DRUTO: Upper-Bounding Silent Data Corruption Vulnerability in GPU Applications

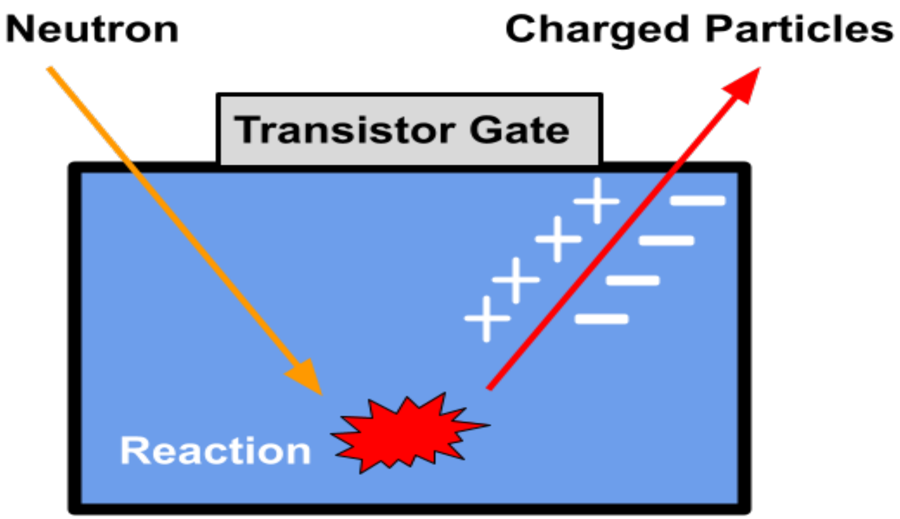
Md Hasanur Rahman, Sheng Di, Shengjian Guo, Xiaoyi Lu, Guanpeng Li, Franck Cappello

Outline

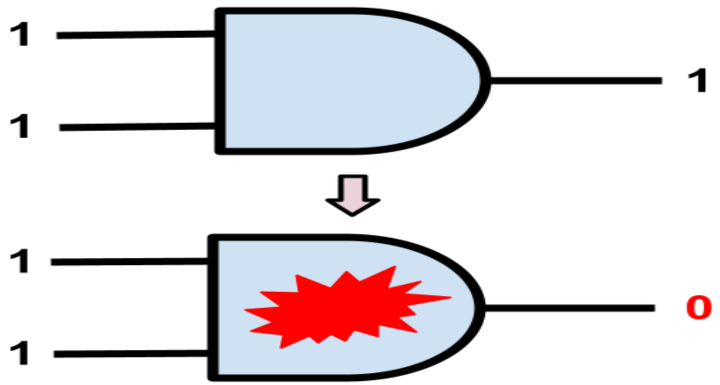
- Silent Data Corruption Due to Soft Errors
- Fault Model
- Limitation of Existing Approaches
- Our Methodology
- Evaluation
- Conclusion

