15th International Workshop on Data Management on Modern Hardware (DaMoN), July 1, 2019, Amsterdam

Performance Analysis and Automatic Tuning of Hash Aggregation on GPUs

Viktor Rosenfeld viktor.rosenfeld@dfki.de
Sebastian Breß sebastian.bress@tu-berlin.de
Steffen Zeuch steffen.zeuch@dfki.de
Tilmann Rabl tilmann.rabl@hpi.de
Volker Markl volker.markl@tu-berlin.de

Introduction

Hash aggregation:

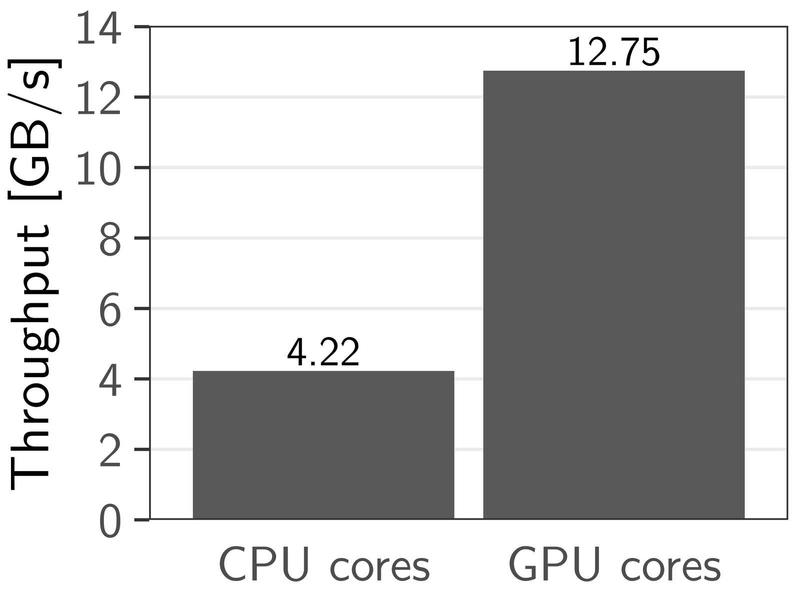
- Used to implement `GROUP BY` and `DISTINCT`.
- Can be significantly accelerated on GPUs.

Example:

- Query:

```
SELECT G, sum(V)
FROM R GROUP BY G
```

- Processor: AMD A10-7850K APU (CPU and GPU cores on same die).
- GPU cores 1.6 – 4.8 times faster.



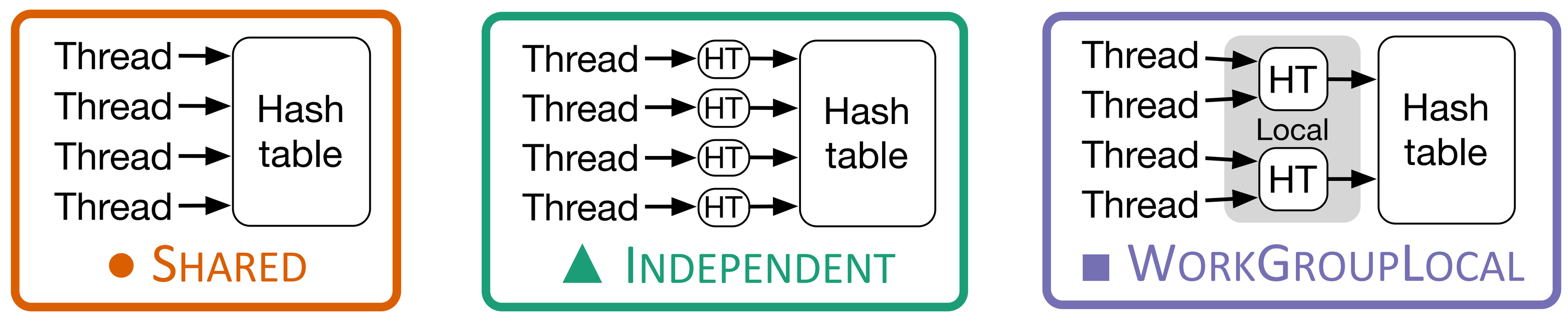
Previous work:

