AUTOTUNING GPU COMPILER PARAMETERS WITH OPENTUNER

Pedro Bruel
phrb@ime.usp.br

Marcos Amarís
amaris@ime.usp.br

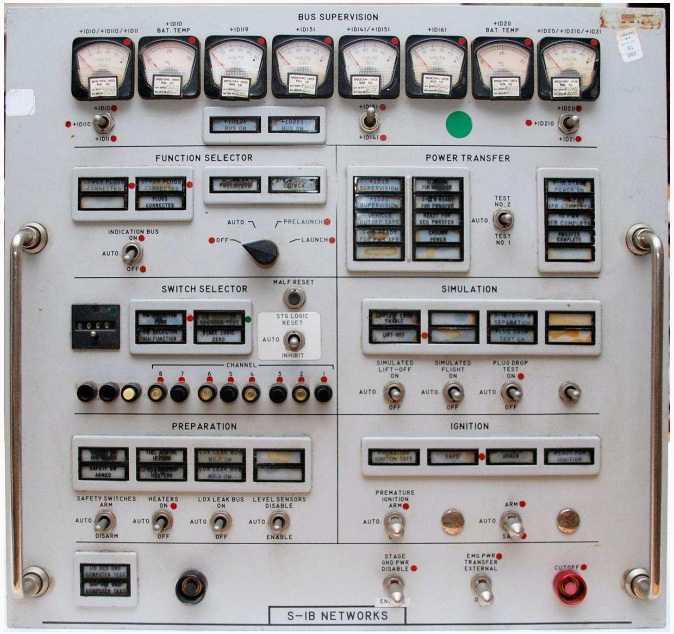
Alfredo Goldman
gold@ime.usp.br
29 de Setembro de 2015

Instituto de Matemática e Estatística
Universidade de São Paulo

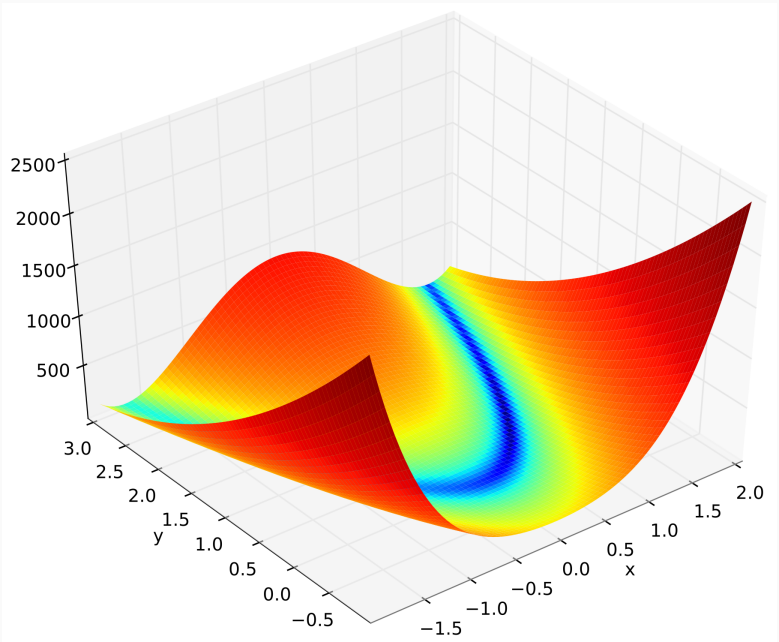
CONTRIBUTION

It is possible to optimize GPU applications for different devices by automatically tuning compilation parameters.

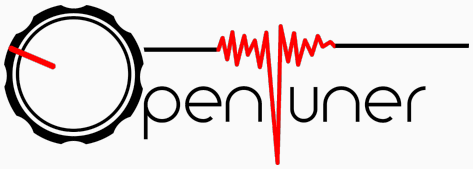
Configurations and Optimizations



Search Space



AUTOTUNING: OPENTUNER



- Autotuning framework
- Implements ensembles of search techniques
- Shares optimization results between techniques

