Analyzing the Complexity of Auto-Tuning Many-Core Accelerators

Alessio Sclocco, Henri E. Bal, Rob V. van Nieuwpoort

NIRICT GPGPU Systems Workshop

October 28, 2016

VU UNIVERSITY AMSTERDAM Faculty of Sciences

netherlands eScience center

Introduction

Tuning, and auto-tuning, are well known optimization strategies

Research focuses mainly on two aspects:

1. Finding optimal configurations
2. Reducing the size of the optimization space

Our goal is to find a way to determine how difficult tuning is

Tuning, More Formally

- K : kernel
- $P = \{p_0, p_1, \ldots p_n\}$: set of K's tunable parameters
- I : valid input for K
- $C = \{c_0, c_1, \ldots c_m\}$: valid configurations of K's parameters
- P : platform

Tuning means finding $c_i \in C$ that optimizes a function, such as

- minimizing execution time
- maximizing throughput
- minimizing energy consumption

Tuning is Great, But it Takes Time

Tuning is great because the execution time of K running on P with input I using the optimal configuration c_i will be lower or equal than the execution time using any other c_j where $j \neq i$

But tuning can take a long time:

- Big optimization space
- Low performance of suboptimal configurations

Is Tuning Worthwhile?

Scenario 1:

- Optimum: 10% better than the average configuration
- Tuning time: 2 hours
- Future use: few times

Scenario 2:

- Optimum: 10% better than the average configuration
- Tuning time: 2 hours
- Future use: in production for 3 years

Scenario 3:

- Optimum: 200% better than the average configuration
- Tuning time: 2 hours
- Future use: few times

Tuning Difficulty

We say that tuning K is difficult (for platform P and input I) if the performance associated with c_i lies far apart from the performance associated with all the other c_j

Open question:

- How to quantify difficulty with just a number?

Knowing how difficult is tuning K makes it possible to:

- answer the question on how worthwhile tuning is
- steer the optimization process, e.g. pruning search space

Optimum Portability

Optimum portability is the degree to which the optimal configuration c_i of K (for platform P and input I) can be reused:

- on other platforms
- for other inputs

Reuse can be total or partial

Optimum portability modifies tuning difficulty of a kernel:

- low portability makes tuning more difficult
- some degree of portability make tuning easier

TuneBench

To test these ideas we needed a benchmark suite with easily tunable kernels

But we could not find one on the Internet

So we wrote our own, and called it TuneBench

- 5 kernels
  - with tunable parameters
- open-source
- available on GitHub[1]

[1] https://github.com/isazi/TuneBench

Experiments

Measuring tuning difficulty and optimum portability in TuneBench

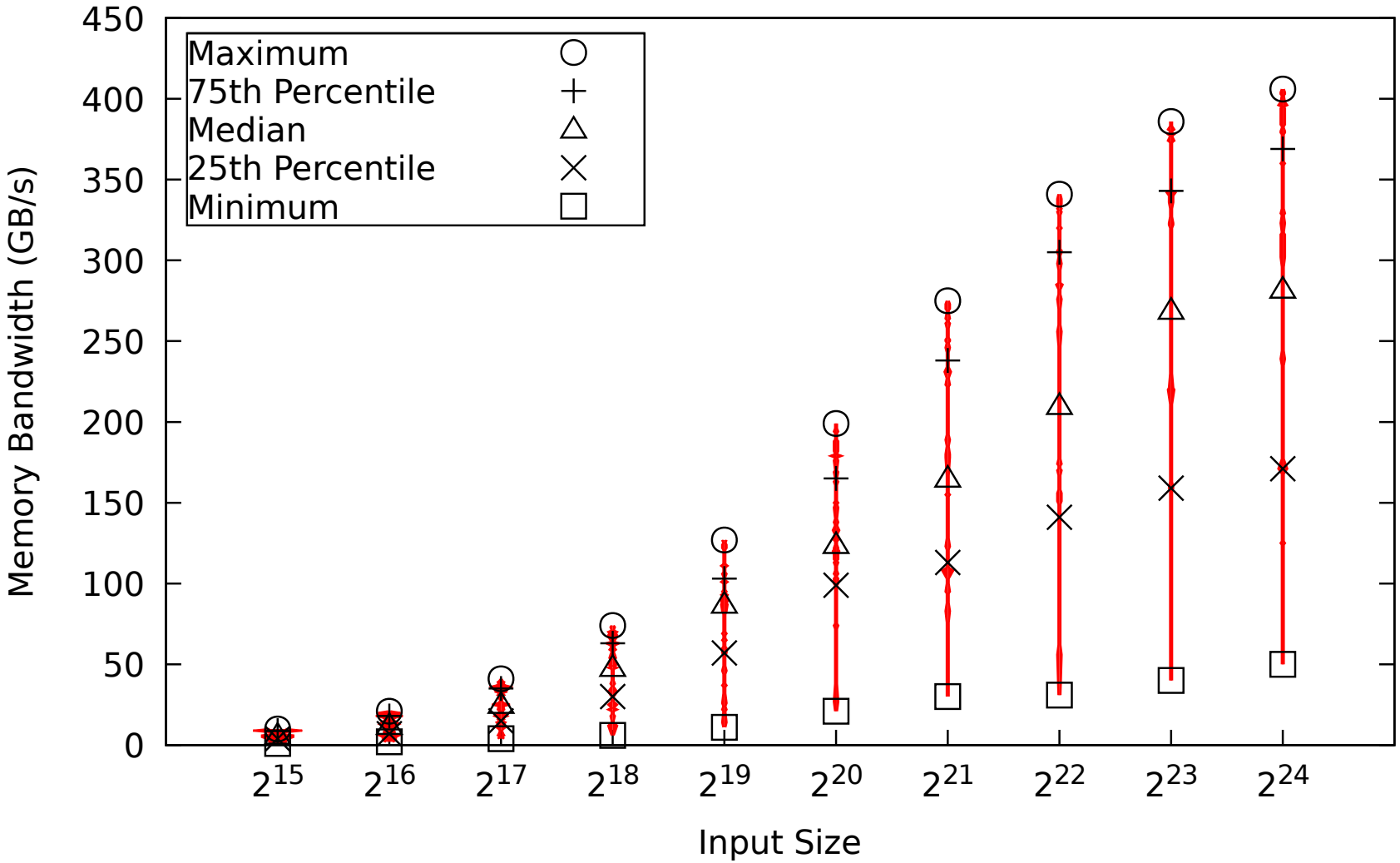
- triad
  - 3 tunable parameters
  - memory-bound
- reduction
  - 4 tunable parameters
  - memory-bound
- stencil
  - 5 tunable parameters
  - data-reuse
- MD
  - 3 tunable parameters
  - data-reuse
- correlator
  - 5 tunable parameters
  - data-reuse

Six different platforms, multiple inputs

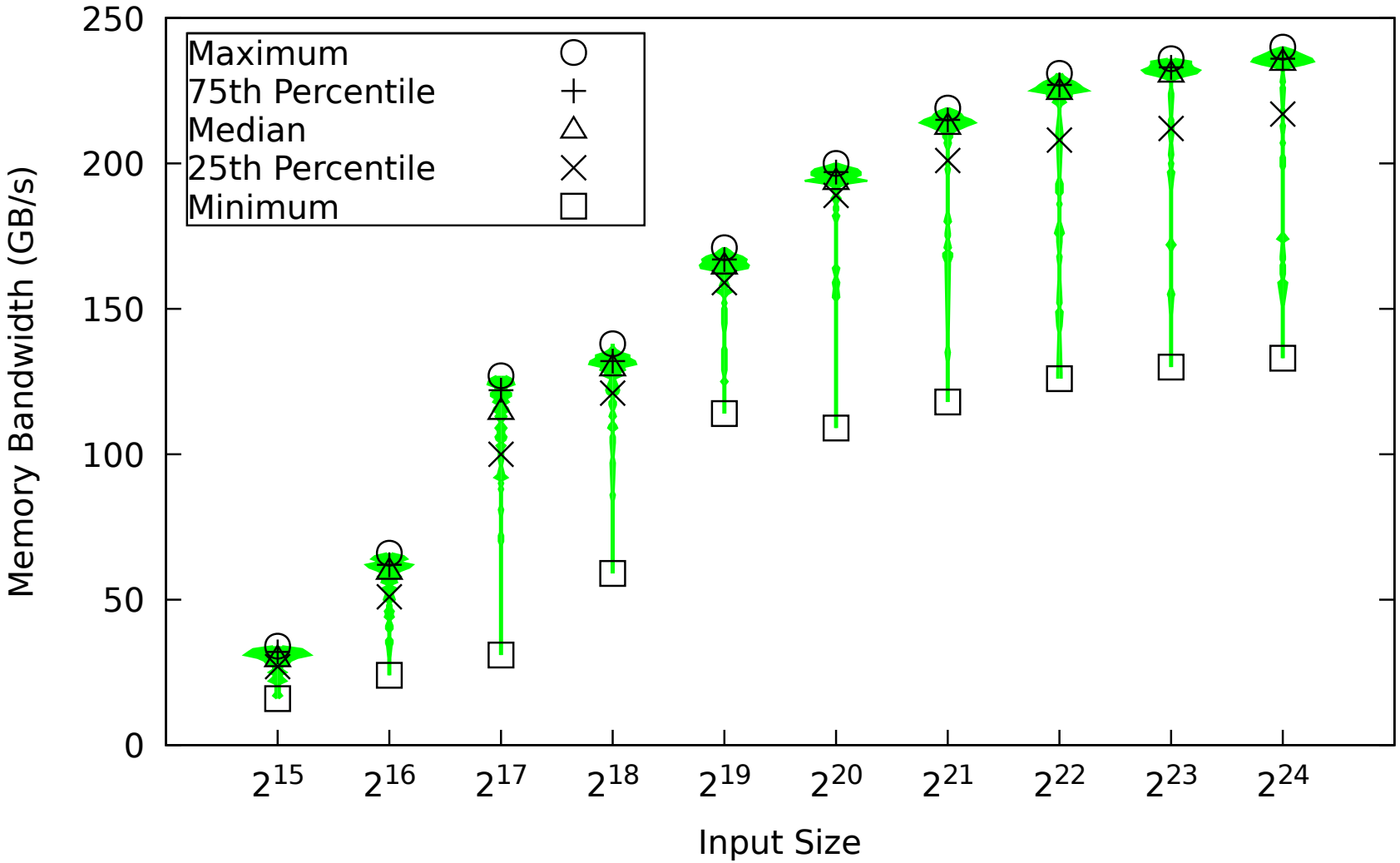
Platforms

Platform	Cores	GFLOP/s	GB/s
AMD FirePro W9100	2816	5237	320
AMD R9 Fury X	4096	8601	512
Intel Xeon Phi 31S1P	112	2006	320
NVIDIA GTX Titan	2688	4499	288
NVIDIA GTX Titan X	3072	6144	336
NVIDIA GTX 1080	2560	8228	320

