Autotuning: a Design of Experiments Approach

Pedro Bruel

phrb@ime.usp.br

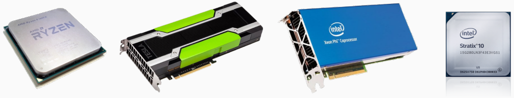
Journée au Vert POLARIS, March 2018

Outline

1. Autotuning
2. Applying Design of Experiments to Autotuning
3. Perspectives

Autotuning: Optimizing Program Configurations

Architectures for High Performance Computing



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, . . .
- . . .

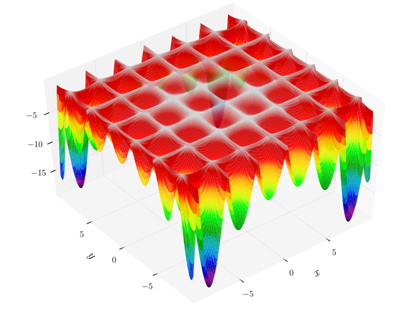
Objectives

- 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



Hölder Table function

Autotuning: Exploring 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

- These approaches assume:
  - A large number of function evaluations
  - Search space "smoothness"
  - Good solutions are reachable
- After optimizing:
  - Learn nothing about the search space
  - Can't explain why optimizations work

Applying Design of Experiments to Autotuning

Our Approach

Using efficient experimental designs to overcome issues related to 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:

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

Mixed-Level Designs:

- Factors have different number of levels
- Many optimality criteria

A Plackett-Burman screening design for 7 2-level factors:

Run	A	B	C	D	E	F	G
1	1	-1	1	-1	-1	1	1
2	1	1	1	-1	1	-1	-1
3	-1	1	-1	-1	1	1	1
4	-1	1	1	1	-1	1	-1
5	1	-1	-1	1	1	1	-1
6	1	1	-1	1	-1	-1	1
7	-1	-1	1	1	1	-1	1
8	-1	-1	-1	-1	-1	-1	-1

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

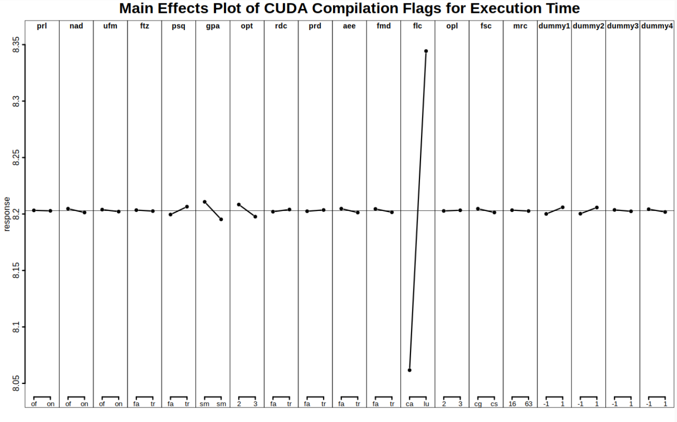
Construction:

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

Looking at Data: CUDA Compiler Flags

CUDA Compiler Flags

- Rodinia Benchmark
- 15 factors, few with multiple levels
- 10^6 combinations
- 1~10s to measure
- Screening Experiment:
  - 15 “2-level” factors
  - 4 “dummy” factors



Mixed-Level Designs

A multi-level design for 1 2-level factor and 3 3-level factors:

Run	A	B	C	D
1	1	1	1	3
2	1	1	2	1
3	1	1	3	2
4	1	2	1	2
5	1	2	2	3
6	1	2	3	1
7	1	3	1	1
8	1	3	2	2
9	1	3	3	3
10	2	1	1	1
11	2	1	2	2
12	2	1	3	3
13	2	2	1	3
14	2	2	2	1
15	2	2	3	2
16	2	3	1	2
17	2	3	2	3
18	2	3	3	1

Mixed-Level Designs

Strategy 1: Contractive Replacement

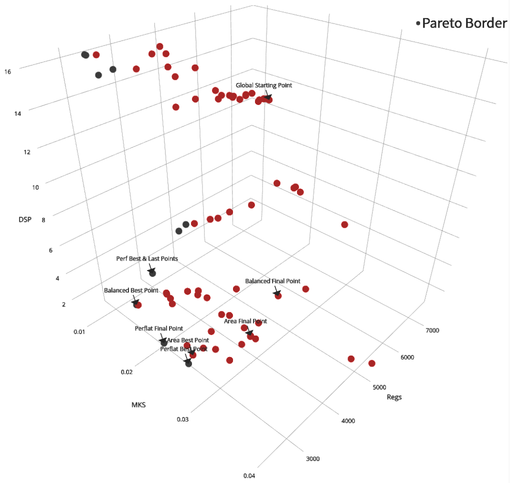
- Find specific sets of k-level columns of a design
- Contract the set into a new factor with more levels
- Maintain orthogonality of the design

Strategy 2: Direct Construction

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

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



Perspectives

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

Takeaway

Target Scenario: FPGA Compiler Parameters

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

Our Approach

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

Main Design Candidates

Screening & Mixed-Level designs

Autotuning: a Design of Experiments Approach

Pedro Bruel
phrb@ime.usp.br
Journée au Vert POLARIS, March 2018