Imperial College London

Barrier Invariants: a Shared State Abstraction for the Analysis of Data-Dependent GPU Kernels

Nathan Chong, Alastair F. Donaldson,
Paul H.J. Kelly, Jeroen Ketema
Imperial College London
{nyc04, afd, phjk, jketema}@imperial.ac.uk

Shaz Qadeer
Microsoft Research
qadeer@microsoft.com

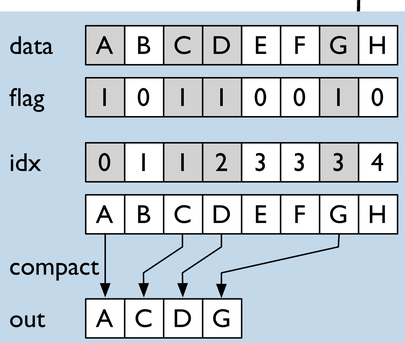
We attack the problem of analysing data-dependent GPU kernels, a tricky and important class of GPU program. Our work enables scalable analysis where existing techniques cannot be applied and where exhaustive symbolic execution is often infeasible.

Data-Dependent GPU Kernels

A GPU kernel is **data-dependent** if the input to the program can change the data or control flow of the program. This class of program is tricky for verification because the access pattern of a single thread may depend on the data or control flow of many other threads.

Consider this Stream Compaction kernel, which filters an input array. In the compact stage the elements to be kept are written contiguously to the output array. The index for each thread is computed using the **parallel prefix sum** primitive.

This kernel is data-dependent since the index written-to by a thread is dependent on the flag result of all 'preceding' threads. *This kernel cannot be verified with existing tools.*



Data-Dependent Stream Compaction

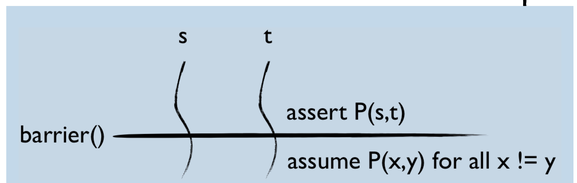
The Problem

Existing verifiers, such as GPUVerify [0] and PUG [1], achieve scalability by exploiting the following observation: if a kernel is race-free for an arbitrary pair of threads then it must be race-free for all possible pairs of threads. Therefore, to establish race-freedom it suffices to consider the behaviour of an *arbitrary* pair of threads with an abstraction to over-approximate the behaviour of other threads.

The simplest abstraction is to make no assumptions about the behaviour of other threads: at each barrier we assume that shared state is arbitrarily updated by setting the contents of shared state nondeterministically. This abstraction is too coarse to correctly reason about data-dependent GPU kernels: consider the contents of the `idx` array in the Stream Compaction kernel, after this abstraction has been applied. The tool will report a false-positive.

Our Solution

Barrier invariants are a new abstraction that retain the scalability of the two-thread reduction, but allow more precise reasoning about shared state. The idea is to annotate a barrier with an invariant that must hold every time the barrier is reached. The property must hold for an arbitrary pair of threads. Then, after the barrier, shared state is set to an arbitrary value that additionally satisfies the barrier invariant.



In the case of the Stream Compaction kernel we can prove the following barrier invariant about the compact stage:
flag[s] ∧ flag[t] ⟶ idx[s] ≠ idx[t] for all distinct threads s and t
which allows us to prove race-freedom for the whole kernel.

Results

In our paper we give a formal semantics and proof of soundness for barrier invariants. Then, suitably equipped, we give a functional correctness result for 3 different parallel prefix sum algorithms, in order to ultimately prove race-freedom for the Stream Compaction kernel. We show that our approach allows us to scale to sizes (more than 32k threads) that are infeasible for dynamic symbolic execution techniques.

Our toolchain and experimental evaluation are available as an artifact.
http://multicore.doc.ic.ac.uk/tools/GPUVerify/OOPSLA13/

Summary

Barrier invariants are a new shared-state abstraction that enable scalable analysis of data-dependent GPU kernels.

References

[0] A. Betts, N. Chong, A. F. Donaldson, S. Qadeer, and P. Thomson. GPUVerify: a verifier for GPU kernels. In OOPSLA, pages 113–132, 2012.

[1] G. Li and G. Gopalakrishnan. Scalable SMT-based verification of GPU kernel functions. In FSE, pages 187–196, 2010.

Talk details

Mobile & Graphics Track
1'30pm Thursday 31st October 2013

This work was supported by the EU FP7 STREP project CARP (project number 287767) and the EPSRC PSL project (EP/I006761/1).