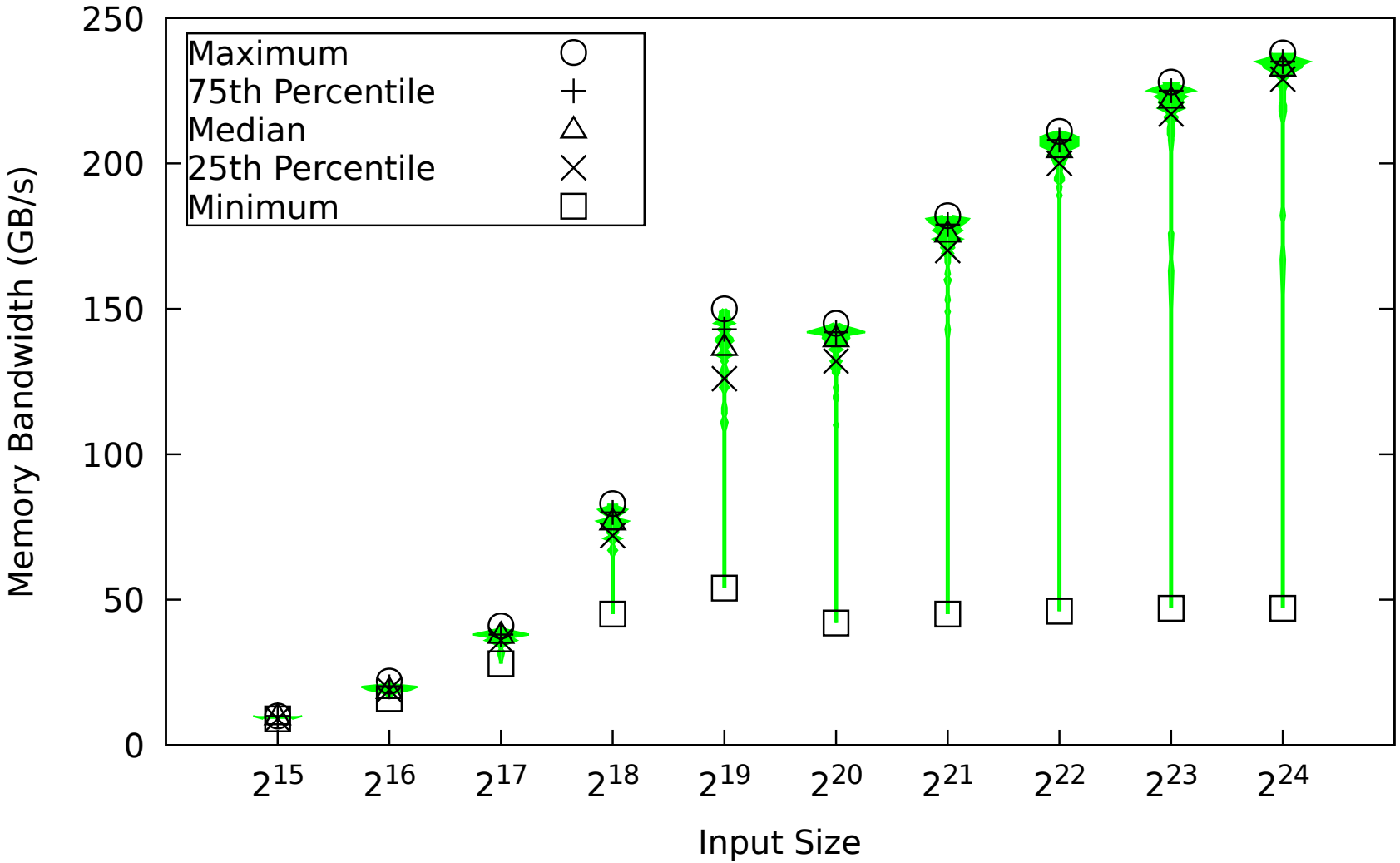
Tuning Difficulty: Fury X, triad



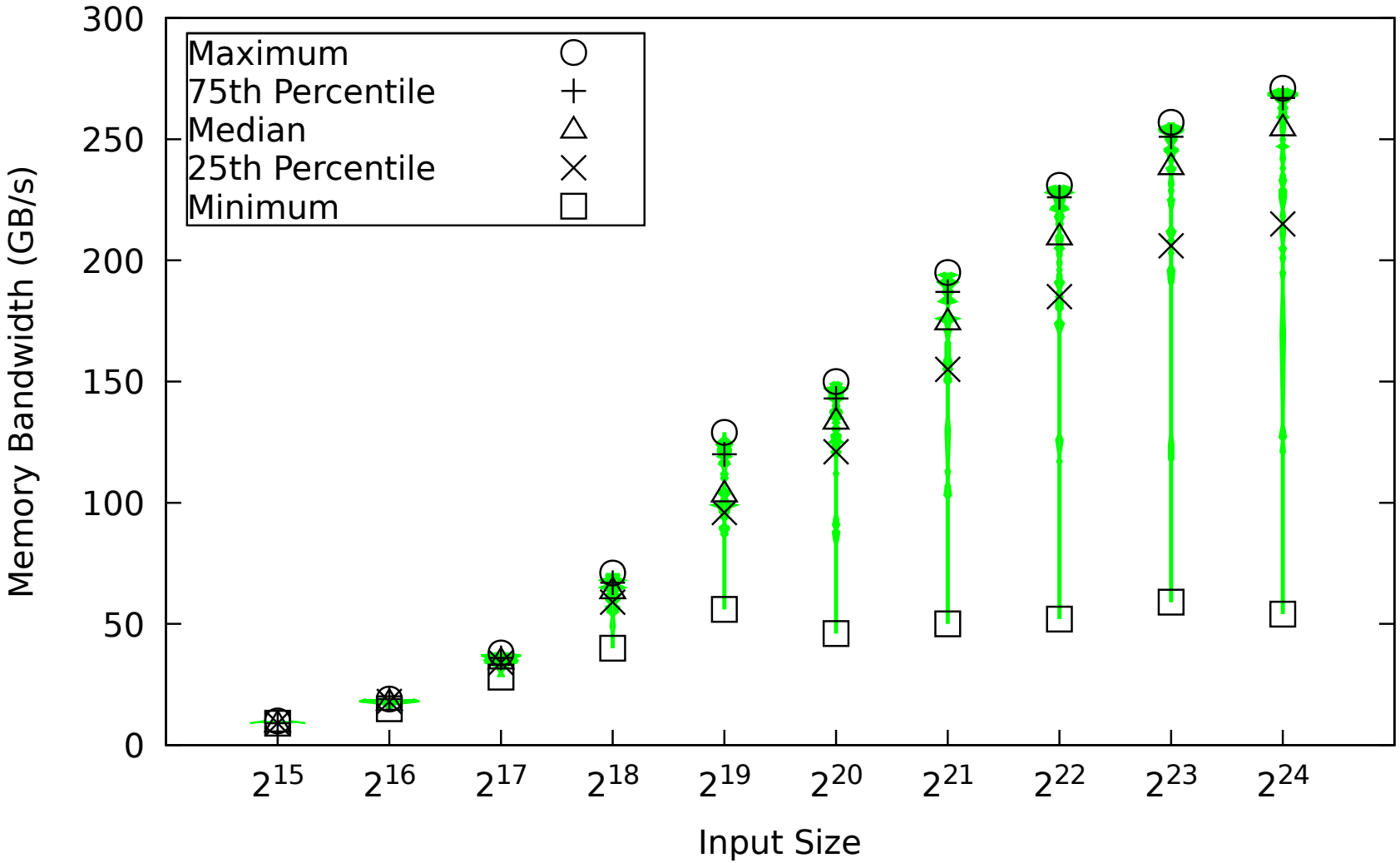
Tuning Difficulty: GTX 1080, triad



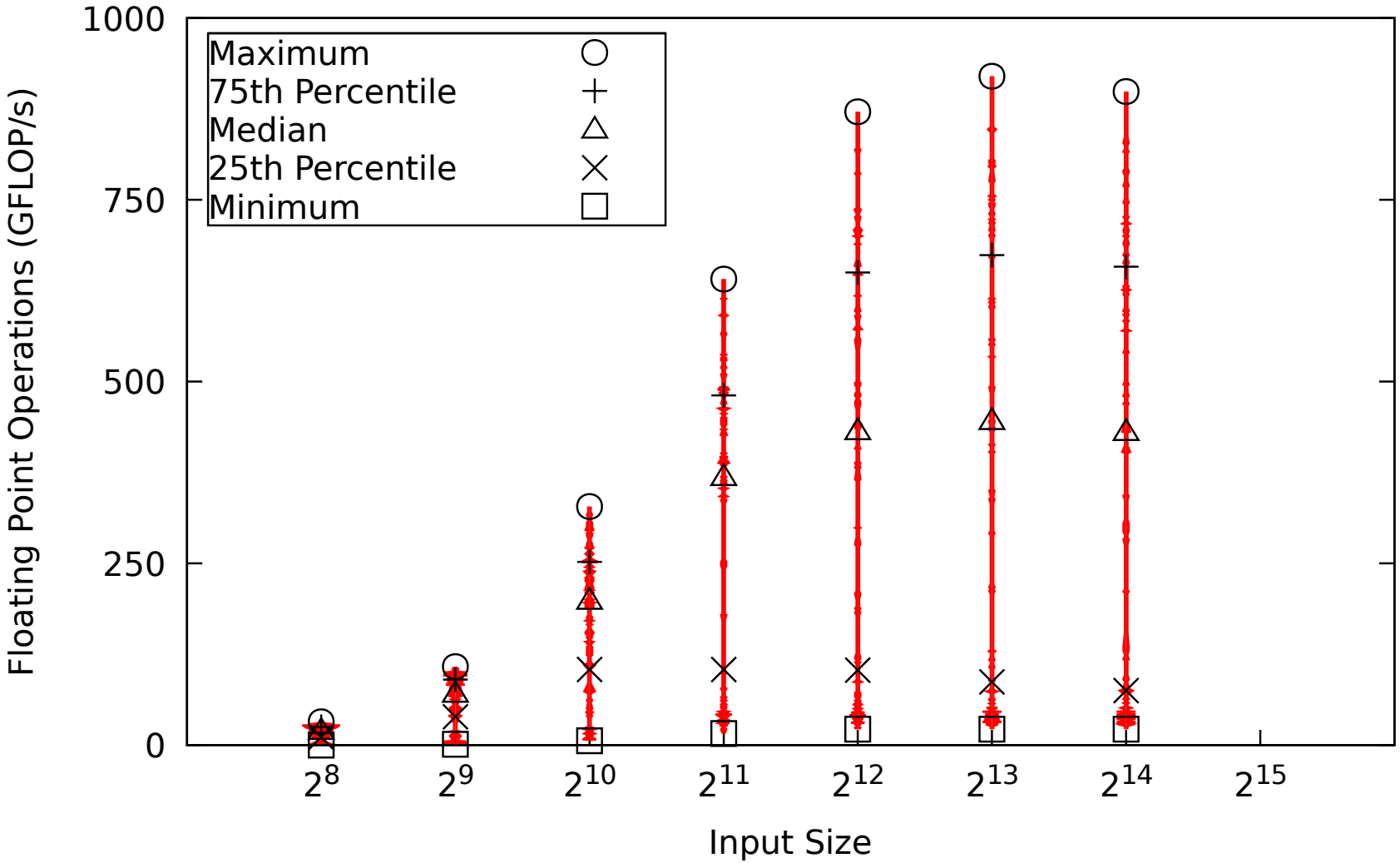
Tuning Difficulty: GTX 1080, reduction



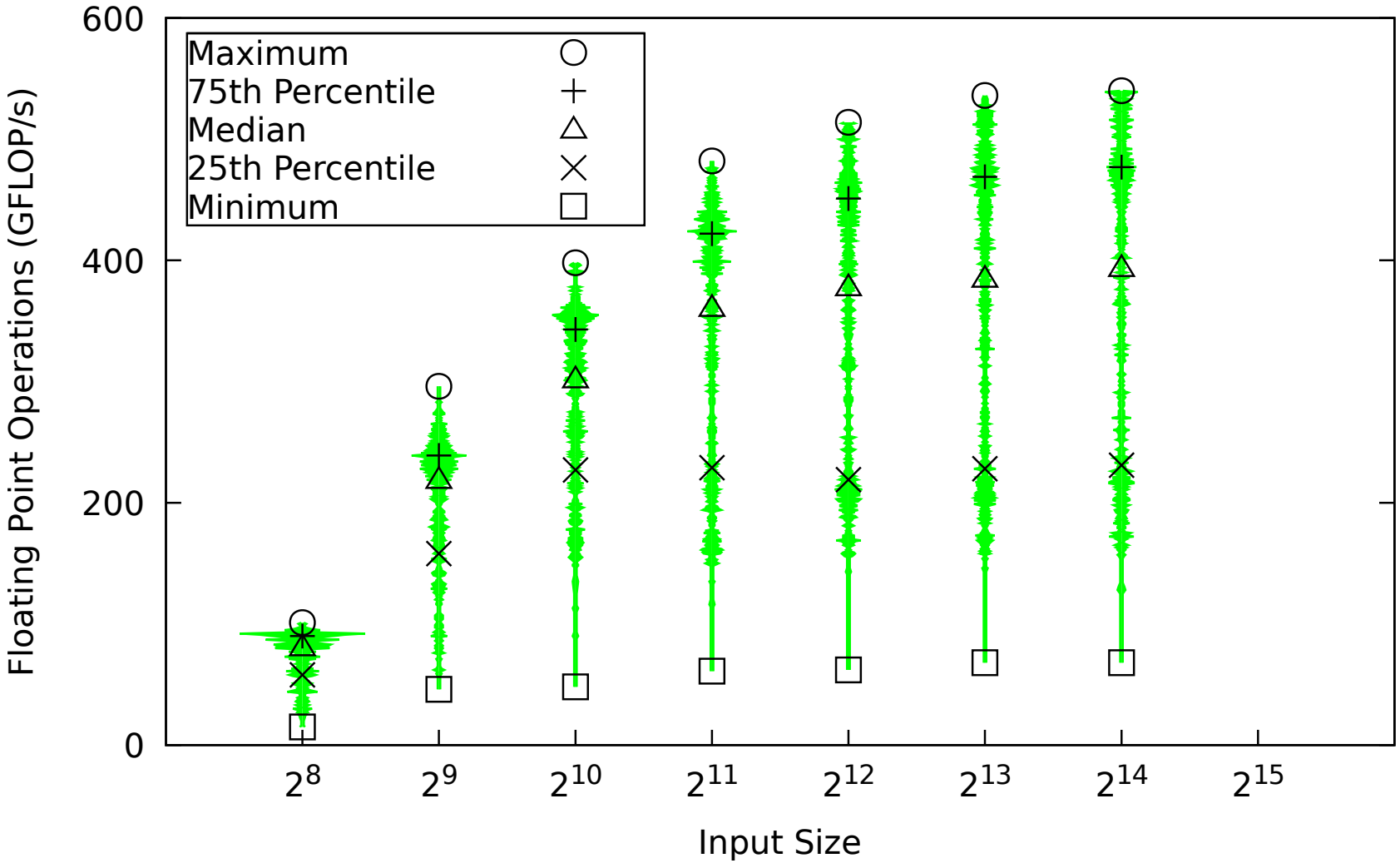
