SC20

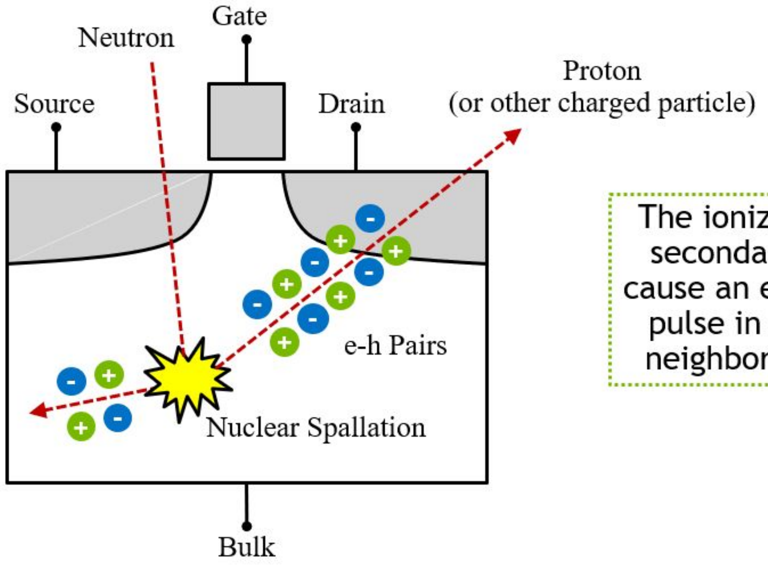
Everywhere we are | more than hpc.

GPU-Trident: Efficient Modeling of Error Propagation in GPU Programs

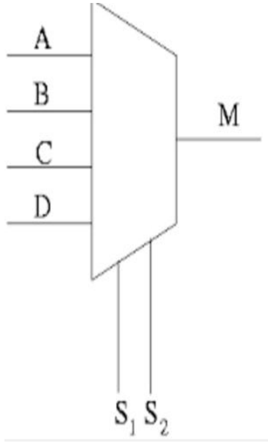
November 19, 2020

Abdul Rehman Anwer
Guanpeng Li
Karthik Pattabiraman
Siva Hari
Michael Sullivan
Timothy Tsai

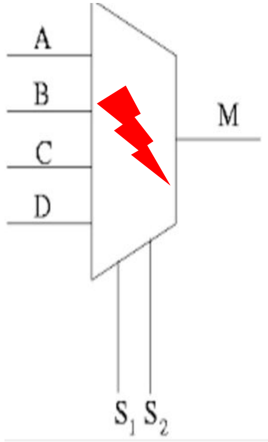
Soft Errors



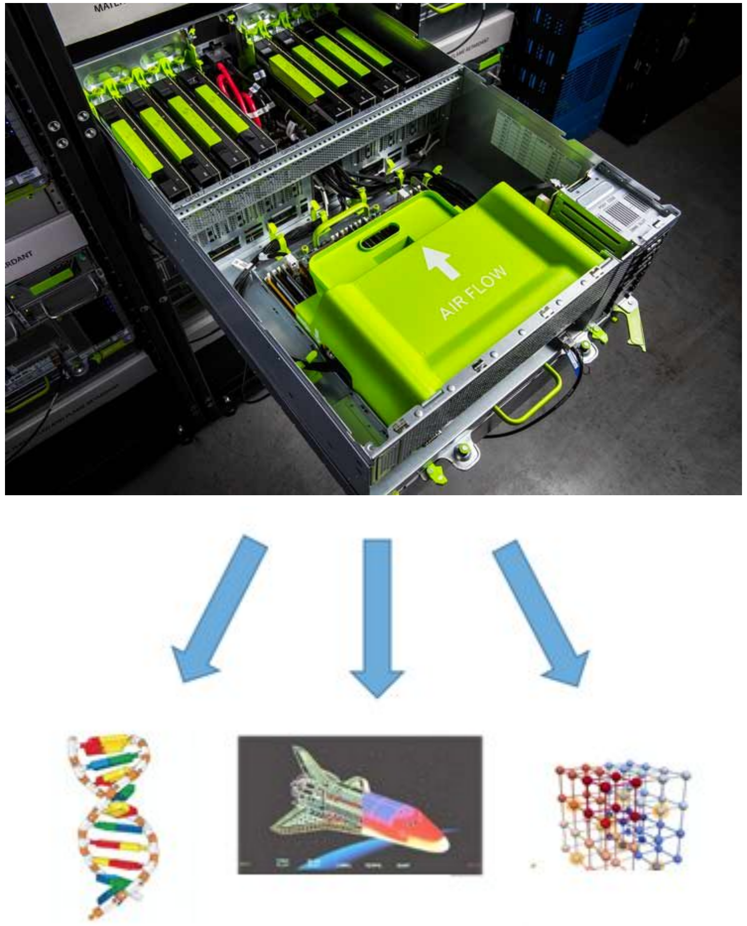
The ionizing track left by secondary particles can cause an erroneous current pulse in one or multiple neighboring transistors.



= 00

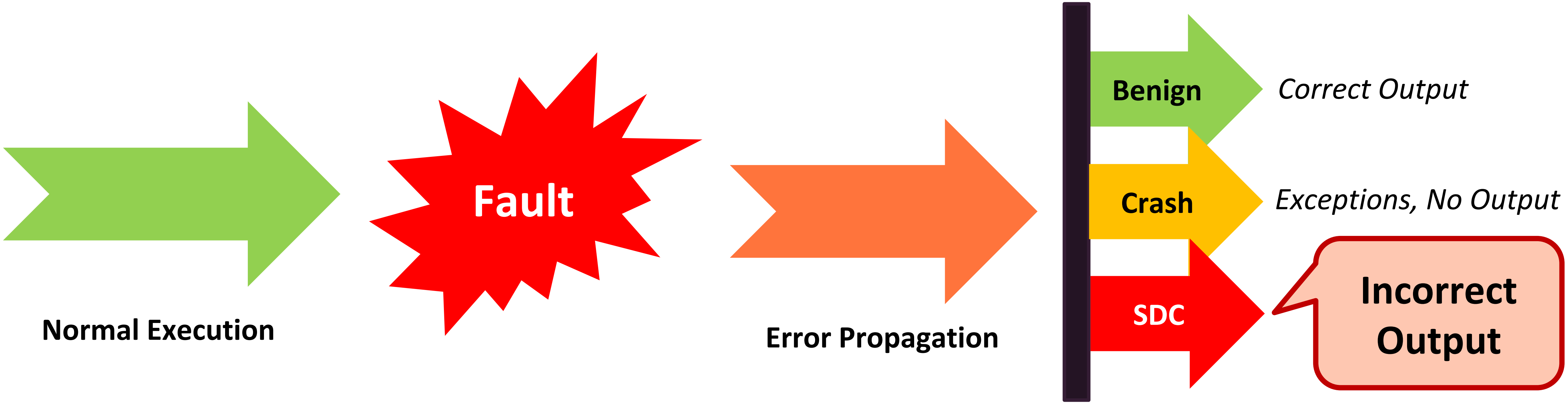


= 01



Silent Data Corruption (SDC)