Soft Errors

How soft errors occur



Example



Soft errors trend

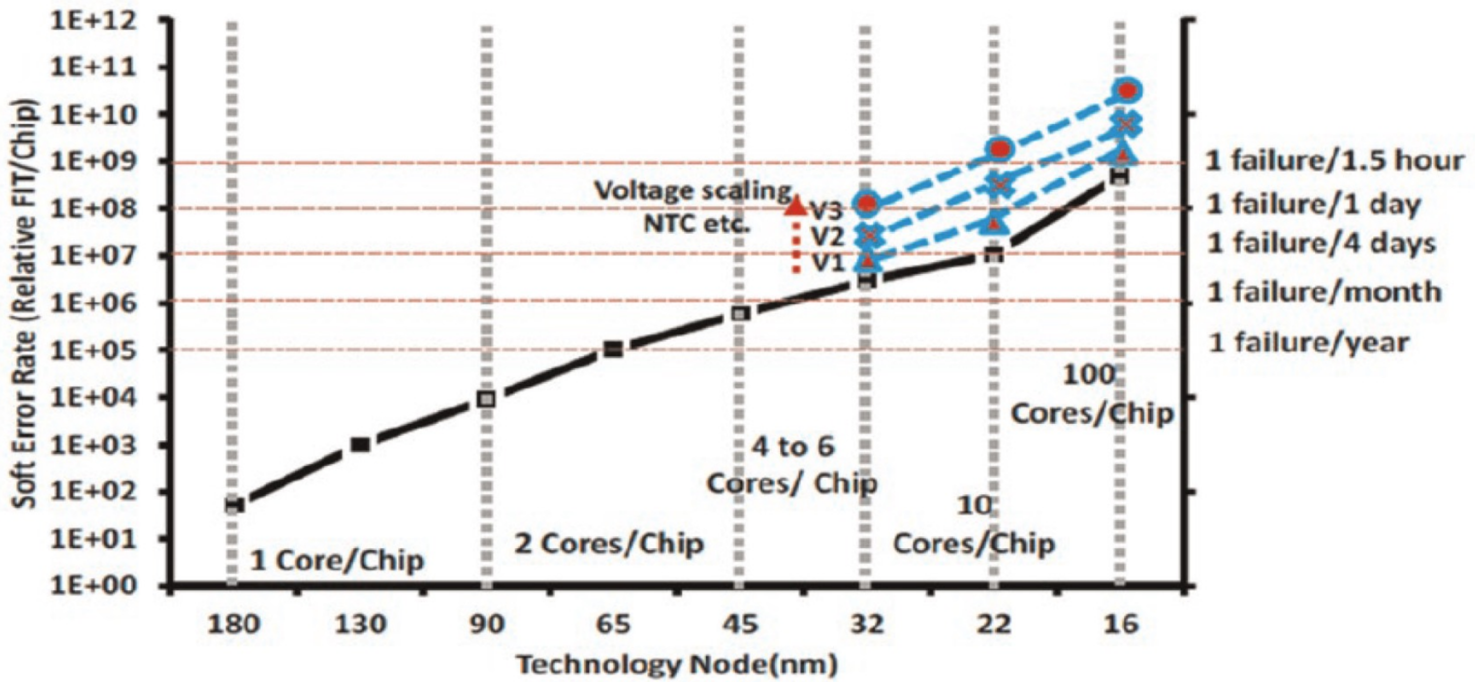
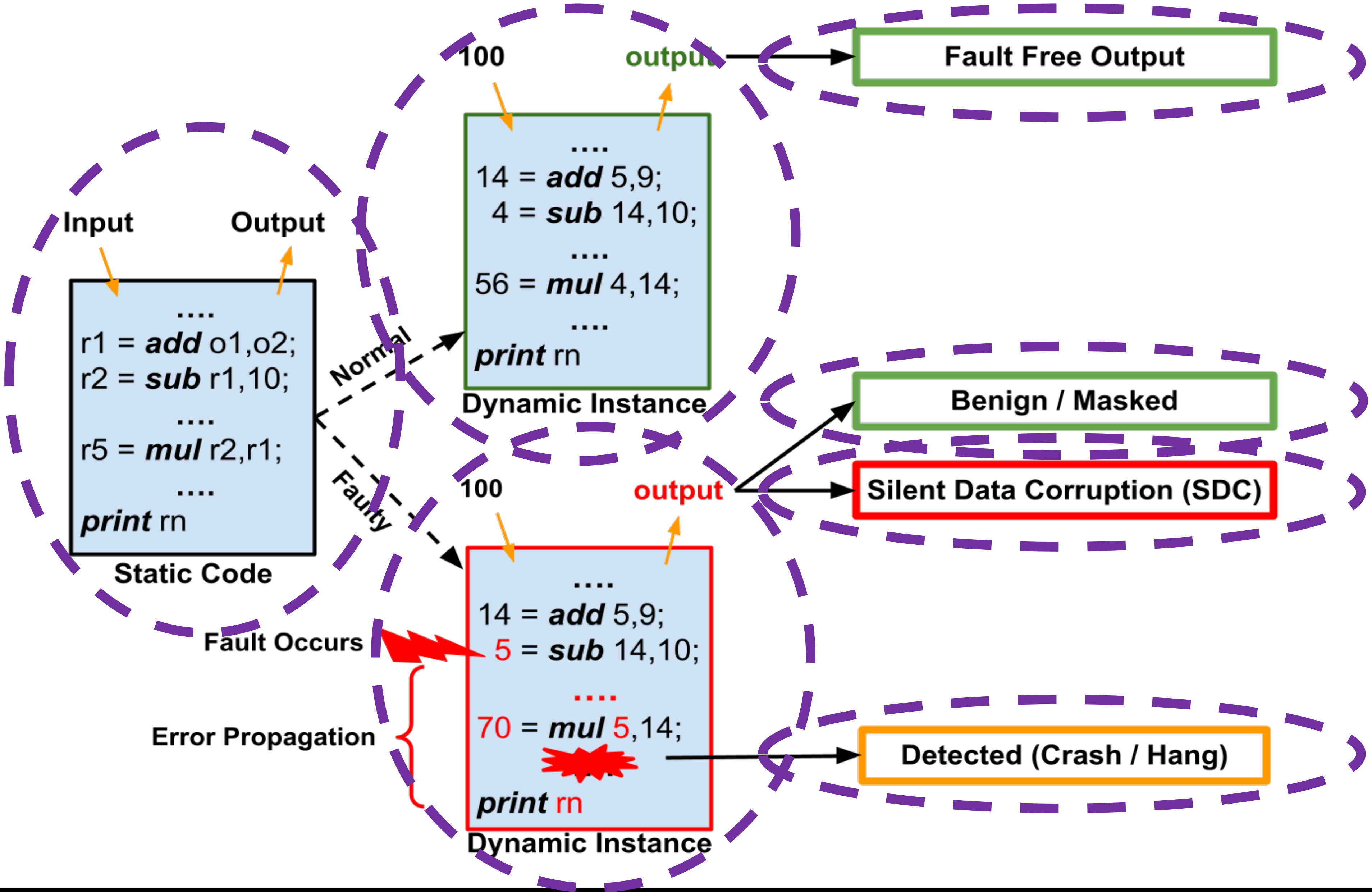


