Automatically Tuning Performance and Power for GPUs

Jeffrey K. Hollingsworth

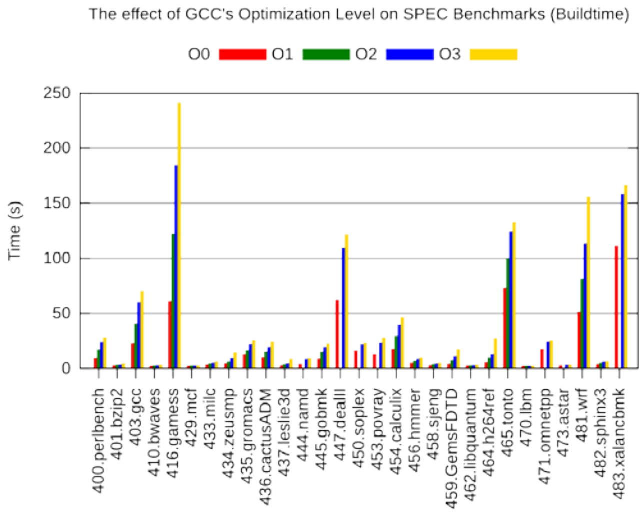
What is Auto-tuning?

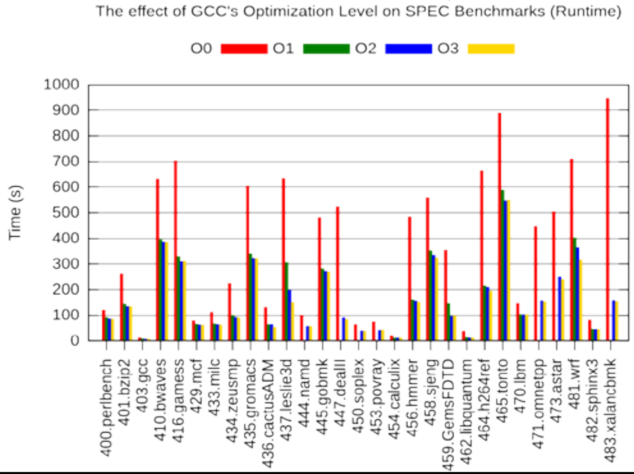
- **Making programs "better" based on empirical feedback from observed runs without programmers having to in the loop.**
- **What to Tune**
  - **Parameters: Application, library, OS, …**
  - **Code: re-compile the program**
  - **Algorithms**
  - **Node or multi-node performance**
- **When to Tune**
  - **Once per architecture/machine/application**
  - **Once per data set**
  - **Once per run of the program**
  - **Continuously during program execution**

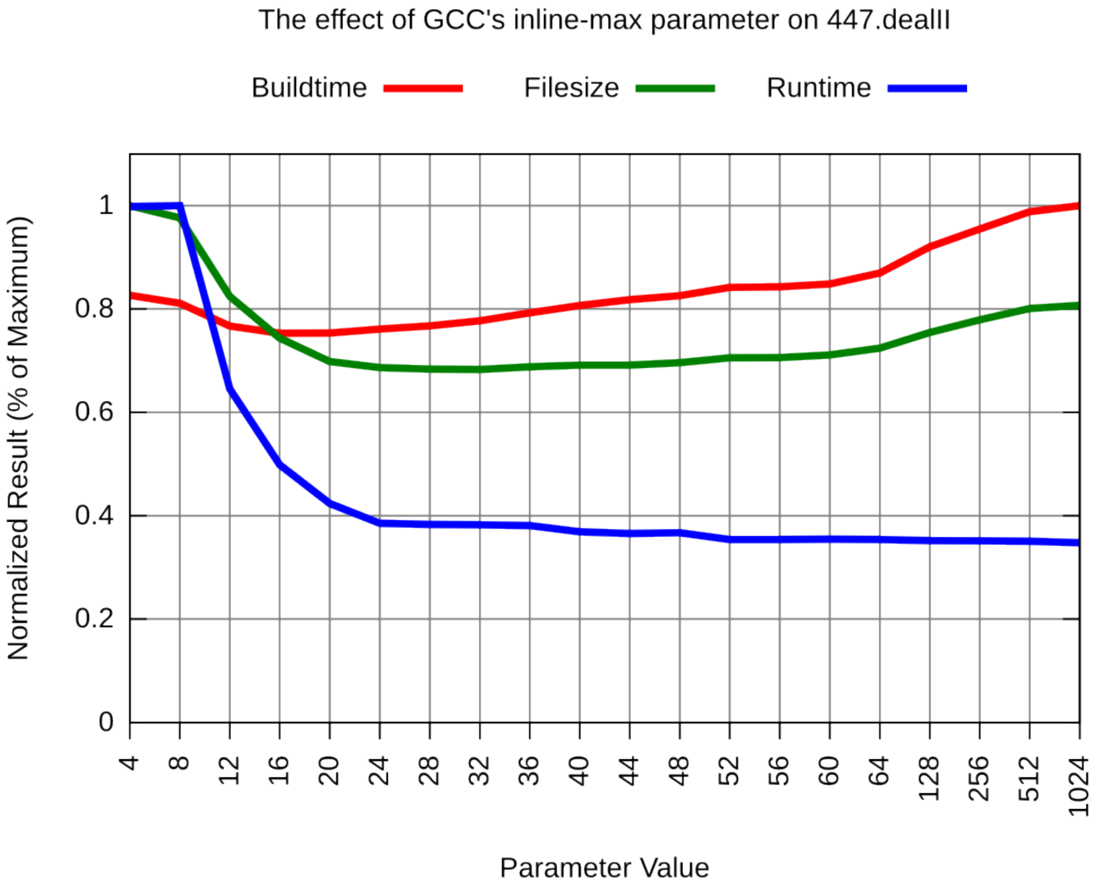
Why Auto-tuning?

