OpenTuner

- A general framework for building autotuners
- A toolbox, not a one-size-fits-all autotuner
- Taking advantage of what we have learned in the 10+ years of using machine learning in compilers
- Available at http://www.opentuner.org

Machine Learning and Compilers

- OpenTuner: An Extensible Framework for Program Autotuning, OOPSLA 2014
- Portable Performance on Heterogeneous Architectures, ASPLOS 2014
- SiblingRivalry: Online Autotuning Through Local Computitions, CASES 2012
- Language and Compiler Support for Auto-Tuning Variable Accuracy Algorithms, CGO 2011
- Autotuning Multigrid with PetaBricks, SC 2009
- PetaBricks: A Language and Compiler for Algorithmic Choice, PLDI 2009.
- Adapting Convergent Scheduling Using Machine. LCPC 2003.
- Meta Optimization: Improving Compiler Heuristics with Machine Learning. PLDI 2003.

Lesson #1

- Configuration representation is critical
- Cartesian coordinates often natural/useful
- Represents things like trees poorly

OpenTuner:

- Custom format with dual interfaces:
  - Primitive Parameters
    - Point in high dimensional space
  - Complex Parameters
    - Dynamic number "moves" can be taken from any current position

Lesson #2

- There is no perfect search technique
- Techniques have differing strengths
  - Experience with many novel techniques
- Exploitation/exploration tradeoff

OpenTuner:

- Library of competing techniques:
  - Ensembles of techniques run in parallel
  - Credit assignment gives larger testing budgets to successful techniques
  - Long term (cross-run) performance informs which techniques are best for each problem

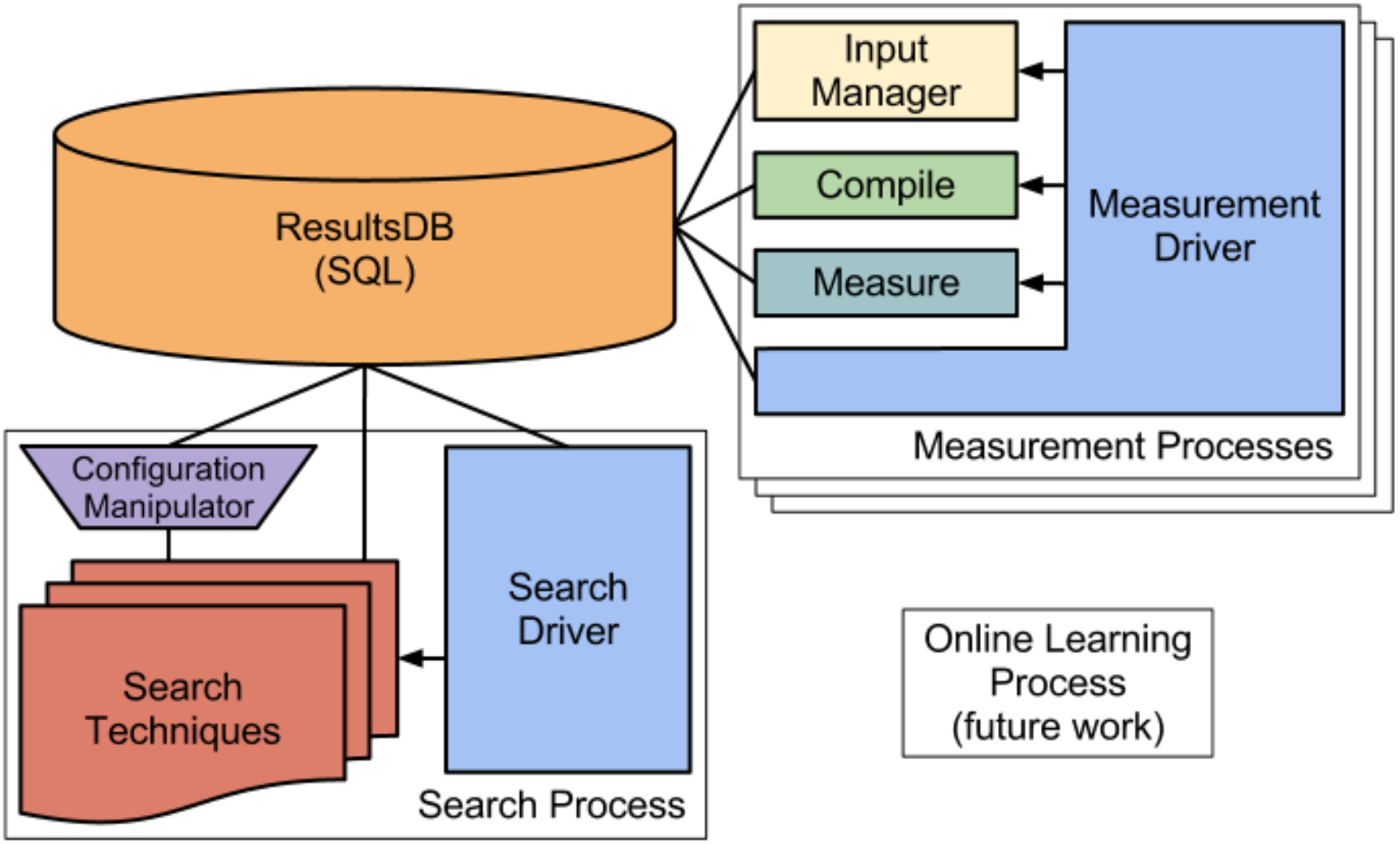
Lesson #3

- Usage, aggregation, and interpretation of results data varies widely
- Often accessed in different ways at different times

