a place of mind

GPU-Qin: A Methodology For Evaluating Error Resilience of GPGPU Applications

Bo Fang , Karthik Pattabiraman, Matei Ripeanu,

The University of British Columbia

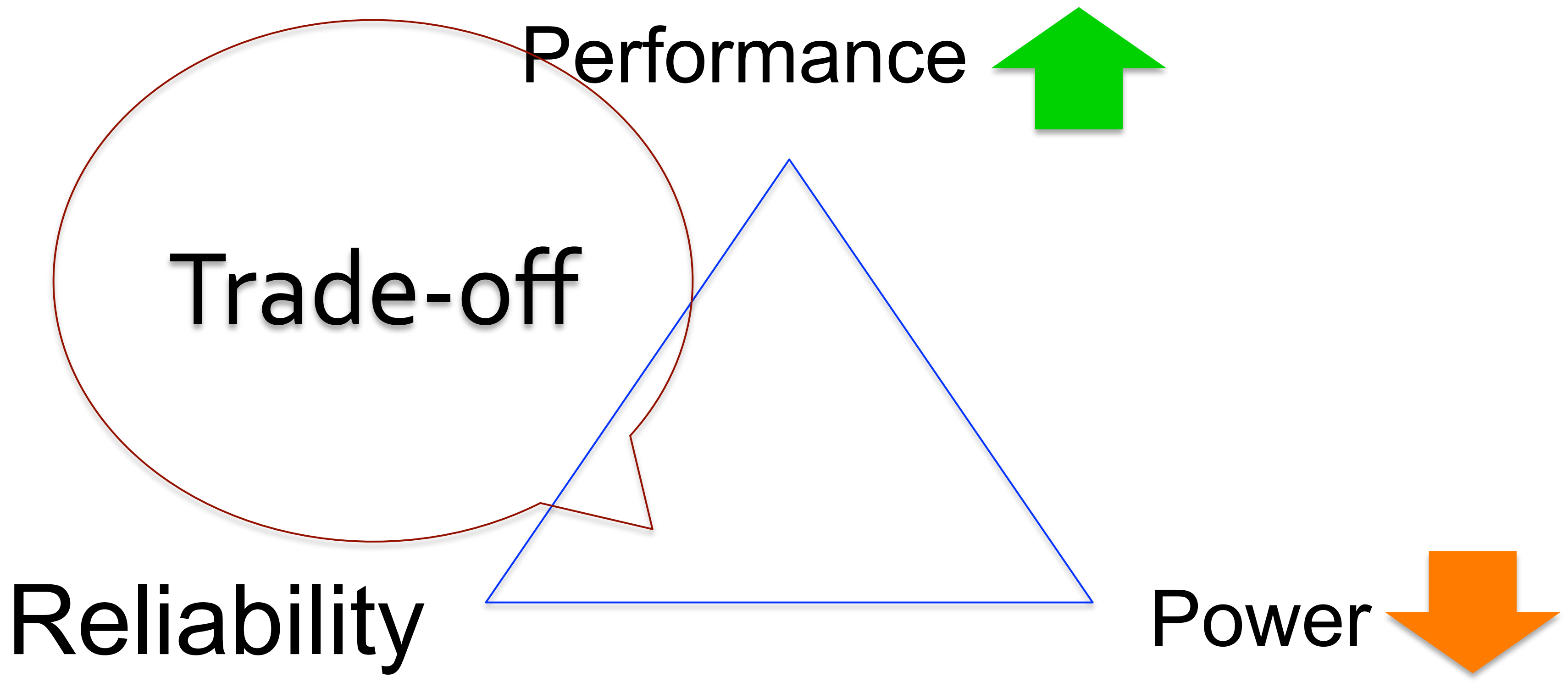
Sudhanva Gurumurthi

AMD Research

a place of mind

Graphic Processing Unit

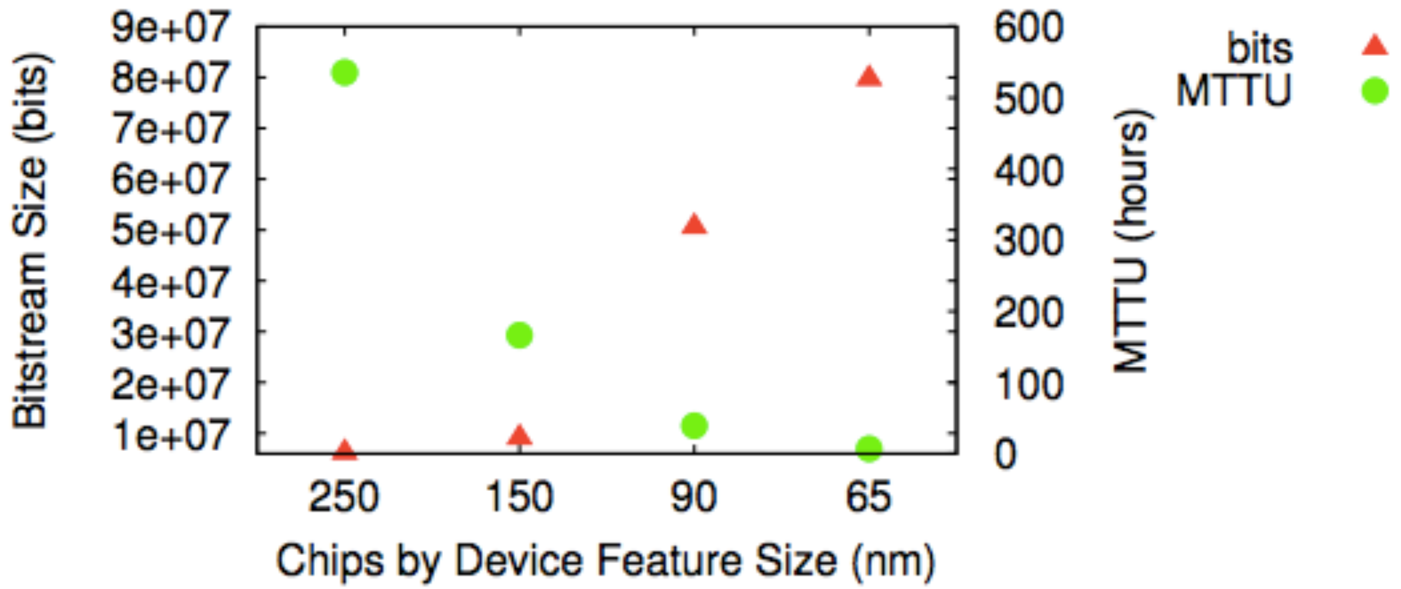
