A Comparative Study of GPUVerify and GKLEE

Anmol Panda[1] Philipp Rümmer[2] Neena Goveas[1]

[1]Department of Computer Science and Information Systems
BITS Pilani K K Birla Goa Campus, India

[2]Department of Information Technology
Uppsala University, Sweden

4th International Conference on Parallel and Distributed Grid Computing, 2016
JUIT, Waknaghat, India

Outline

1 Background
- Challeneges in GPU Computing
- Verification tools: GPUVerify and GKLEE

2 Objectives and Scope

3 Experiments and Results
- Benchmarks
- Experimental Setup
- Results

4 Analysis
- Differences in Bugs Reported
- Difference in Runtime

5 Conclusion

Background

- CUDA, Opencl, OpenAMP

Background

- CUDA, Opencl, OpenAMP
- Lack of backward compatibility
- Absence of dedicated cache memory
- Difficult to optimize performance
- Making efficient software requires significant time and resources

Background

- CUDA, Opencl, OpenAMP
- Lack of backward compatibility
- Absence of dedicated cache memory
- Difficult to optimize performance
- Making efficient software requires significant time and resources
- Bugs like Data races and diverging barriers

Background

- Need for verification

Background

- Need for verification
- GPUVerify: Developed by Alastair Donaldson from Imperial College London and Shaz Qadeer from Microsoft as a portable verifier of Opencl and CUDA kernels

Background

- Need for verification
- GPUVerify: Developed by Alastair Donaldson from Imperial College London and Shaz Qadeer from Microsoft as a portable verifier of Opencl and CUDA kernels
- GKLEE: Developed by the Gauss Research group as a concolic (concrete and symbolic) verifier-cum-analyzer of CUDA programs for GPUs

Objectives and Scope

- Objectives
  - Compare GPUVerify and GKLEE for factors like bugs reported, execution time and system portability
  - Understand their usability, learn-ability and and preferred usage

Objectives and Scope

- Objectives
  - Compare GPUVerify and GKLEE for factors like bugs reported, execution time and system portability
  - Understand their usability, learn-ability and and preferred usage
- Scope
  - Within the scope: Performance aspects of the tools
  - Beyond the scope: Theoretical aspects of these tools

Experiments and Results

- Number of benchmarks: 26
- Number of OpenCl benchmarks: 6
- Number of CUDA becnhmarks: 20
- Source of benchmarks: Open source Github repositories and GKLEE test samples
- Type of benchmarks: Image processing, data mining, mathematical operations, etc.
- Test conducted: GPUVerify - 6 Opencl, 14 CUDA, GKLEE - 16 CUDA

Experiments and Results

Sr No	Property	Type / Value
1	CPU	Intel ®Core ™ i7-3770
2	Clock Speed	3.40 GHz
3	Number of Cores	8
4	Graphics	Intel ®IvyBridge Desktop
5	Operating System	Ubuntu 14.04 LTS
6	OS Type	64 bit
7	System Memory	8 GB
8	Disk Size	483.8 GB

Table: System Specifications

Experiments and Results

Id	Benchmark	Data Race	Barrier Divergence	Time (seconds)
6	N-Body Computation	2	2	39.7
7	PI Estimation	3	0	3.9
8	MatrixMultiply2	8	0	6.7
9	Image Blur	0	0	0.7
10	Pairwise sums timed	4	0	1.6
11	GPU kmeans	8	0	4.5
12	Vector Sums	1	0	1.2
13	Matmul	0	0	1.4
14	Pairwise sums	4	0	1.7
15	Cube	1	0	1.1
16	Square	1	0	1.1
17	Deadlock0	3	1	1.4
18	Deadlock2	0	1	1.3
19	Seive1	2	0	1.5