e.g. Amazon S3 Incident



Traditional Solutions

- Error Correction Code (ECC)
- Hardware means
  - **Circui**

    Waste … i.e.,

    betwe

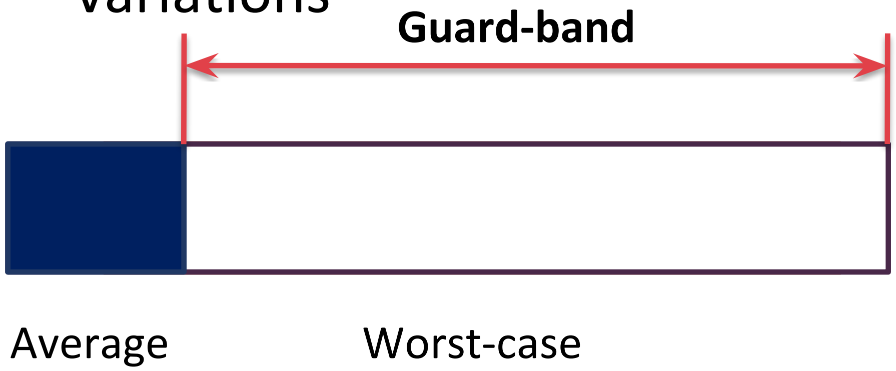
    worst-case widens due to

    variations

    …gy

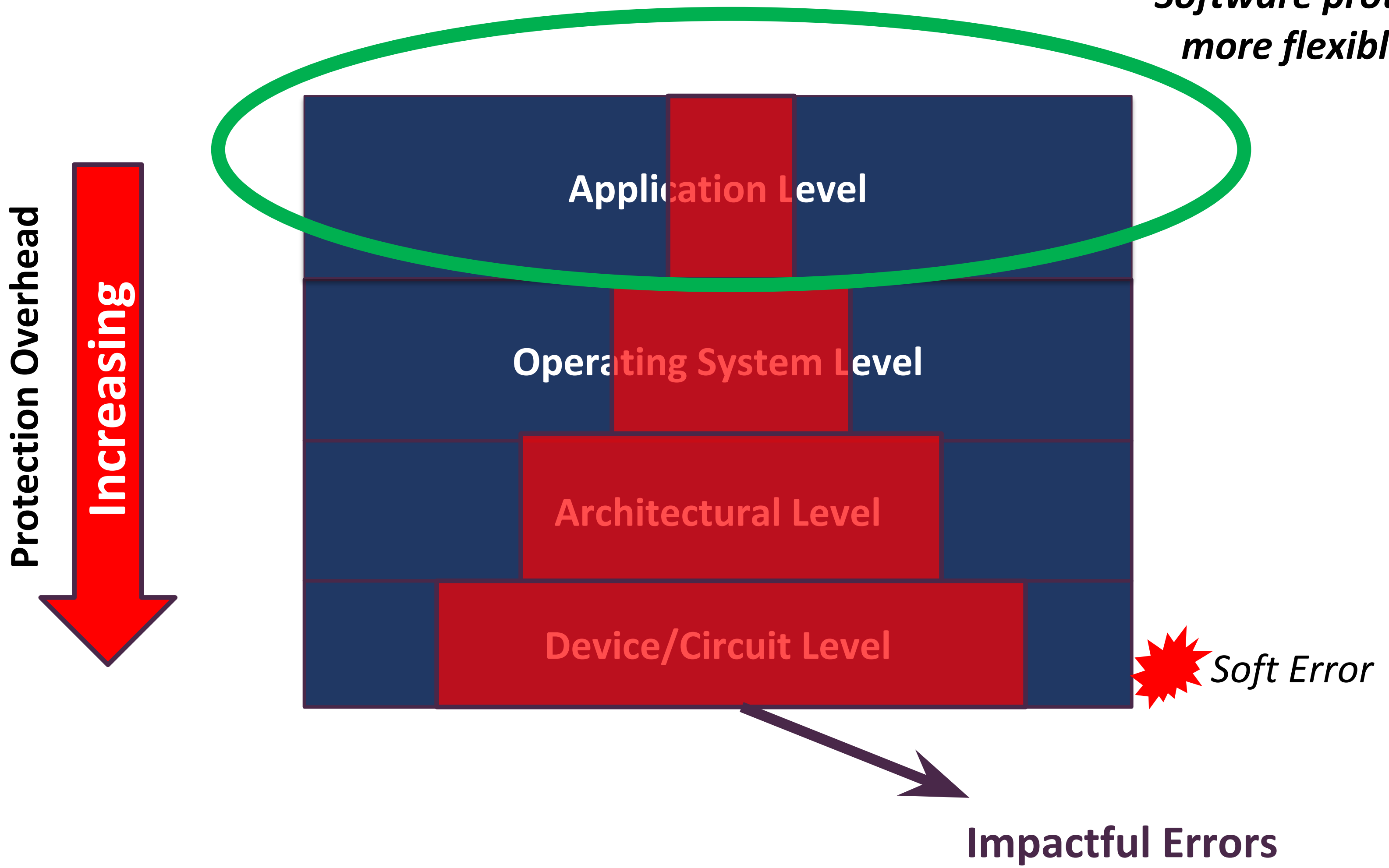
    consumption

Very expensive to deploy in practice!

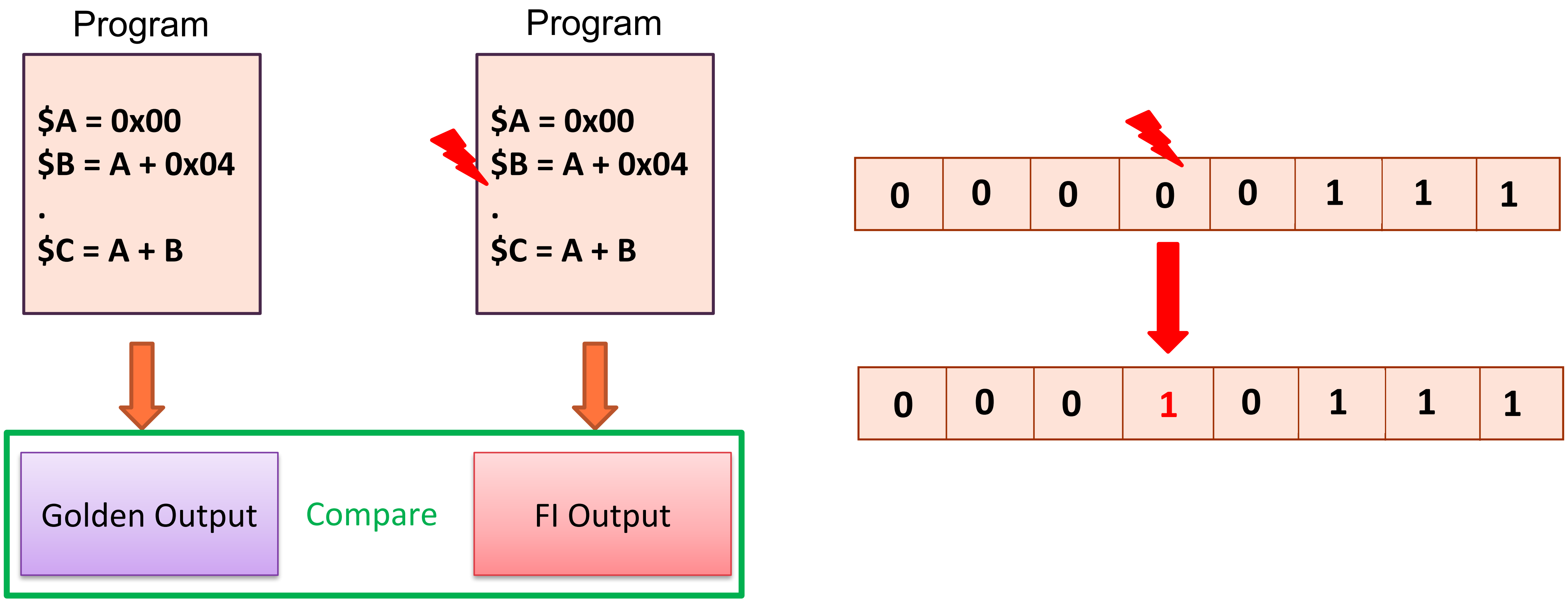


Software Solutions

Software protection techniques are more flexible and cost-effective!



