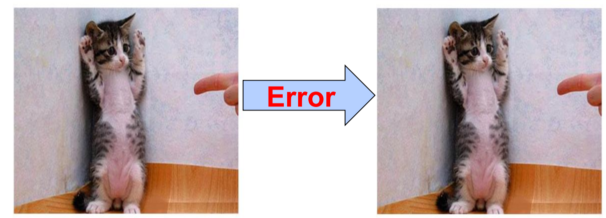
Towards Building Error Resilient GPGPU Applications

Bo Fang, Jiesheng Wei, Karthik Pattabiraman, Matei Ripeanu
Department of Electrical and Computer Engineering
University of British Columbia

Motivation

- GPUs have been used for error-resilient workload
  - E.g. Image Rendering



- Today, GPUs are used in error-sensitive applications, i.e. GPGPU applications
  - Scientific Computing: E.g. DNA Processing, Physics Simulation, etc.

ATATTTTTTCTTGTT
TTTTATATCCACAAA
CTCTTTTCGTACTTT
TACACAGTATATCGT
GT

Error

ATATTTTTTCTTGTT
TTTTATATCCACAAT
CTCTTTTCGTACTTT
TACACAGTATATCGT
GT

Motivation

- 2/3 of 50,000 GPUs exhibit transient faults in memory or logic [Haque-CCGRID10]′
- Since Fermi, NVIDIA introduces ECC protected GPU models.
- Is ECC a panacea?
  - ECC is usually not enabled for HPC systems
  - Computation units are not covered by ECC.
  - Portion of area going to Computation unit: GPU > CPU

Research Question:

How to tolerate hardware faults in GPU?

– Focus on software based techniques

- First step: two tasks

  Task 1: Understand the reliability characteristics of GPGPU applications

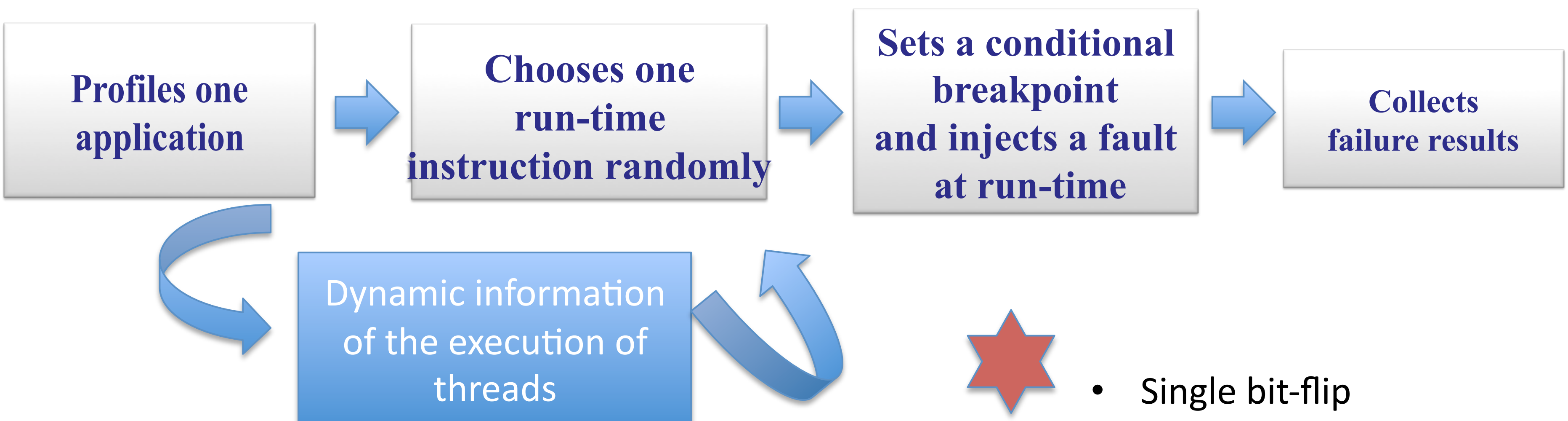
  Task 2: Detect hardware faults at software level

Contributions

- Study the behaviour of GPGPU applications
  - Build a fault-injection tool
  - Discover the reliability "hotspots" of the programs
- Develop heuristics to protect GPGPU applications based on the "hotspots"
  - Focus on silent data corruptions (SDCs)
- Evaluate the efficacy and cost of the detectors

