Performance Evaluation of OpenCL Standard Support (and Beyond)

Tyler Sorensen, Princeton University
Sreepathi Pai, University of Rochester
Alastair F. Donaldson, Imperial College London

IWOCL 2019

Tyler Sorensen

Background

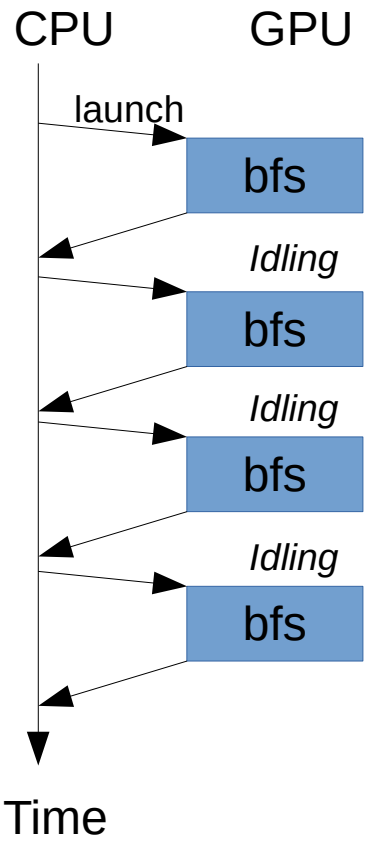
- IrGL: DSL for (Irregular) Graph Analytic Kernels
  - Wrote a compiler from IrGL to CUDA
  - Fastest graph kernels
  - Pai and Pingali, “A compiler for throughput optimization of graph algorithms on GPUs”, OOPSLA 2016
- Teamed up with Tyler and Ally to target OpenCL

IrGL Key Insight

- Graph algorithms suffer 3 bottlenecks
- Need 3 key optimizations for high performance
  - Iteration Outlining
  - Nested Parallelism (*not* OpenCL NP)
  - Cooperative Conversion

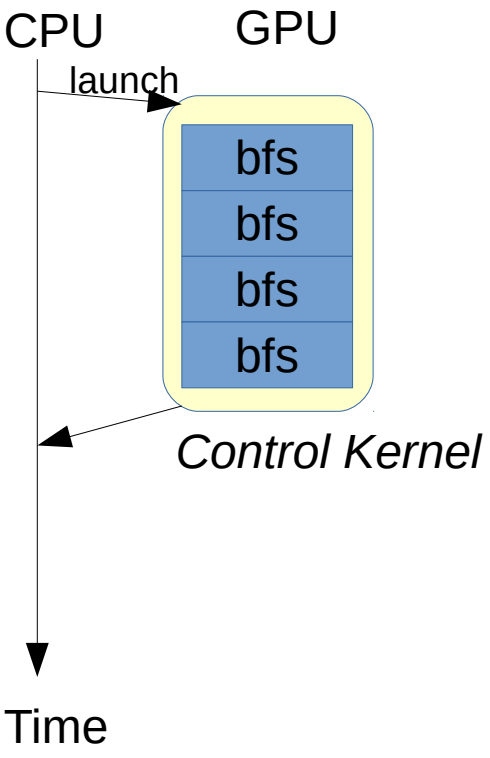
Problem: CPU—GPU Launch

- Most graph algorithms are iterative
  - Repeat until fixpoint
- If time per iteration is small (average ~20us for BFS), launch throughput can’t keep up



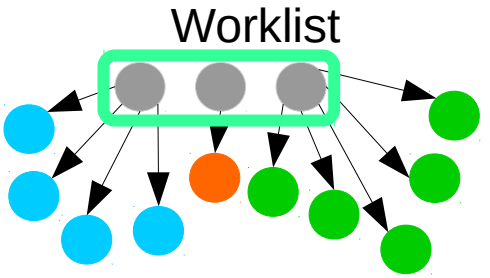
Optimization #1: Iteration Outlining

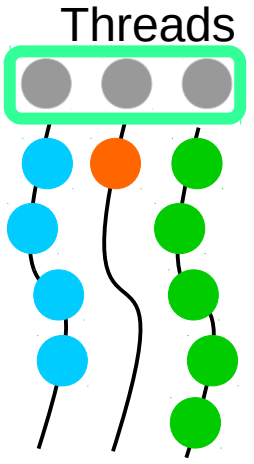
- Compiler generates a control kernel that “launches” child kernels
  - Actually inlines them
- Need a global barrier between “kernel invocations” (now function calls)
- OpenCL Device-side Enqueue (Nested Parallelism) not a good fit



