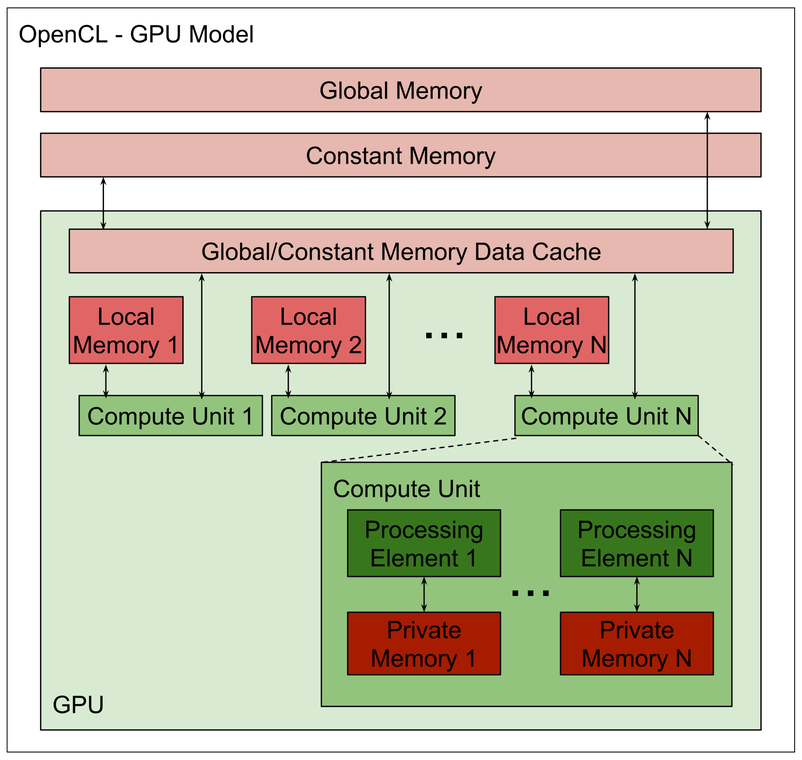
CLTestCheck: Measuring Test Effectiveness for GPU Kernels

Chao Peng, Ajitha Rajan
School of Informatics, University of Edinburgh, UK



Motivation

- Writing correct GPU programs is challenging due to the huge number of threads clustered into **work-groups** and the **three-level memory hierarchy**.
- Existing work focuses on program verification by static and dynamic code analysis.
- There exist no means of measuring effectiveness of tests developed for GPU kernels.

Contributions

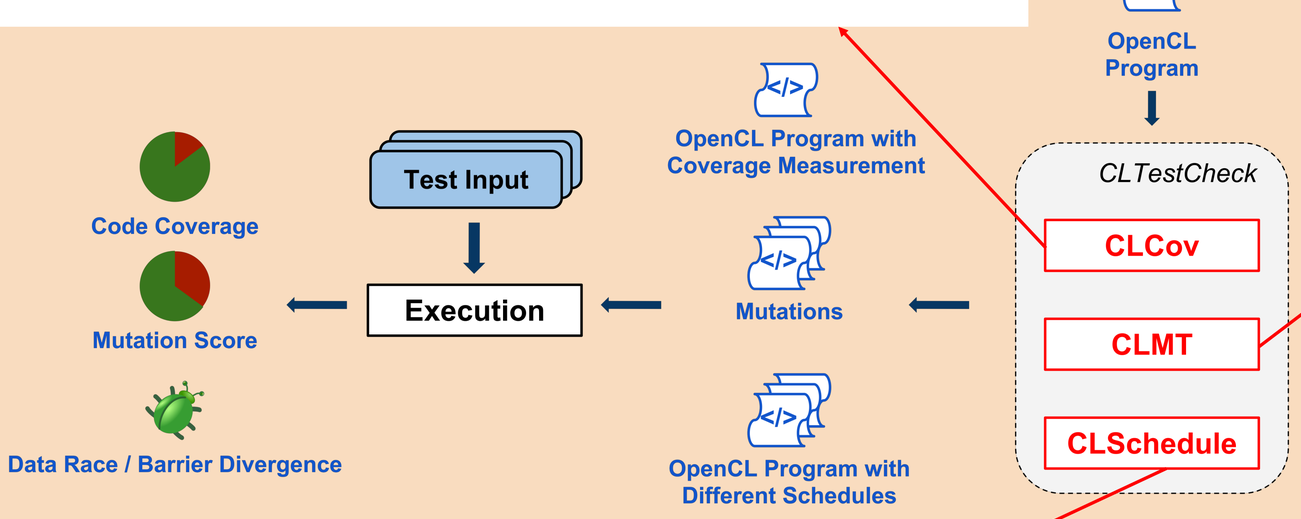
- Definition of three OpenCL code **coverage metrics** inspired by real faults.
- Automate **measuring code coverage** achieved by test suites.
- Automate **measuring fault finding** capability of test suites.
- Work-group **schedule amplifier** to check for potential deadlocks and data races with a given test suite that may be caused by inter work-group dependencies.

Our Approach

Code Coverage Measurement

Coverage criteria: *Branch Coverage, Loop Boundary Coverage, Barrier Coverage*

The framework automatically instruments the GPU program and code coverage is recorded by the inserted data structure and code blocks.



Fault Seeding

The framework generates mutations using:

- **Operator mutants**: mutating arithmetic, relational, bitwise, logical and assignment operators;
- **Barrier mutants**: deleting barrier function call;
- **Image coordinate mutants**: changing coordinates of image accesses;
- **Loop boundary mutants**: changing the boundary value in loop conditions.

Schedule Amplification

- We implemented the first OpenCL *Schedule Amplifier* to generate different work-group schedules.
- In each schedule, one work-group is selected to execute first and when it finishes execution, other work-groups are enabled to proceed.
- The selected work-group ID is uniformly distributed in the space of work-groups.

Case Studies

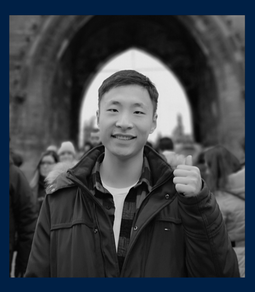
4 industry standard OpenCL benchmark suites:

- Parboil
- Rodinia
- Scan
- PolyBench

Totaling 82 OpenCL kernels.

Results

- CLTestCheck is able to automatically measure coverage achieved by test inputs and reveal uncovered code blocks that may need further verification.
- CLTestCheck is able to automatically generate different types of mutations. Fault finding and total coverage were highly correlated for these benchmarks.
- Schedule amplification revealed data races in two subject programs.



Chao Peng
chao.peng@ed.ac.uk
chao-peng.github.io

Ajitha Rajan
arajan@ed.ac.uk
homepages.inf.ed.ac.uk/arajan/

THE UNIVERSITY of EDINBURGH
informatics