- Main accelerators for general purpose applications (e.g. TITAN , 18,000 GPUs)



a place of mind

Reliability trends

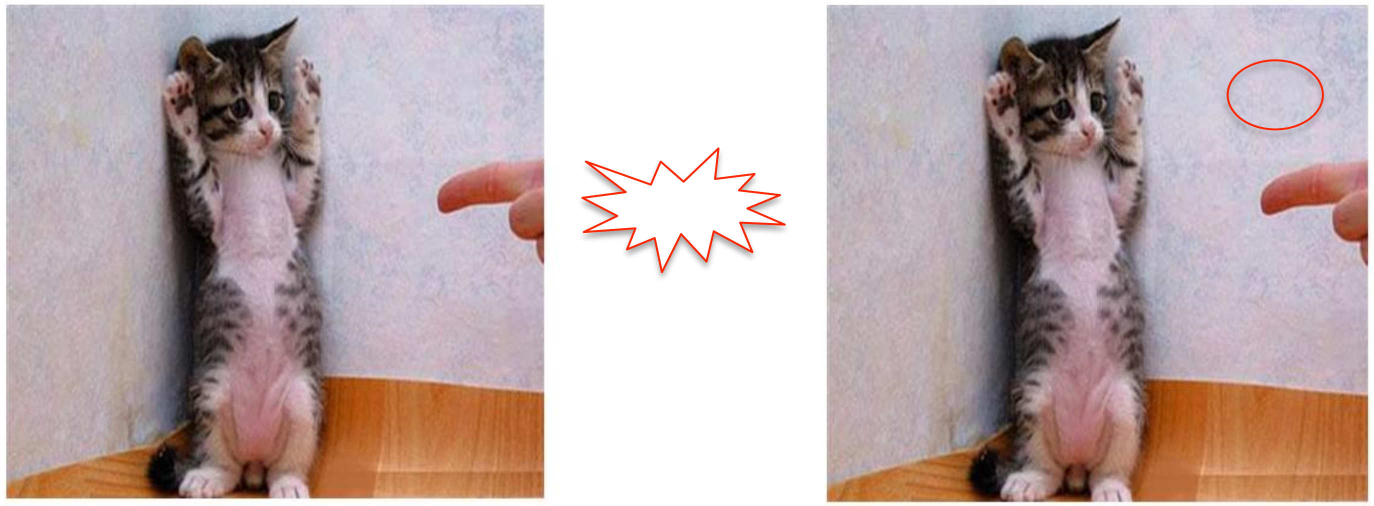
- The soft error rate per chip will continue to **increase because of larger arrays**; probability of soft errors within the combination logic will also increase. [Constantinescu, Micro 2003]



