NEMO: Autotuning power and performance

Jeffrey K. Hollingsworth
A Word About Power

• What can you do with 14MW?
  – Make Snow for Snow Summit (13.6 MW)
  – There are > 400 ski resorts in the US

• What can you do with 75 MW?
  – Make steel (60-90MW is typical)
  – 8 such mills in Malaysia

• Does that mean exascale is more useful than a ski resort and less than a steel mill?
One more Thought on Power

• US has spent $100M on exascale to date
  – Perhaps for little/no return
• Could have purchased 20MW for 5 years
  – So a 40 MW machine could have worked
• In the future power may really matter, but
  – Let mobile market solve that for us
  – Let the Search/booksellers experts work on power
    • We could then work on the science & impact exascale can have to society
Generalized Auto-Tuning

1. Search Strategy
2. Candidate Points
3. Evaluated Performance
4. Client Application
Onion Model Workflow

- Allows for paired functionality, but either hook is optional.
- Fetch hooks executed in ascending order.
- Report hooks executed in descending order.
- Point values cannot be modified (directly).
Search Strategies

• Allow for new algorithms to be tried
• All other plugins still work
• Current Search Strategies
  – Exhaustive
  – Parallel Rank Order
  – Nelder-Mead
Plug-in Example: Constraints

- Support for non-rectangular parameter spaces.
  - Implemented as plug-in #2 using REJECT workflow.
  - $y \leq x$
Auto-Tuning Objectives

• Single Objective
  – “More apples is better.”
  – The best solution is easy to select

• Multi-Objective
  – “Is more apples, or more oranges 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
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.dealII

Normalized Result (% of Maximum)

Parameter Value
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
NEMO Example

NEMO Search Phase 1
(No Thresholds Available)

NEMO Search Phase 2
(Penalized for threshold violations of Objective 1)

NEMO Search Phase 3
(Penalized for threshold violations of Objective 1 & 2)
Preliminary Results
Conclusions

• Need to efficiently support multi-objective search
  – At least 2 objectives, likely more
  – NEMO is a promising option for this