- Example, a dense matrix multiple kernel
- Various Options:
  - Original program: 30.1 sec
  - Hand Tuned (by developer): 11.4 sec
  - Auto-tuned of hand-tuned: 15.9 sec
  - Auto-tuned original program: 8.5 sec
- What Happened?
  - Frustration for the person trying to auto-tune!
  - Hand tuning prevented analysis
    - Auto-tuned transformations were then not possible

Impact of Compiler Options







Active Harmony Approach to Automated Performance Tuning

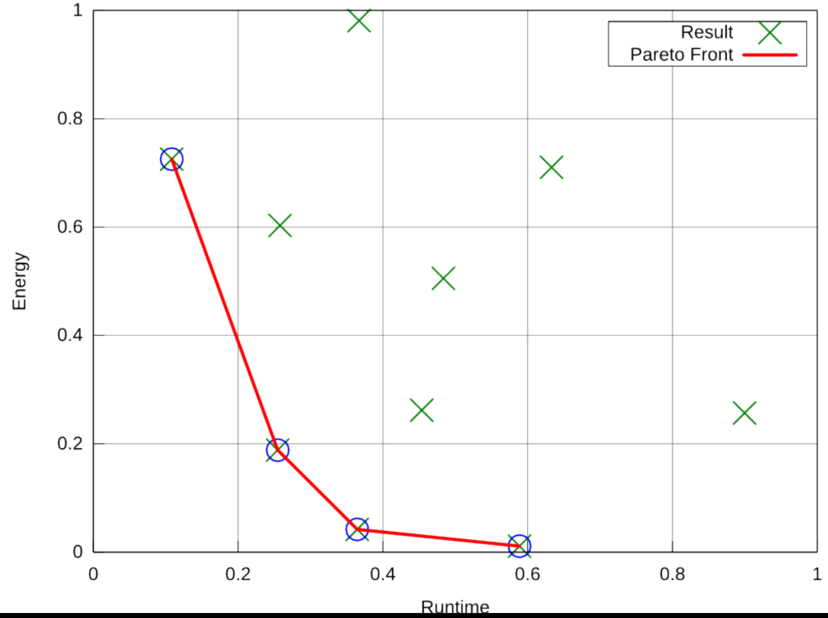
- Goal: Improve performance without training runs
- Problems:
  - Large number of parameters to tune
  - Shape of objective function unknown
  - Multiple libraries and coupled applications
  - Analytical model may not be available
- Requirements:
  - Runtime tuning for long running programs
  - Don’t try too many configurations
  - Avoid gradients

Auto-Tuning Objectives

- Single Objective
  - “More money is better.”
  - The best solution is easy to select
- Multi-Objective
  - “Is more money or more free-time better?”
  - Multiple different, but equally good solutions
  - The best solution becomes a subjective choice

Multi-Objective Example

- Minimize both energy and runtime
- Pareto set formed by non-dominated solutions
  - Solutions cannot be strictly improved upon



Existing Approaches

- Use experiments to find entire Pareto set
  - Algorithms judged by accuracy and efficiency
  - Evolutionary algorithms are widely used
- Provide set to users for final selection
  - This step is unacceptable for auto-tuning

Introducing NEMO

- Non-Evolutionary Multi-Objective Search Algorithm
- Goal:
  - Return a single solution, not a set of solutions
- Inputs:
  - Objective preference ranking
    - "When in conflict, I prefer runtime to be optimized over power."
  - Objective leeway percentage
    - "The search may stray up to 20% from the best known runtime."