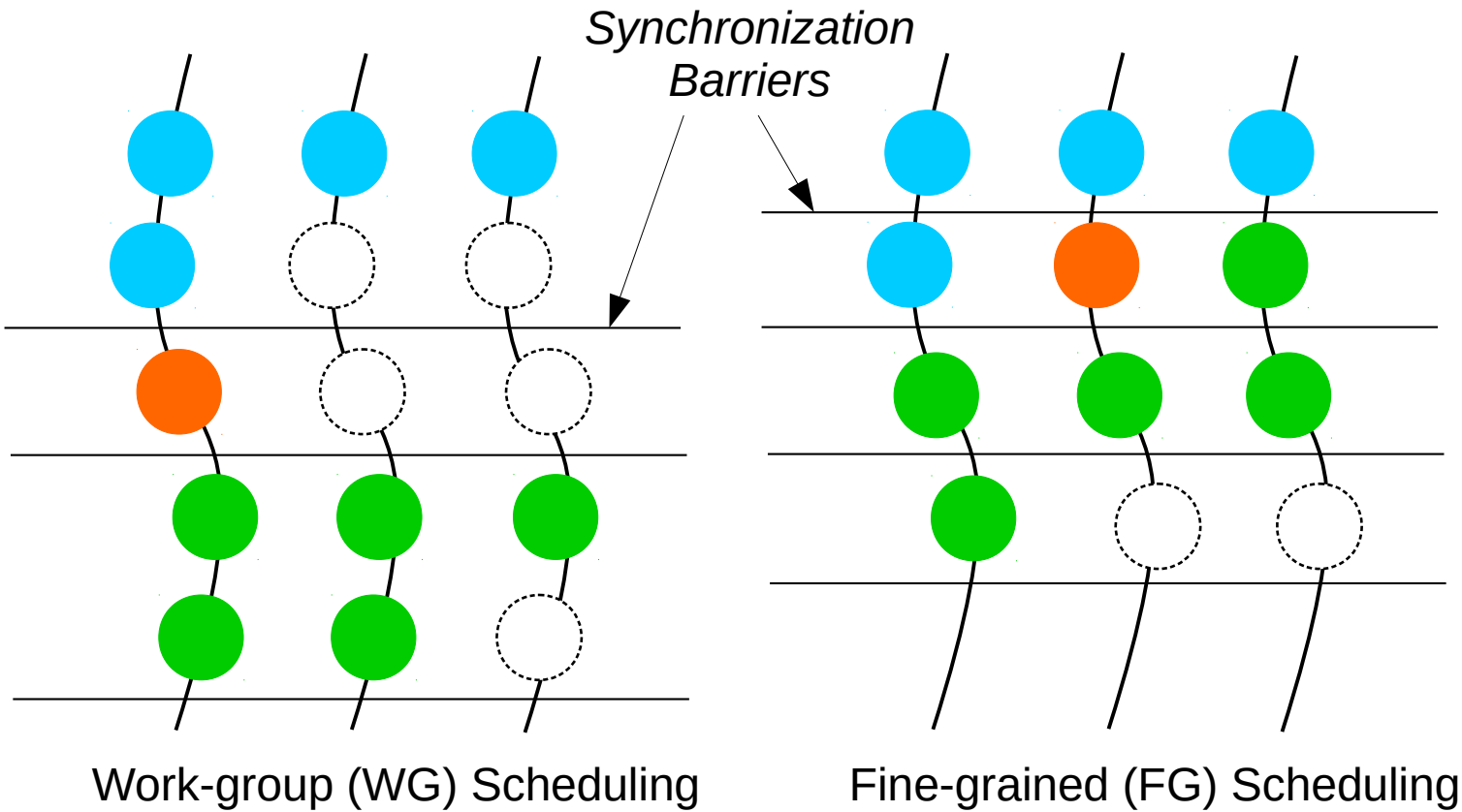
Problem: Load Imbalance

- Graph algorithms usually two *parallel* nested loops
  - Outer loop over vertices
  - Inner loop over edges of a vertex
- Graph edge distribution can be very skewed
  - Social network graphs
- Leads to load imbalance if statically scheduled