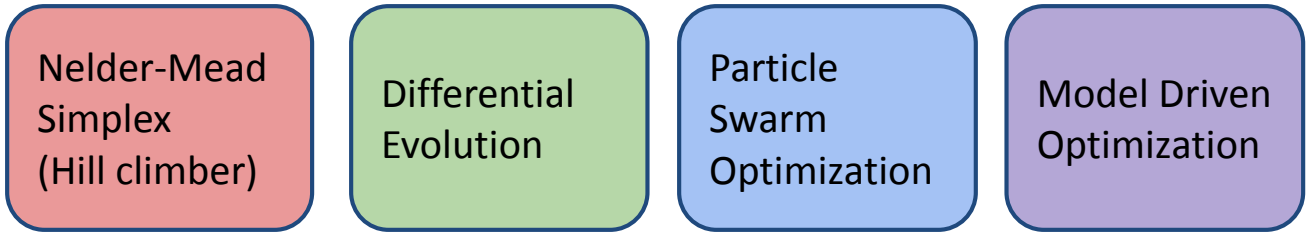
OpenTuner:

- Fully featured database of results (SQL):
  - Cross cutting access and mining of results data
  - Supports transactional parallelism
  - Long term knowledge sharing between runs

OpenTuner Modules/Processes

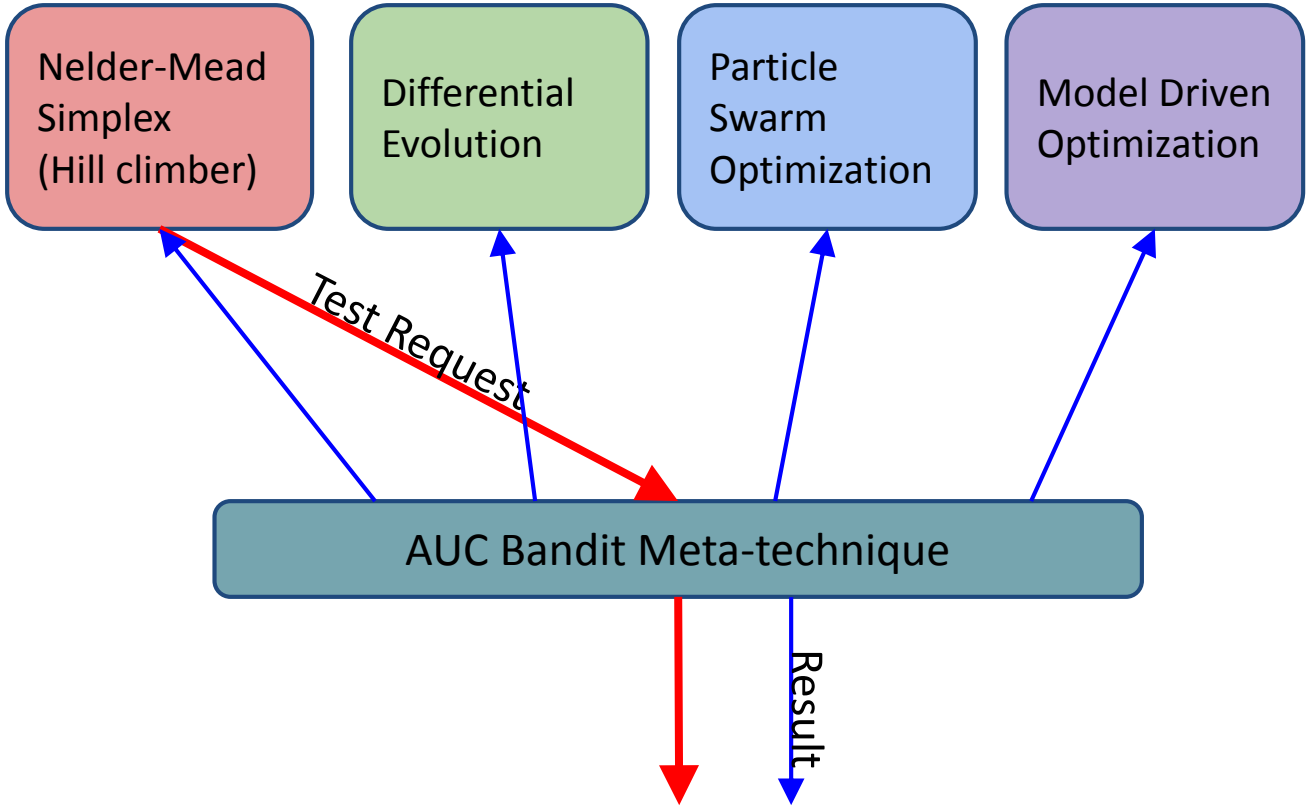


OpenTuner: Combining ensembles of techniques



- Many different techniques
- Each best suited to solve different problems
- Hard to write a single autotuner that performs well in different domains
- Can we make these techniques work together?

OpenTuner: Combining ensembles of techniques



- Meta-technique divides testing budget between sub-techniques
- Results are shared between all techniques

Exploitation versus exploration: How to allocate testing budget?

	Nelder-Mead Simplex (Hill climber)	Differential Evolution	Particle Swarm Optimization	Model Driven Optimization
Exploitation: estimated payoff probabilities based on recent results	0.62	0.46	0.21	0.03

Exploration: based on optimal solution to multi-armed bandit problem
(n_t = number of times technique t has been tried)

$$\sqrt{\frac{2\lg \sum_k n_k}{n_t}}$$

Pick the technique to maximize weighted combination of exploitation and exploration terms