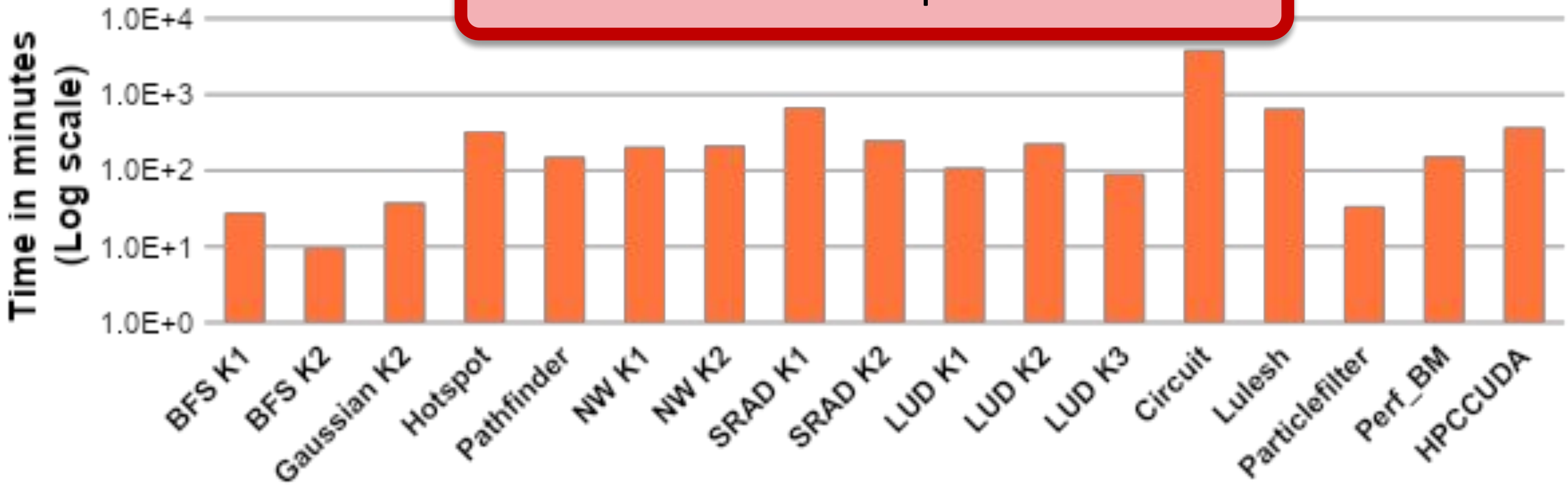
Fault Injection



FI injection – Overhead

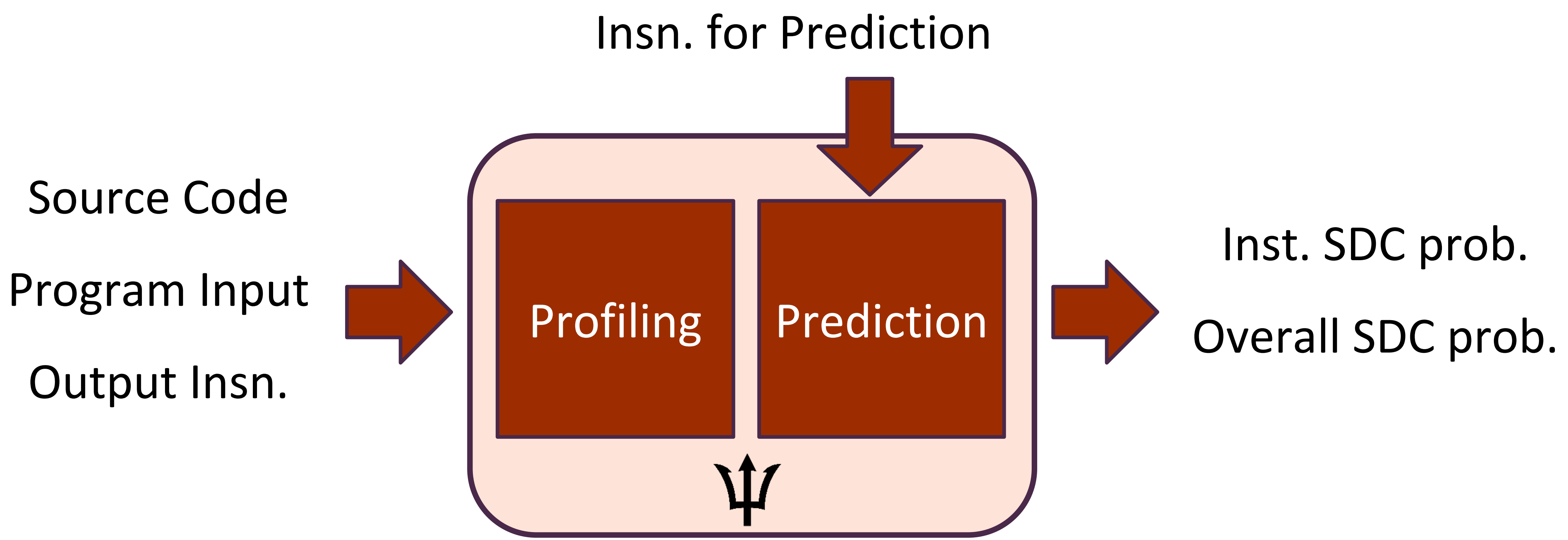
- GPU-Qin [ISPASS’14, Fang], LLFI-GPU [SC’16,Li], SASSIFI [ISPASS’17, Hari]
- Highly inefficient, as it has to be repeated if application is updated

~7 hrs. for 100 faults per instruction



*Timings obtained from our experiments using LLFI-GPU, other works report similar timings

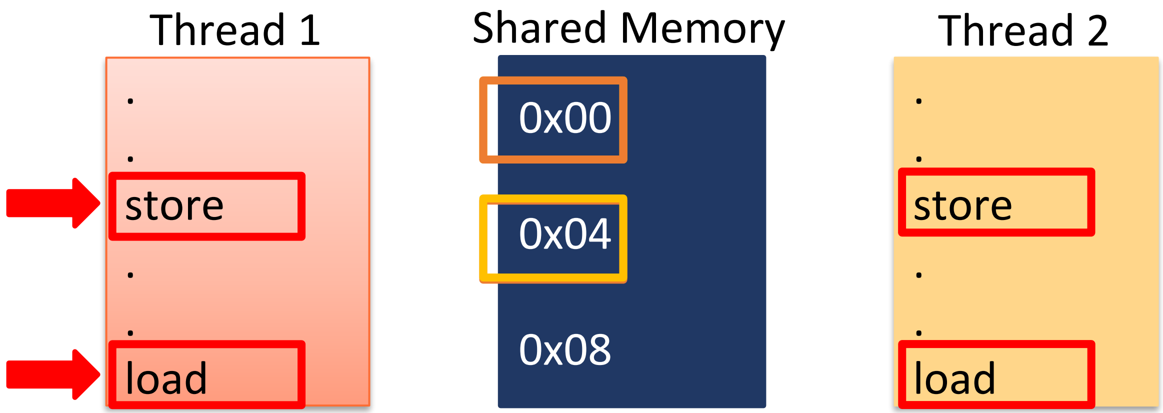
Trident – for CPU (DSN'18, Li)



GPU - Challenges

- Execution of GPU applications is inherently multi threaded
- Threads frequently communicate with each other