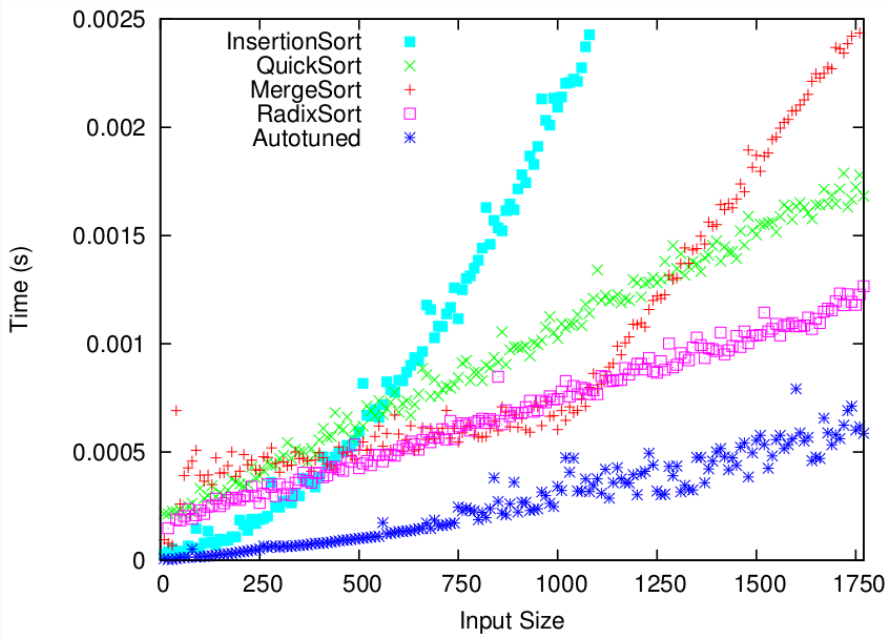


Figure 1: Autotuning recursive sorting algorithms for an 8-core machine.

Figure 1: Ansel, Jason, et al. "Opentuner: An extensible framework for program autotuning." Proceedings of the 23rd ICPAC. ACM, 2014.

HP THE MACHINE: POSSIBLE CONTRIBUTION

Optimizing programs for The Machine will cost a lot of time.
Autotuning can help the programmer by:

- Adapting existing algorithms
- Pointing the way to the best optimizations

COMPILER FLAGS

Step	Options
NVCC	`prec-sqrt`, `relocatable-device-code`, `no-align-double`, `use-fast-math`, `gpu-architecture`, `ftz`, `prec-div`
PTX	`def-load-cache`, `opt-level`, `fmad`, `allow-expensive-optimizations`, `maxrregcount`
NVLINK	`preserve-relocs`

Options	gpu-architecture	opt-level	def-load-cache	maxrregcount
Values	`sm_20`, `sm_21`, `sm_30`, `sm_32`, `sm_35`	`0 - 1`	`ca`, `cg`, `cv`, `cs`	`16 - 64`

GPU TESTBED

Model	c.c.	Global Memory	Bus	Bandwidth	L2	SM/Cores	Clock
GTX-680	3.0	2 GB	256-bit	192.2 GB/s	512 KB	8/1536	1006 Mhz
Tesla-K20	3.5	4 GB	320-bit	208 GB/s	1280 KB	13/2496	706 Mhz
Tesla-K40	3.5	12 GB	384-bit	276.5 GB/s	1536 KB	15/2880	745 Mhz

EXPERIMENTS

All the results' data and the code for the experiments, the autotuner and the figures is hosted at github.com/phrb/gpu-autotuning, under the GNU GPLv3 license.

EXPERIMENTS: MATRIX MULTIPLICATION

Four optimizations of square matrix multiplication ($N = 1024$):