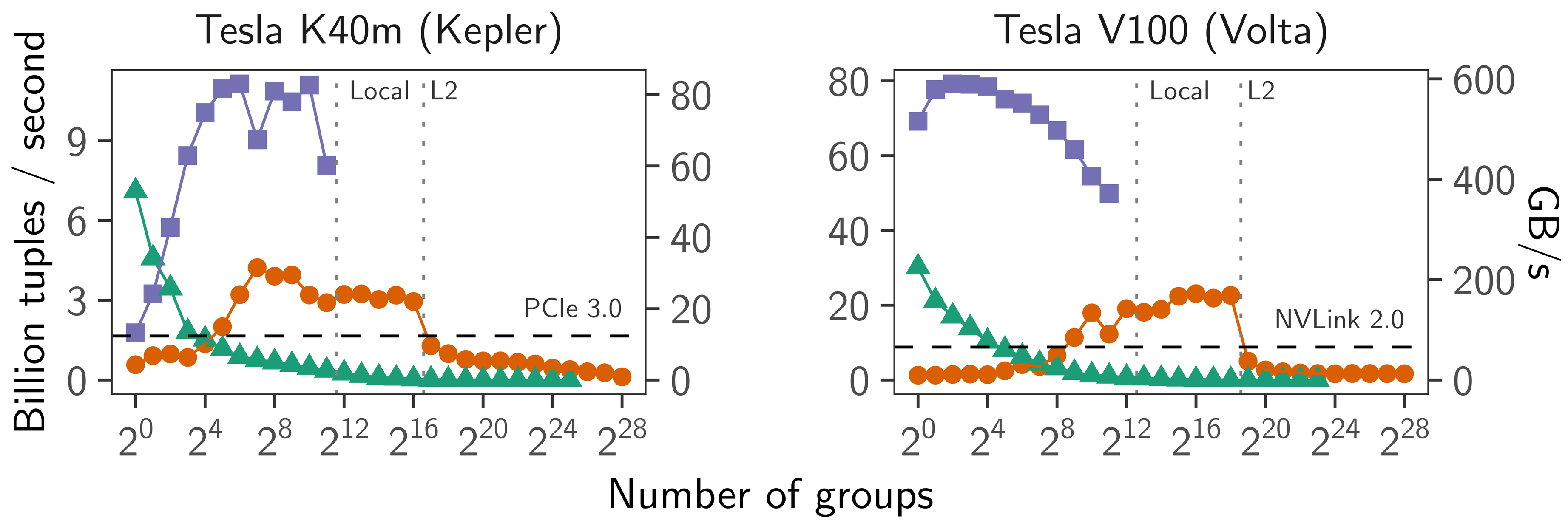
- Analyzed influence of parallelization strategy and thread configuration on aggregation performance.
- Formulated heuristics based on analysis of single NVIDIA Kepler GPU.

Our contributions:

1. Performance analysis on six diverse AMD and NVIDIA GPUs.
2. Automatic tuning of execution parameters at runtime.

Parallelization strategies





(1) INDEPENDENT aggregation is not competitive on newer GPUs which implement fast atomics on local memory.

(2) Aggregation throughput limited by global GPU memory latency (and not transfer bandwidth) when hash table exceeds L2 cache.

Thread configuration search space

Work items per work group (rows) × Work groups per compute unit (columns)

