C

- Autotuning systems were made portable by using C as code generation language.
- Many optimizations could be performed in C:
  - Tiling, unrolling, software pipelining, data structure transformations, “register allocation” (explicit temps)
  - Although use of a C compiler adds a level of uncertainty into performance
    - Search over compiler flags and code versions
- C is used as the code generation language in FFTW, Atlas, OSKI, and in several research compilers:
  - Inclusion of machine-specific pragmas, intrinsics, and even assembly code may also be permitted.
How much coverage do autotuning systems have?
Phillip Colella’s “Seven dwarfs”

High-end simulation in the physical sciences = 7 numerical methods:

1. Structured Grids (including Adaptive Mesh Refinement)
2. Unstructured Grids
3. Spectral Methods (e.g., FFTs)
4. Dense Linear Algebra
5. Sparse Linear Algebra
6. Particles (including n^2, tree codes, and particle/mesh)
7. Monte Carlo

- defined A dwarf is a pattern of computation and communication
- Dwarfs are well-defined targets from algorithmic, software, and architecture standpoints

*Slide from “Defining Software Requirements for Scientific Computing”, Phillip Colella, 2004*
Dwarf Use (Red Important → Blue Not)

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 Particles
10 MapReduce
11 Backtrack/ B&B
12 Graphical Models
13 Unstructured Grid
Autotuned Library Instances

- Spectral algorithms: Spiral, FFTW, UHFFT,…
- Dense linear algebra: Atlas, PHiPAC
- Sparse linear algebra: OSKI, Sparsity
- Structured grids: In progress (Datta), Sketching (Bodik et al)
Compiler Level Autotuning Example

- GCO (Generic Code Optimizations) infrastructure
  - UT: Haihang You, Jack Dongarra, Shirley Moore, and Keith Seymour

  - Application to Multiresolution Analysis Kernel from MADNESS using a Specialized Code Generator
    - Extract matrix-vector multiplication kernel from doitgen routine
    - Design specific code generator for small matrix-vector multiplication
    - Tune block size and unrolling factor separately for each input size
Automatic Tuning at many levels

- Libraries: generate code without analyzing or transforming source
- Compilers: source-to-source transformations; whole application optimizations

Guidance controls tuning

- Performance models, measurements, annotations, assertions,…

PERI automatic tuning framework
Questions for Discussion

- Compilers: what they can/should do
- Cache oblivious vs. cache-aware
- Can autotuners help get us over the multicore hurdle?
- How much parallelism can autotuners handle? Does search space get too large?
- If cores get simpler, will autotuning be less important
- Performance models: are they predictive enough?
Agenda Days 1 and 2

Day 1 - Monday, July 9
- 8:00 Continental Breakfast

Overview and Libraries 1 (Spectral)
- 8:30 Kathy Yelick, Berkeley/LBNL
- 9:15 Matteo Frigo, Cilk Arts
- 10:00 Break
- 10:30 Jeremy Johnson, Drexel
- 11:15 Franz Franchetti, CMU
- 12:00 Lunch on your own (per diem)

Processors 1
- 13:30 Richard Johnson, NVIDIA
- 14:15 Adrian Tate, Cray
- 15:00 Break
- 15:30 Sam Williams, UC Berkeley/LBNL
- 16:15 Michael Houston, Stanford
- 18:00 Dinner Reception

Day 2 - Tuesday, July 10
- 8:00 Continental Breakfast

Libraries 2 (Dense LA etc)
- 8:30 Paolo Bientenisi, Duke
- 9:15 Nikos Pitsianis, Duke
- 10:00 Break
- 10:30 Dan Terpstra, U. Tennesse
- 11:15 Clint Whaley, UT San Antonio
- 12:00 Lunch on your own

Compilers
- 13:30 Keith Cooper, Rice
- 14:15 Rudi Eigenmann, Purdue
- 15:00 Break
- 15:30 Mary Hall, USC/ISI
- 16:15 Jackie Chame, USC/ISI
- 17:00 Break
- 18:00 Dinner on your own
- 19:30 Experiences, controversies, discussion
Agenda Days 3 and 4

Day 3 - Wednesday, July 11
- 8:00 Continental Breakfast
Libraries 3 (Sparse LA)
- 8:30 Richard Vuduc, LLNL/GeorgiaTech
- 9:15 Kaushik Datta, UC Berkeley
- 10:00 Break
- 10:30 Jim Demmel, UC Berkeley
- 11:15 Experiences, discussion
- 2:00 Lunch on your own
- 13:30 Enjoy the Snowbird area
- 15:00 Break
Processors 2
- 15:30 Mattan Erez, UT Austin
- 16:15 Michael Frank, AMD
- 17:00 Break
- 18:00 Banquet

Day 4 - Thursday, July 12
- 8:00 Continental Breakfast
Tools for Autotuning
- 8:30 Qing Yi, UT San Antonio
- 9:15 Guojing Cong, IBM
- 10:00 Break
- 10:30 Dan Terpstra, U. Tennessee
- 11:15 Yaoqing Gao, IBM
- 12:00 Lunch on your own
Wrapup Discussion
- 13:30 Community building, engaging applications
- 15:00 Break
- 15:30 Informal discussions with specific subgroups
- 17:00 Workshop ends