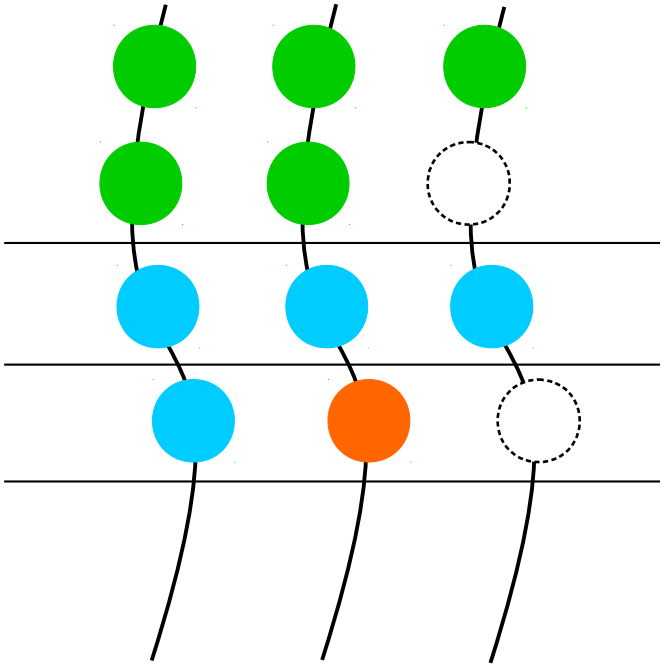
Optimization #2: Nested Parallelism



NP: Multiple Schedulers

Use Work-group (WG) for high-degree nodes

Use fine-grained (FG) for low-degree nodes

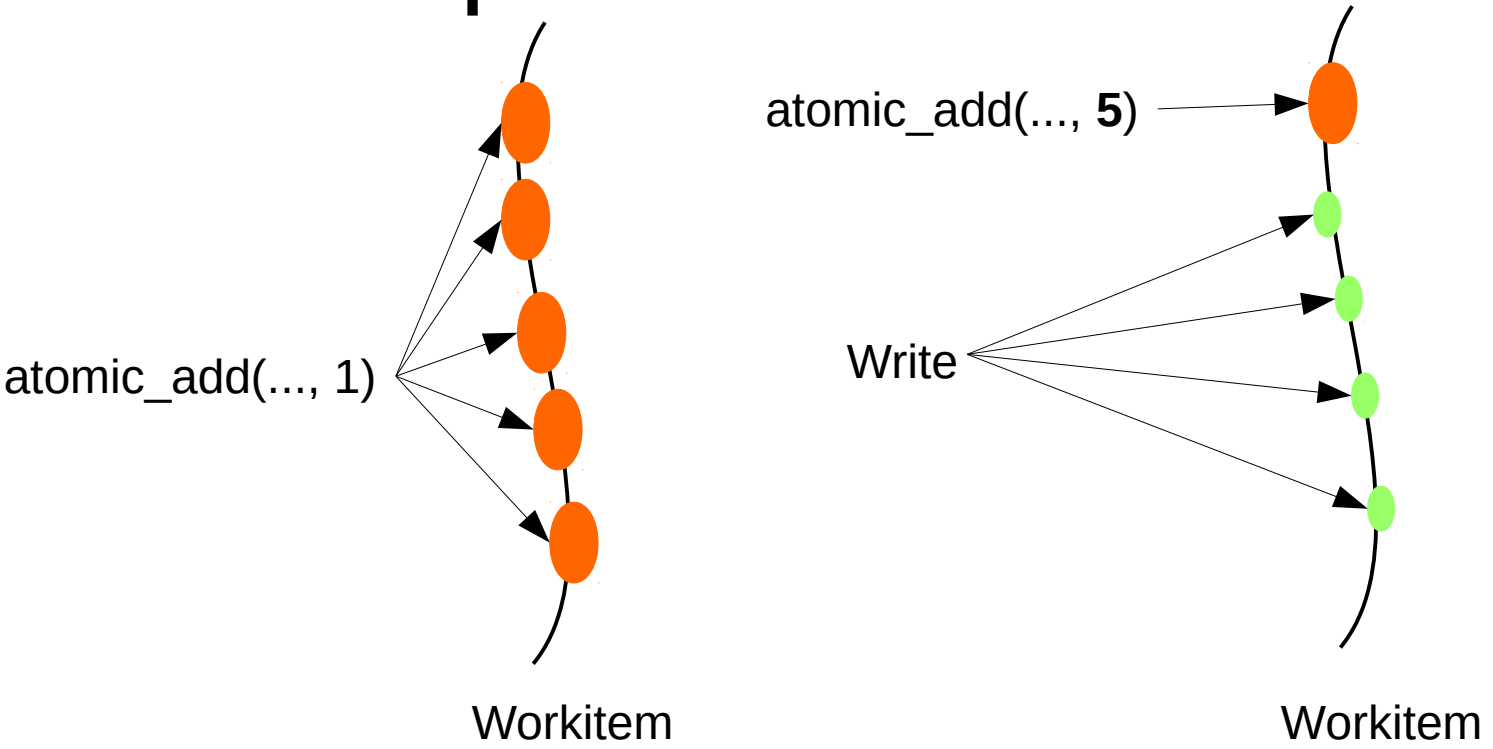


Work-group (WG) + Finegrained (FG) Scheduling

Implementing IrGL NP in OCL