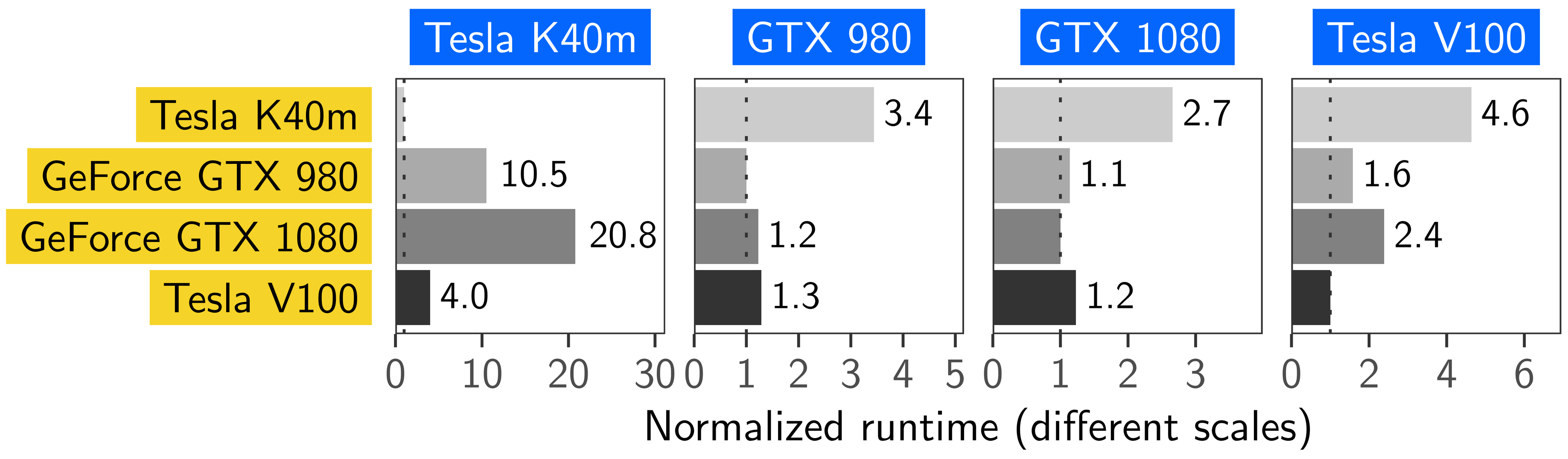
	1		4		16		64		256		1024
1024	1.42	1.51	1.50	1.51	1.50	1.51	1.49	1.48	1.49	1.46	1.49
	1.10	1.42	1.51	1.51	1.51	1.51	1.51	1.50	1.49	1.49	1.48
256	1.14	1.10	1.42	1.51	1.52	1.52	1.51	1.51	1.51	1.50	1.51
	1.59	1.14	1.09	1.42	1.52	1.52	1.52	1.51	1.50	1.50	1.50
64	3.08	1.63	1.14	1.10	1.42	1.42	1.41	1.41	1.41	1.40	1.40
	6.04	3.13	1.63	1.14	1.10	1.10	1.10	1.09	1.10	1.10	1.09
16	8.71	4.52	2.34	1.30	1.01	1.01	1.01	1.00	1.00	1.01	1.00
	13.54	6.97	3.64	1.90	1.14	1.14	1.13	1.13	1.13	1.12	1.12
4	21.18	10.68	5.54	2.88	1.54	1.53	1.50	1.50	1.50	1.50	1.49
	33.65	16.86	8.62	4.43	2.31	2.28	2.25	2.25	2.24	2.24	2.22
1	54.43	27.12	13.69	7.00	3.60	3.56	3.54	3.53	3.52	3.51	3.48

- Number of threads determines execution runtime.
- **Performance plateaus:** Adjacent thread configurations have similar runtimes.
- AMD GPUs exhibit high degree of runtime variation.

(3) Single local minimum when we account for runtime variation.

Performance penalty

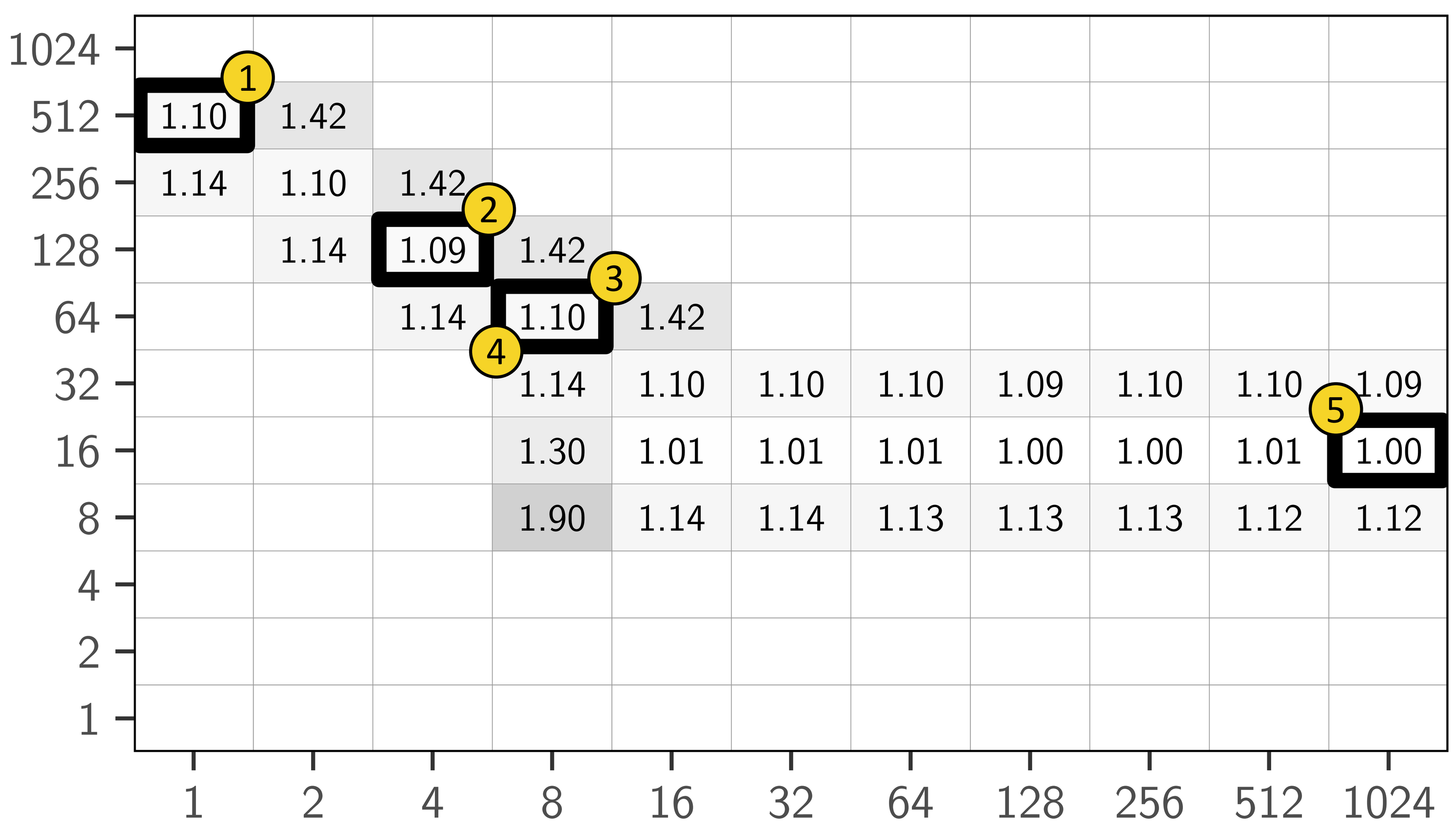
A configuration optimized for a specific GPU is executed on another GPU.



