Autotuning: a Design of Experiments Approach

Pedro Bruel

phrb@ime.usp.br

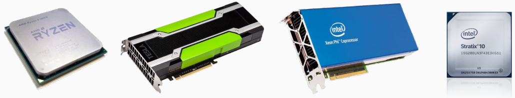
Journée au Vert POLARIS, March 2018

Outline

1. Autotuning
2. Applying Design of Experiments to Autotuning
3. Looking at Data
4. Summary & Perspectives

Autotuning: Optimizing Program Configuration

Architectures for HPC



How to write efficient code for each of these?

Autotuning

The process of automatically finding a configuration of a program that optimizes an objective

Configurations

- Program configuration
  - Algorithm, block size, . . .
- Source code transformation
  - Loop unrolling, tiling, rotation, . . .
- Compiler configuration
  - −O2, vectorization, . . .
- . . .

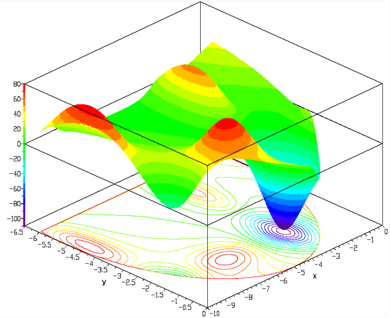
Objectives

- Execution time
- Memory & power consumption
- . . .

Search Spaces

Represent the effect of all possible configurations on the objectives

Can be difficult to explore, with multiple local optima and undefined regions

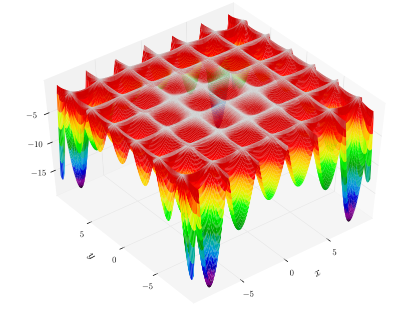


Mishra's Bird function

Search Spaces

Represent the effect of all possible configurations on the objectives

Can be difficult to explore, with multiple local optima and undefined regions



Hölder Table function

Autotuning: Search Spaces

Issue 1: Exponential Growth

Simple factors can generate large spaces:

- 30 *boolean* factors
- 2^{30} combinations

Issue 2: Geometry

- Discrete or continuous factors
- “Smoothness”
- Interactions between factors

Issue 3: Measurement Time

Time to compile:

- Benchmark GPU applications: 1~10s
- Benchmark FPGA applications: 1~10min
- Industrial FPGA applications: 1~10h

Autotuning: Multiple Approaches

Popular Approaches

- Exhaustive
- Meta-Heuristics
- Machine Learning

System	Domain	Approach
ATLAS	Dense Linear Algebra	Exhaustive
INSIEME	Compiler	Genetic Algorithm
Active Harmony	Runtime	Nelder-Mead
ParamILS	Domain-Agnostic	Stochastic Local Search
OPAL	Domain-Agnostic	Direct Search
OpenTuner	Domain-Agnostic	Ensemble
MILEPOST GCC	Compiler	Machine Learning
Apollo	GPU kernels	Decision Trees

Main Issues

- Optimized function is a black-box:
  - Learn nothing about the search space
  - Can’t explain why optimizations work
- These approaches assume:
  - A large number of function evaluations
  - Search space “smoothness”
  - Good solutions are reachable

Applying Design of Experiments to Autotuning

Our Approach

Using efficient experimental designs to overcome the issues of exponential growth, geometry, and measurement time

Design Requirements

- Support a large number of factors (Exponential Growth)
- Support continous and discrete factors (Geometry)
- Minimize function evaluations (Measurement Time)

Main Design Candidates

Screening Designs:

- Assume interactions are negligible
- Estimate main effects
- Aim to minimize runs

Mixed-Level Designs:

- Factors have different number of levels
- Many optimality criteria

Screening Designs

Plackett-Burman designs for 2-level factors:

- Orthogonal arrays of strength 2
- Estimate the main effects of n factors with $n+1$ runs

Construction:

- For $n+1$ multiple of 4
- Identical to a fractional factorial design if $n+1$ is a power of two

Mixed-Level Designs

Strategy 1: Contractive Replacement

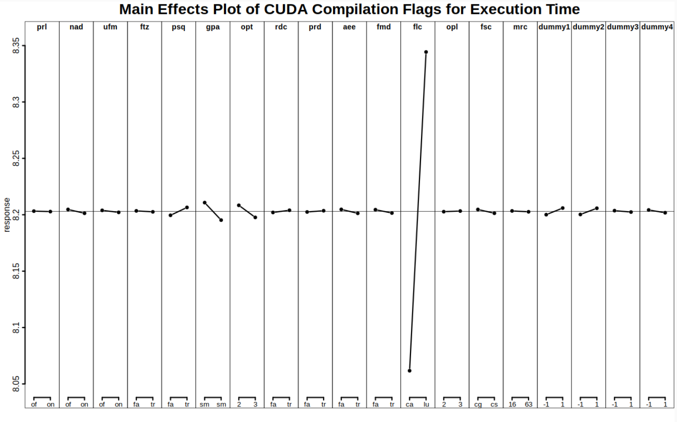
- Find specific sets of k-level columns of a design
- Contract the set into a new factor with more levels
- Maintain orthogonality of the design

Strategy 2: Direct Construction

Directly generate small mixed-level designs by solving Mixed Integer Programming problems

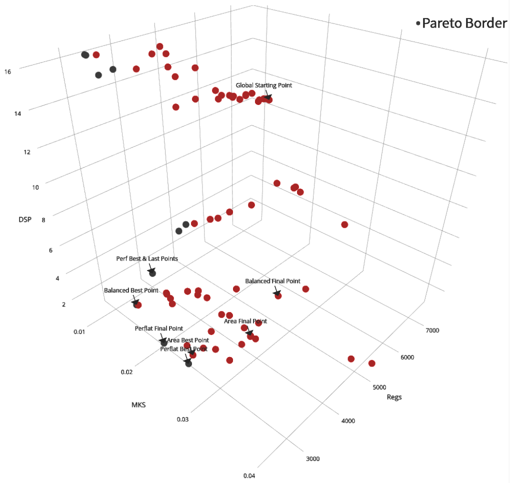
CUDA Compiler Flags

- Rodinia Benchmark
- 16 factors, few with multiple levels
- 10^6 combinations
- 1~10s to measure
- Screening Experiment:
  - 16 "2-level" factors
  - 4 "dummy" factors



FPGA Compiler Parameters

- CHStone Benchmark
- 141 factors, most with multiple levels
- 10^{128} combinations
- 1~10min to measure
- Multiple objectives
- Search with Meta-Heuristics:
  - Unstructured data difficults analysis
  - We are working on obtaining more data



Summary & Perspectives

Our Approach

Using efficient experimental designs to overcome the issues of exponential growth, geometry, and measurement time

Main Design Candidates

Screening & Mixed-Level designs

Target Scenario: FPGA Compiler Parameters

- Large search space
- Factors with multiple levels
- Large measurement time

Perspectives

- Short term:
  - Generate small, balanced, orthogonal multi-level designs for large numbers of factors
- Long term:
  - Use such designs to autotune industrial-level FPGA applications
- Longer term:
  - Iteratively drop least significant factors with user input
  - Provide an autotuning shared library to applications

Autotuning: a Design of Experiments Approach

Pedro Bruel

phrb@ime.usp.br

Journée au Vert POLARIS, March 2018