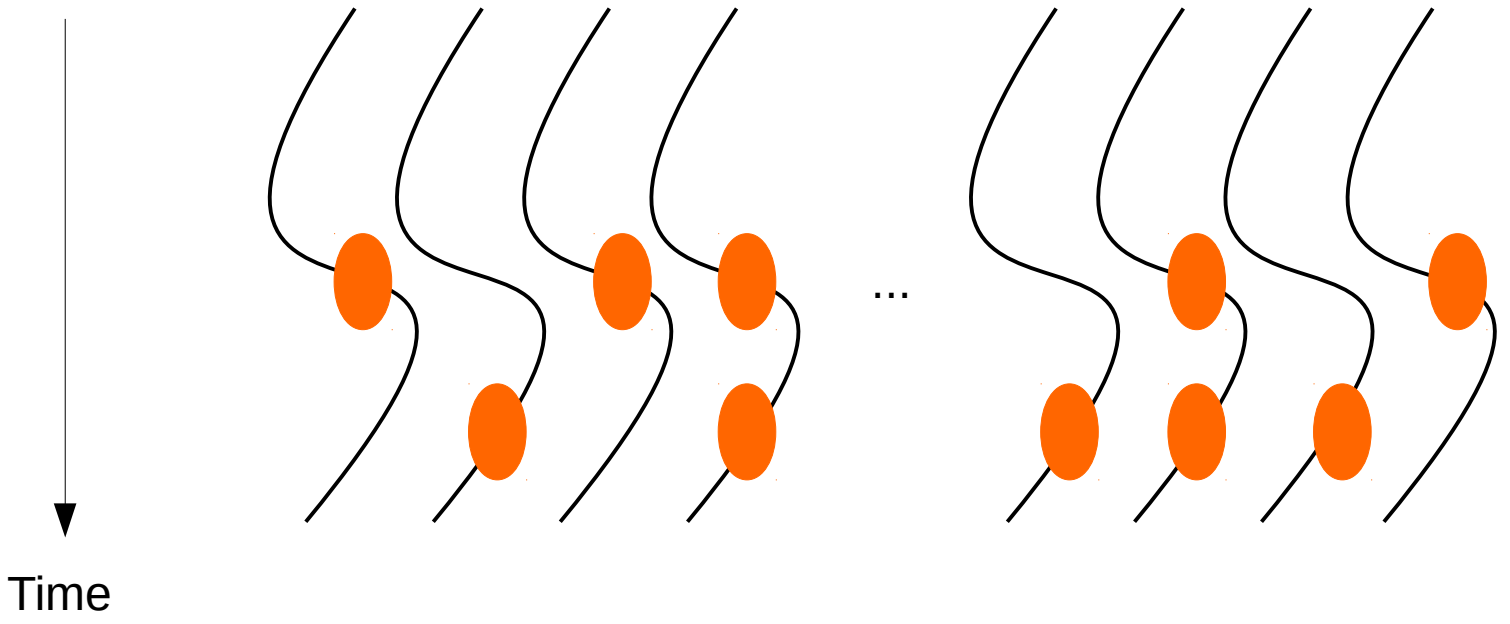
- Support any combination of *three* schedulers
  - workgroup (wg)
  - subgroup (sg)
  - finegrained (fg)
- Workgroup and Finegrained schedulers require:
  - local memory and workgroup barriers
- Subgroup scheduler requires:
  - subgroup barriers and reductions