Figure Taken from Venkatesha et al. [5]

Silent Data Corruption (SDC)



Motivation

- Fault Injection (FI) is usually conducted to simulate soft error
- SDC Evaluation via FI has been typically conducted with reference inputs
- Higher-than-expected SDC is observed in Meta and Google Research [4,29]
- Goal: Find the SDC-bound input that approximates SDC upper-bounding in GPU
- Challenges
  - Input search space is huge
  - SDC characteristics vary at both kernel and thread level

Fault Model

- Where (fault site)
  - Inject faults in processor registers and core functional units
  - Memories and caches are protected by error correcting codes (ECC)
  - Errors in control logic can be checked by control flow checking techniques
- What (type)
  - A single bit flip in the destination register of a target dynamic instruction [45]
- How (to simulate)
  - Pause execution at target fault site, inject a bit flip, and let the execution to completion

Existing Approaches and Limitations

Random FI

- Evaluate each possible input with thousands of random FIs
- Single FI takes full program cycle plus injecting fault at target fault site
- Fault injection is time consuming
- Millions of inputs; Impractical approach

Existing Approaches and Limitations

Peppa-X [25]

- Finds SDC-bound input for a CPU program
- Insight: some program regions are always vulnerable regardless of input changes
- Dynamic analysis-based fuzzing technique
- GPU program is different from CPU
- Method does not hold for GPU programs

Pathfinder	Needle	Particlefilter	CoMD	Hpccg	Xsbench	FFT
0.92	0.79	0.90	0.90	0.96	0.59	0.77

Table: Correlation Coefficient between Ranking of Per-instruction SDC Probabilities with Different Inputs in CPU [25].

Pathfinder	Needle-K1	Needle-K2	Particlefilter	FFT	Backprop-K1	Backprop-K2
0.59	0.37	0.33	0.79	0.60	0.27	0.64
BFS-K1	**BFS-K2**	**Jmeint**	**BlackScholes**	**2DCONV**	**GEMM**	**MVT-K1**
0.84	0.74	0.73	0.25	0.69	0.75	0.77

Table: Correlation Coefficient between Rankings of Per-instruction SDC Probabilities with Different Inputs in GPU.

Existing Approaches and Limitations

How to efficiently build the input ranking?

How to avoid FI to assess SDC vulnerability?

Is there any relation between thread and kernel level SDC characteristics?