Task 1: Study on Reliability Characteristics of GPGPU Applications

- Method: fault injection techniques
  - Built a breakpoint-based fault injection tool (cuda-gdb)
  - On real GPU hardware



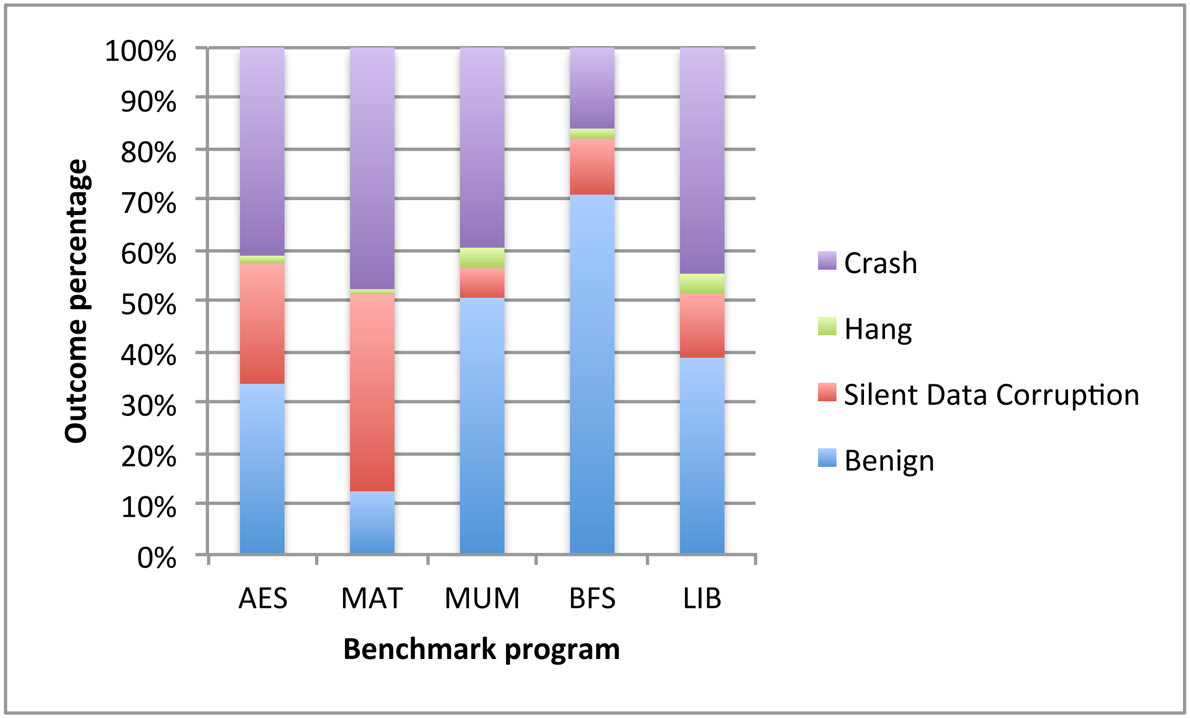
- Single bit-flip
- Dest registers or memory address for ld/st
- One fault per run

Evaluation Setup

- NVIDIA Telsa C2075
- CUDA Toolkit 4.1
- Benchmarks

Benchmark	Domain	Computation/IO
AES Cryptography	Cryptography	Computation
Breadth First Search	Graph processing	IO
MUMmerGPU	Bio-information	IO
LIBOR Monte Carlo	Finance	Regular
Matrix Multiplication	Linear Algebra	Computation

Characteristic Study Results



- SDCs rate is upto 40%

- Root causes of SDCs:
  - Branches with thread id, etc.

Task 2: Heuristic-Based Fault Detection

What we learn from characteristic study?

