Efficient GPU Synchronization without Scopes: Saying No to Complex Consistency Models

Matthew D. Sinclair, Johnathan Alsop, Sarita V. Adve
University of Illinois @ Urbana-Champaign
hetero@cs.illinois.edu

Motivation

Heterogeneous systems now used for a wide variety of applications
Emerging applications have fine-grained synchronization

BUT current GPUs have sub-optimal consistency and coherence

	Consistency	*Coherence*
Defacto	Data-race-free (DRF) Simple	High overhead on synchs Inefficient
Recent	Heterogeneous-race-free (HRF) Scoped synchronization Complex	No overhead for local synchs Efficient for local synch

This work: simple consistency + efficient coherence

Motivation (Cont.)

Do GPU models (HRF) need to be more complex than CPU models (DRF)?

NO! Not if coherence is done right!

DeNovo+DRF: Efficient AND simpler memory model

- **Comparable or better results vs. GPU+DRF and GPU+HRF**

complex consistency models

Outline

- Motivation
- Coherence Protocols and Consistency Models
  - Classification
  - GPU Coherence
  - DeNovo Coherence
  - Coherence and Consistency Summary
- Results
- Conclusion

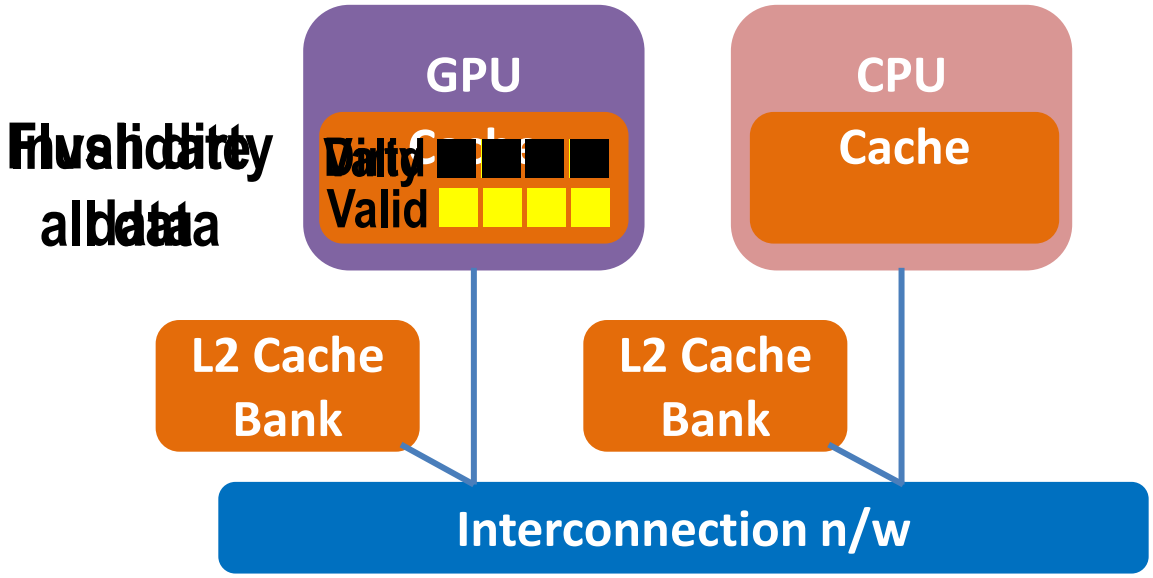
A Classification of Coherence Protocols

- Read hit: Don't return stale data
- Read miss: Find one up-to-date copy

		Invalidator	
		Writer	Reader
Track up-to-date copy	Ownership	MESI	DeNovo
	Writethrough		GPU

- Reader-initiated invalidations
  - No invalidation or ack traffic, directories, transient states
- Obtaining ownership for written data
  - Reuse owned data across synchs (not flushed at synch points)