Table: GPUVerify results time for CUDA benchmarks

Id	Benchmarks	Errors		Performance Bugs			Time (seconds)
		DR #	BD #	BCR %	WDR %	MCR %	
10	Pairwise-sums timed	1	0	0	2, 45	100	1m 21.9s
11	GPU kmeans	1	0	0	0, 25	96, 75	1m 33.8s
12	Vector Sums	1	0	0	0	100	0m 0.9s
13	Matmul	1	0	0	50	100	0m 3.7s
14	Pairwise sums	1	0	0	50	100	0m 0.5s
15	Cube	1	0	0	0	100	0m 5.2s
16	Square	1	0	0	0	100	0m 3.4s
17	Deadlock0	0	1	NA	NA	NA	0m 1.4s
18	Deadlock2	0	1	0	50	100	0m 2.8s
19	Seive1	1	0	0	100	100	0m 6.3s
21	Interblock race	1	0	0	0	100	0m 0.5s
23	Bank Conflict	0	0	100	0	100	0m 1.4s
26	SumMatrix-2D grid 2D block	0	0	0	100	100	1m15.8s

Table: GKLEE results for CUDS benchmarks

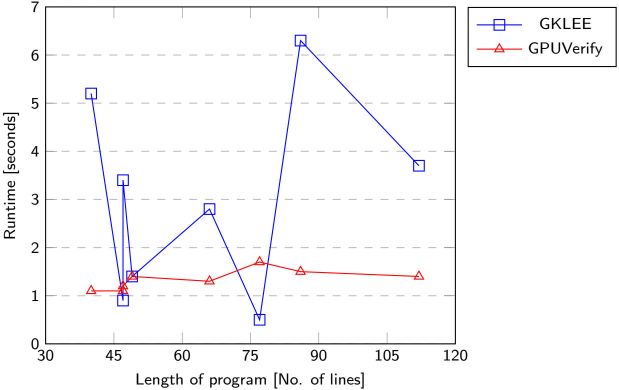
Analysis

Id	Benchmark	Number of Data Races detected		Remarks
		GPU Verify	GKLEE	
10	Pairwise sums timed	4	1	GKLEE exits after first data race is detected
11	GPU Kmeans	8	1	GKLEE exits after first data race is detected
13	Matmul	0	1	GKLEE reports a benign data race
14	Pairwise sums	4	1	GKLEE exits after first data race is detected
17	Deadlock0	3	0	GKLEE exits after reporting a potential deadlock (barrier divergence)
18	Deadlock2	0	0	Neither tool reports any data races
19	Seive1	2	1	GKLEE exits after first data race is detected

Table: Comparative analysis of data races reported by GPUVerify and GKLEE

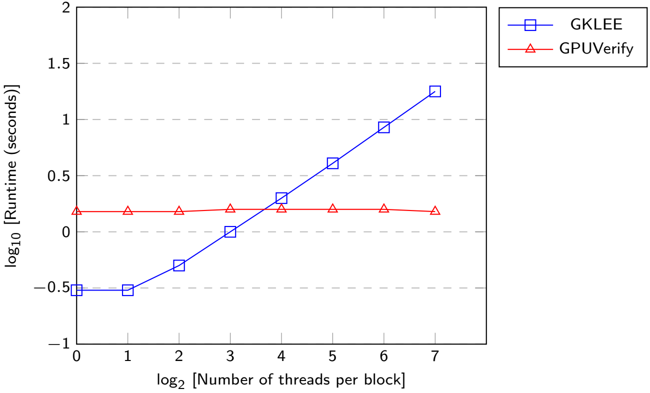
Analysis

Graph 5.2: Variation in runtime with length of code



Analysis

Graph 5.3: Runtime of GKLEE and GPUVerify for Pairwise Sums



Conclusion

- Scope of the software
- Portability, learn-ability and usability issues
- Execution time
- Recommended use

Summary

- GPUVerify and GKLEE provide much needed and useful mechanisms to verify GPU software
- GPUVerify takes less time and is more portable
- GKLEE provides detailed results and reports performance pitfalls

- Future work
  - False positive and negatives
  - Qualitative classification of benchmarks