(4) The optimal thread configuration is highly GPU-dependent.

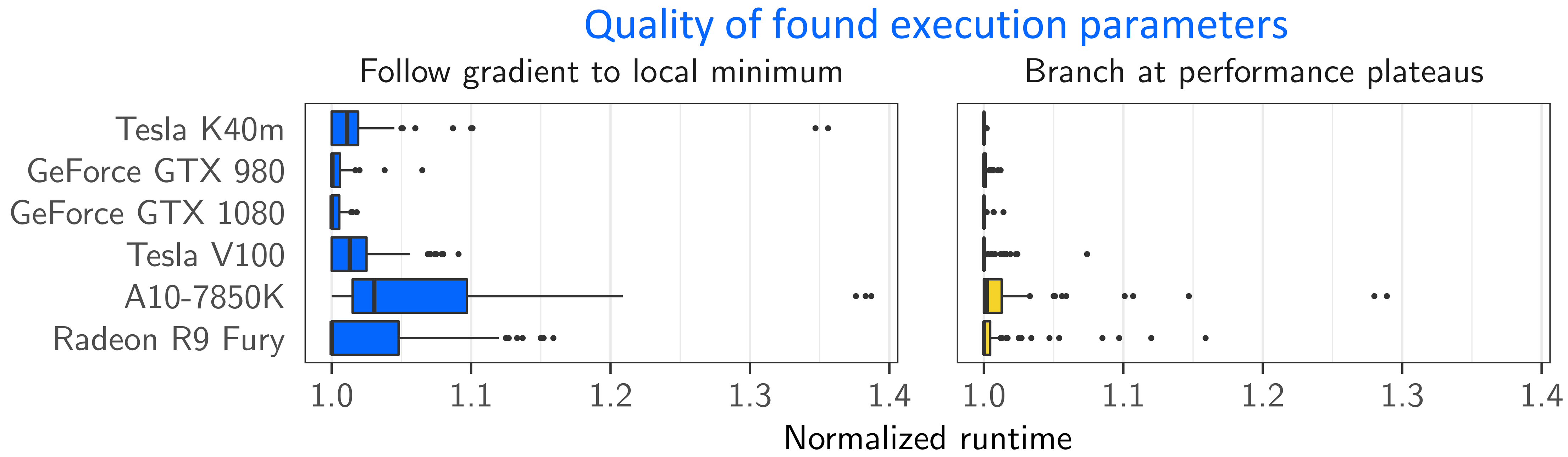
(5) Suboptimal configurations lead to large performance penalties.