GPU Coherence with DRF

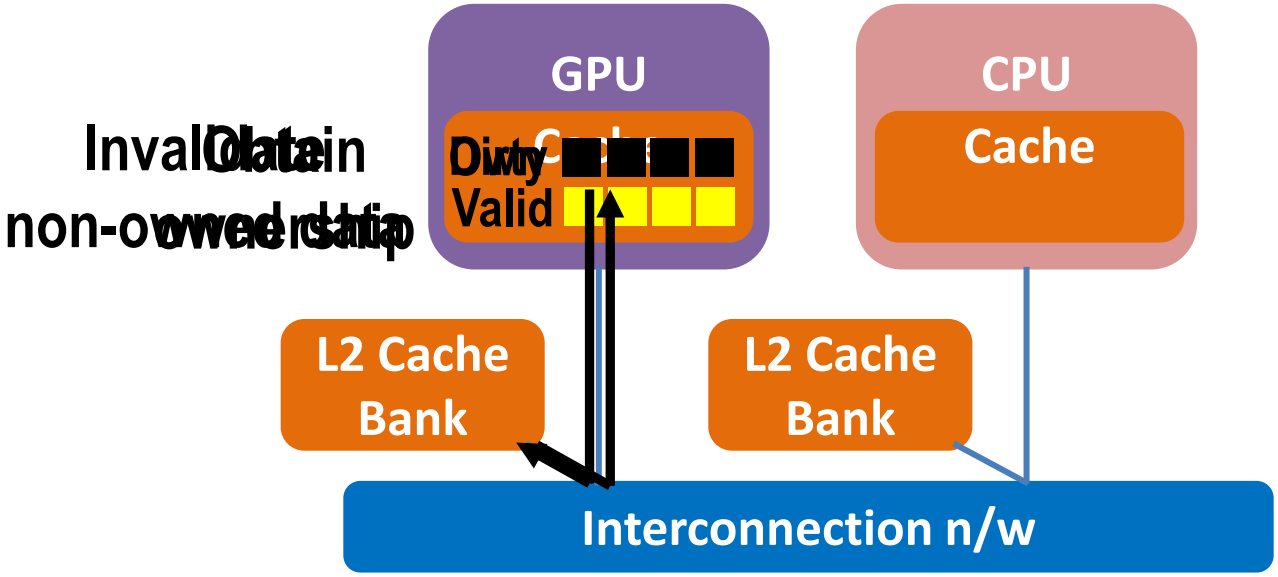


- With data-race-free (DRF) memory model
  - No data races; synchs must be explicitly distinguished
  - At all synch points
    - Flush all dirty data: Unnecessary writethroughs
    - Invalidate all data: Can't reuse data across synch points
  - Synchronization accesses must go to last level cache (LLC)

GPU Coherence with HRF

- With ~~data-~~race-free (~~DRF~~) memory model [ASPLOS '14]
  - No ~~data~~ races; synchs must be explicitly distinguished
  - At all synch points
    - Flush all dirty data: Unnecessary writethroughs
    - Invalidate all data: Can't reuse data across synch points
  - Synchronization accesses must go to last level cache (LLC)
  - No overhead for locally scoped synchs
- But higher programming complexity

DeNovo Coherence with DRF



- With data-race-free (DRF) memory model
  - No data races; synchs must be explicitly distinguished
  - At all synch points
    - ~~Flush all dirty data~~ Obtain ownership for dirty data
    - Invalidate all non-owned data

    } Can reuse owned data

    can be performed at L1
  - Synchronization accesses ~~must go to last level cache (LLC)~~
- 3% state overhead vs. GPU coherence + HRF

DeNovo Configurations Studied

- **DeNovo+DRF:**
  - Invalidate all non-owned data at synch points
- **DeNovo-RO+DRF:**
  - Avoids invalidating read-only data at synch points
- **DeNovo+HRF:**
  - Reuse valid data if synch is locally scoped

Coherence & Consistency Summary

Coherence + Consistency	Reuse Data		Do Synchs at L1
	Owned	Valid	
GPU + DRF (GD)	X	X	X
GPU + HRF (GH)	local	local	local
DeNovo + DRF (DD)	✓	X	✓
DeNovo-RO + DRF (DD+RO)	✓	read-only	✓
DeNovo + HRF (DH)	✓	local	✓

