Tuning for Multiple Objectives: Power versus Performance

SUPER
INSTITUTE FOR SUSTAINED PERFORMANCE, ENERGY, AND RESILIENCE

Prasanna Balaprakash, Paul Hovland (lead), Stefan Wild
Argonne National Laboratory

Jeff Hollingsworth
University of Maryland

Ananta Tiwari
San Diego Supercomputer Center

Abstract

When a single objective, such as execution time, is available, the autotuning search problem can be posed as a numerical optimization problem. However, it is increasingly common to have multiple objectives, such as execution time, energy consumption, resilience to errors, power demands, and memory footprint. When the relative weights or constraints on these objectives are not known at search time, one must pose the autotuning search problem as a multi-objective optimization problem. We describe the optimization framework for search, discuss some of the potential tradeoffs among multiple objectives, and provide empirical evidence that such tradeoffs do exist in practice.

Optimization as Optimization

The problem of empirically optimizing a code can be posed as the mathematical optimization problem

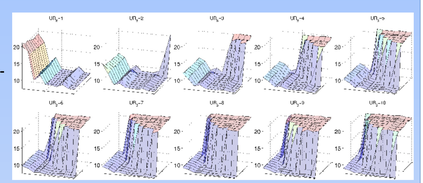
$$\min_{x} \{ f(x) : x = (x_{\mathcal{I}}, x_{\mathcal{B}}, x_{\mathcal{C}}) \in \mathcal{D} \subset \mathbb{R}^n \}$$

THE OBJECTIVE

- A single, real-valued performance objective
- Can capture average, median, quantile (e.g., worst-case) empirical performance
- Often stochastic/noisy (from measurement and/or run)
- Depends on machine and input size (or distribution over inputs)
- *Examples*: run time, failure rate, expected run time

THE DECISIONS

- Binary (compiler type, +examples)
- Integer (unroll factor, register tiling, +examples)
- "Continuous" (algorithmic parameters, internal tolerances)
- Each *x* generates a code variant (through source-to-source/compiler-based transformation, etc.)



THE CONSTRAINTS

- Ensuring feasibility of transformation
- Correctness of output, maximum temperature, etc.

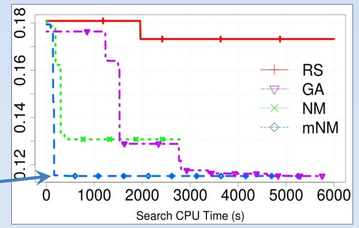
Kernel	Operation	Transformations n_t	n_b			$\lvert\mathcal{D}\rvert$
linear algebra kernels						
ATAX	matrix transpose & vector multiplication	13	UJ, CT, RT, LPM	6	SR, AC, LV, OMP	1.65e+14
DGEMV	scalar, vector & matrix multiplication	38	UJ, CT, RT, LPM	11	SR, AC, LV, OMP	2.73e+30
FDTD4d2d	finite-difference time-domain kernel	25	UJ, CT, RT, LPM	5	SR, AC, LV, OMP	7.06e+24
GEMVER	vector multiplication & matrix addition	18	UJ, CT, RT, LPM	6	SR, AC, LV, OMP	7.26e+17
GESUMMV	scalar, vector, & matrix multiplication	8	UJ, CT, RT, LPM	3	SR, LV, OMP	1.56e+08
HMC	Hessian matrix computation kernel	7	UJ, CT, RT, LPM	8	SR, AC, LV, OMP	1.01e+08
MM	matrix multiplication	10	UJ, CT, RT, LPM	4	SR, AC, LV, OMP	1.83e+12
MVT	matrix vector product & transpose	6	UJ, CT, RT, LPM	6	SR, AC, LV, OMP	1.38e+08
Tensor	tensor matrix multiplication	17	UJ, CT, RT, LPM	3	SR, LV, OMP	5.49e+16
TRMM	triangular matrix operations	20	UJ, CT, RT, LPM	5	SR, LV, OMP	5.33e+19
linear algebra solvers						
BiCG	sub kernel of BiCGStab linear solver	9	UJ, CT, RT, LPM	4	SR, AC, LV, OMP	9.33e+09
LU	LU decomposition	9	UJ, CT, RT, LPM	5	SR, AC, LV, OMP	1.86e+10
stencil codes						
ADI	matrix subtraction, multiplication, & division	16	UJ, CT, RT, LPM	4	SR, AC, LV, OMP	6.05e+15
Jacobi-1d	1-D Jacobi computation	8	UJ, CT, RT, LPM	3	SR, LV, OMP	1.55e+08
Seidel	matrix factorization	12	UJ, CT, RT, LPM	3	SR, LV, OMP	6.86e+11
Stencil3d	3-D stencil computation	24	UJ, CT, RT, LPM	5	SR, AC, LV, OMP	2.35e+23
data mining						
COR	correlation computation	16	UJ, CT, RT, LPM	4	SR, AC, LV, OMP	6.05e+15
COV	covariance computation	20	UJ, CT, RT, LPM	5	SR, AC, LV, OMP	5.33e+19