Source: final report for CCC cross-layer reliability visioning study (http://www.relxlayer.org/)

GPGPUs vs Traditional GPUs

- Traditional: image rendering



- GPGPU: DNA matching

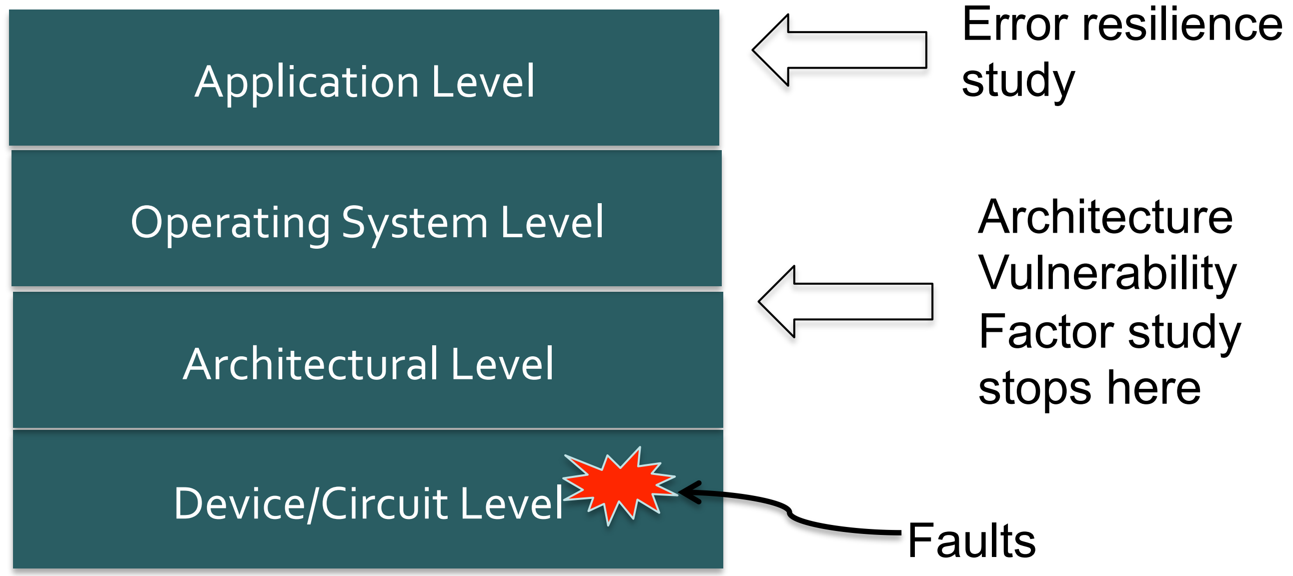
ATATTTTTTCTTGTTTT
TTATATCCACAAACTC
TTTTCGTACTTTTACA
CAGTATATCGTGT

ATATTTTTTCTTGTTT
TTTATATCCACAATCT
CTTTTCGTACTTTTAC
ACAGTATATCGTGT

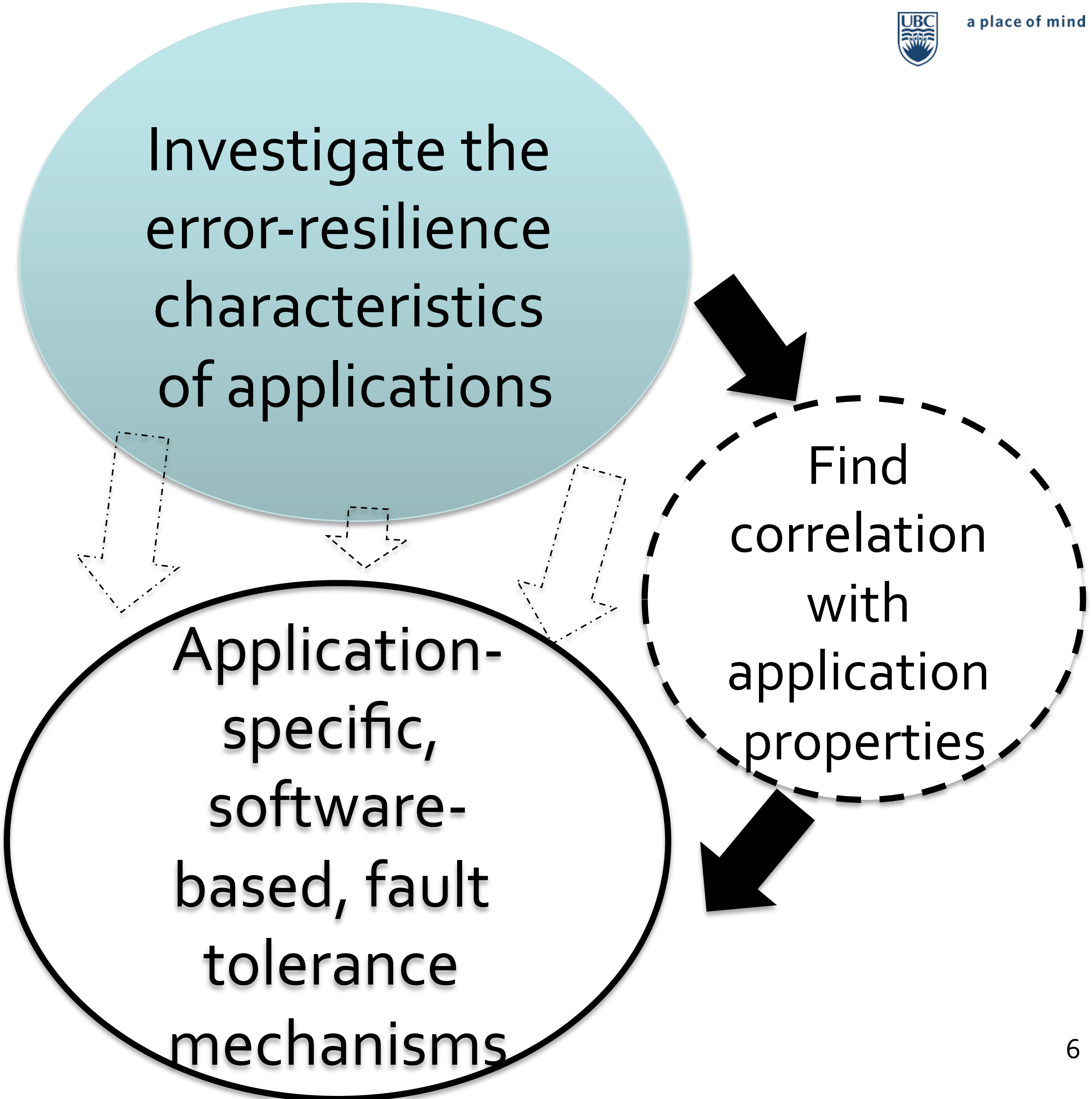
Serious result !

Vulnerability and error resilience

- Vulnerability: *Probability that the system experiences a fault that causes a failure*
  - Not consider the behaviors of applications
- Error Resilience: *Given a fault occurs, how resilient the application is* ✔



Goal


Investigate the error-resilience characteristics of applications
Find correlation with application properties
Application-specific, software-based, fault tolerance mechanisms

GPU-Qin: a methodology and a tool

- Fault injection: introduce faults in a systematic, controlled manner and study the system’s behavior
- GPU-Qin: on real hardware via cuda-gdb
  - Advantages:
    - Representativeness
    - efficiency