Observations

SDC Variation among Inputs

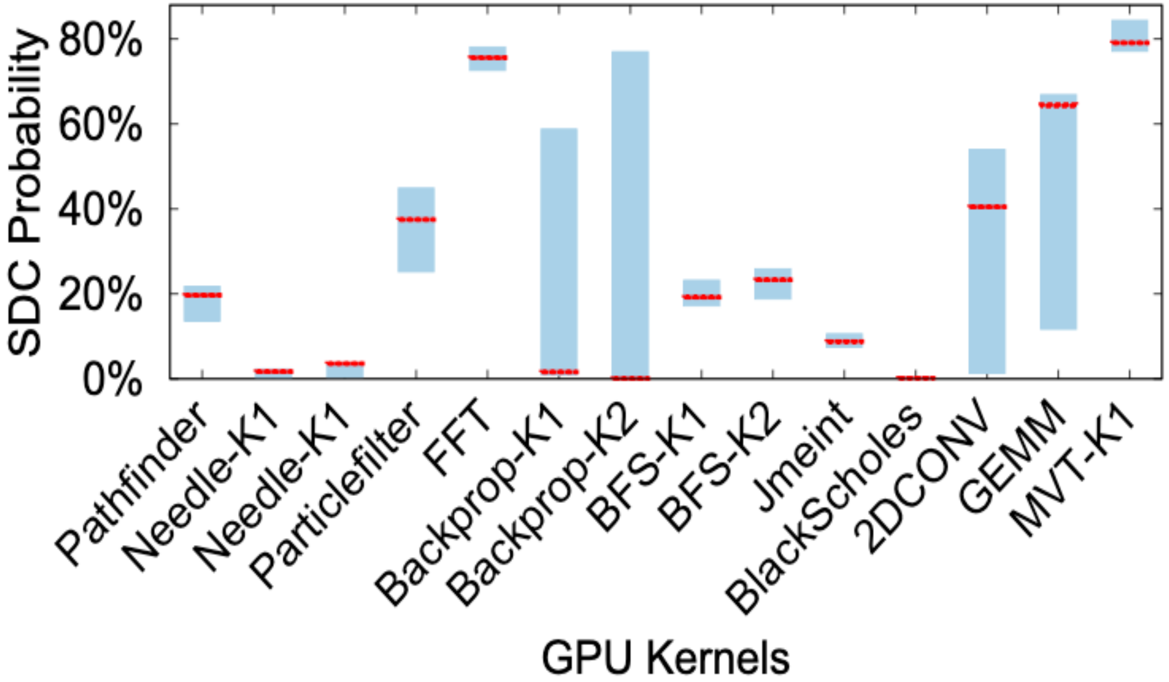


Figure: Overall SDC Probability Range for GPU Kernels.

Observations

SDC Variation among Inputs

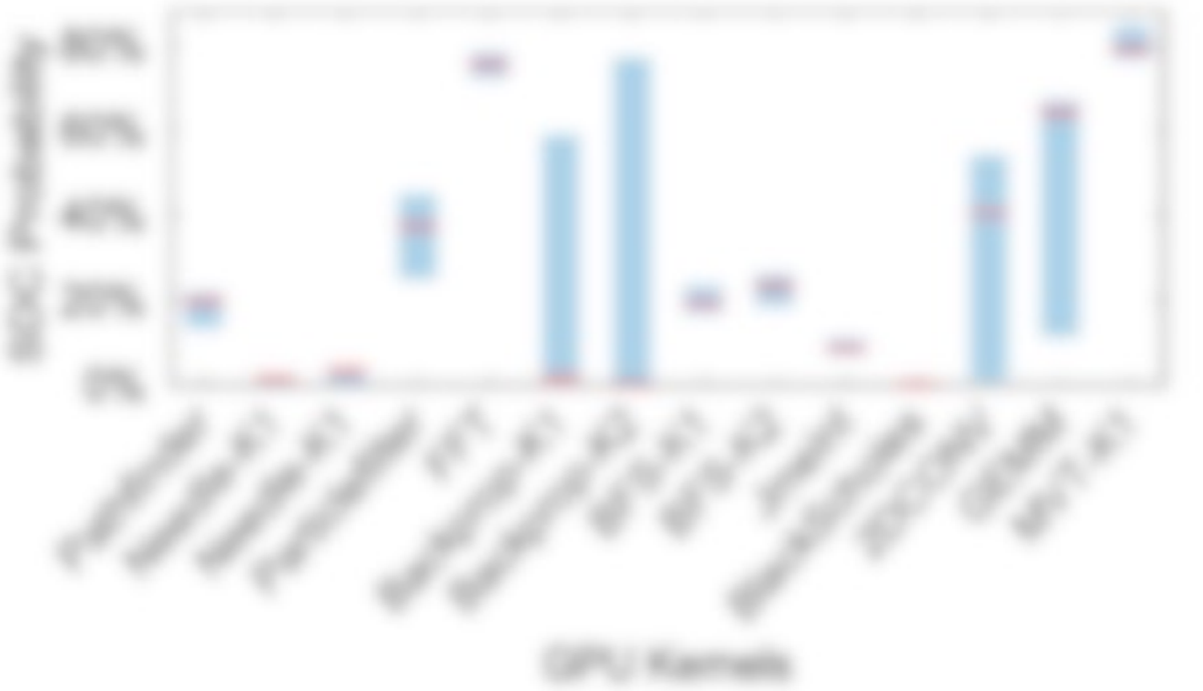


Figure: Overall SDC Probability Range for GPU Kernels.

O1: Dominant Thread SDC and Kernel SDC

SDC probability of a kernel under different inputs can be ranked by SDC probability of its dominant thread under those inputs.

Pathfinder	Needle-K1	Needle-K2	Particlefilter	FFT	Backprop-K1	Backprop-K2
0.77	0.78	0.75	0.83	0.83	0.86	0.86

BFS-K1	BFS-K2	Jmeint	BlackScholes	2DCONV	GEMM	MVT-K1
0.92	0.81	0.87	0.80	0.80	0.81	0.71

Table: Correlation Coefficient between SDC Probability of Dominant Thread and Kernel across Inputs.