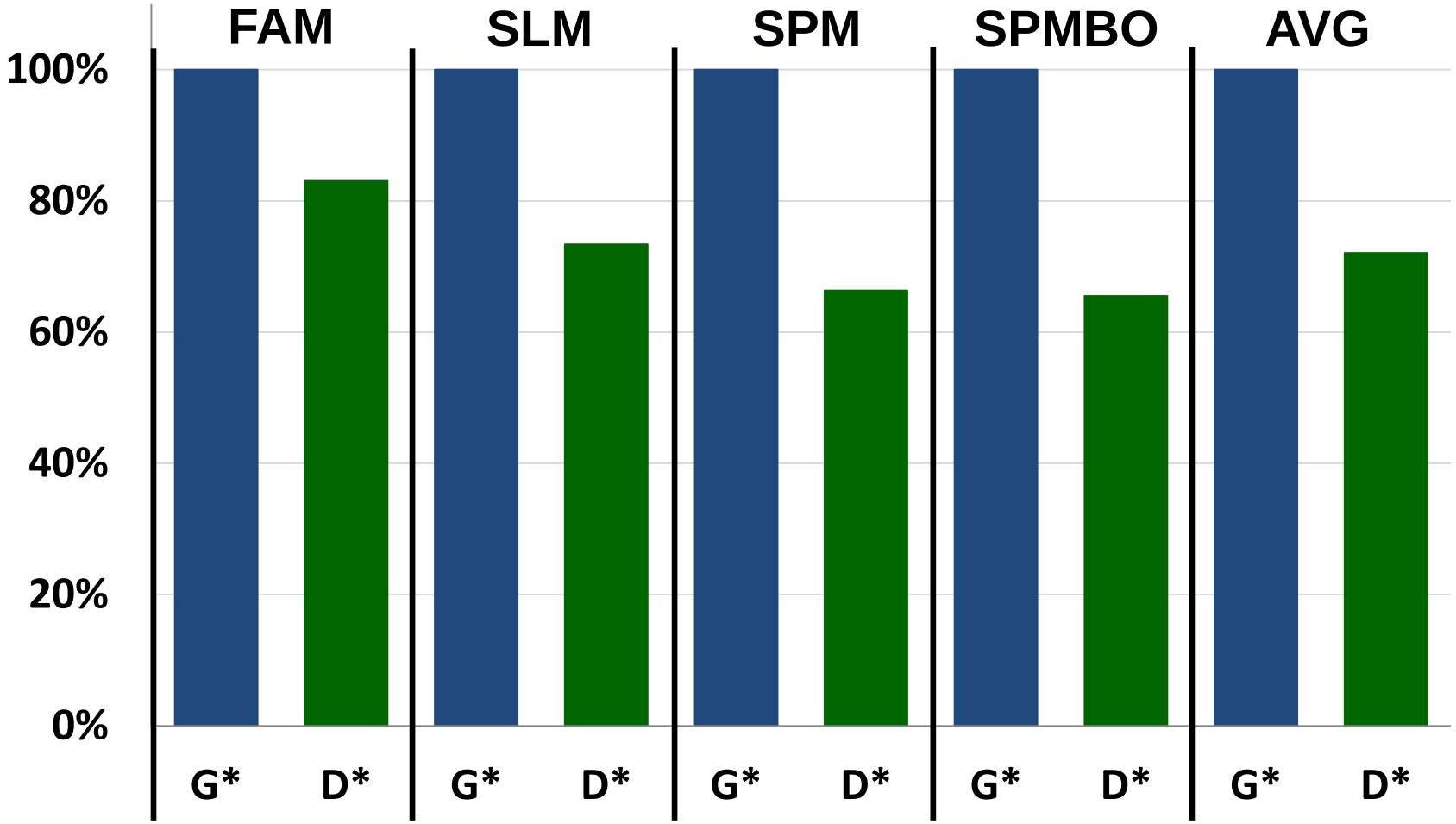
Outline

- Motivation
- Coherence Protocols and Consistency Models
- Results
- Conclusion

Evaluation Methodology

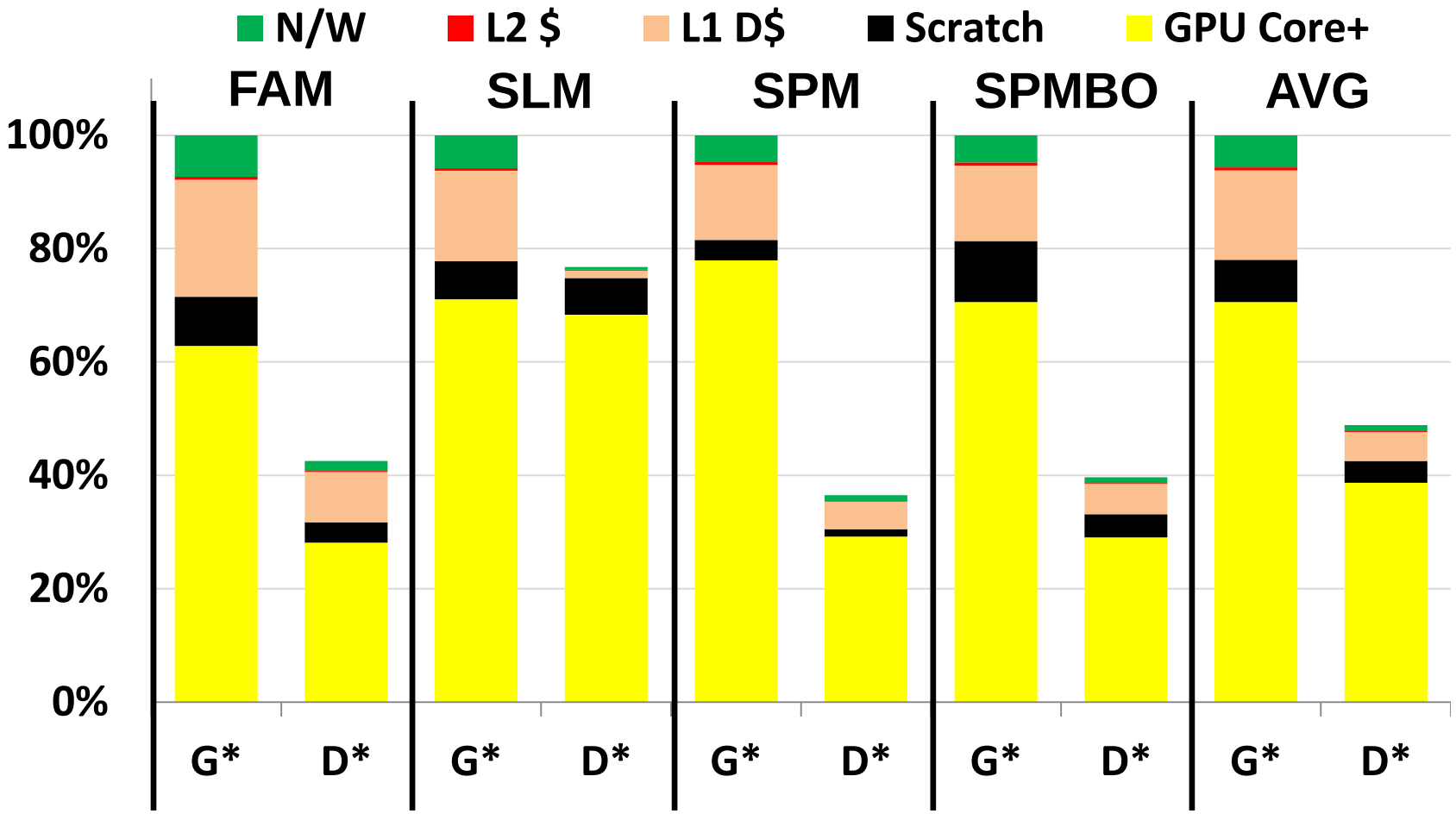
- 1 CPU core + 15 GPU compute units (CU)
  - Each node has private L1, scratchpad, tile of shared L2
- Simulation Environment
  - GEMS, Simics, Garnet, GPGPU-Sim, GPUWattch, McPAT
- Workloads
  - 10 apps from Rodinia, Parboil: no fine-grained synch
    - DeNovo and GPU coherence perform comparably
  - UC-Davis microbenchmarks + UTS from HRF paper:
    - Mutex, semaphore, barrier, work sharing
    - Shows potential for future apps
    - Created two versions of each: globally, locally/hybrid scoped synch

