TOP TEN QUESTIONS ON AUTOTUNING

David Padua
Department of Computer Science
University of Illinois at Urbana-Champaign
The many faces of autotuning

- Different concepts fall under autotuning.
  - No standard definition
    - Wikipedia: “Auto-Tune is a proprietary audio processor created by ....”

- Library generators, compilers, runtime systems make use of autotuning techniques.
A definition

- **Autotuning** is an automatic process for selecting one out of several possible solutions to a computational problem.

  - **Wikipedia**: In theoretical computer science, a *computational problem* is a mathematical object representing a collection of questions that computers might want to solve. [e.g.] "Given a positive integer $n$, find a nontrivial prime factor of $n$.

- Perhaps too wide a definition because it includes compiler optimizations.
The versions may differ in the
- algorithm (e.g. merge-sort vs. quicksort) or
- implementation (unroll loop $n$ times).

The versions may result from
- transformations (unroll, tile, interchange MMM) or
- pre-specified collection (e.g. sorting algorithms [today, we cannot generate one from the other]).

The versions could be generated by
- programmer (via directives or coding)
- compiler / rewriting system
Search could be

- **Exhaustive**
- Guided by **heuristics**
  - Orthogonal, random, genetic algorithms, dynamic programming, ...
- Guided by **model**
  - Formula → best version (no search)
    - Initial approximation for steepest descend search ...

- Statistical techniques
Evaluation

- **Empirical**
- Based on prediction

Search could be

- Dynamic (e.g. IBM Testarossa, CPO)
- Static (most compilers)

Search could be

- Data independent (e.g. ATLAS, SPIRAL)
- Data dependent
  - Berkeley’s Bebop.
Questions on what to target
Question 1: What library routines to generate?
Should applications be autotuned or only libraries?

- **Multiple levels of abstraction.**
  - From kernel level (BLAS) all the way to applications.
  - The lower the level the widest the applicability, but optimizations are more local.

- Libraries not always cover the bulk of computation.
Question 2: Can library routines be generated with input data representation as a parameter?

- E.g. sparse input could be represented in different formats.
Questions on tools
Question 3: Are there useful programming notation for autotuning codes and library generators?

- ATLAS, FFTW,… were written from scratch in conventional languages.
- But there are many commonalities.
- Is there a notation (language, libraries,… ) that would facilitate development?
- Two approaches: Generators vs. transformers:

Question 4: How to use the compiler?
What should be done by the compiler and what by the generator?

- Need compilers for
  - portability, machine independence.
  - uniform results across machines.
- Some autotuning can be done automatically without programmer’s intervention.
- What should be done by the compiler and what by the generator?
- The more that is done to the compiler, the better.
Search for optimization not new. Typically, exhaustive search with a predictor:
- Unrolling factor
- Interchange

Could bring ideas from autotuning.
- E.g. machine learning …

Most important open question in compilers today.
Question 5 (cont.): How to implement compiler optimizations?

- Some have suggested that machine learning techniques would improve quality of compiler optimization.

- More important: give the compiler room to maneuver/more choices.
  - Higher level notations
  - Directives/black-belt notations?
  - Language-compiler co-design
    - Solution to autoparallelization?
Question 6: How to interact with the programmer? How to debug?

- Multiple versions seem to complicate the problem of debugging.
- Exponential growth if naïve.
Questions on search and optimality
Question 7: Is there a methodology to develop performance models?

- To guide the search, we need models that map solutions to performance (perhaps relative performance).
- Parameters could be algorithmic, or from the code (tile size).
Question 8: Given the code, can we build reliable performance predictors?

- Saves time by avoiding actual execution
- Difficult, especially when performance is data dependent.
  - Can we predict sorting behavior?
Question 9: How to determine what parameters are important for performance?

- To train, or just tune, need to vary observe variation of parameters.
- Can we automatically determine what affects performance?
- In sorting: standard deviation of data, degree of sortedness, others?
- In solvers: Size, degree of diagonal dominance, …
Question 10: Can we know how close to the optimum?

- The space of possibilities is fixed
- Exhaustive search will give us the optimum, but too expensive in many cases.