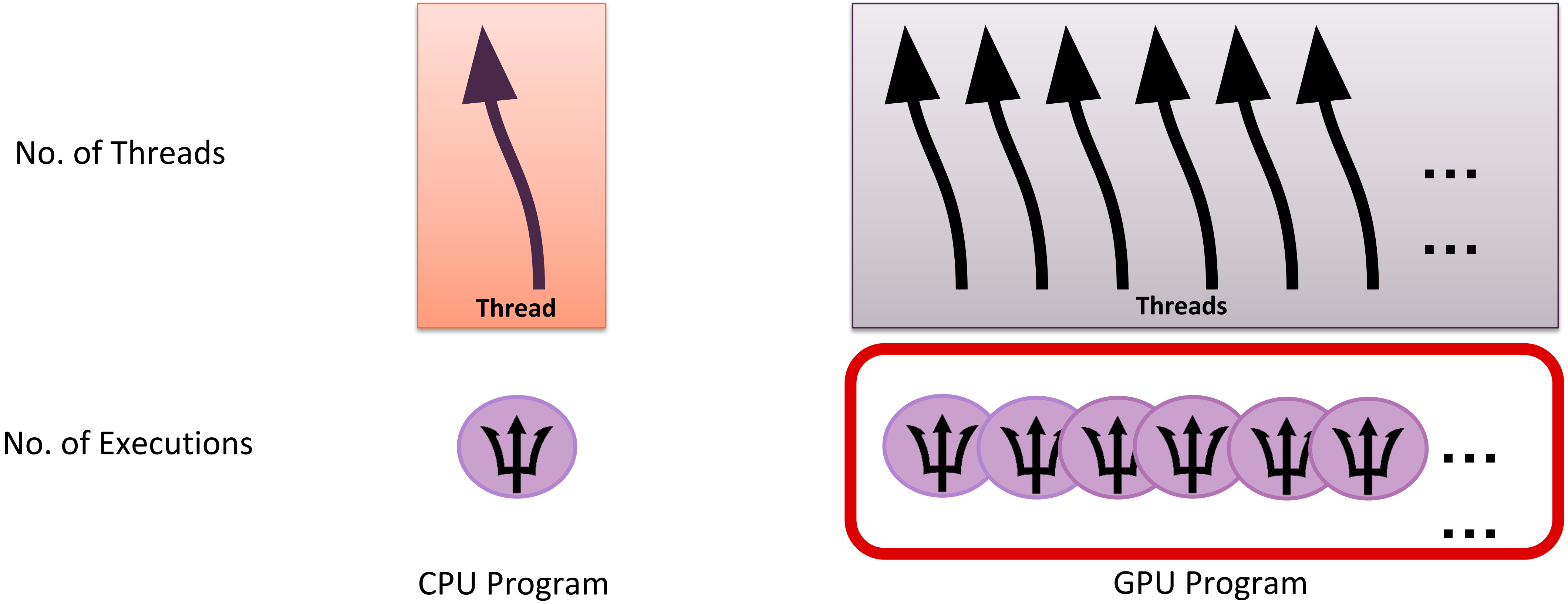
Interleaving dependencies complicate error propagation



[SC’16, Li]

GPU - Challenges

- *Average of ~5 years, Max 17 years*
- > 5GB, data to be profiled in Circuit
- *Inaccuracies in model accumulate*



Challenges - Summary

1. Tracking error across threads

2. Huge amount of states to profile

3. Accumulation of inaccuracies

Challenges - Summary

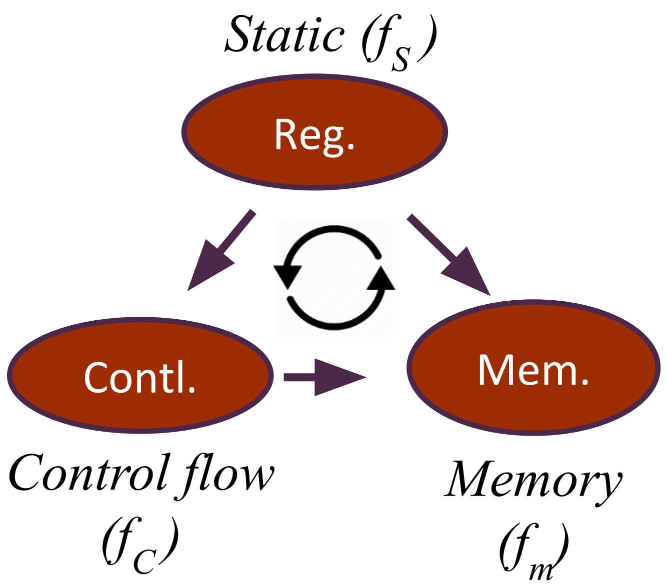
1. Tracking error across threads

2. Huge amount of states to profile

f_m

3. Accumulation of inaccuracies

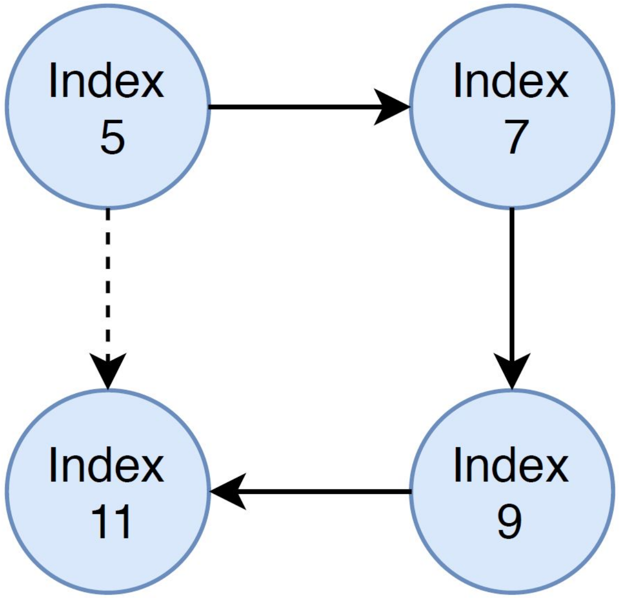
f_c, f_s



Updating f_m

- f_m constructs a memory dependency graph between instructions.
- We construct graph of whole kernel, instead of individual threads.

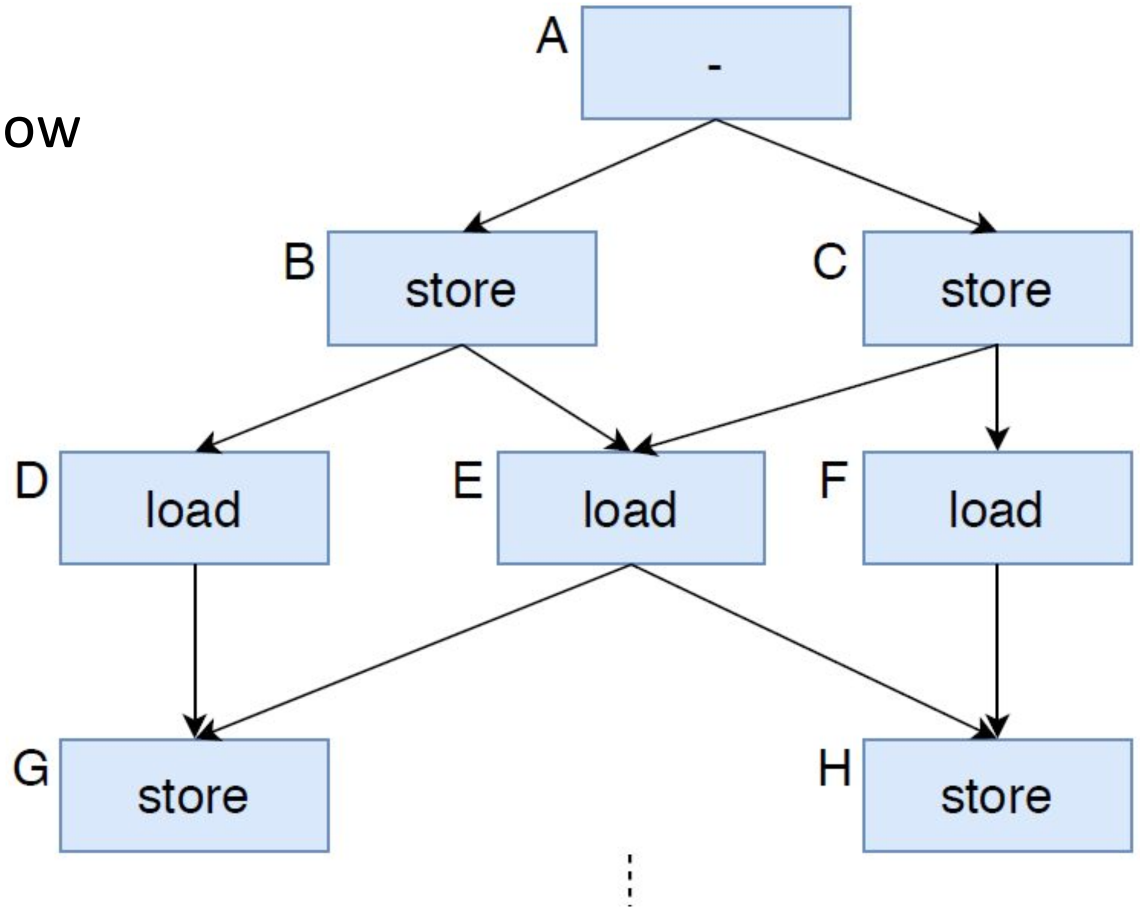
```
__global__ void staticReverse(int *d, int n)
{
  __shared__ int s[64];
  int t = threadIdx.x;
  int tr = n-t-1;
  s[t] = d[t];
  __syncthreads();
  d[t] = s[tr];
}
```