Tuning Difficulty: Titan X, reduction



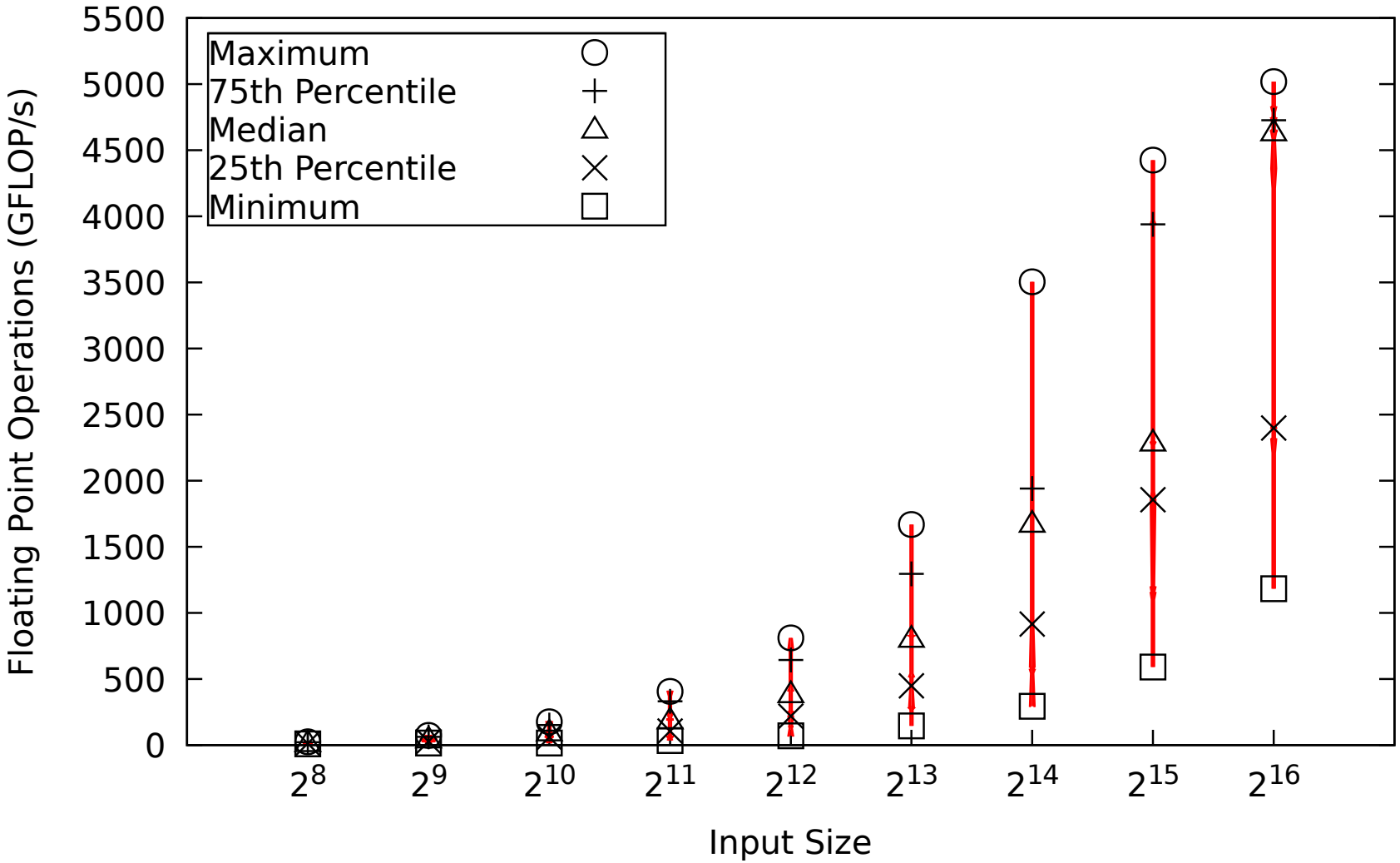
Tuning Difficulty: Fury X, stencil



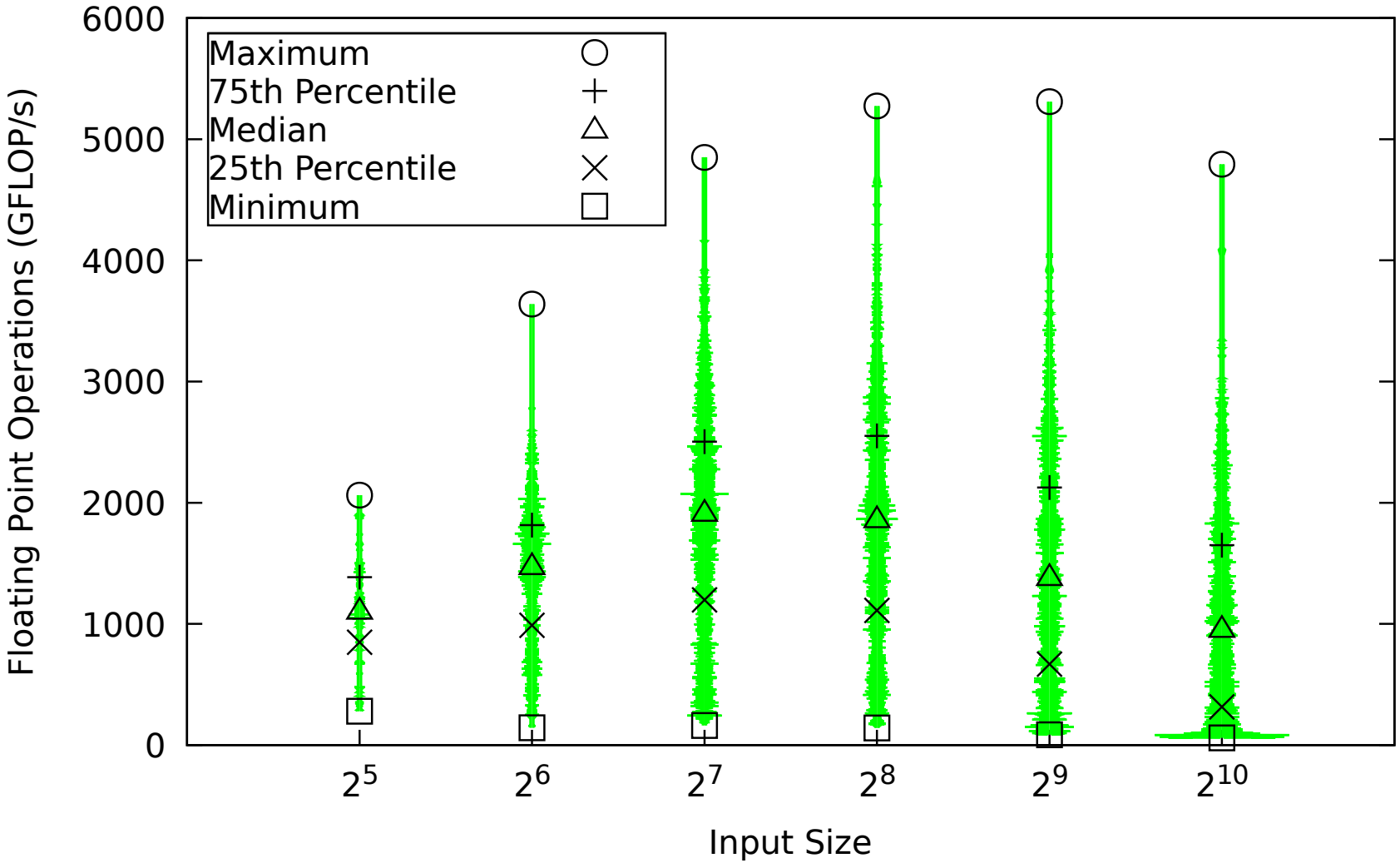
Tuning Difficulty: GTX 1080, stencil



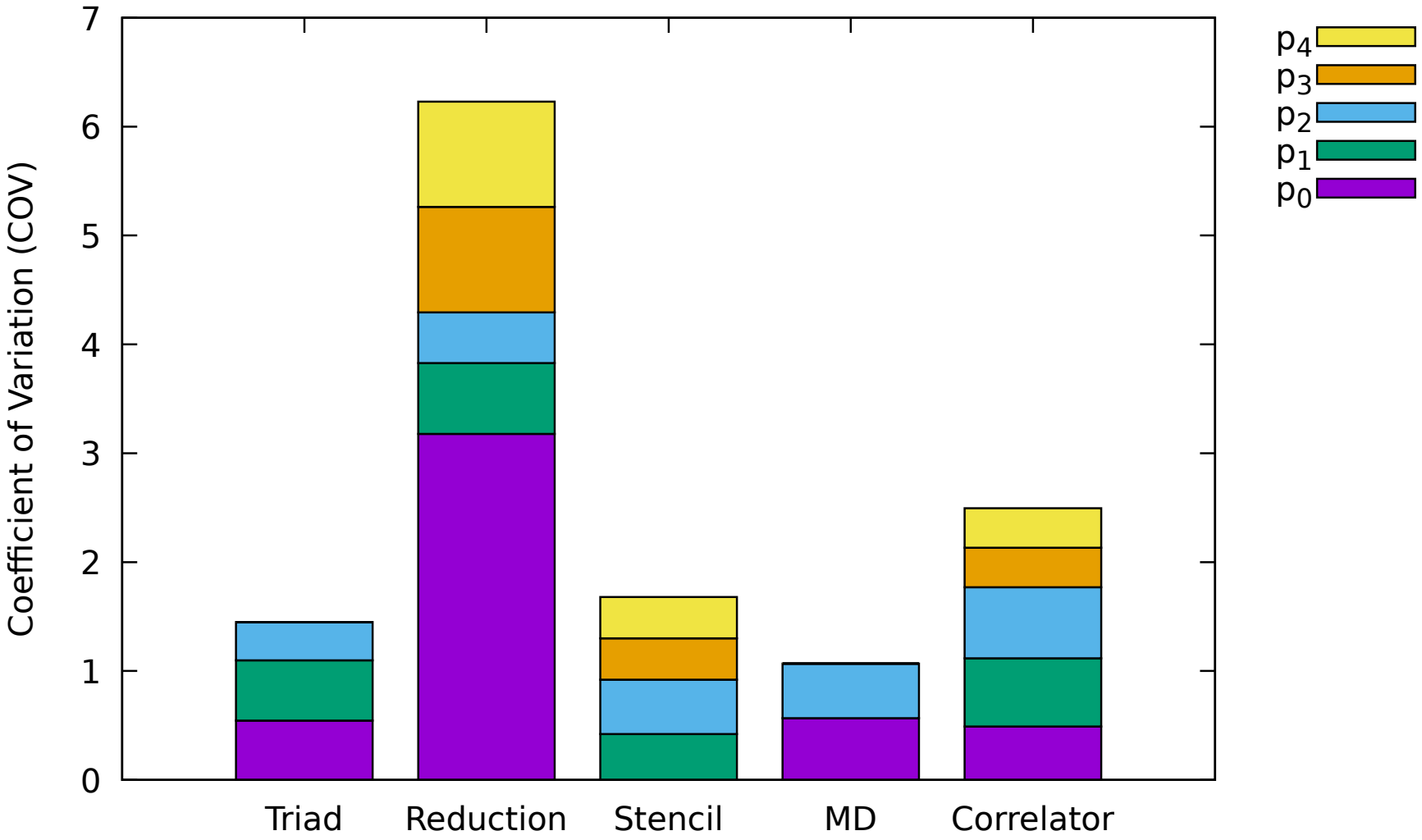
Tuning Difficulty: Fury X, MD



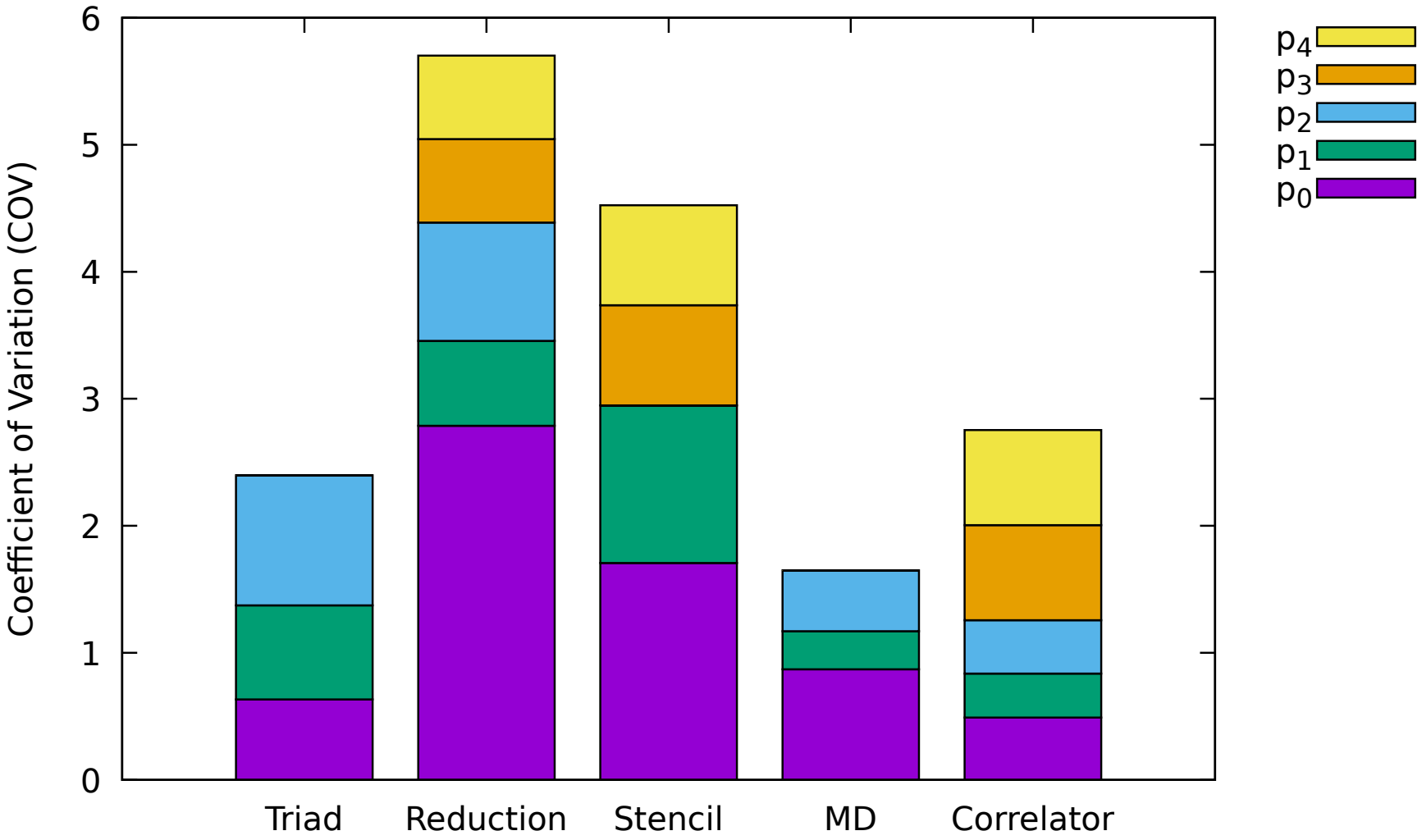
