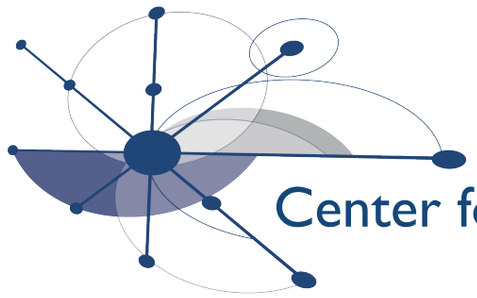


Center for Scalable Application Development Software

Automatic tuning for petascale systems

Keith Cooper (Rice)

Richard Vuduc (Georgia Tech)

Kathy Yelick (UCB/LBNL), Jack Dongarra (UTK)

Proverb

A movement
begins as a **vision**,
runs as a **business**, and
ends as a **racket**.

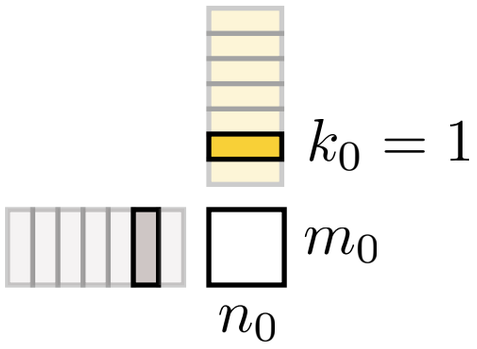
Question: In what stage are we?

“Automatic tuning” – Early seedlings

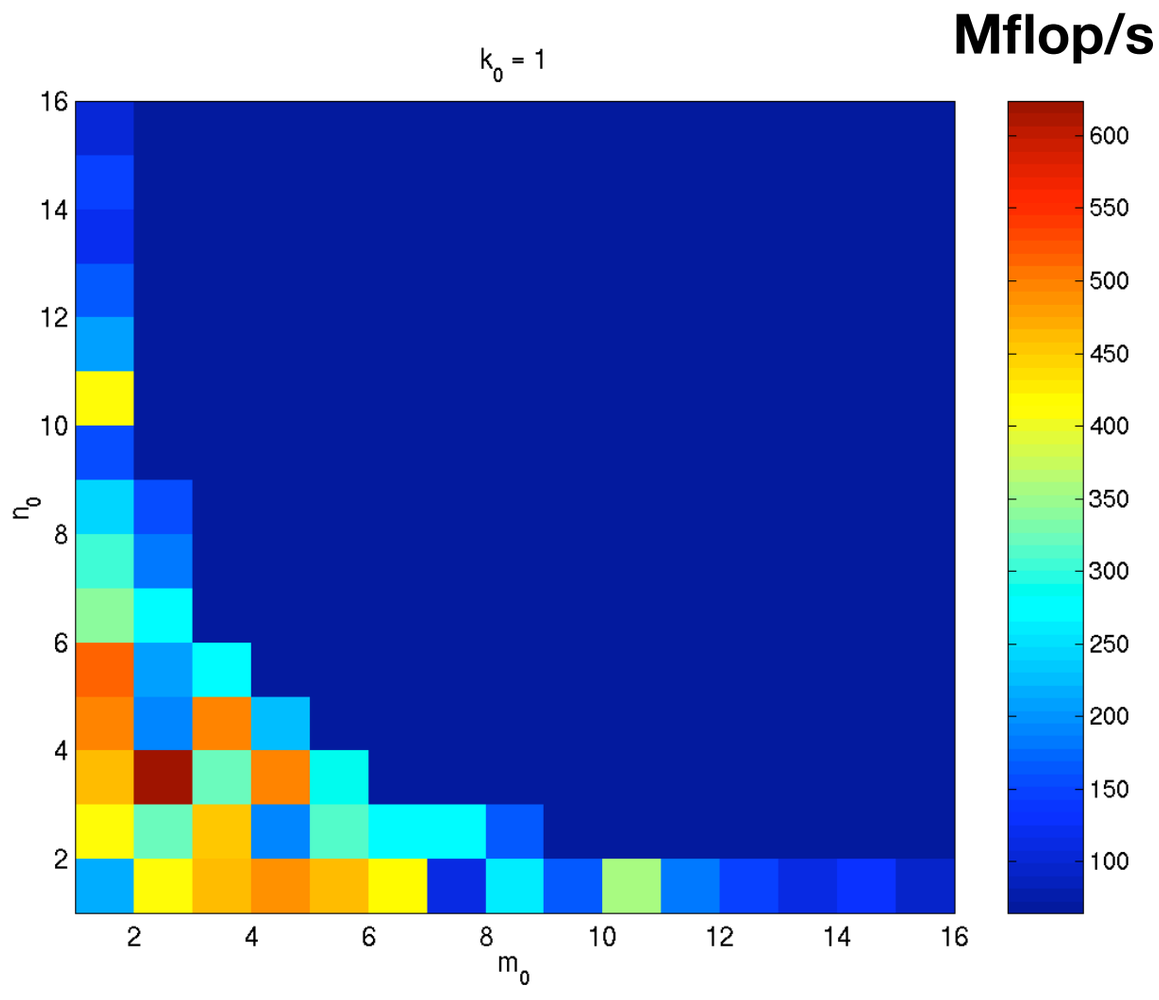
- **Poly-algorithms**: John R. Rice (Purdue)
  - (1969) “A polyalgorithm for the automatic solution of nonlinear equations”
  - (1976) “The algorithm selection problem”
- **Profiling and feedback-directed compilation**
  - (1971) Knuth, “An empirical study of FORTRAN programs”
  - (1982) Graham, et al., gprof
  - (1987) Massalin, “superoptimizer”
  - (1991) Chang, Mahlke, Hwu: “Using profile information to assist classic code optimizations”
- **Code generation from high-level representations**
  - (1989) J. Johnson, R.W. Johnson, D. Rodriguez, R. Tolimieri: “A methodology for designing, modifying, and implementing Fourier Transform algorithms on various architectures.”
  - (1992) M. Covell, C. Myers, A. Oppenheim: “Computer-aided algorithm design and arrangement” (1992)