Observations

O2: Thread SDC and Dynamic Count

Threads' ranking based on dynamic instruction counts approximately follows threads' ranking based on SDC prob. with inputs

Pathfinder	Needle-K1	Needle-K2	Particlefilter	FFT	Backprop-K1	Backprop-K2
0.77	0.60	0.69	0.62	0.78	0.79	0.75
BFS-K1	**BFS-K2**	**Jmeint**	**BlackScholes**	**2DCONV**	**GEMM**	**MVT-K1**
0.75	0.69	0.78	0.61	0.92	0.68	0.75

Table: Average Correlation Coefficient between Thread SDC Probability and Its Dynamic Execution Count across Inputs.

Observations

O2: Thread SDC and Dynamic Count

Threads' ranking based on dynamic instruction counts approximately follows threads' ranking based on SDC prob. with inputs

Table: Average Correlation Coefficient between Thread SDC Probability and Its Dynamic Execution Count across Inputs.

Overall Relationship for Approximation

Dynamic instruction count of a kernel's dominant thread can approximate the relative ranking of an input in terms of kernel SDC probability

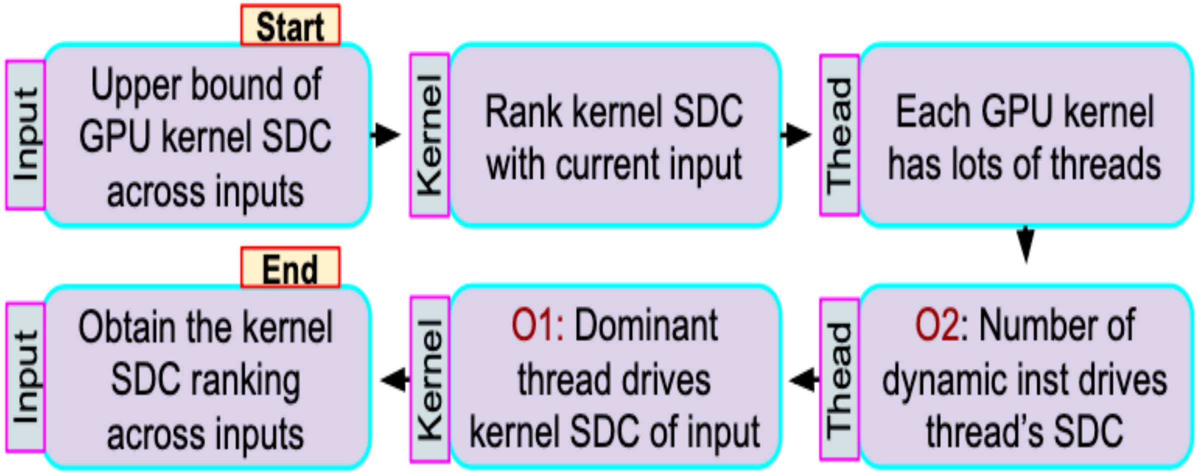


Figure: Contribution of Each Observation to Approximating Upper Bound Kernel SDC Prob. across Program Inputs.