Tuning Difficulty: GTX 1080, correlator



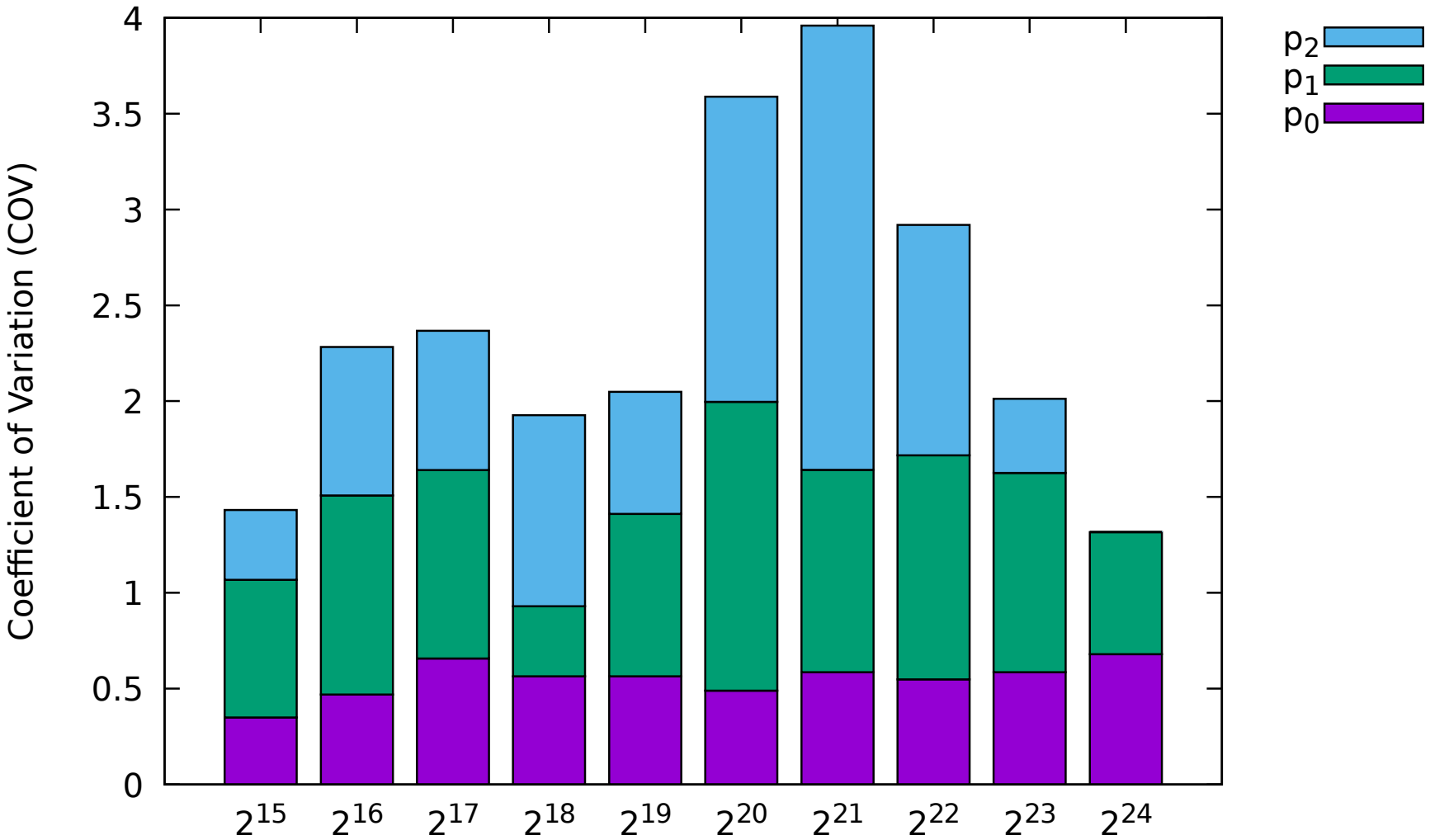
Optimum Portability: Fury X



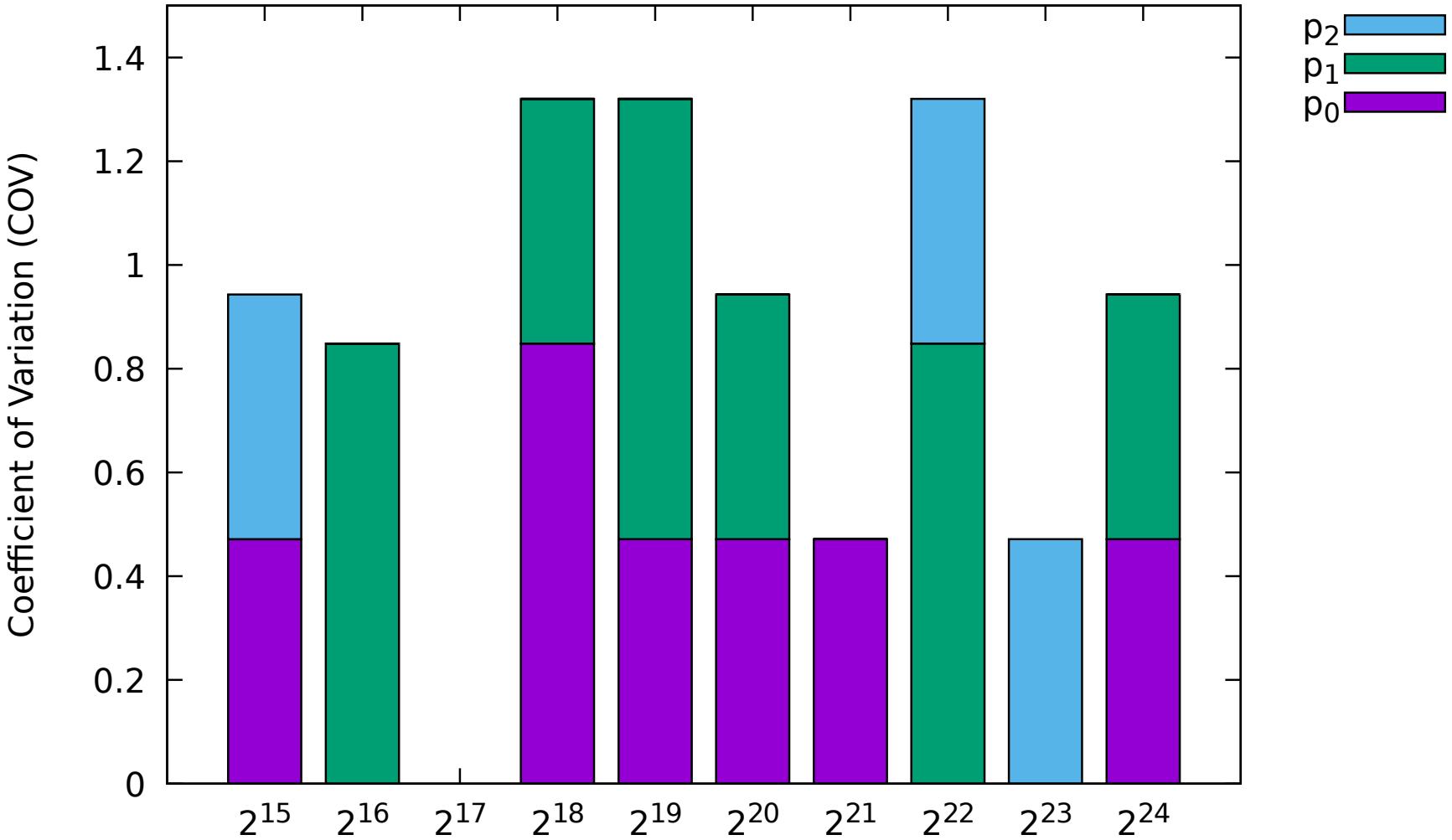
Optimum Portability: GTX 1080



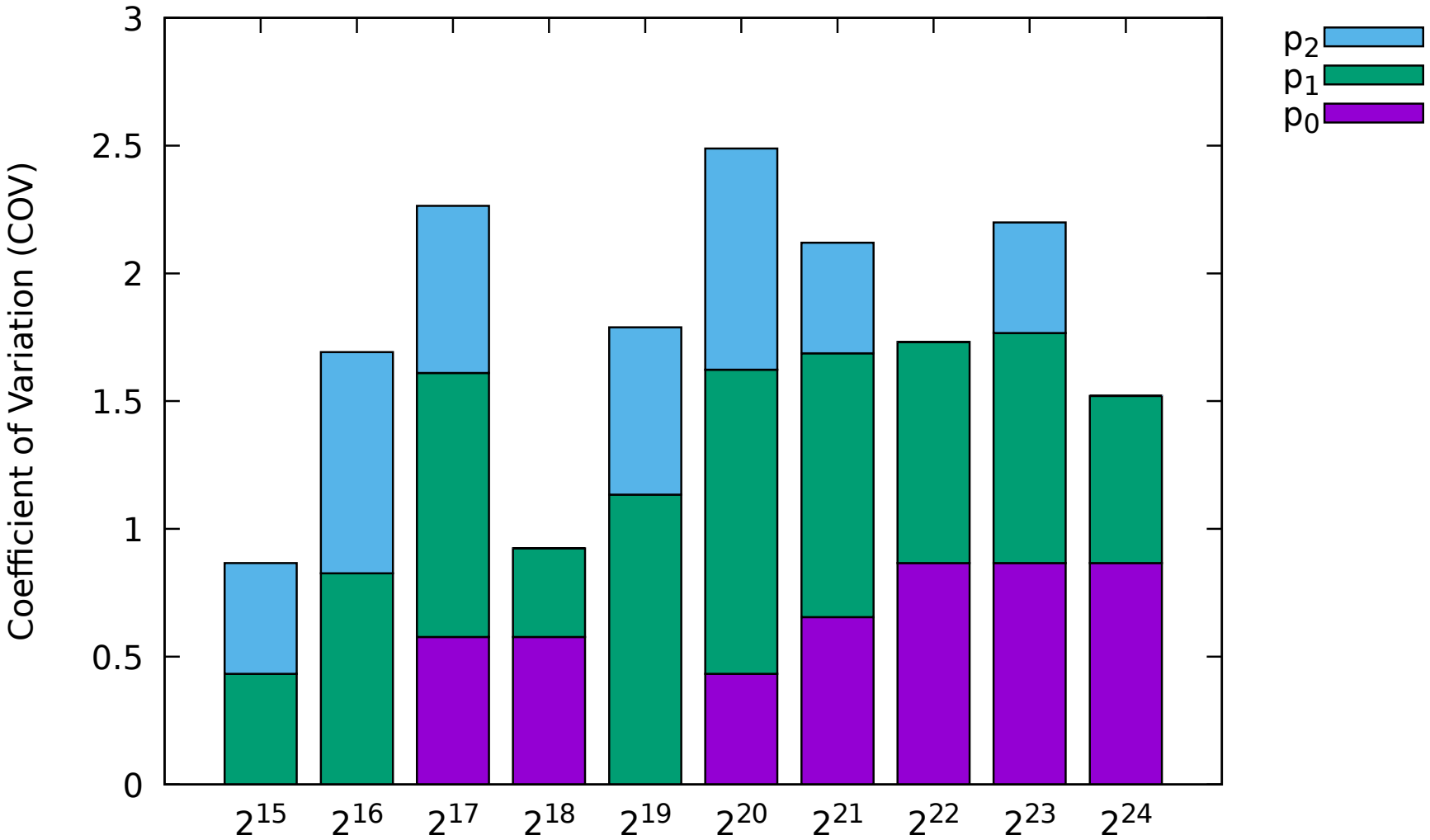
Optimum Portability: triad