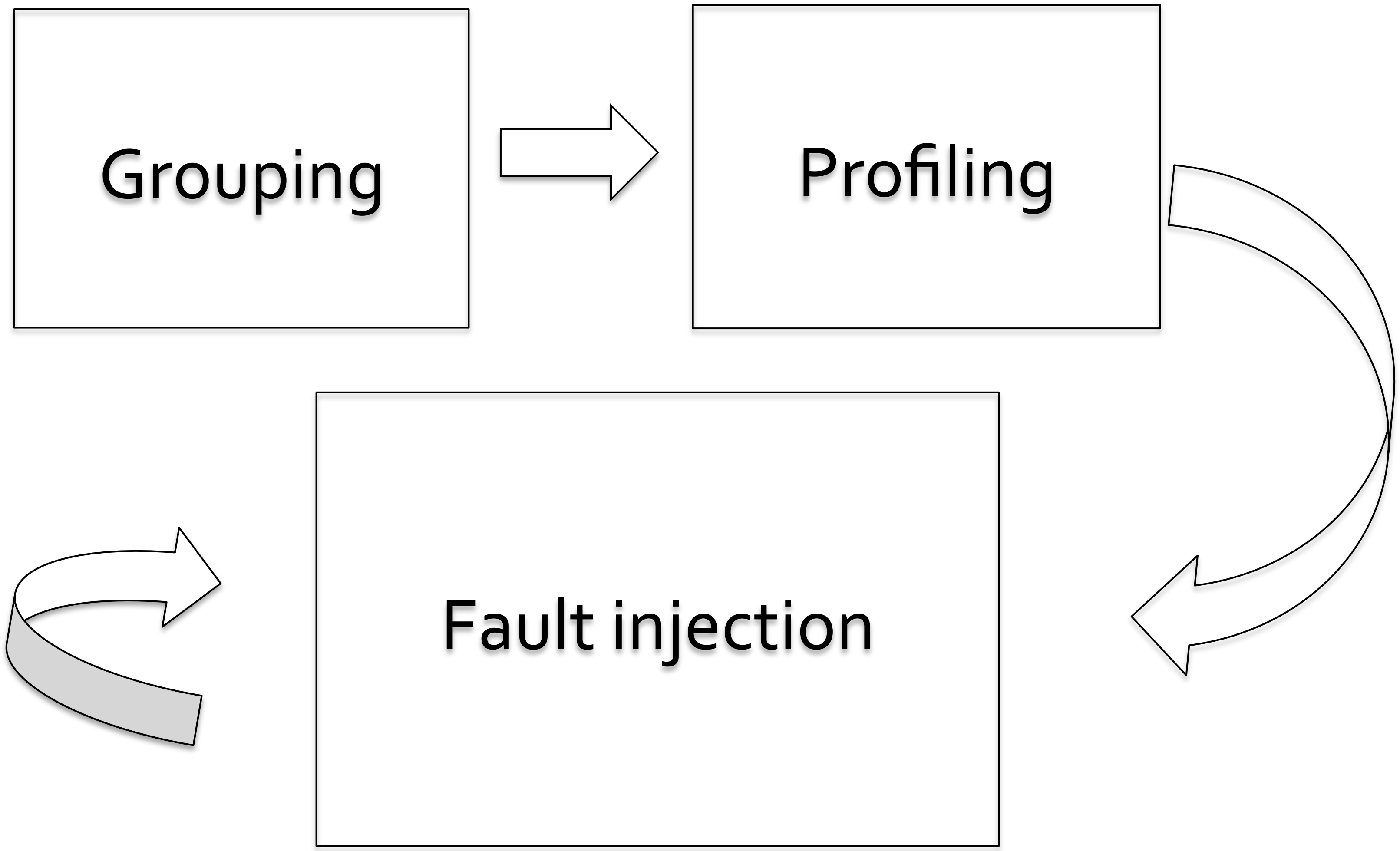
Challenges

- High level: massive parallelism
  - Inject into all the threads: impossible!
- In practice: limitations in cuda-gdb
  - Breakpoint on source code level, not instruction level
  - Single-step to target instructions

Fault Model

- Transient faults (e.g. soft errors)
- Single-bit flip
  - No cost to extend to multiple-bit flip
- Faults in arithmetic and logic unit(ALU), floating point Unit (FPU) and the load-store unit(LSU). (Otherwise ECC protected)

Overview of the FI methodology



Grouping

Grouping

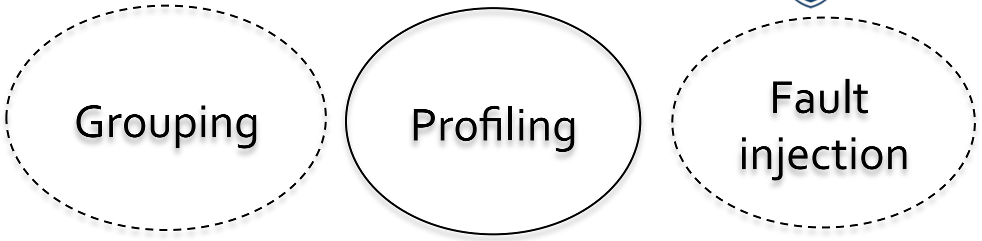
Profiling

Fault injection

- Detects groups of threads with similar behaviors (i.e. same number of dynamic insts)
- Pick representative groups (challenge 1)

