GPGPU in HPC

GPGPU

Scientific Applications
Soft Errors

$2^0 = 1$

Bauman et al. [TDMR, 2005]
Traditional Method: DMR

Dual Modular Redundancy (DMR)

• Run 2 copies

• Compare for divergence

Too much energy consumption!
Software Solutions

Advantages

- No hardware modification
- Errors can be masked
- Allow selective protection

Impact
Challenges for GPGPU Resilience

• Different architecture and programming model from CPUs

• No scalable fault injection tools for HPC GPGPU applications
Our Contributions

LLFI-GPU: Scalable Fault Injector → Characterization of Error Propagation → Implications on Error Mitigations
Existing Publicly Available GPU Fault Injectors

• Hauberk [IPDPS, 2011]
  • Source code level fault injection
  • Not representative for hardware errors

• GPU-Qin [ISPASS, 2014]
  • Debugger-based
  • Execution is slow

• GPGPU-Sim based fault injector [DSN, 2015]
  • Not full system simulation
Goals of LLFI-GPU

• **Native Speed**
  - Program-level fault injection
  - Compile to binary

• **Full system simulation**
  - Execute on real hardware
  - Able to simulate different failure outcomes

• **Representativeness**
  - LLVM IR level fault injection
  - Close to assembly, yet preserve high-level program symbols
LLVM (Low Level Virtual Machine)

LLFI for CPU: https://github.com/DependableSystemsLab/LLFI
LLFI-GPU: Overview

... 
R0 = add R1, R2
R0 = injectFault(R0)
R4 = mul R0, R3
...
Advantages of LLFI-GPU

• Compile on large GPGPU programs
  • 1000x faster compared to GPU-Qin (MatrixMul)

• Represented simulation
  • Full system simulation of soft errors

• Open-source
  • https://github.com/DependableSystemsLab/LLFI-GPU
Experiment Setup: Nvidia K20

• 12 Benchmarks
  • Rodinia & Parboil suites
  • Lulesh (LLNL), Barns-Hut (Texas State Univ.), Fiber (Northeastern Univ.), Circuit Solver (Rice Univ.) and NMF (UC Berkley)

• Fault Injection
  • 10,000 per application (Error bar: 0.22%-2.99% , 95% confidence level)

• Fault Model
  • Single bit-flip
  • Transient faults in execution units
Failure Outcomes

• Silent Data Corruption (SDC)
  • Mismatch in program outputs from golden run and fault injection run

• Crash
  • CUDA exceptions (e.g., illegal memory address)
  • Cause kernel execution to halt

• Benign
  • No effect on program output
Our Contributions

LLFI-GPU: Scalable Fault Injector

Characterization of Error Propagation

Implications on Error Mitigations
Research Question 1

What is the percentage of SDCs in different memory states?
Memory State

... 
**cudaMalloc( M1 )**
**cudaMalloc( M2 )**
**cudaMemcpy(M1, ...)**
**cudaMemcpy(M2, ...)**
...
Kernel<<<>>> , ...
...
**cudaMemcpy(..., M2)**
...
**Foreach(M2): if(ele>0) {print(ele)}**
...
### SDC in Different Memory States

#### SDC of States

<table>
<thead>
<tr>
<th></th>
<th>bfs</th>
<th>barneshut</th>
<th>nmf</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SDC_{TM} - SDC_{RM}$</td>
<td>0.00%</td>
<td>0.20%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

**Average of $SDC_{(TM-RM)}$ in all benchmarks:** 0.09%

**Most of the faults in TM propagate RM**

#### Size of States

<table>
<thead>
<tr>
<th></th>
<th>bfs</th>
<th>barneshut</th>
<th>nmf</th>
</tr>
</thead>
<tbody>
<tr>
<td>RM</td>
<td>14.29%</td>
<td>37.50%</td>
<td>0.03%</td>
</tr>
<tr>
<td>TM</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

**Average size of RM in all benchmarks:** 13.56%

**Checking RM reduces ~86% overhead while retaining coverage**
Example of Checkers

- Check value range of particular states
  - Calculating angle: if (angle > 60 or angle < 0) {error detected}

- Overhead is directly proportional to the number of states checked
  - Checking RM reduces ~86% overhead
  - Small loss of coverage

Pattabiraman et al. [TDSC, 2011]
Hari et al. [DSN, 2012]
Research Question 2

How long do errors take to propagate to the RM?
Metrics: Kernel Call

... to measure propagation time of error

Error detection latency is 2
Tracking Error Propagation

... 
Kernel1

DumpToDisk(TM) 
DumpToDisk(RM) 
DumpToDisk(OM)

Kernel2

... 

Compared with golden copy for any data corruptions
Checking RM provides short detection latency
Implications

• RM is a narrow tunnel where faults frequently propagate through
  • Checking RM for SDC is a better trade-off

• Crash-causing faults rarely propagate across kernel calls
  • Deploying high frequency checkpoints for GPGPU can avoid checkpoint corruptions

• Studied on 2 GPGPU platforms (Nvidia GTX 960 & Nvidia K20)
  • Results are statistically indistinguishable

• Investigated in error spread & masking etc
  • ... more interesting findings can be found in the paper!
Summary

- Designed a scalable fault injector for GPGPUs: LLFI-GPU
- Characterized error propagation patterns in GPGPU applications
- Discussed their implications on error mitigation techniques

- Name: Guanpeng (Justin) Li (gpli@ece.ubc.ca)
- Website: ece.ubc.ca/~gpli
- LLFI-GPU:
- Results:
  - https://www.dropbox.com/s/xrvojidskkcrj4y/FI_data.xlsx?dl=0
Acknowledgements