Optimization #3: Cooperative Conversion



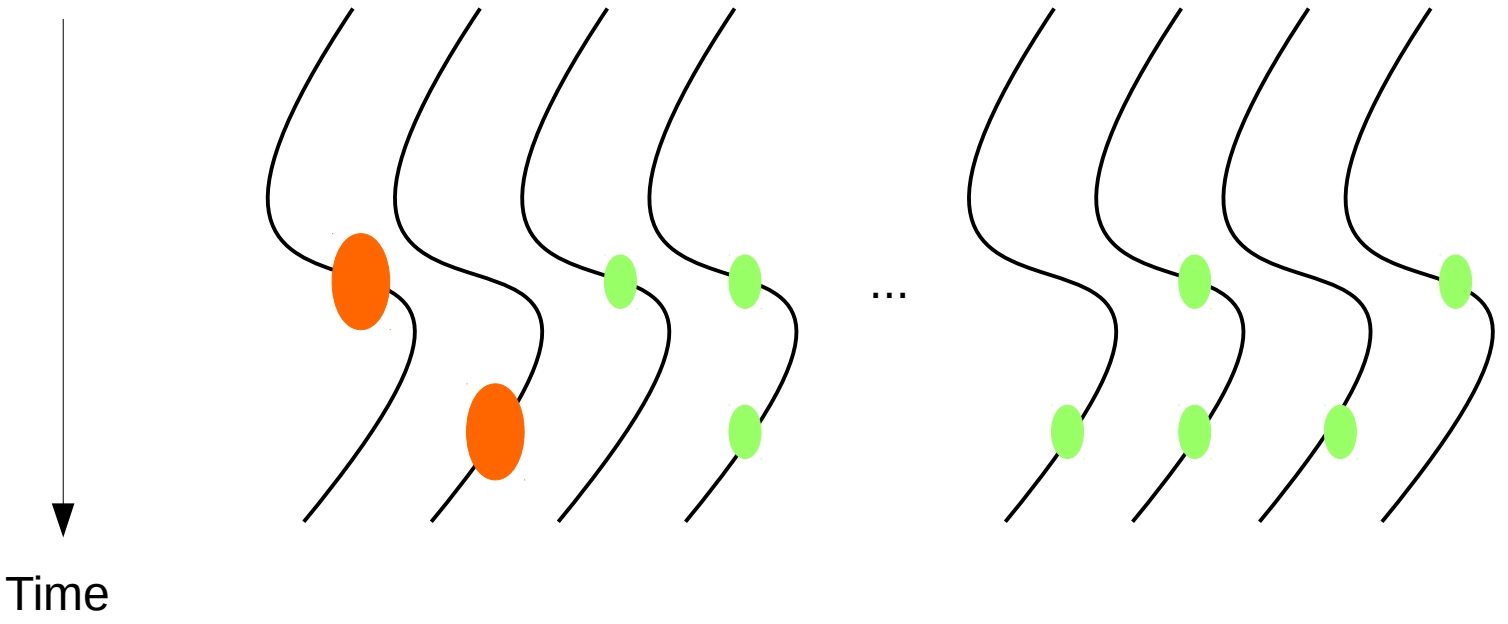
Dynamic aggregation problem

```
if(edge.dst.level == INF)
    Worklist_push(edge.dst)
```



Aggregate across workitems

```
if(edge.dst.level == INF)
    Worklist_push(edge.dst)
```



Implementing Coop-Conv in OCL

- Supports aggregation:
  - within a workitem
  - across a subgroup
  - across all workitems of a workgroup
- Workitems use prefix scans for aggregation, requiring barriers
  - Unlike CUDA, even subgroup aggregation requires barriers – OCL does not support lockstep execution
- Need to place barriers safely to avoid deadlock
  - Extended our prior compiler analysis to find subgroup uniform branches