DRUTO Design Workflow

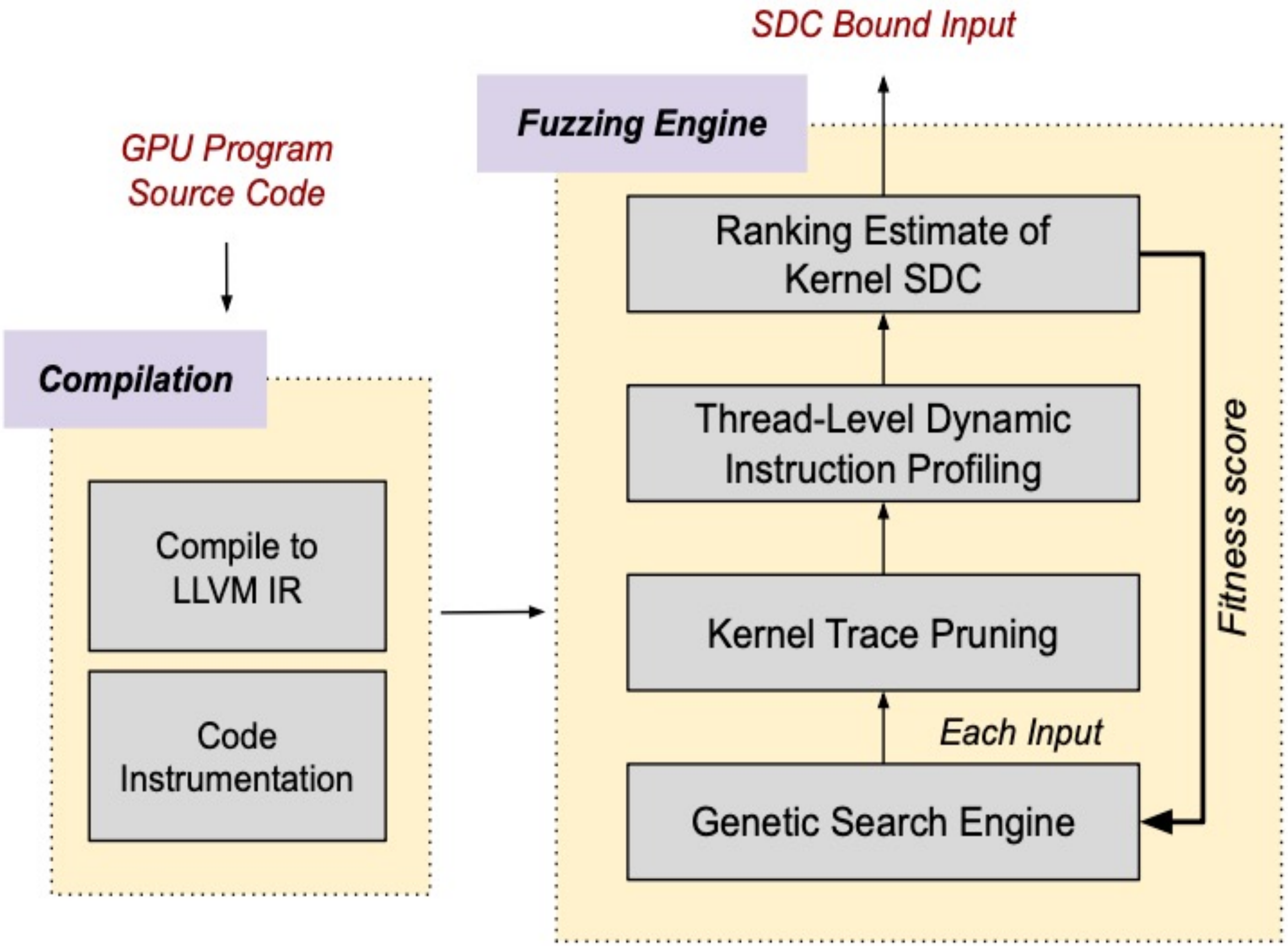


Figure: DRUTO Design Workflow.