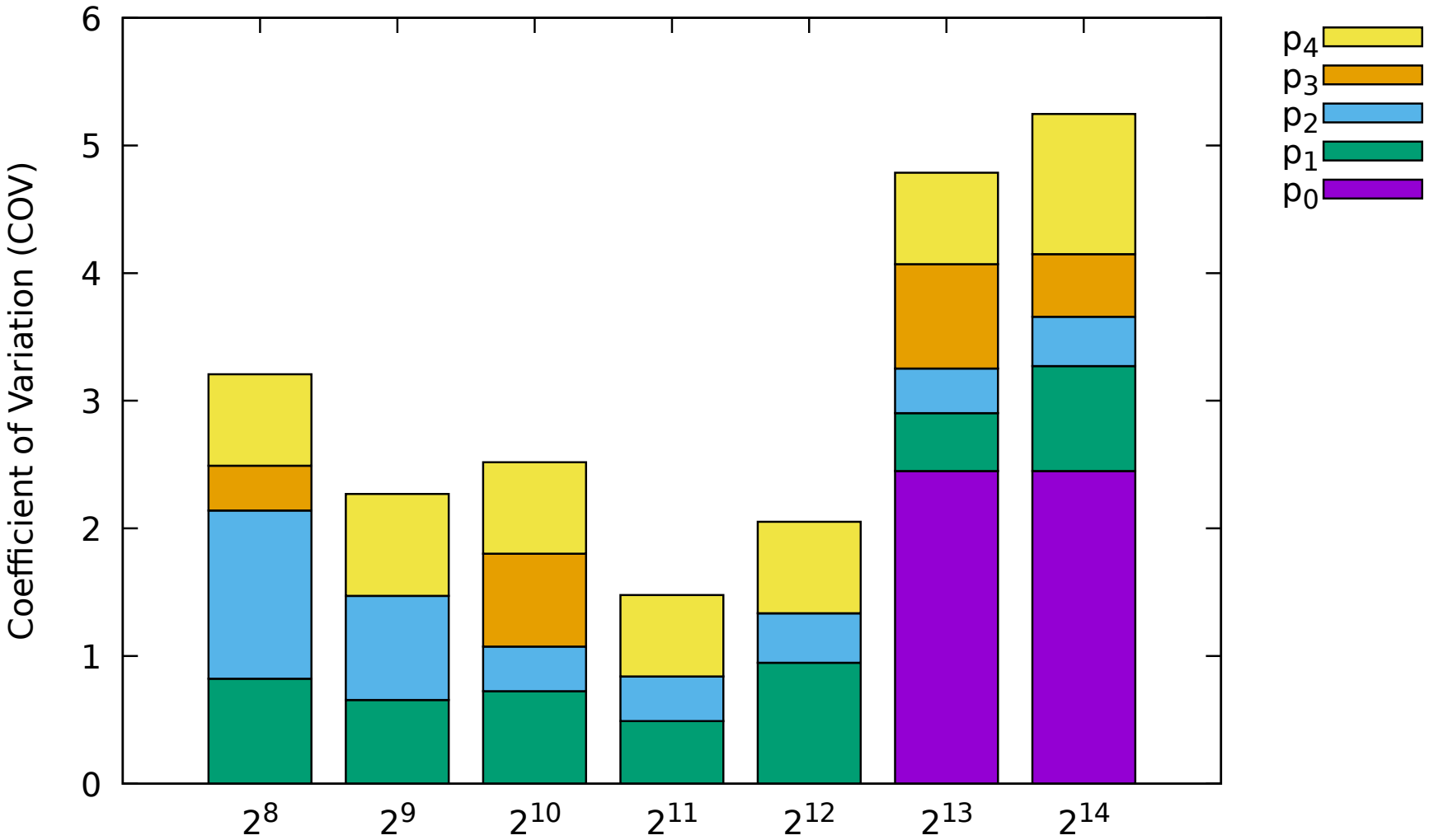
Optimum Portability: AMD GPUs, triad



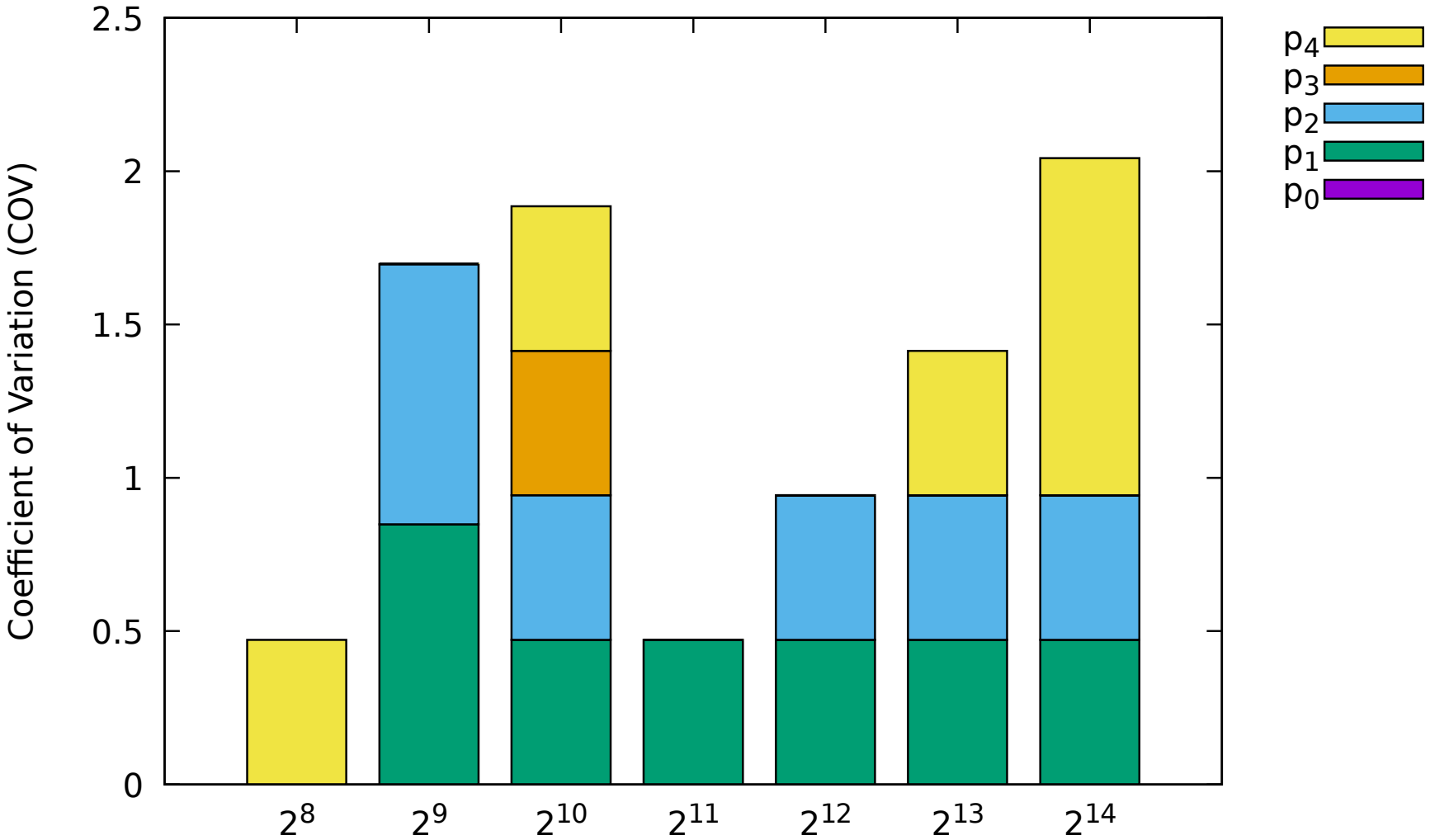
Optimum Portability: NVIDIA GPUs, triad



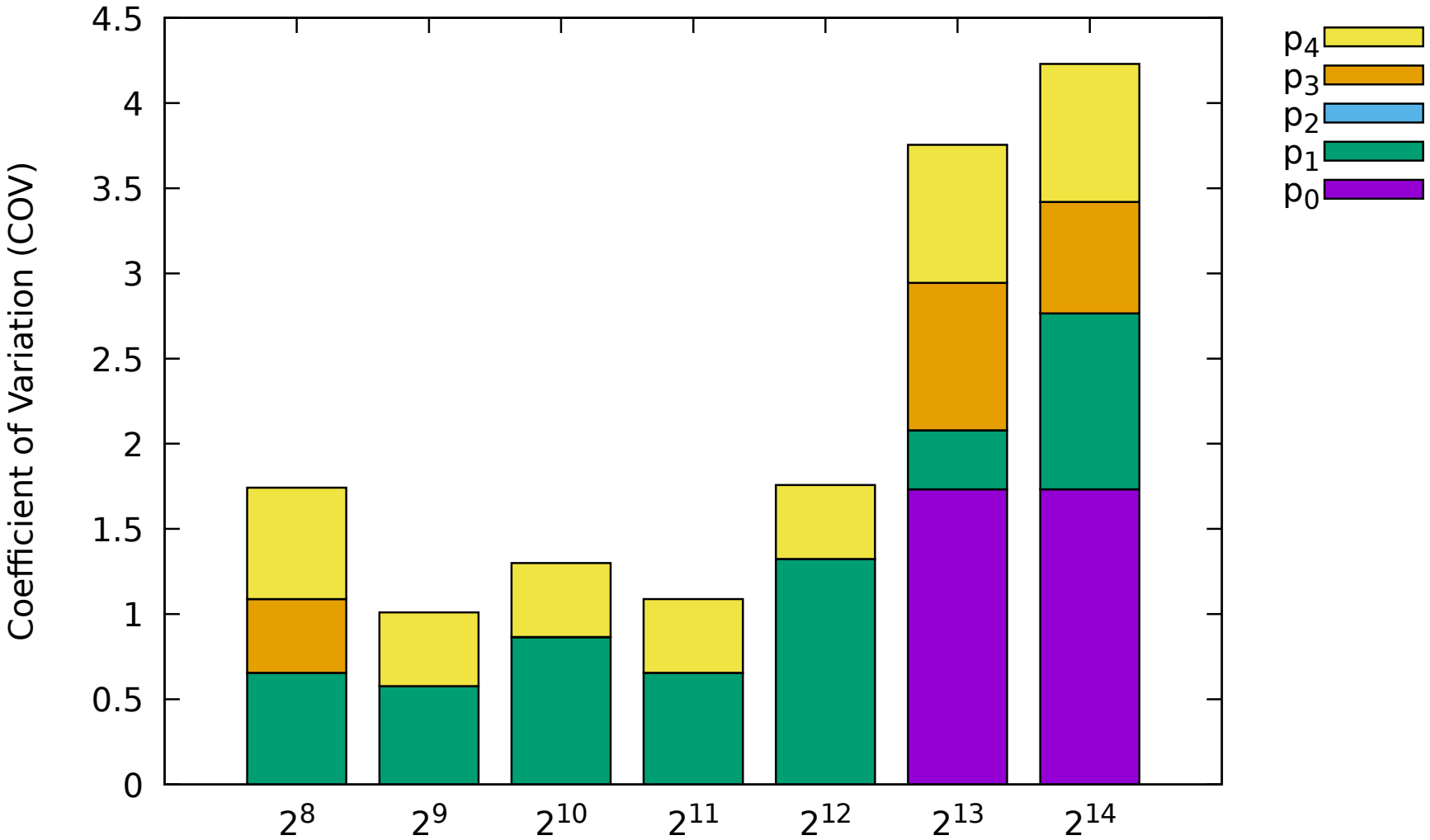
Optimum Portability: stencil



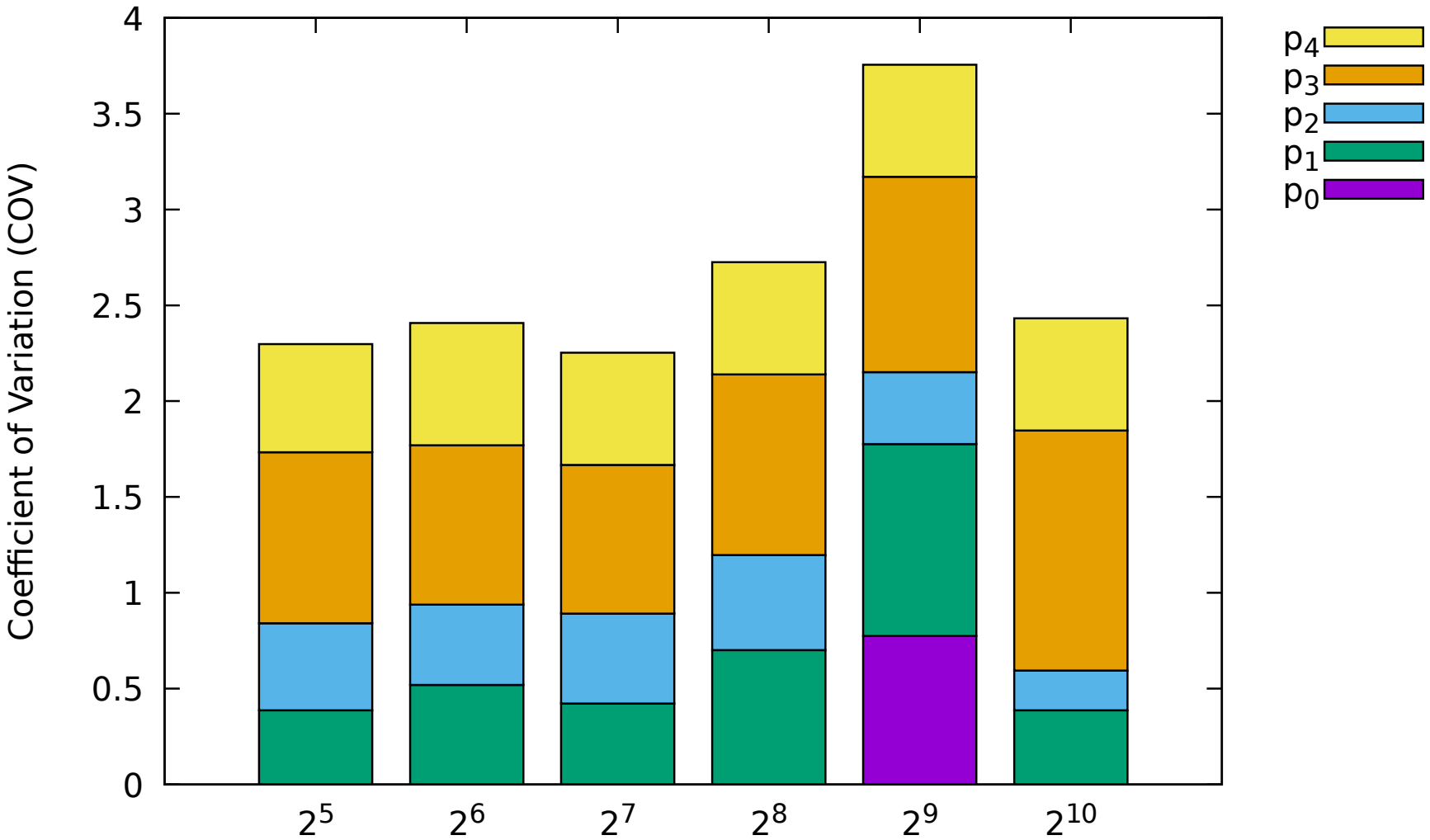
Optimum Portability: AMD GPUs, stencil



