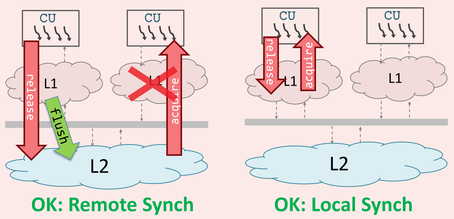
hLRC: Lazy Release Consistency for GPUs

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

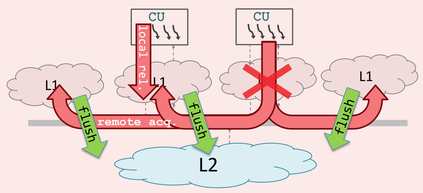
Scoped synchronization improves efficiency, but…

- Complicates memory model
- Limits communication patterns (e.g. work stealing)

Existing Solutions

Remote Scope Promotion (Orr et al. ASPLOS'15)

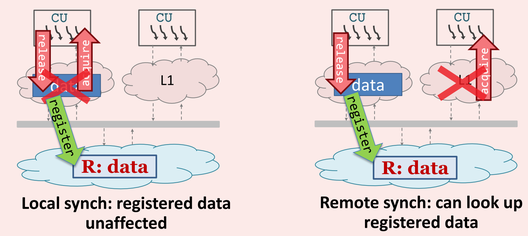
- Add new remote scope semantics
- Used to promote remote access to shared level
- Implemented using broadcast flush/invalidate



No actions for local synch!
Adds complexity
Broadcasts scale poorly

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

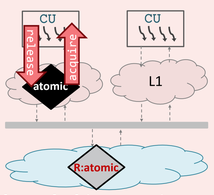


No heterogeneous races!
Improved reuse in modified data
Limited reuse in non-modified data
Adds registration overhead

hLRC Design

Like Remote Scope Promotion:

- Avoids coherence actions for local synch
- Moves synch overhead to non-local (steal) access



No actions for local synch!
Detected automatically

Like DeNovo:

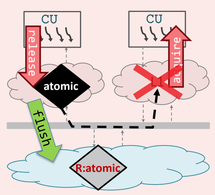
- Eliminates heterogeneous races
- Uses DeNovo registration to track atomics

Heterogeneous Lazy Release Consistency

- Track atomic accesses using DeNovo registration
- Atomic registration change = possible remote synch
- Invalidate cache when registration is obtained
- Flush cache when registration is lost

Like Lazy Release Consistency for DSM:
(Keleher et al. ISCA'92)

- Automatically detects local synchronization
- Only performs coherence actions for remote synch



No heterogeneous races!
Lower registration overhead
Sensitive to synch locality…
…but can mitigate with scopes

Evaluation

Simulation Environment

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

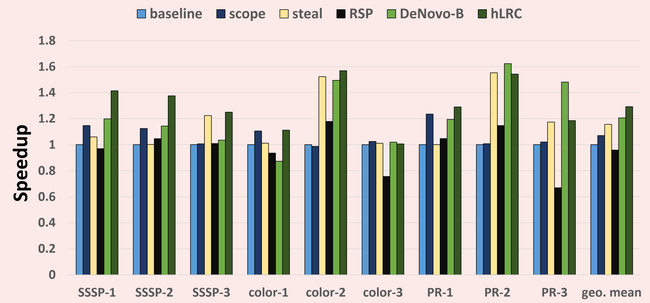
Workloads

- Benchmarks from Pannotia benchmark suite (Che et al. IISWC'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

Scenario	Coherence	Steal enabled?	Pop scope	Steal scope
baseline	Scopes	no	agent	N/A
scope-only	Scopes	no	wg	N/A
steal-only	Scopes	yes	agent	agent
RSP	Scopes+RSP	yes	wg	remote agent
DeNovo-B	DeNovo*	yes	N/A	N/A
hLRC	hLRC	yes	N/A	N/A

*Block-granularity DeNovo is used, with no coherence regions

Performance Comparison



- Broadcasts cause RSP to harm performance on average
- Reg. overhead causes DeNovo to occasionally harm performance
- hLRC matches or exceeds best of scope-only, steal-only
  - 29% avg speedup relative baseline
  - 7% avg speedup relative DeNovo

Optional Scopes to Improve hLRC Performance

If synch has minimal locality,
hLRC causes excessive flush/inval
⇒Use broader scope

Remote steals are expected to exhibit less locality
⇒Try agent scope (hLRC-scoped)
Speedup is comparable
⇒Synch locality exists in steals