Evaluation: Accuracy

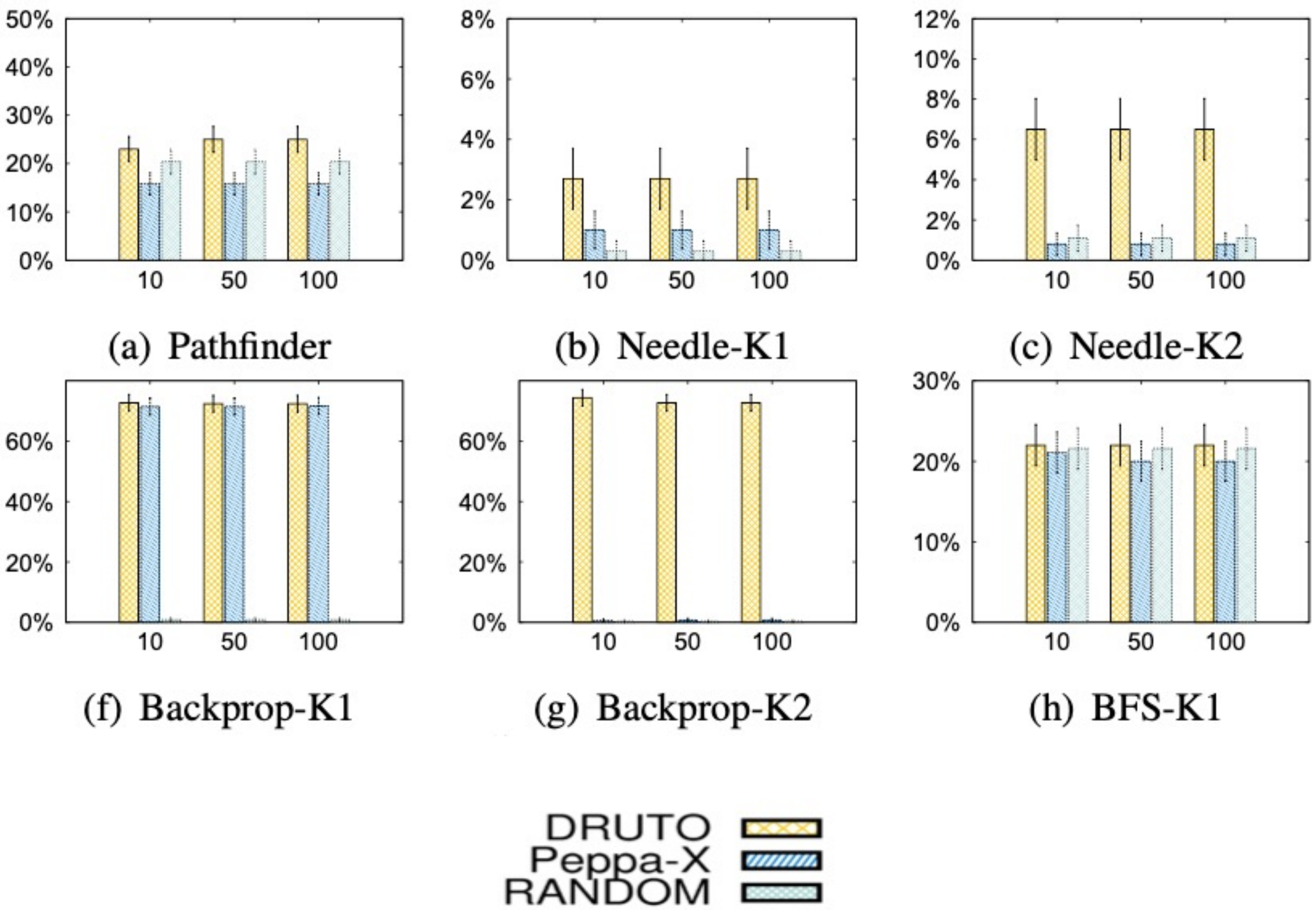


Figure: The Comparison of Approximating Upper-Bounding Kernel SDC Probabilities of DRUTO, PEPPA-X and RANDOM.

Evaluation: Accuracy

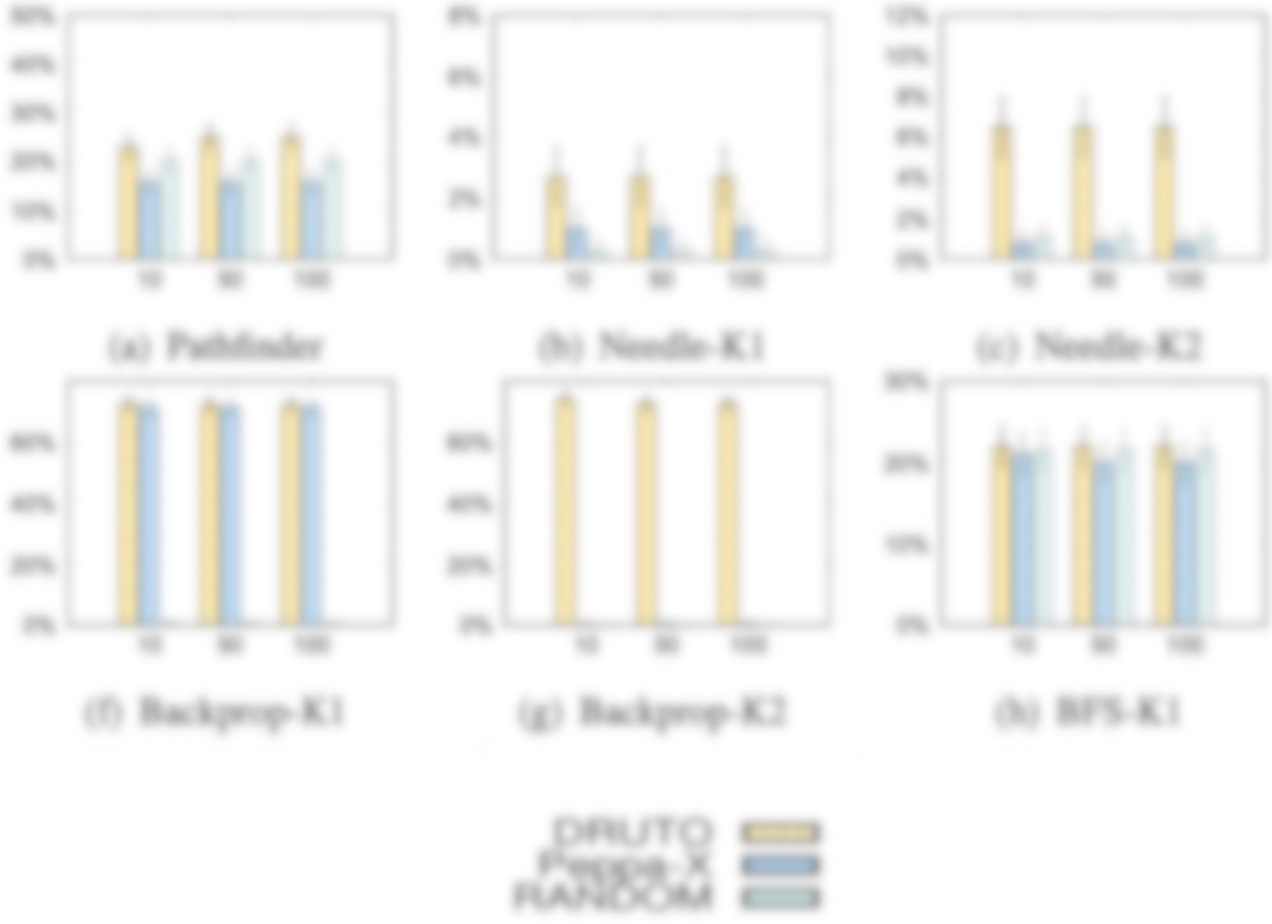


Figure: The Comparison of Approximating Upper-Bounding Kernel SDC Probabilities of DRUTO, PEPPA-X and RANDOM.