Global Synch – Execution Time



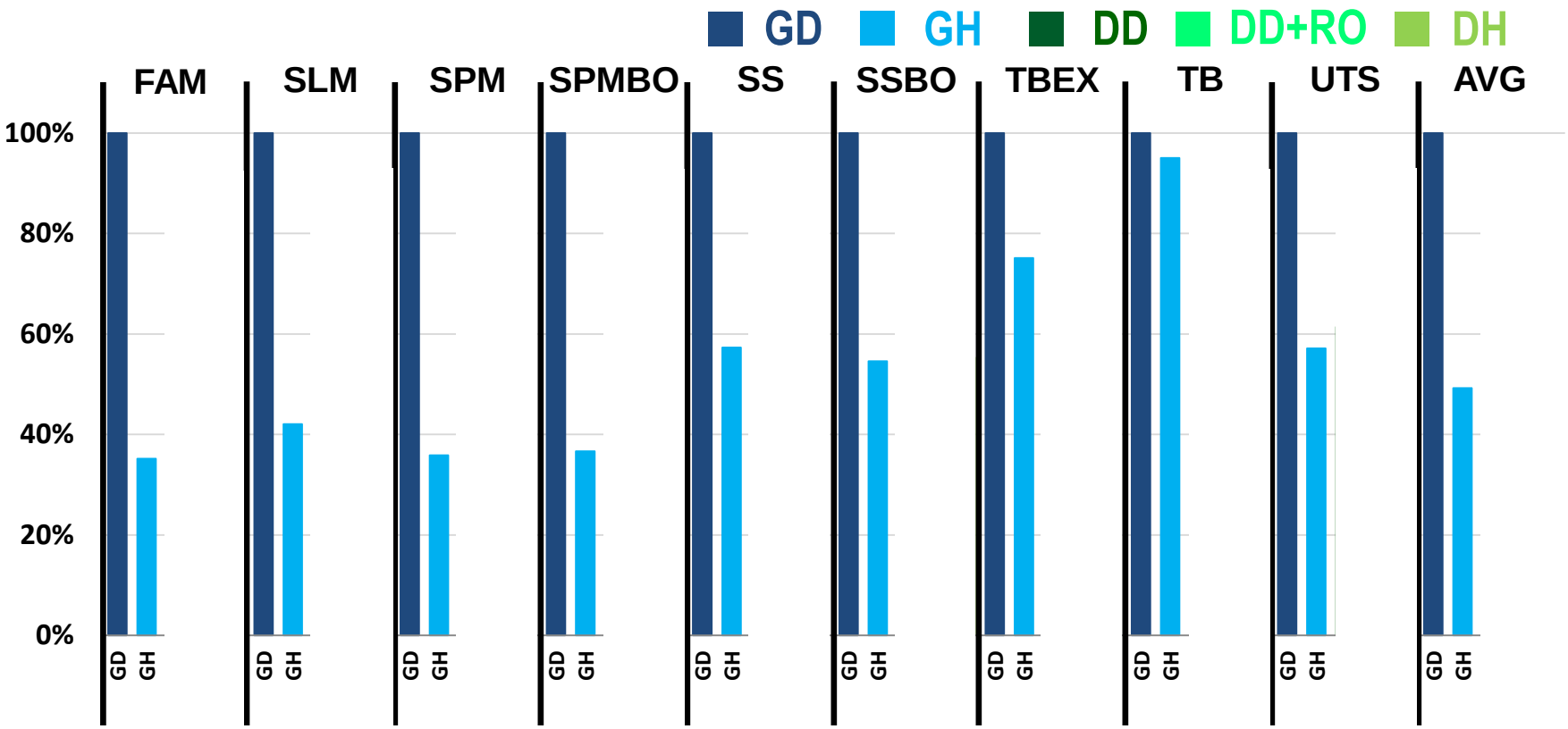
DeNovo has 28% lower execution time than GPU with global synch