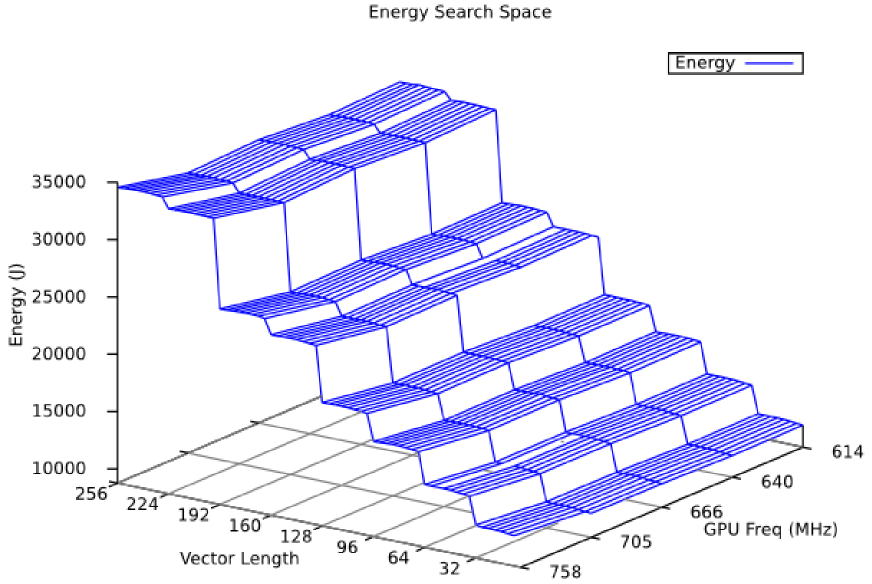
NEMO Algorithm

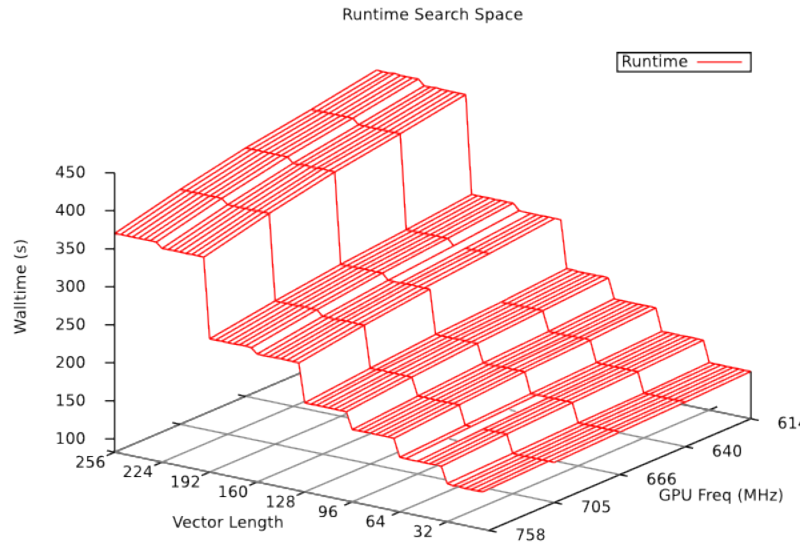
- Consider the first objective in isolation
  - Search using single objective search algorithm
  - Nelder Mead used in our experiments
- Record a threshold for first objective using leeway
  - Penalize any future searches that exceed threshold
- Repeat for objectives 2 through N
  - Search "landscape" changes with each iteration
  - Final landscape affected by all prior thresholds
    - Single objective search led to proper multi-objective solution

Comparison of Search Techniques

	Function Evaluations	Search Quality			Search Efficiency		
		Rand	ParEGO	NEMO	Rand	ParEGO	NEMO
kno1	62	2.385	2.950	**2.081**	11.665	7.711	**6.800**
oka1	53	1.605	2.783	**0.833**	4.294	5.124	**2.444**
oka2	59	2.294	2.978	**2.068**	5.903	5.891	**3.391**
vlmop2	56	0.119	**0.055**	0.377	0.627	0.637	**0.326**
vlmop3	87	**1.172**	1.371	1.231	7.808	4.567	**2.378**
dtlz1a	141	67.889	3.368	**0.840**	220.07	118.498	**30.346**
dtlz2a	209	0.893	0.935	**0.665**	0.963	0.925	**0.651**
dtlz4a	241	1.080	**0.678**	1.415	1.792	1.499	**1.451**
dtlz7a	223	12.992	8.351	**6.327**	12.721	**9.135**	11.646

GPU Tuning


Energy Search Space
Energy
Energy (J)
35000
30000
25000
20000
15000
10000
256
224
192
160
128
96
64
32
Vector Length
614
640
666
705
758
GPU Freq (MHz)


Runtime Search Space
Runtime
Walltime (s)
450
400
350
300
250
200
150
100
256
224
192
160
128
96
64
32
Vector Length
614
640
666
705
758
GPU Freq (MHz)

Conclusions

- Programmers don't want to tune programs
  - Lets computers do that
- Auto-tuning is ready to meet this need
- Need to efficiently support multi-objective search
  - At least 2 objectives, likely more
  - NEMO is a promising option for this