GPUs We Used

Vendor	GPU	OpenCL version
ARM	Mali 4	1.2
NVIDIA	GTX 1080	1.2
	Quadro M4000	1.2
AMD	R9	2.0
Intel	HD 5500	2.0
	Iris 6100	2.0

Portable C11-style Atomics (OCL2)

- Required for: coop-conv, wg, and oitergb
  - Supported by Intel and AMD (already OpenCL 2.0)
- NVIDIA: Hardware support available, so just use PTX intrinsics
- ARM: Use memory fences and hope for the best
  - Verify correctness using application test suites

Portable subgroups (OCL2.1)

- Required for:
  - coop-conv,
  - nested parallelism (sg)
- NVIDIA: Use PTX intrinsics
- Intel and AMD: Use vendor-specific extensions
  - Intel subgroup sizes can vary per kernel!
- ARM: Assume subgroup size of 1

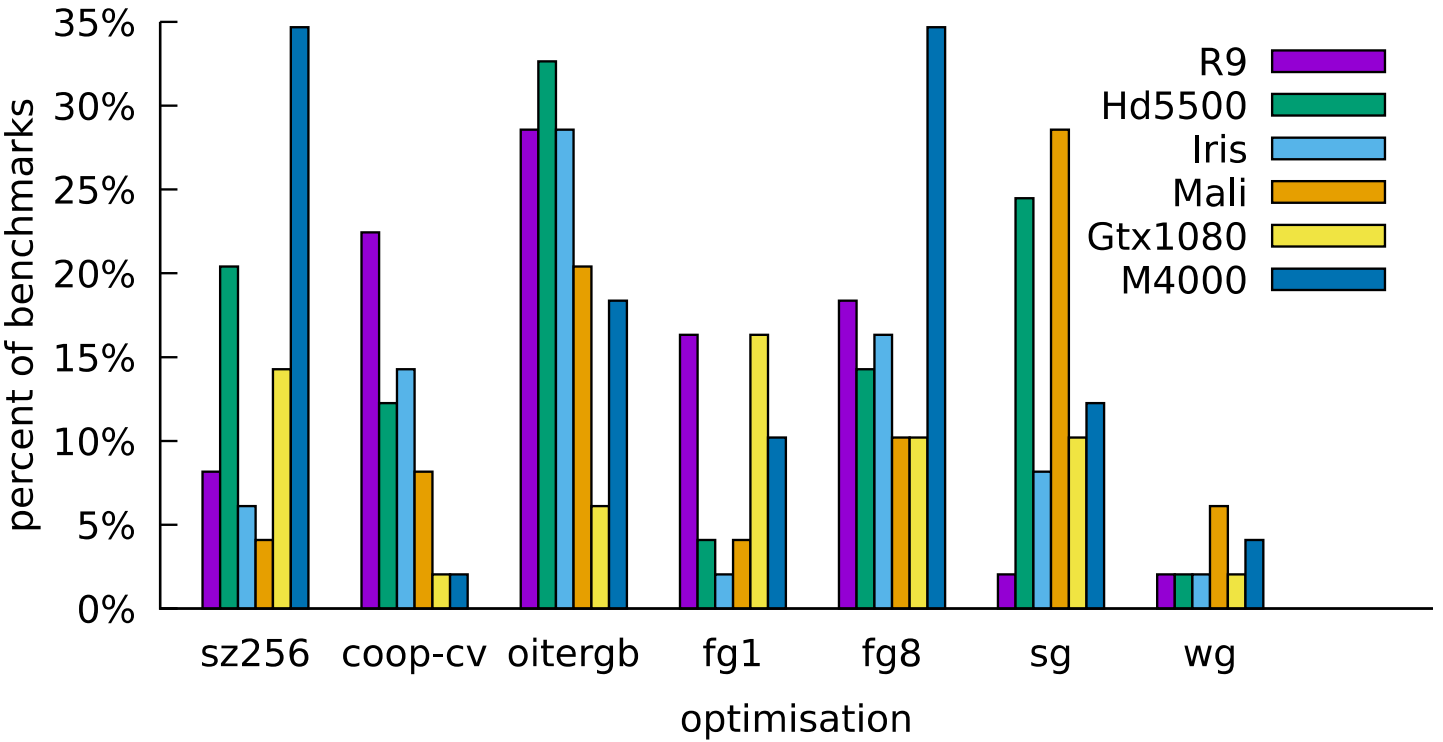
Portable global barriers (OCL-FP)