A notion of autotuning:

- Identify + generate a space of implementations
- Search space to find "best," using models + experiments



- Platform: Sun Ultra IIi
- 16 double regs
- 667 Mflop/s peak
- Unrolled, pipelined inner-kernel
- Sun cc v5.0 compiler



Source: PHiPAC Project at UC Berkeley (1997)

A notion of autotuning:

- Identify + generate a space of implementations
  - "Identify" – What goes into the space?
  - "Generate" – IRs? Infrastructures?
  - How much is automatable? What is composable?
- Search space to find "best," using models + experiments
  - Static vs. dynamic?
  - Limits of models? Composability?
  - How much and what to measure?
- What is "best?" (Metrics of success?)

CScADS Goals

- Conduct research leading to **software tools and systems** that help apps scale to petascale and beyond
- Catalyze activities in computer science
  - Enable **interactions** among vendors, developers
  - Sponsor workshops, create "visions"
- Foster development of new software through support of **common software infrastructures and standards**

CScADS Participants

- Funded by DOE SciDAC Program
- Rice U. (lead): Mellor-Crummey & Cooper
- Argonne: Beckman (site dir.), Gropp, Lusk
- Berkeley: Yelick
- U. Tenn. Knoxville: Dongarra
- U. Wisconsin: Miller

- Acknowledgements: Staff at Rice (Darnell Price), ANL (Lori Swift), Snowbird (Kelly Wilkins)

Format

- "Meeting of minds"
  - Architects, compiler writers, library developers
  - Industry, labs, academia
- Discussion and debate
  - Topic questions, but make up your own
  - **No holds barred – push buttons!**
  - Community building
- Day 1: "Guests," industry, libraries
- Day 2, 3 (half): Compilers, libraries, run-time

Example: DARPA AACE

- "Architecture-Aware Compiler Environment"
  - Build a self-assembling, self-tuning compiler that generates code with peak performance in zero-compilation time on any architecture, including one "it" has never seen before
- **Proposition**: Compilers will never do this.

- Today's autotuning work does/doesn't address the challenges of petascale.
- How do we measure success for tuning? Performance? Productivity?
- What architectures/platforms should we target?
- "Parameter tuning" is the wrong focus for our area, as it suggests only incremental improvements.
- Self-tuned libraries will always outperform compiler-generated code.
- What improvements should we expect from autotuning? From compilers? libraries?
- Simple performance models (e.g., cache-oblivious, simple cores) will be the right models in the future, obviating "search."
- Traditional boundaries between apps, libs, compilers, and OSes are too rigid.
- What issues are we as a community ignoring?
- Common infrastructures?

- Today's autotuning work does/doesn't address the challenges of petascale.
- How do we measure success for tuning? Performance? Productivity?
- What architectures/platforms should we target?
- "Parameter tuning" is the wrong focus for our area, as it suggests only incremental improvements.
- Self-tuned libraries will always outperform compiler-generated code.
- What improvements should we expect from autotuning? From compilers? libraries?
- Simple performance models (e.g., cache-oblivious, simple cores) will be the right models in the future, obviating "search."
- Traditional boundaries between apps, libs, compilers, and OSes are too rigid.
- What issues are we as a community ignoring?
- Common infrastructures?

Who is not here?

- Many recent "autotuning" meetings
  - SIAM Parallel Processing '08 special sessions
  - DOE High Perf. Comp. Sci. Week
    - http://www.hpcsw.org/presentations/workshops/autotuning/
- iWAPT ('06–'08), in Japan
- Others?