hLRC: Lazy Release Consistency for GPUs
John Alsop†, Marc Orr‡§, Brad Beckmann§, David Wood‡§
†UIUC, ‡UW-Madison, §AMD Research

Motivation

GPUs coherence is inefficient
- Release Synch: flush cache
- Acquire Synch: invalidate cache
- Poor reuse when synchronization is frequent

Scoped synchronization improves efficiency, but...
- Complicates memory model
- Limits communication patterns (e.g. work stealing)

Existing Solutions

Remote Scope Promotion (Orr et al. ASPLOS’15)
- Add new remote scope semantics
- Used to promote remote access to shared level
- Implemented using broadcast flush/invalidate

DeNovo Coherence for GPUs (Sinclair et al. MICRO’15)
- Scalable registration tracks locally modified data
- Release Synch: register non-registered dirty data
- Acquire Synch: invalidate non-registered cache data

hLRC Design

Like Remote Scope Promotion:
- Avoids coherence actions for local synch
- Moves synch overhead to non-local (steal) access

Like DeNovo:
- Eliminates heterogeneous races
- Uses DeNovo registration to track atomics

Like Lazy Release Consistency for DSM:
(Keleher et al. ISCA’92)
- Automatically detects local synchronization
- Only performs coherence actions for remote synch

Evaluation

Performance Comparison

Simulation Environment
- Extended version of AMD gem5 APU simulator
- 128 CUs
- 16KB private L1, 4MB shared L2

Workloads
- Benchmarks from Pannotia benchmark suite (Chen et al. ISWC-13)
  - Single Source Shortest Path (SSSP)
  - Graph Coloring (color)
- Pagerank (PR)
- Graph inputs from Florida sparse matrix collection (Davis et al. TOMS’11)
- Added per-CU task queues and work stealing

Optional Scopes to Improve hLRC Performance

If synch has minimal locality, hLRC causes excessive flush/invalidate
⇒ Use broader scope
Remote steals are expected to exhibit less locality
⇒ Try agent scope (hLRC-scope)
Speedup is comparable
⇒ Synch locality exists in steals