- Required for: Iteration Outlining (oitergb)
- Assume occupancy-bound architecture
- Kernel includes a prologue to detect how many workgroups are running (can undercount)
  - All workgroups with ID greater than this count exit
- Details in: Sorensen et al., “Portable inter-workgroup barrier synchronisation for GPUs”, OOPSLA 2016

Experimental Setup

- 17 graph algorithms
- 3 graph inputs: Road, Random, R-MAT
- 6 GPUs, from 4 vendors (NVIDIA, Intel, ARM and AMD)

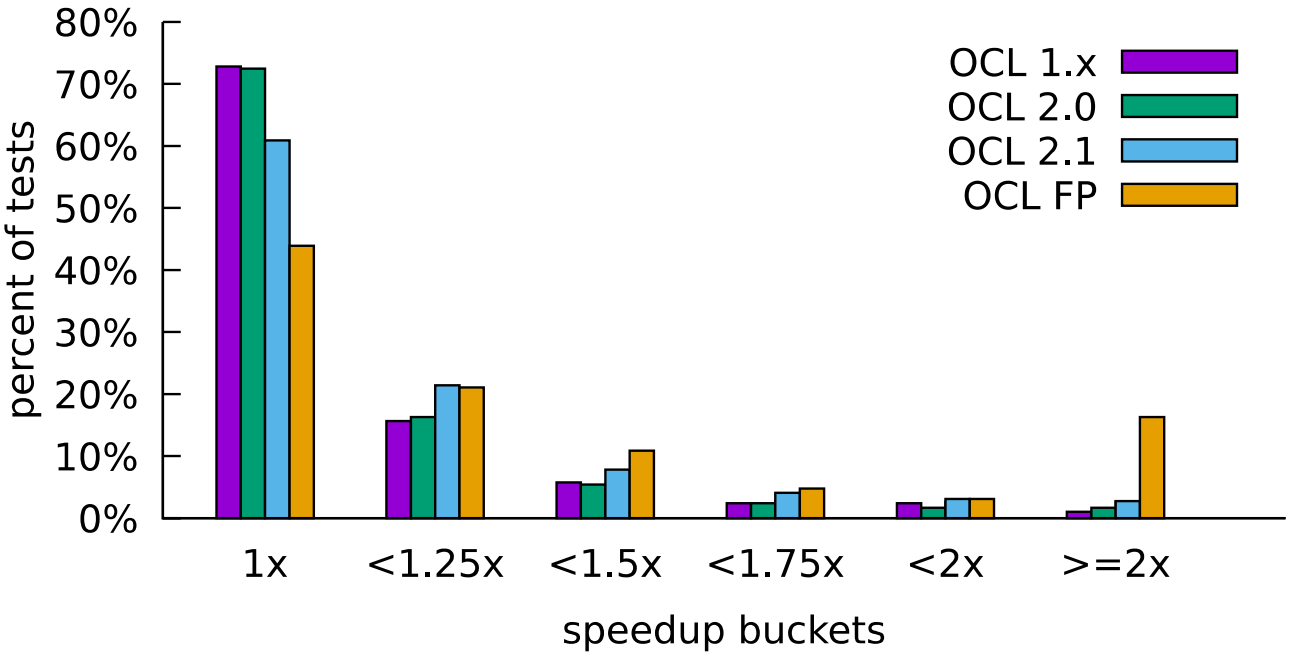
Portability of Optimisations



Portability of Optimisations (contd.)

- All optimisations required to achieve speedup for some fraction of architecture + benchmark + input combinations
- Optimisations are not NVIDIA-specific

OpenCL Speedup Classes



Conclusion

- Newer OpenCL features lead to better performance
- Best performance obtained when OCL-FP features are used – though these are not yet supported

Thank you!

- For more details, see Tyler’s PhD Thesis

sree@cs.rochester.edu
https://cs.rochester.edu/~sree