Global Synch – Energy



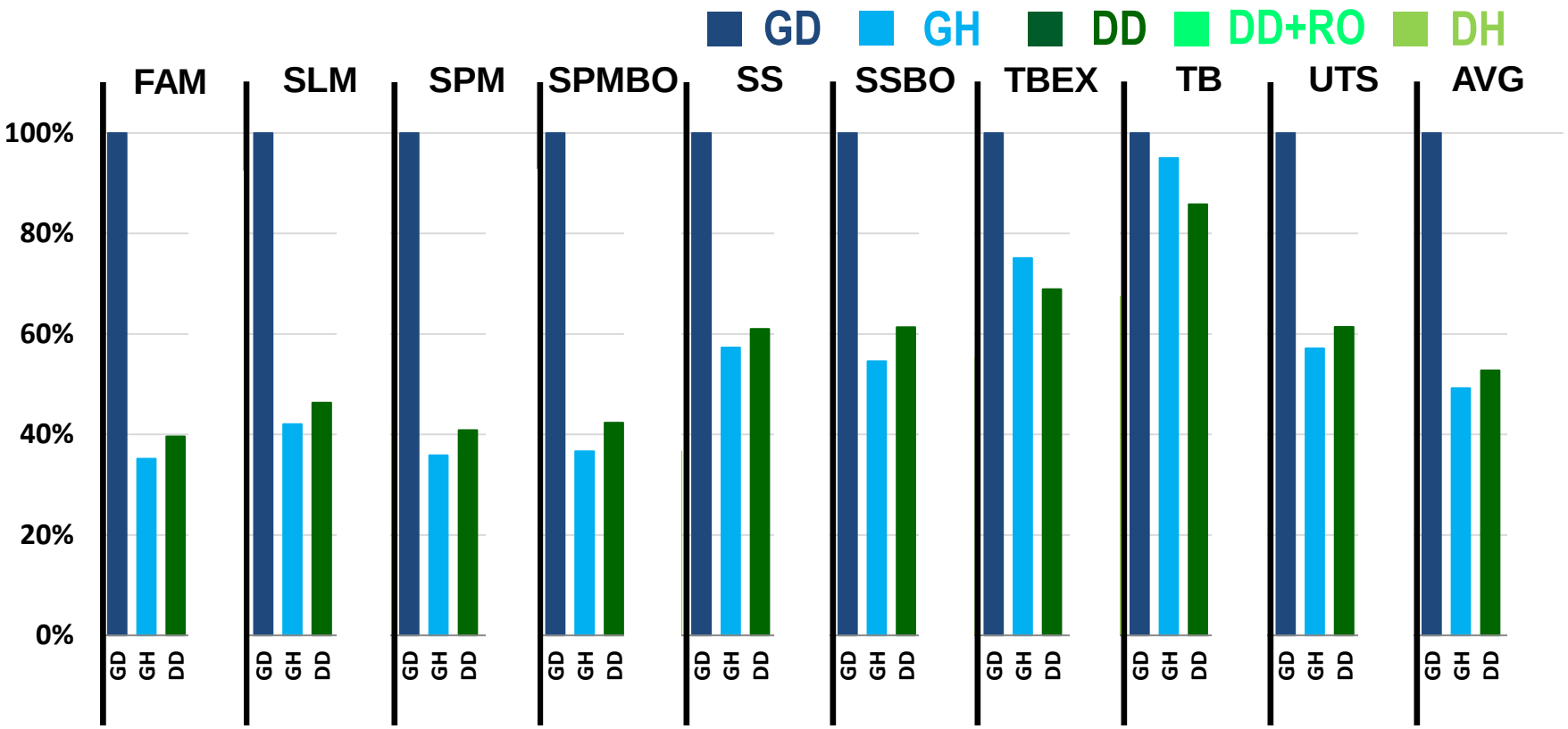
DeNovo has 51% lower energy than GPU with global synch