– faults in GPGPU applications lead to a high rate of SDCs

Focus on protecting programs from SDC

– There are several root causes in the program

Develop heuristics to detect faults – selectively error detection

Heuristic Categories

- Category I: Loop conditions.
- Category II: Branches with block or thread identifier
- Category III: Computation statements that pertain to:
  a. Computations involving block/thread id or loop iterators
  b. Data movement between global memory and other memory regions

Example

// Category I
for (
//Do
}

// C
if (t
//Do sor
}

// Categ
unsigned i ;

// Categor
// C is an a
C[c + wB *

// Category III (a)
unsigned int tid =blockIdx.x*MAX_THREADS_PER_BLOCK + threadIdx.x;
*//Detector: gpuAssert(tid == blockIdx.x*MAX_THREADS_PER_BLOCK*
+th

// Category III (b)
// C is an array allocated in global memory
C[c + wB * ty + tx] = Csub;
*//Detector: gpuAssert(C[c+WB*ty+tx] == Csub);*

// Category II
if (tid<no_of_nodes && g_graph_mask[tid]{
//Detector: gpuAssert(tid < no_of_nodes);
//Detector: gpuAssert(g_graph_mask[tid] == true);
}

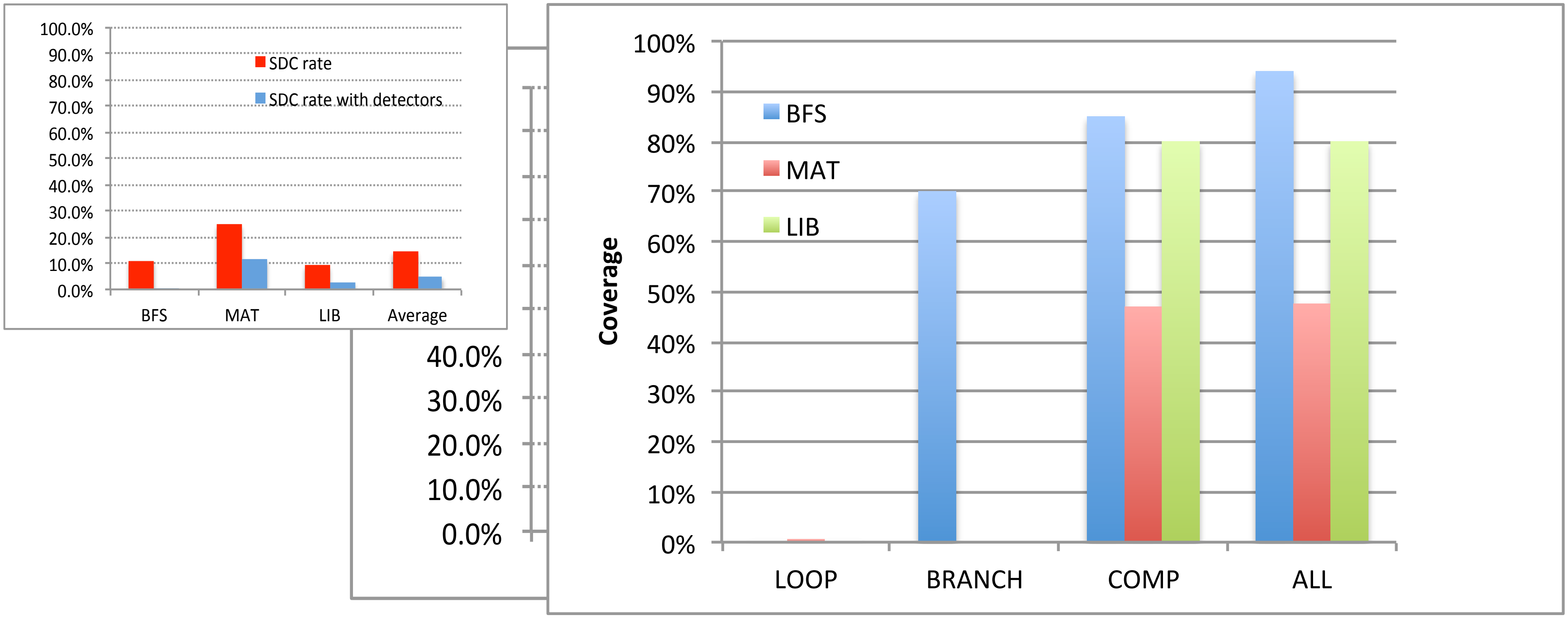
Evaluation Setup

- We manually insert error detectors in the programs based on heuristics
  - The process can be automated via data-flow analysis

Benchmark	LOC of kernel	Number of detectors for each category			
		LOOP	BRANCH	COMP	Total
BFS	44	2	7	9	17
MAT	91	3	0	11	14
LIB	392	36	0	47	83