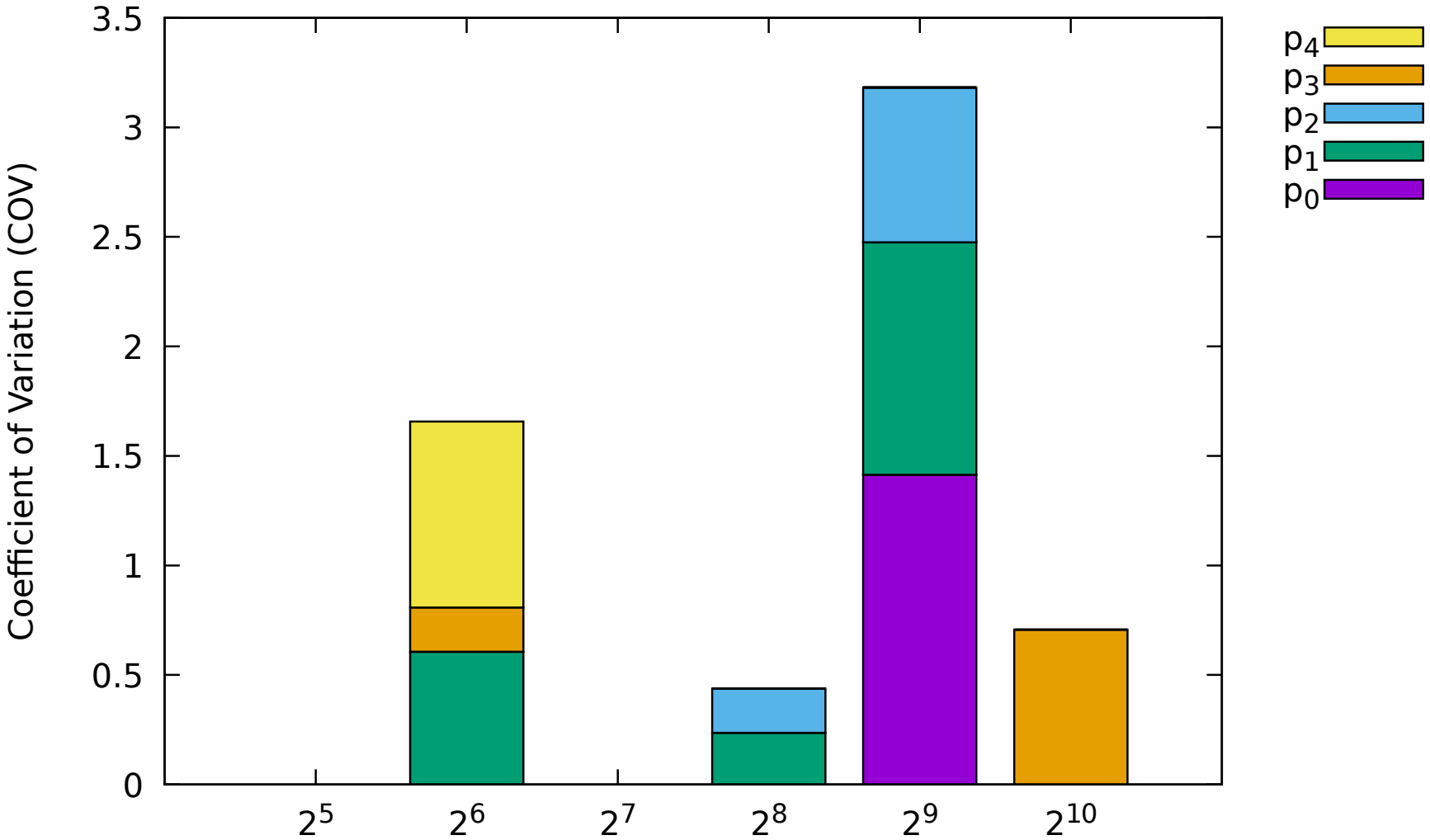
Optimum Portability: NVIDIA GPUs, stencil



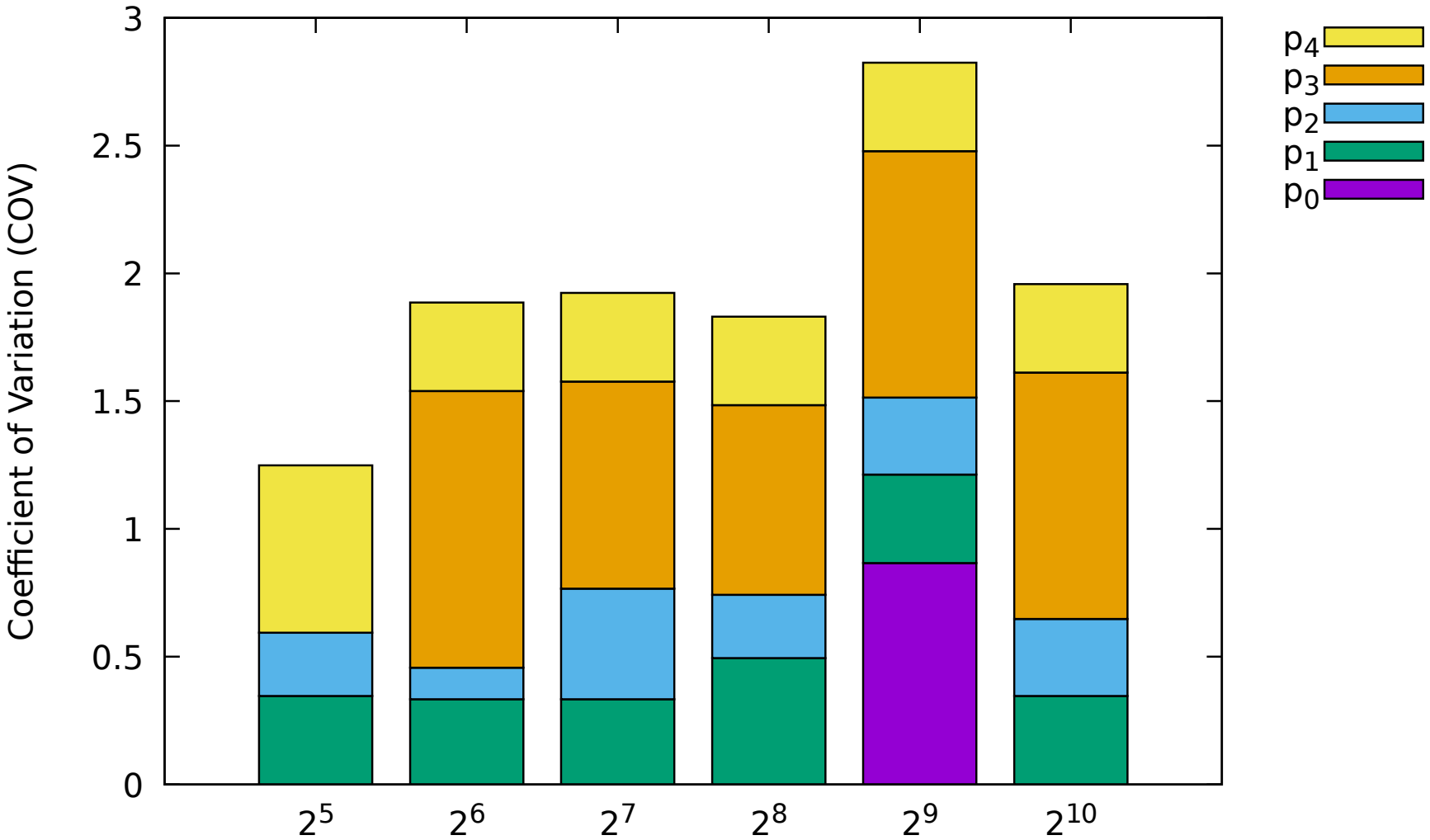
Optimum Portability: correlator



Optimum Portability: AMD GPUs, correlator



Optimum Portability: NVIDIA GPUs, correlator



Conclusions

- Tuning memory-bound applications on GPUs can be easy
  - NVIDIA GPUs are almost trivial to tune
  - It is not always the case with the Xeon Phi
- Tuning applications with data-reuse is more difficult
  - It applies to both GPUs and the Xeon Phi
  - Difficulty can be a function of the input size (e.g. MD)

- Optimum Portability is generally low
  - No much difference between memory-bound and data-reuse
  - Optimums are rarely portable among different platforms
    - But they are more portable for a single vendor
  - Optimums can be portable among inputs on the same platform