Local Synch – Execution Time



GPU+HRF is much better than GPU+DRF with local synch [ASPLOS '14]

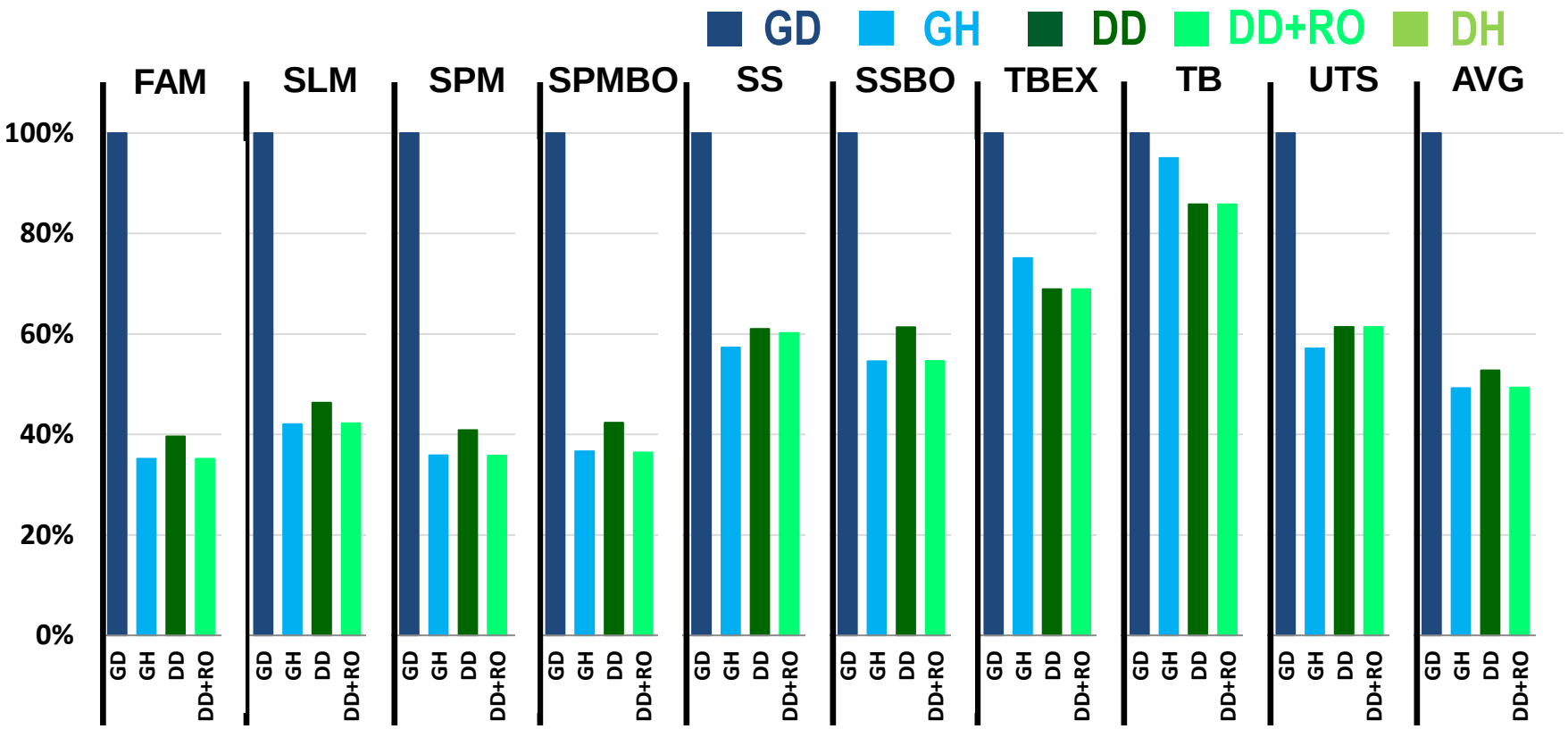
Local Synch – Execution Time



GPU+HRF is much better than GPU+DRF with local synch [ASPLOS '14]