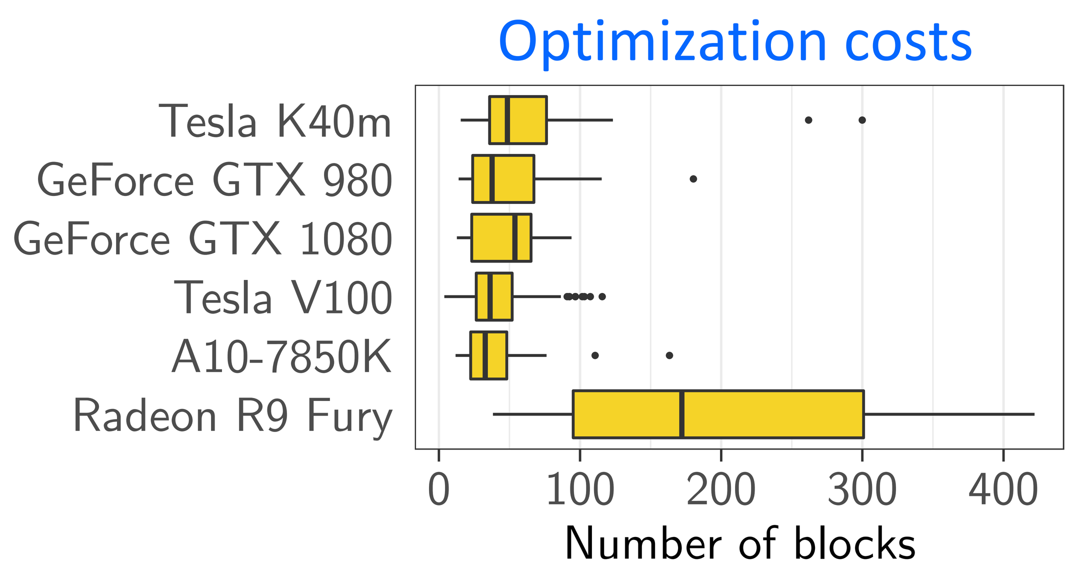
Optimization algorithm

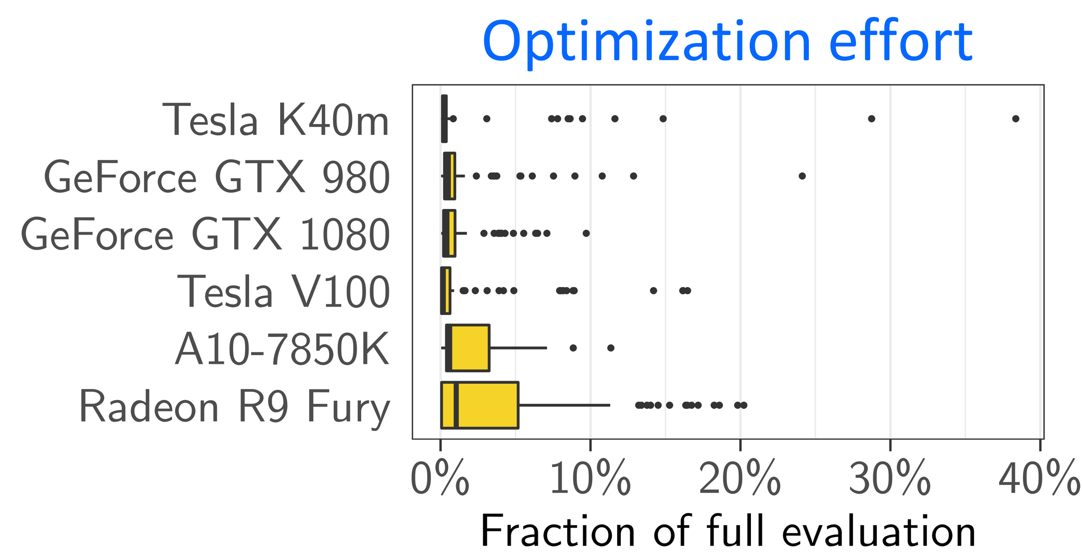


1. Start with initial thread configuration.
2. **Follow gradient** in search space to local minimum.
3. **Branch** search path **at performance plateaus**.
4. Prune slow branches.
5. Stop at minimum when there are no more branches.

Evaluation







Main takeaways

(1) Heuristics to select execution parameters, which are derived from analyzing a single GPU, are not generalizable to other GPUs.

(2) We can optimize execution parameters by following the gradient in the search space and branching the search at performance plateaus.

Technische Universität Berlin
German Research Center for Artificial Intelligence
This work was funded by the EU project E2Data (780245), the DFG Priority Program Scalable Data Management for Future Hardware (MA4662-5), and the German Ministry for Education and Research as BBDC II (01IS18025A).
SPP2037 SCALABLE DATA MANAGEMENT FOR FUTURE HARDWARE
BERLIN BIG DATA CENTER