Search spaces are enormous, examining all possible variants unnecessary and highly inefficient (if not impossible).

GOAL:

Use formulation as a mathematical problem to leverage modern derivative-free optimization algorithms for obtaining approximate minimizers while examining a tiny number of variants.

1.5X speedup evaluating 40 of 10^7 possible variants

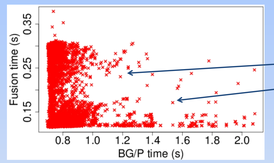


"Can search algorithms save large-scale automatic performance tuning?" Balaprakash, Wild, & Hovland, *IWAPT 2011.*
" Online Adaptive Code Generation and Tuning ." Tiwari & Hollingsworth, *IPDPS 2011.*
"SPAPT (Search Problems in Automatic Performance Tuning)." Balaprakash, Wild, & Norris, *ICCS 2012.*
"An Experimental Study of Global and Local Search Algorithms in Empirical Performance Tuning." Balaprakash, Wild, & Hovland, *VECPAR 2012.*

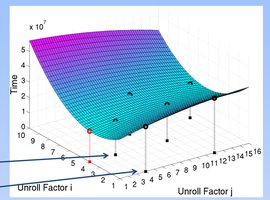
Multi-objective Optimization

Arises when several objectives need to be to optimized simultaneously

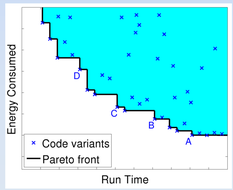
- No weights given a priori
- Sometimes objectives are correlated and satisfied simultaneously; otherwise there are tradeoffs
- Code variants now live both in a decision space and in an objective space



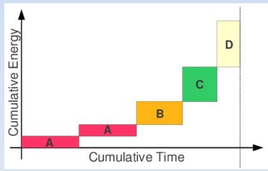
Objective space



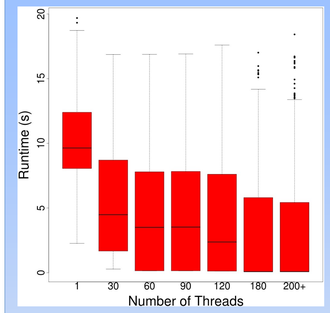
Decision space



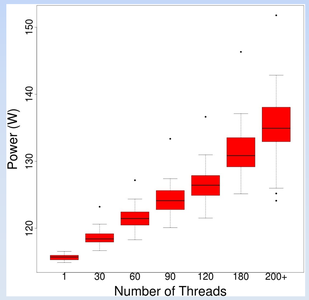
- Code variants for which no other variant is better in all objectives are said to be *nondominated* or *Pareto optimal*
- Pareto optimal variants can be used to optimally schedule jobs on leadership computing facilities given constraints on time or energy

