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

The effect of GCC's Optimization Level on SPEC Benchmarks (Buildtime)

The effect of GCC's inline-max parameter on 447.deall

Buildtime  | Filesize  | Runtime

Normalized Result (% of Maximum)

Parameter Value

4  8  12  16  20  24  28  32  36  40  44  48  52  56  60  64  128  256  512  1024
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

<table>
<thead>
<tr>
<th>Function</th>
<th>Search Quality</th>
<th>Search Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rand</td>
<td>ParEGO</td>
</tr>
<tr>
<td><strong>kno1</strong></td>
<td>62</td>
<td>2.385</td>
</tr>
<tr>
<td><strong>oka1</strong></td>
<td>53</td>
<td>1.605</td>
</tr>
<tr>
<td><strong>oka2</strong></td>
<td>59</td>
<td>2.294</td>
</tr>
<tr>
<td><strong>vlmop2</strong></td>
<td>56</td>
<td>0.119</td>
</tr>
<tr>
<td><strong>vlmop3</strong></td>
<td>87</td>
<td><strong>1.172</strong></td>
</tr>
<tr>
<td><strong>dtlz1a</strong></td>
<td>141</td>
<td>67.889</td>
</tr>
<tr>
<td><strong>dtlz2a</strong></td>
<td>209</td>
<td>0.893</td>
</tr>
<tr>
<td><strong>dtlz4a</strong></td>
<td>241</td>
<td>1.080</td>
</tr>
<tr>
<td><strong>dtlz7a</strong></td>
<td>223</td>
<td>12.992</td>
</tr>
</tbody>
</table>
GPU Tuning

Energy Search Space

Runtime Search Space
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