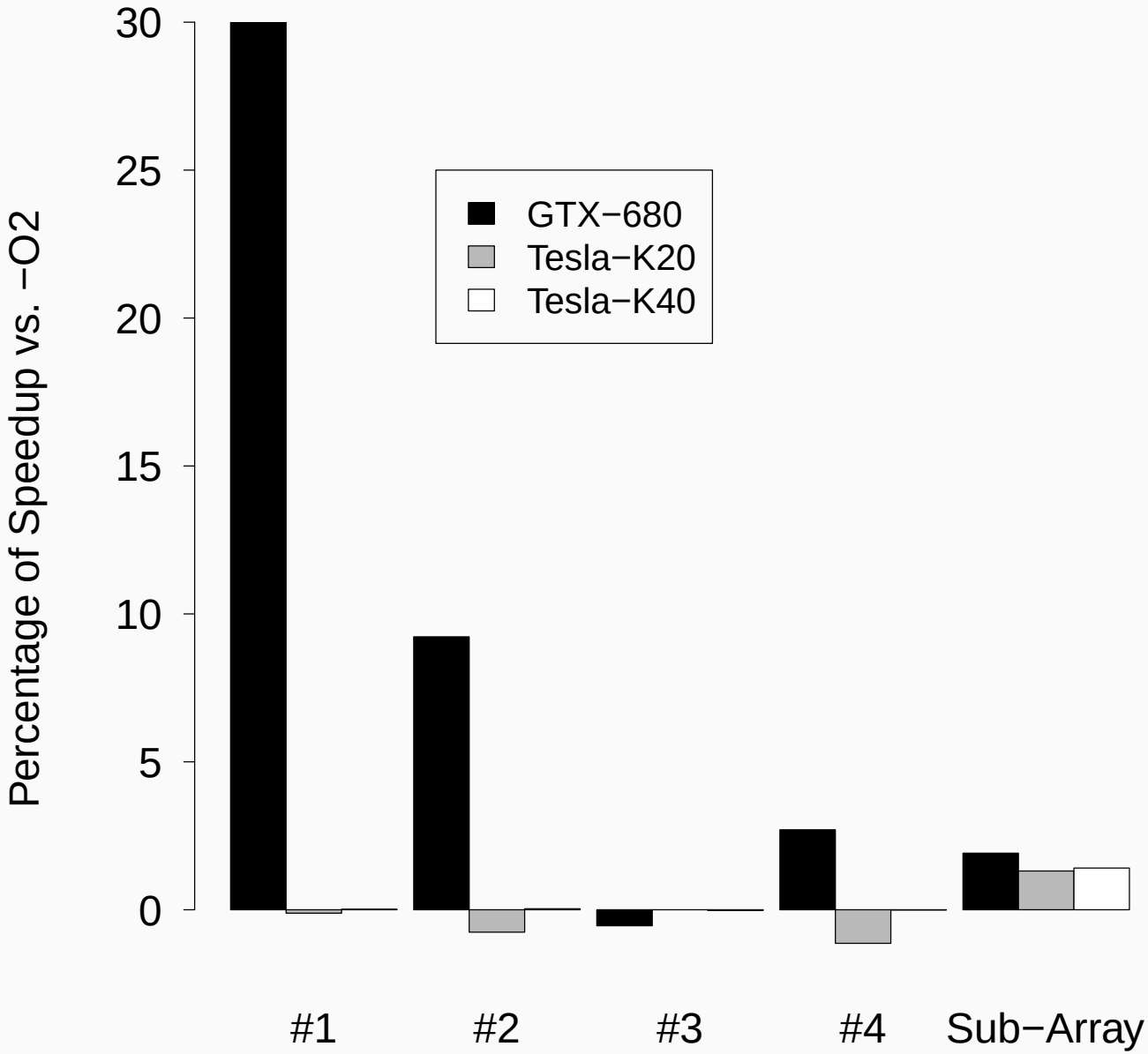
- #1: Non-Coalesced accesses to Global Memory
- #2: Coalesced accesses to Global Memory
- #3: #1, plus Shared Memory
- #4: #2, plus Shared Memory

EXPERIMENTS: MAXIMUM SUBARRAY

Find the maximum subsequence sum of an array ($N = 134217728$):