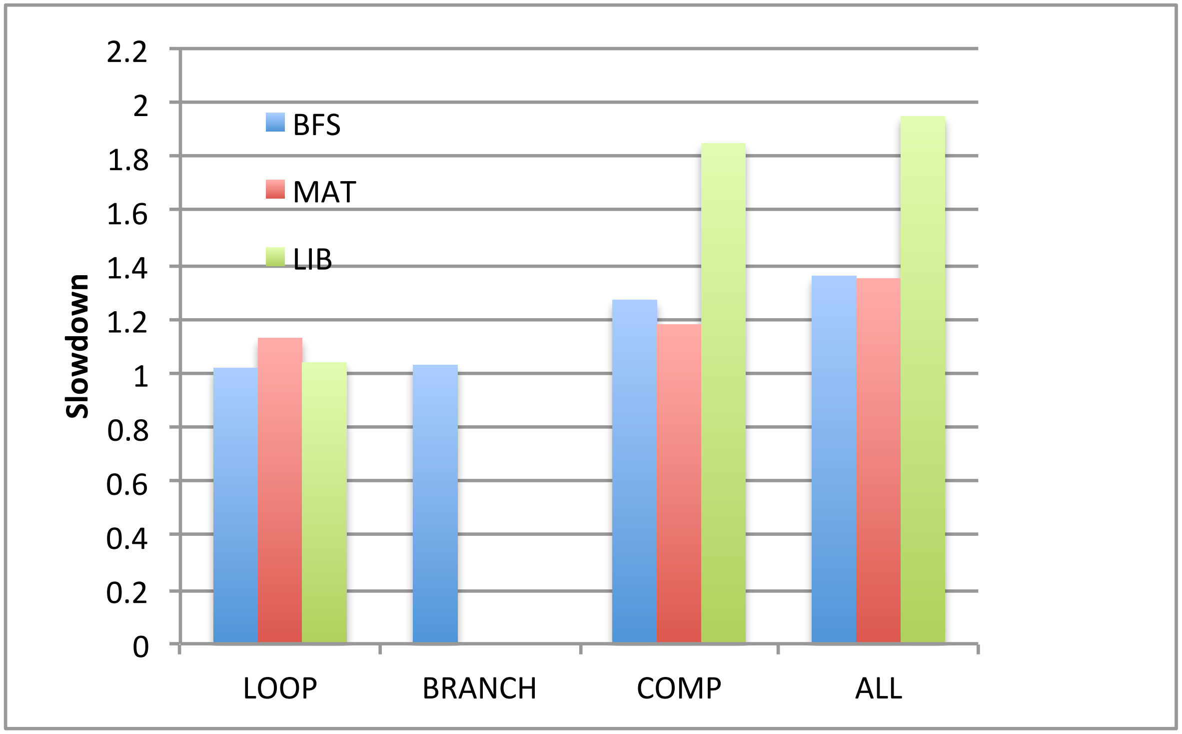
- We rerun the fault injection experiment with instrumented applications

Error Detection Results



- On average the fault detectors succeeded in reducing SDC rate from 14.5% to 5.8%
- COMP is the most effective category across all benchmarks (85%, 47% and 80%)
- LOOP is ineffective in providing any protection

Performance Overhead



- On average the instrumented application is running 1.5x slower
- BRANCH is the most cost-efficient category with incurring 3% overhead but catching 70% of SDCs for MAT

Conclusions

- Characteristic study shows transient faults in GPU lead to high Silent Data Corruption
- Heuristic-based error detection mechanism reduces SDCs for GPGPU applications, with a reasonable performance overhead

Contact: bof@ece.ubc.ca

a place of mind

Thank You

Networked System Lab http://netsyslab.ece.ubc.ca/

Radical Lab http://radical.ece.ubc.ca/

Contact: bof@ece.ubc.ca

Future Work

- Expand to more GPGPU applications
- Optimize the heuristics
  - E.g. Develop more efficient error detectors
- Investigate automated techniques to instrument programs with error detectors

References

[Haque-CCGRID10] *Haque et al,Hard Data on Soft Errors: A Large-Scale Assessment of Real-World Error Rates in GPGPU, CCGRID 10*