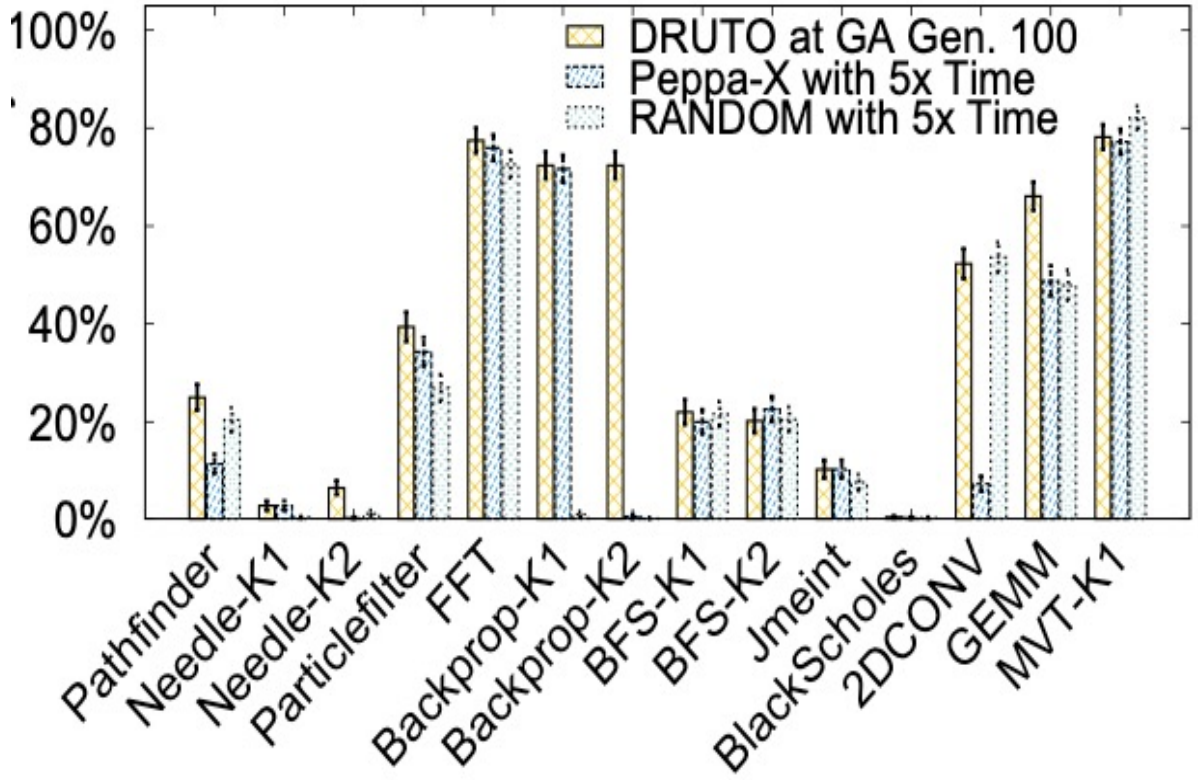


Figure: Accuracy Comparison between DRUTO and Baselines with 5x More Search Time.

Evaluation: Efficiency

Kernel	Path-finder	Needle-K1	Needle-K2	Particle-filter	FFT	Backprop-K1	Backprop-K2
DRUTO	8.74	5.89	5.85	3.35	2.44	10.04	10.28
PEPPA-X	269.88	376.63	300.74	26.20	454.58	847.42	395.32

Kernel	BFS-K1	BFS-K2	Jmeint	Black-Scholes	2DCONV	GEMM	MVT-K1
DRUTO	4.60	4.50	7.45	3.05	6.60	3.24	3.43
PEPPA-X	13.58	7.76	633.82	197.51	40.14	15.78	56.78

Figure: Comparison of Average Per-input Evaluation Time (in sec) by DRUTO and Peppa-X up to 100 Generations in Genetic Algorithm.

Conclusion

- Propose an automated compiler-based search technique DRUTO
- Can approximate the upper-bounding of GPU kernel SDC
- Leverage resilience characteristics of representative threads
- Does not need fault injection during the entire search
- Existing techniques are given 5x more search time
- They still cannot reach the level of DRUTO SDC upper-bounding