Constructing the dependency graph

- Memory dependency, based on control-flow
- Limited possible control-flows

Solution: *Sample threads with unique control-flows for profiling e.g. 3,840 out of 592,640 threads profiled for Pathfinder*



Challenges

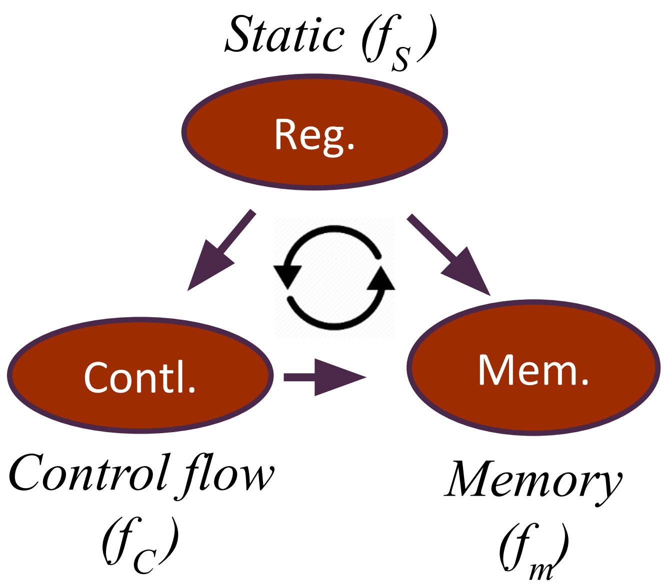
1. Tracking error across threads

2. Huge amount of states to profile

f_m

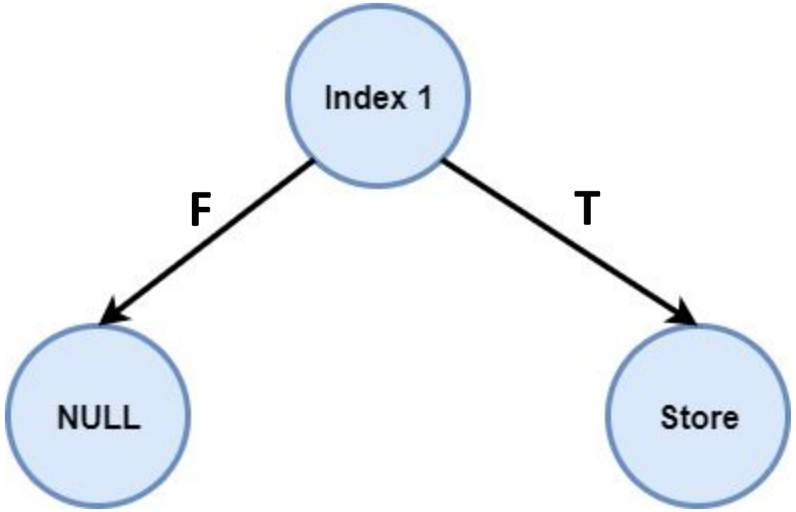
3. Accumulation of inaccuracies

f_c, f_s



Lucky Stores

- If a memory contains the same data we want to store in it
- Missing that store won't result in any SDC



Solution: *Update f_c to modify propagation probability of comparisons dominating output stores.*