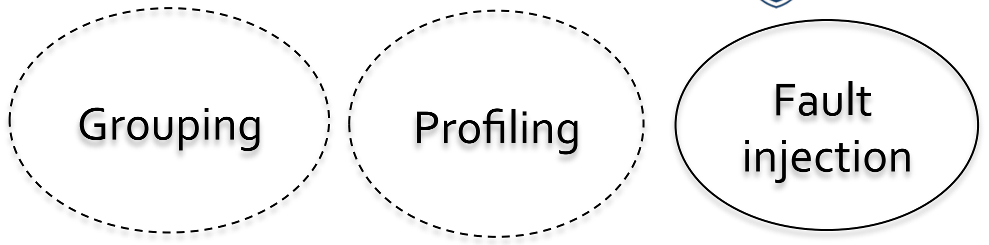
Category	Benchmarks	Groups	Groups to profile	% of threads in picked groups
Category I	AES,MRI-Q,MAT,Mergesort-ko, Transpose	1	1	100%
Category II	SCAN, Stencil, Monte Carlo, SAD, LBM, HashGPU	2-10	1-4	95%-100%
Category III	BFS	79	2	>60%

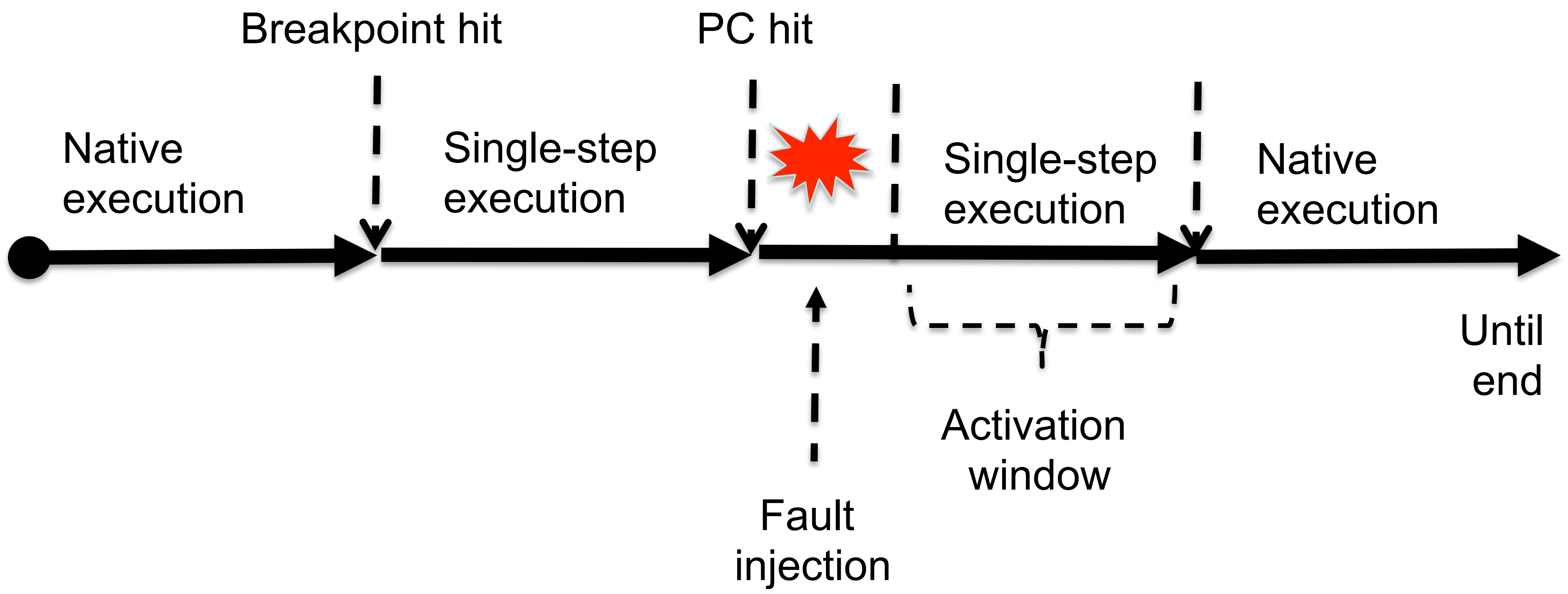
Profiling



- Finds the instructions executed by a thread and their corresponding CUDA source-code line (challenge2)
- Output:
  - Instruction trace: consisting of the program counter values and the source line associated with each instruction