- 4096 threads
- 32 blocks of 128 threads

RESULTS: OPTIMIZATION

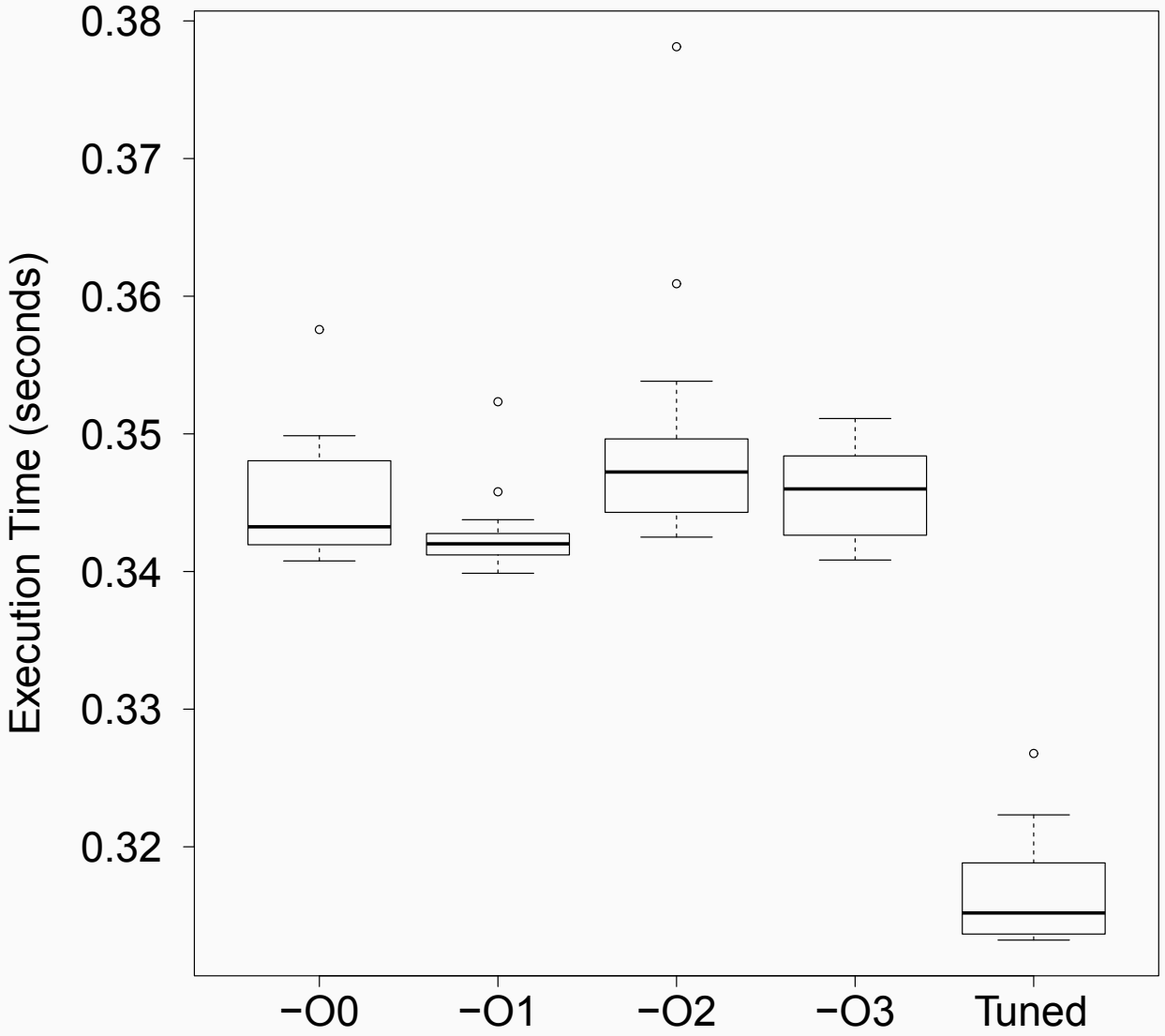


RESULTS: OPTIMIZATION

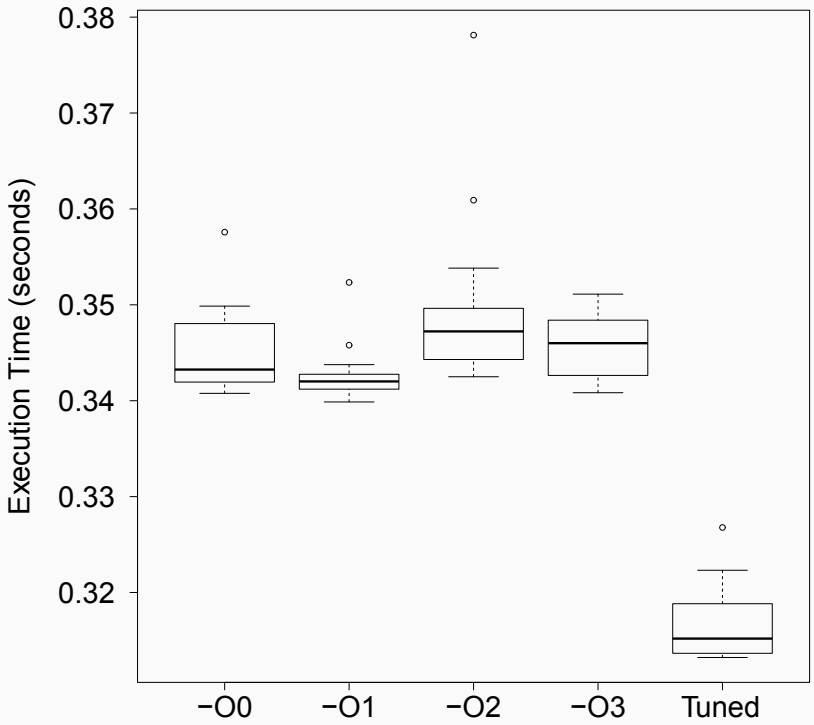
Why did the GTX-680 have the best results?

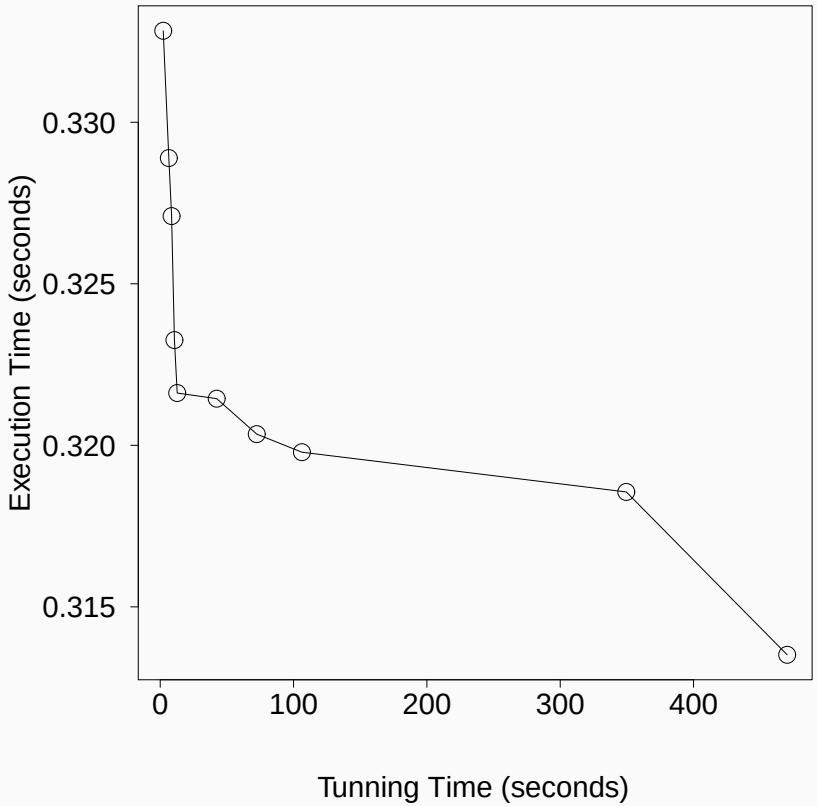
Model	c.c.	Global Memory	Bus	Bandwidth	L2	SM/Cores	Clock
GTX-680	3.0	2 GB	256-bit	192.2 GB/s	512 KB	8/1536	1006 Mhz
Tesla-K20	3.5	4 GB	320-bit	208 GB/s	1280 KB	13/2496	706 Mhz
Tesla-K40	3.5	12 GB	384-bit	276.5 GB/s	1536 KB	15/2880	745 Mhz

Results for optimization #2 in the GTX-680:

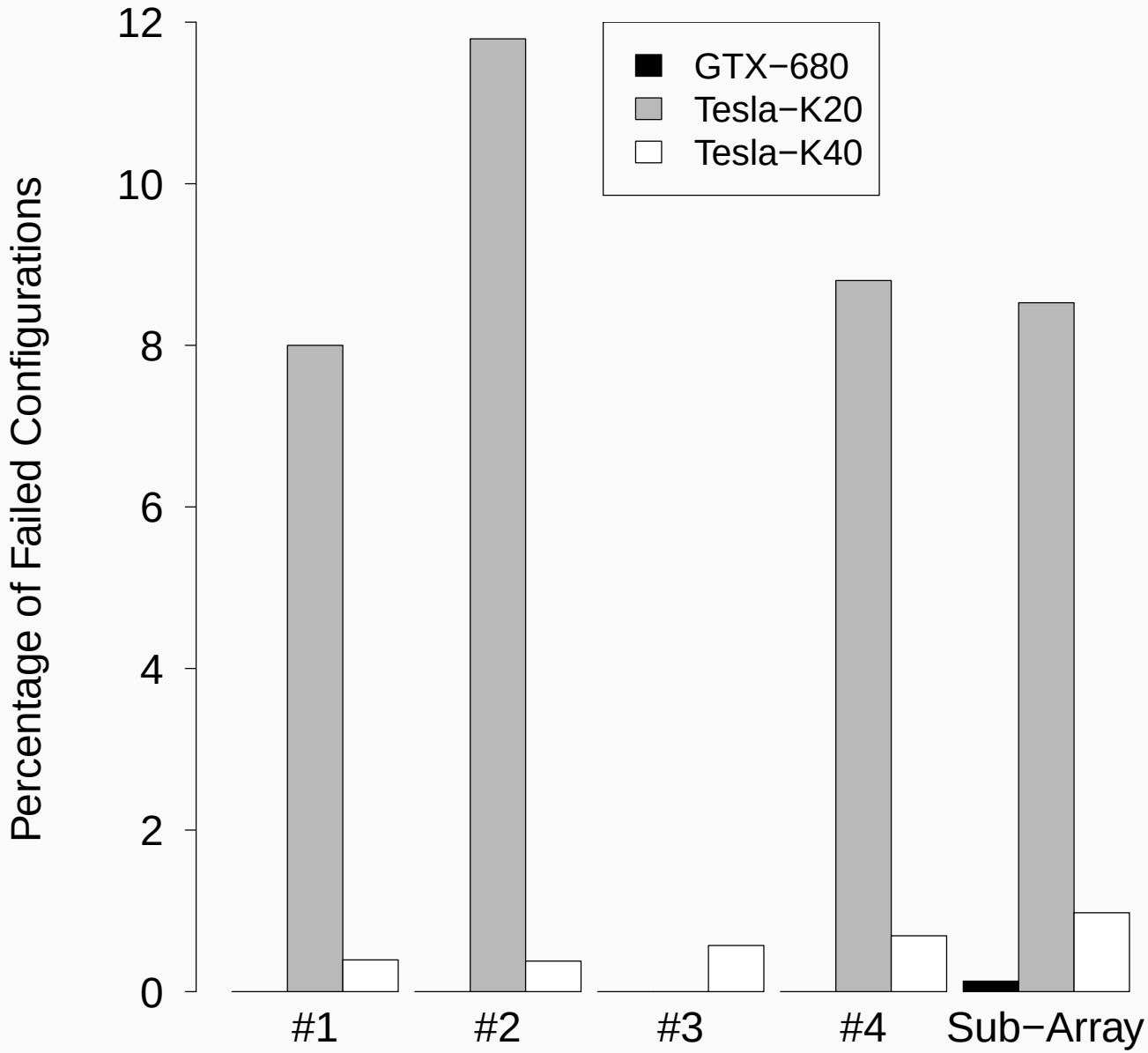


Results for optimization #2 in the GTX-680:





RESULTS: FAILED CONFIGURATIONS



RESULTS: FAILED CONFIGURATIONS

Who's guilty?

- sm_32, for #1 in the K20 and K40

CONCLUSION

- 30% speedup for #1 in the GTX-680

CONCLUSION

- 30% speedup for #1 in the GTX-680
- Different parameters for each GPU

CONCLUSION

- 30% speedup for #1 in the GTX-680
- Different parameters for each GPU
- Always assert the results

Thank you!

AUTOTUNING GPU COMPILER PARAMETERS WITH OPENTUNER

Pedro Bruel
phrb@ime.usp.br

Marcos Amarís
amaris@ime.usp.br

Alfredo Goldman
gold@ime.usp.br
29 de Setembro de 2015

Instituto de Matemática e Estatística
Universidade de São Paulo