Challenges

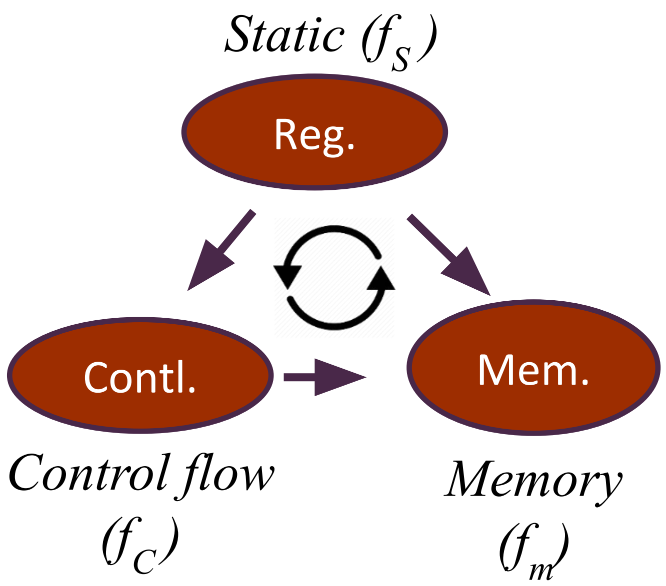
1. Tracking error across threads

2. Huge amount of states to profile

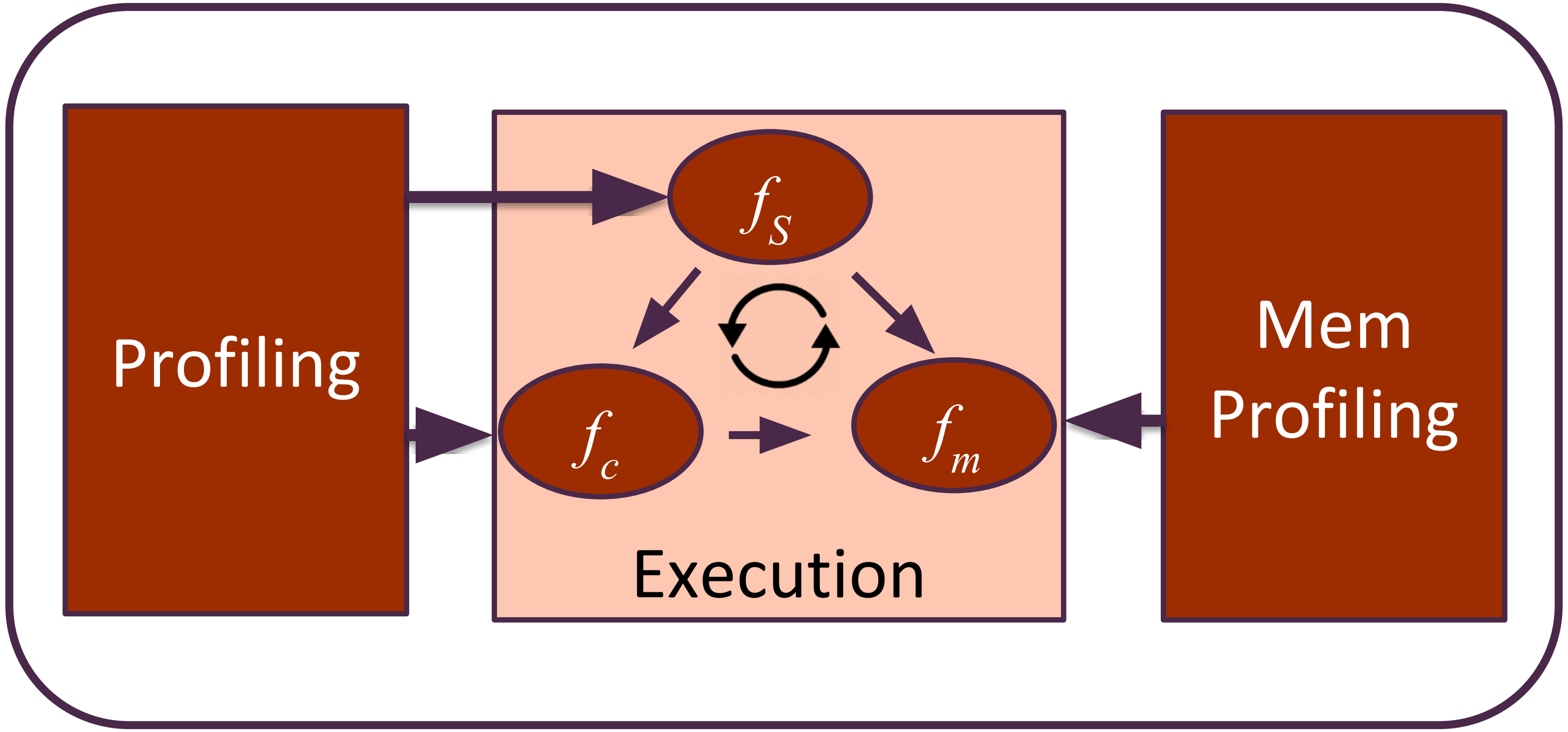
f_m

3. Accumulation of inaccuracies

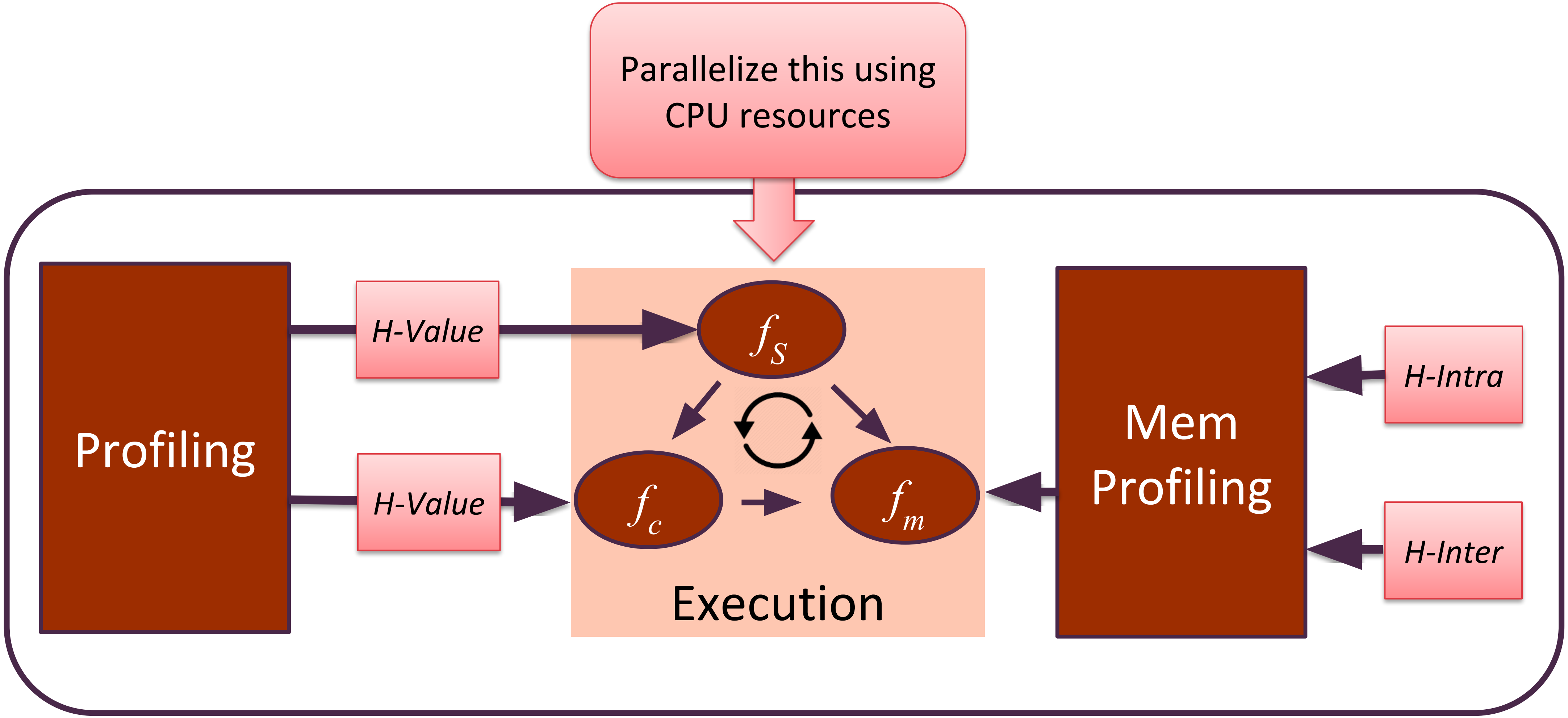
f_c, f_s



Trident - Workflow



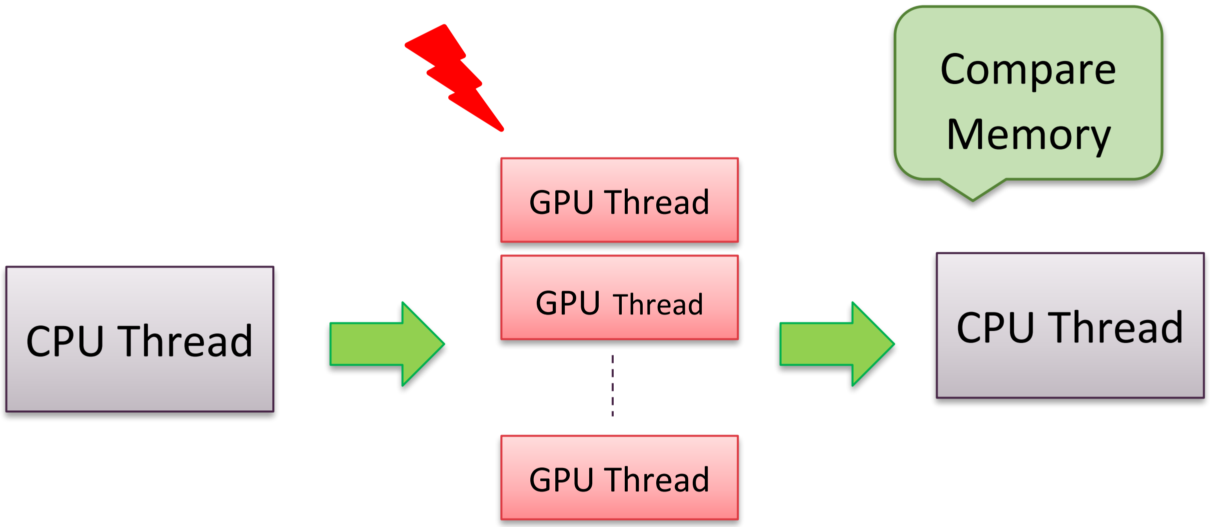
GPU-Trident - Workflow



Experiments - Overview

- GPU-Trident: A set of LLVM passes, driven by python scripts
  URL: https://github.com/DependableSystemsLab/GPU-Trident
- Use LLFI-GPU for FI and use it as a baseline
- Predict SDC probability with GPU-Trident, compare with FI
- Evaluate GPU-Trident for 17 kernels (Rodinia and OSS HPC) at
  - Kernel level
  - Instruction level

FI in GPU applications



$A = 0x00
$B = A + 0x04
.
$C = A + B

Evaluating Kernel SDC probability

T1	*T2*		*Tn*
\$A = 0x00 \$B = A + 0x04 . \$C = A + B	\$A = 0x00 \$B = A + 0x04 . \$C = A + B	. . .	\$A = 0x00 \$B = A + 0x04 . \$C = A + B

0	0	0	1	0	1	1	1

5,000 random FI trials per kernel

Evaluating Instruction SDC probability

T1

$A = 0x00
$B = A + 0x04
.
$C = A + B

T2

$A = 0x00
$B = A + 0x04
.
$C = A + B

...