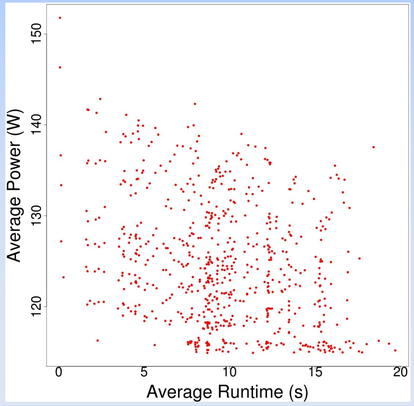


Collecting Objective Metrics



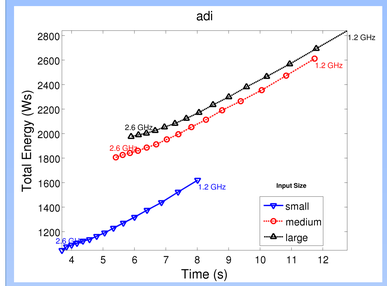
- Properties such as monotonicity can often be observed with respect to some decisions
  - Time decreases in a monotone fashion in number of threads, for many ranges of thread count
  - Power increases in a monotone fashion in number of threads
- Tradeoffs between power and time can be observed for many different tuning spaces
- Finding Pareto optimal points can be made faster by exploiting known properties, such as monotonicity, as well as by discovering latent structure in the objectives

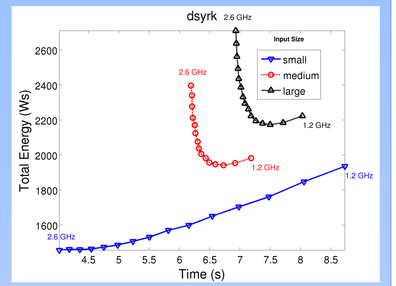


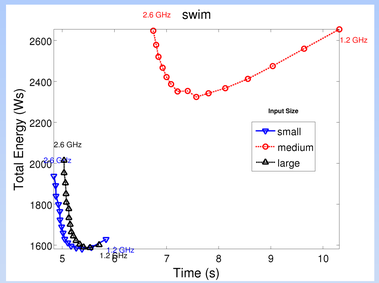


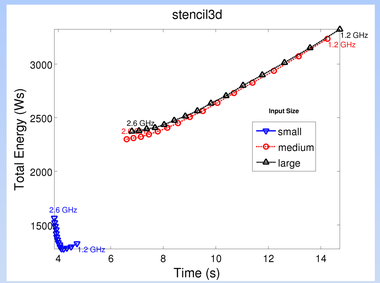
Exploring Tradeoffs in SUPER

Whether tradeoffs between energy and time exist or "race-to-idle" conditions are present depends on computational workloads, architectural features, and the tunable decisions



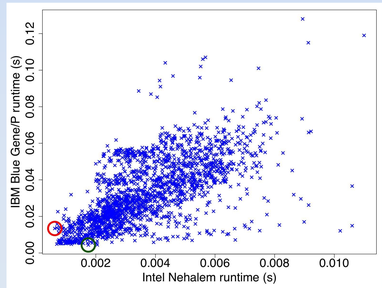




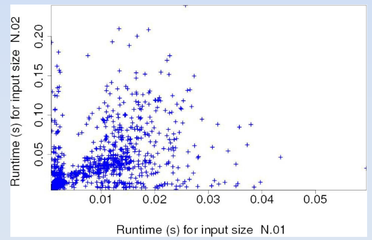


Many other sets of performance and power-based simultaneous objectives are of interest

Architecture



Kernel input size



Future Investigations

- Develop multiobjective optimization algorithms for autotuning search
- Indentify appropriate use cases
- Study other tradeoffs
  - Resilience versus memory footprint
  - Resilience versus execution time
  - Memory footprint versus execution time
  - Memory footprint versus energy

Support for this work was provided through the Scientific Discovery through Advanced Computing (SciDAC) program funded by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research.