Fault injection





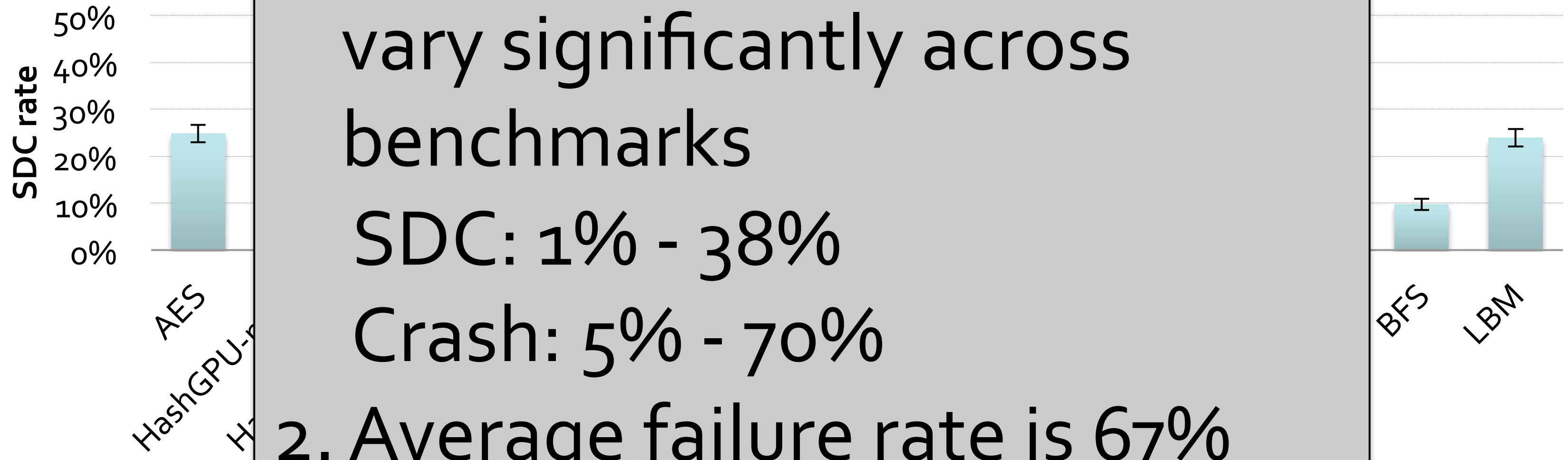
Heuristics for efficiency

- A large number of iterations in loop
    - Limit the number of loop iterations explored
- Single-step is time-consuming
    - Limit activation window size

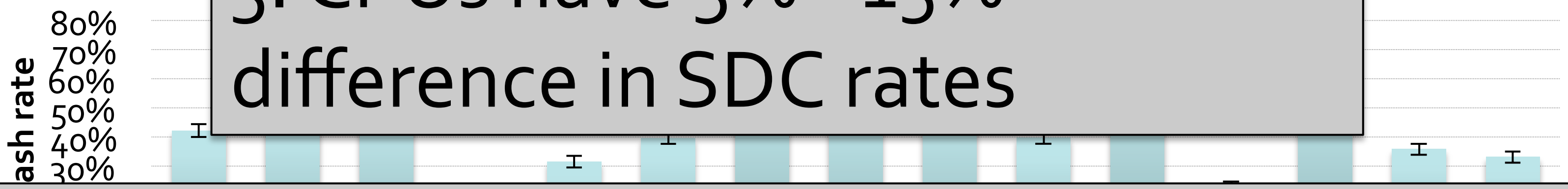
Characterization study

- NVIDIA Tesla C 2070/2075
- 12 CUDA benchmarks, 15 kernels
  - Rodinia, Parboil and CUDA SDK
- Outcomes:
  - ***Benign***: correct output
  - ***Crash***: exceptions from the system or application
  - ***Silent Data Corruption (SDC)***: incorrect output
  - ***Hang***: finish in considerably long time

Error cs



1. SDC rates and crash rates vary significantly across benchmarks
   SDC: 1% - 38%
   Crash: 5% - 70%
2. Average failure rate is 67%
3. CPUs have 5% - 15% difference in SDC rates