Tn

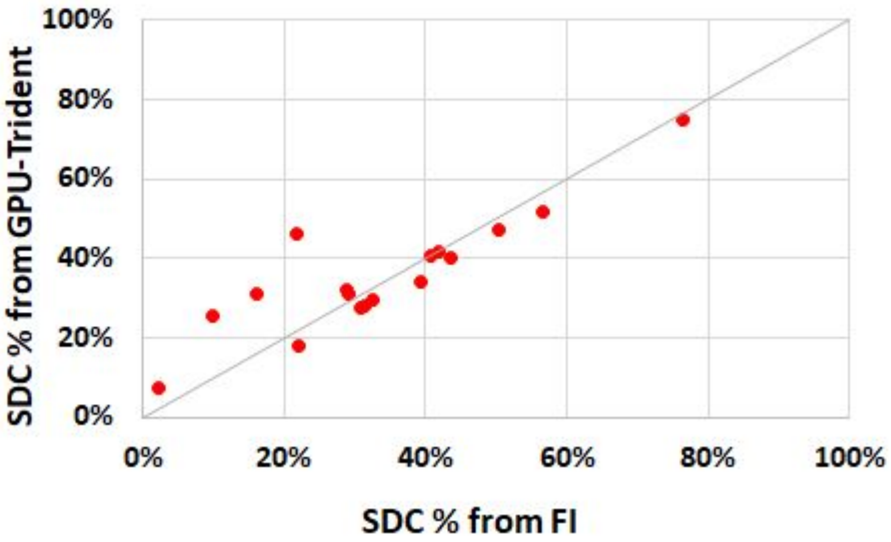
$A = 0x00
$B = A + 0x04
.
$C = A + B

0	0	0	1	0	1	1	1

100 random FI trials per instruction

Accuracy - Summary

- Mean absolute error for kernel SDC is **5.7%** (Trident has error of **4.75%**)



- Pearson correlation coefficient for kernel SDC is **0.88** (Without outliers **0.99**)

- Average Pearson correlation coefficient for instructions is **0.83**

Scalability - Summary

- Kernel SDC probability

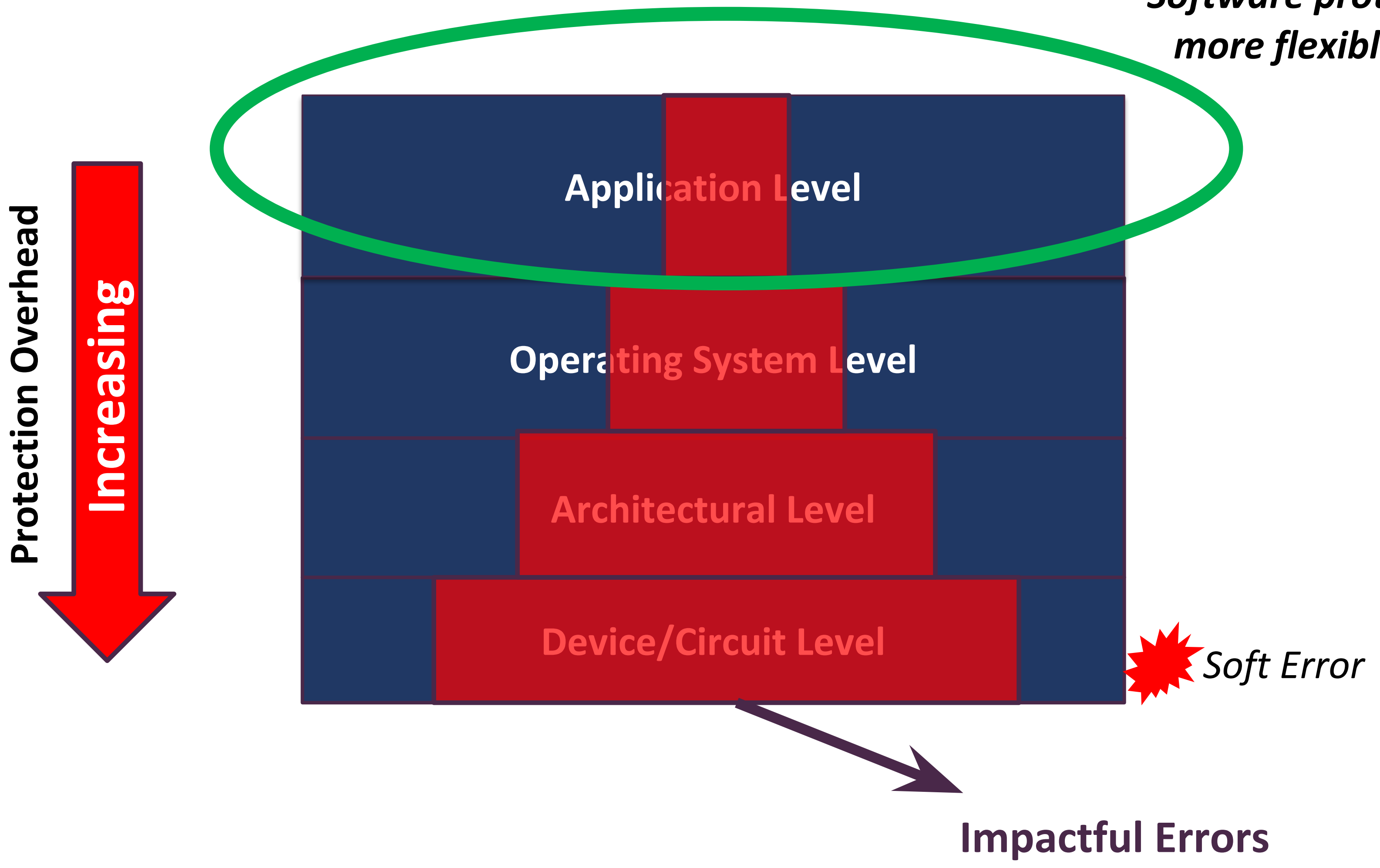
FI trials	Speed up
1000	11x
3000	33x
5000	55x

- Instruction SDC probability
  - GPU-Trident is **2 orders of magnitude (~100x)** faster than FI
  - FI takes **7 hrs**, while GPU-Trident takes less than **5 minutes**

GPU-Trident needs to construct model once, while each FI trials requires an application run

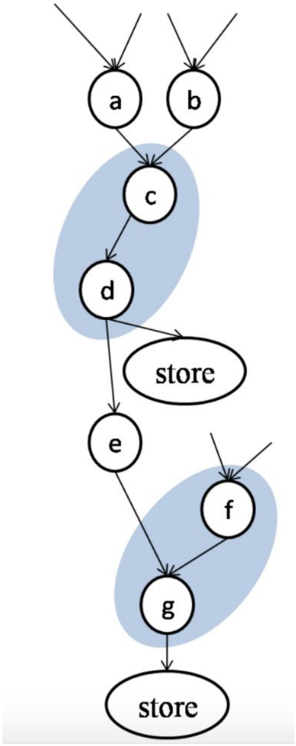
Software Solutions

Software protection techniques are more flexible and cost-effective!

