Motivation

- Provide insight into runtime behaviour of CUDA threads
- Help programmers optimize their code by discovering hidden bottlenecks caused by non-optimal memory accesses
- Design an easy to use CUDA instrumentation framework - current tools are difficult to set up and break with new framework releases
- Record all accesses - current tools provide only aggregated data

Contribution

- CUDA profiling library that records memory accesses
- Analytical and visualization tool
- Key features:
  - detailed thread memory access recording
  - shared memory conflict detection
  - visualization of memory access patterns
  - simple setup (just include a single header file)
  - independent of CUDA SDK version

Implementation

- LLVM plugin that finds memory accesses in CUDA functions
- Accesses are wrapped with code that records their value, size, address, data type and thread ID at runtime
- GPU allocations are tracked to provide detailed address space information
- Accesses are exported to JSON or Protobuf files and can be optionally compressed
- Recorded data is visualized in a React web app, which provides filtering of accesses and displays shared memory conflicts and memory access strides

Conclusion

- Memory access anomalies become obvious when visualized
- Runtime behaviour inspection is useful for memory optimization experiments
  - Working open-source tool tested on official CUDA samples
  - Created library also serves as a maintainable CUDA instrumentation framework
  - Could be combined with CPU access recording and integrated into an IDE

https://github.com/kobzol/cuda-profile