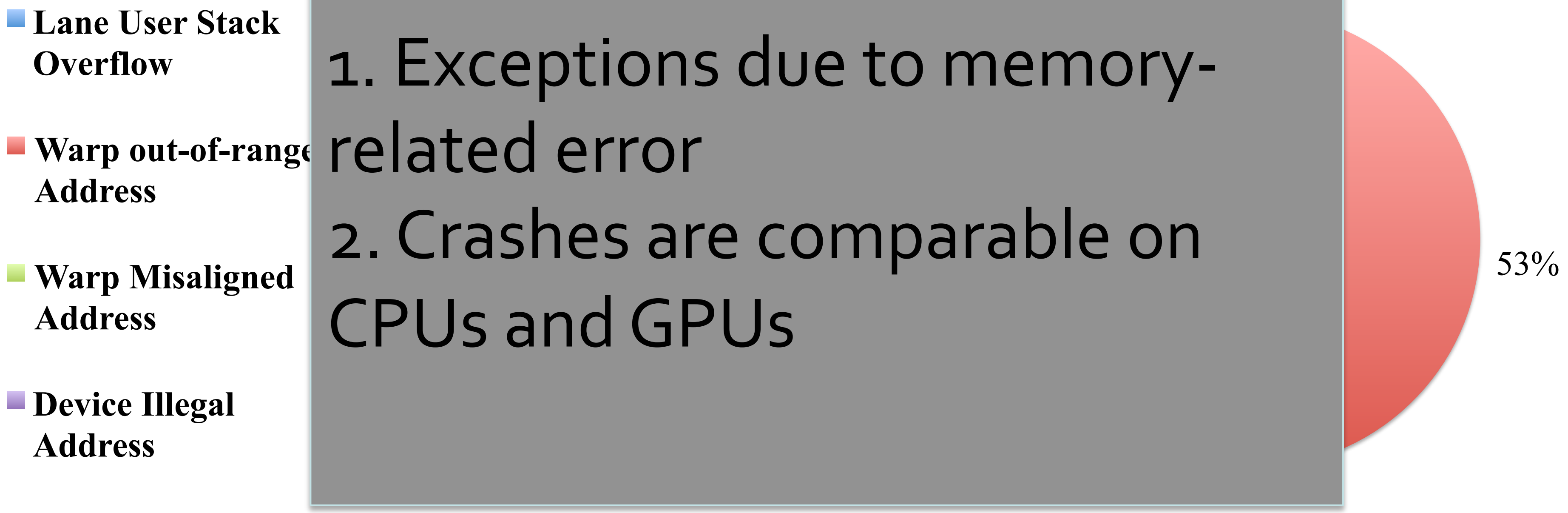


We need application-specific FT mechanisms

Hardware error detection mechanisms

- Hardw

1. Exceptions due to memory-related error
2. Crashes are comparable on CPUs and GPUs



AES (Crash rate: 43%)

MAT (Crash rate: 30%)

Crash latency – measure the fault pro

- T

1. Device illegal address have highest latency in all benchmarks except Stencil
2. Warp out-of-range has lower crash latency otherwise
3. Crash latency in CPUs is shorter (thousands of cycles) [Gu, DSN04]

We can use crash latency to design application-specific checkpoint/recovery techniques

Crash latency in milliseconds

Resilience Category

Resilience Category	Benchmarks	Measured SDC	Dwarfs
Search-based	MergeSort	6%	Backtrack and Branch+Bound
Bit-wise Operation	HashGPU, AES	25% - 37%	Combinational Logic
Average-out Effect	Stencil, MONTE	1% - 5%	Structured Grids, Monte Carlo
Graph Processing	BFS	10%	Graph Traversal
Linear Algebra	Transpose, MAT, MRI-Q, SCAN-block, LBM, SAD	15% - 25%	Dense Linear Algebra, Sparse Linear Algebra, Structured Grids

Conclusion

- GPU-Qin: a methodology and a fault injection tool to characterize the error resilience of GPU applications
  - We find that there are significant variations in resilience characteristics of GPU applications
  - Algorithmic characteristics of the application can help us understand the variation of SDC rates

Thank you !

Project homepage:

http://netsyslab.ece.ubc.ca/wiki/index.php/FTGPU

Code:

https://github.com/DependableSystemsLab/GPU-Injector