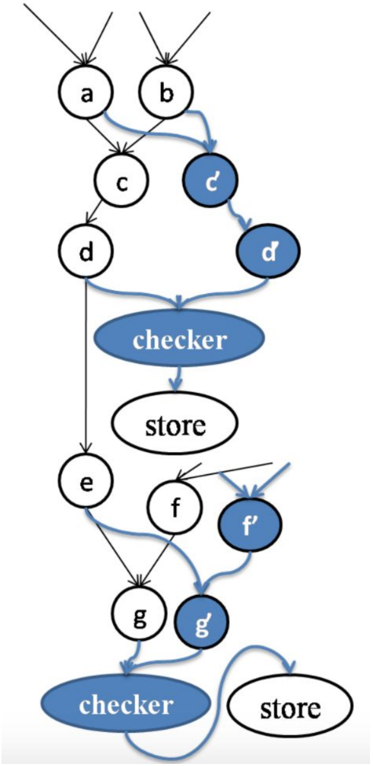


Selective Instruction Duplication

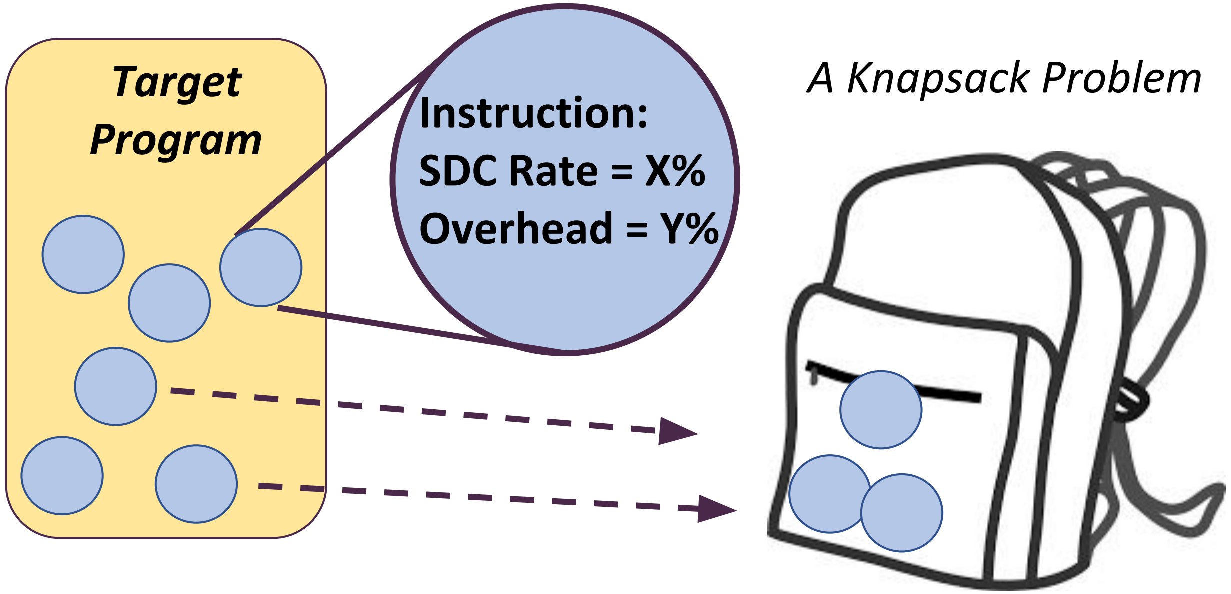
- Proposed by [DAC’09, Reddi], [ASPLOS’10, Feng], [CGO’16, Laguna]



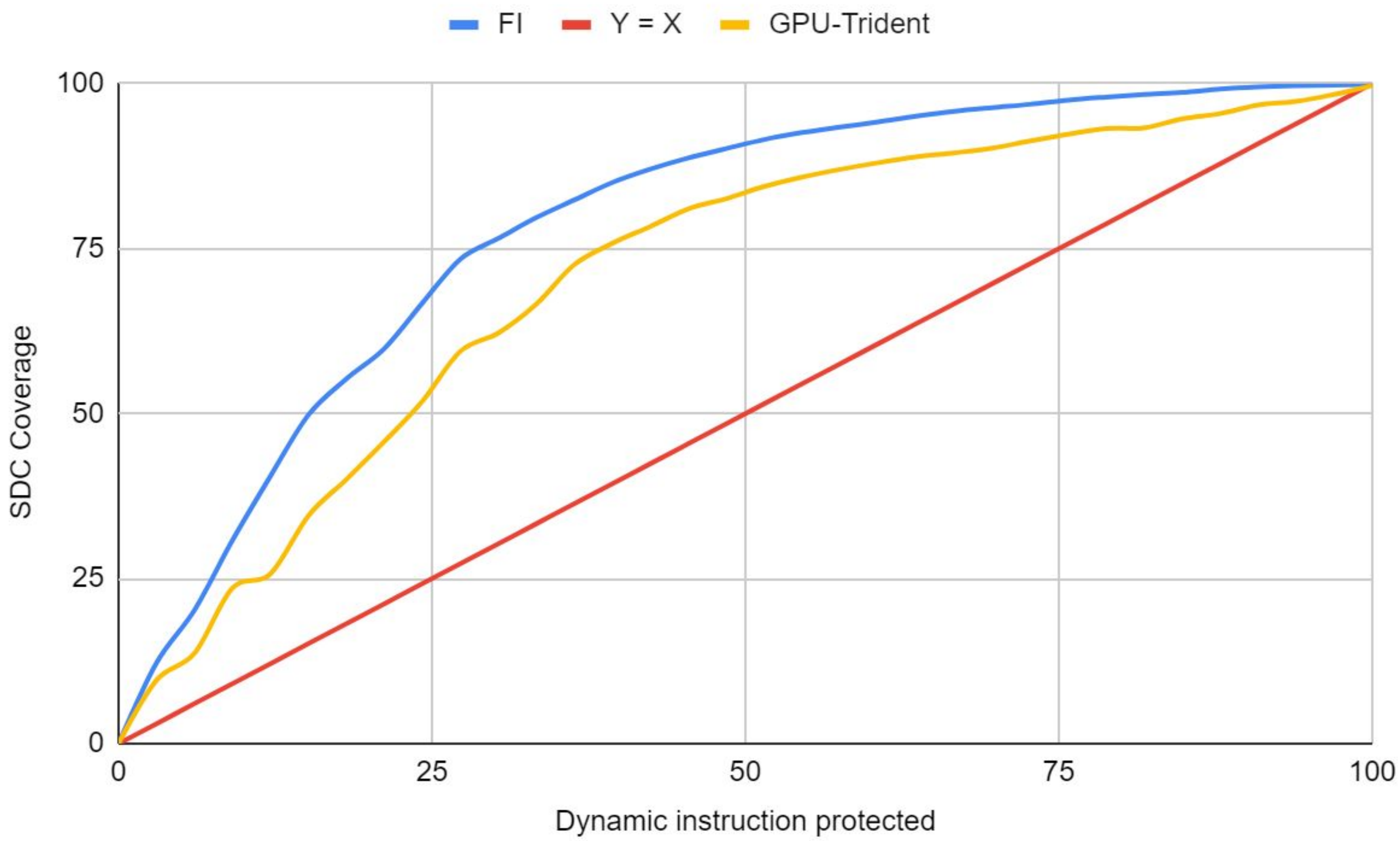
Instruction Sequence



Instruction Duplication



Selective Instruction Duplication



Summary

- Modeling error propagation in GPU applications is challenging due to *scale* and *inaccuracy*
- We develop heuristics, based on *memory access* and *data patterns*
- Experiments show our techniques are *accurate* and *scalable*
- *GPU-TRIDENT: Efficient Modeling of Error Propagation in GPU Programs (https://github.com/DependableSystemsLab/GPU-Trident)*
- *For questions: arehman.anwer1@gmail.com*