DeNovo+DRF comparable to GPU+HRF, but simpler consistency model

Local Synch – Execution Time

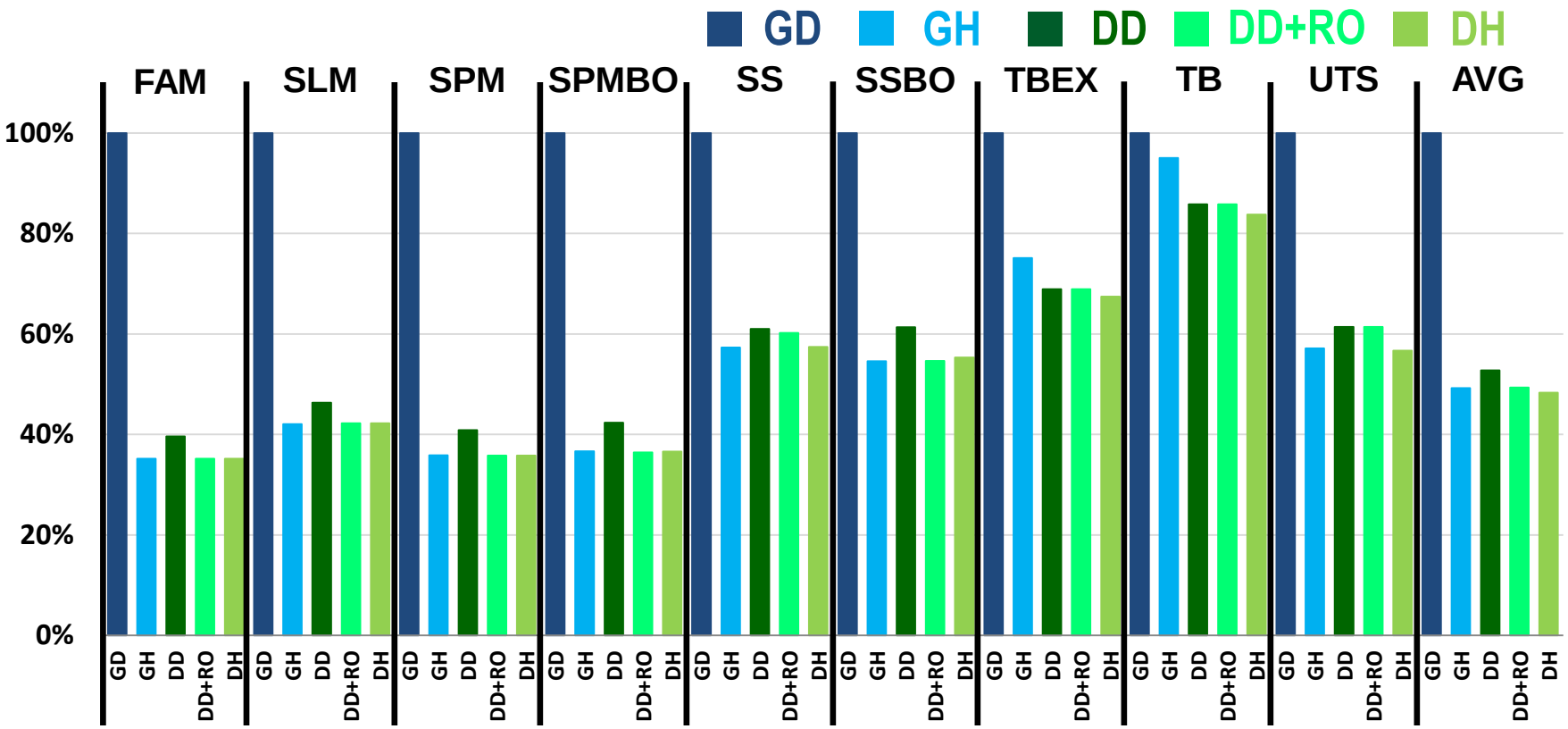


GPU+HRF is much better than GPU+DRF with local synch [ASPLOS '14]

DeNovo+DRF comparable to GPU+HRF, but simpler consistency model

DeNovo-RO+DRF reduces gap by not invalidating read-only data

Local Synch – Execution Time



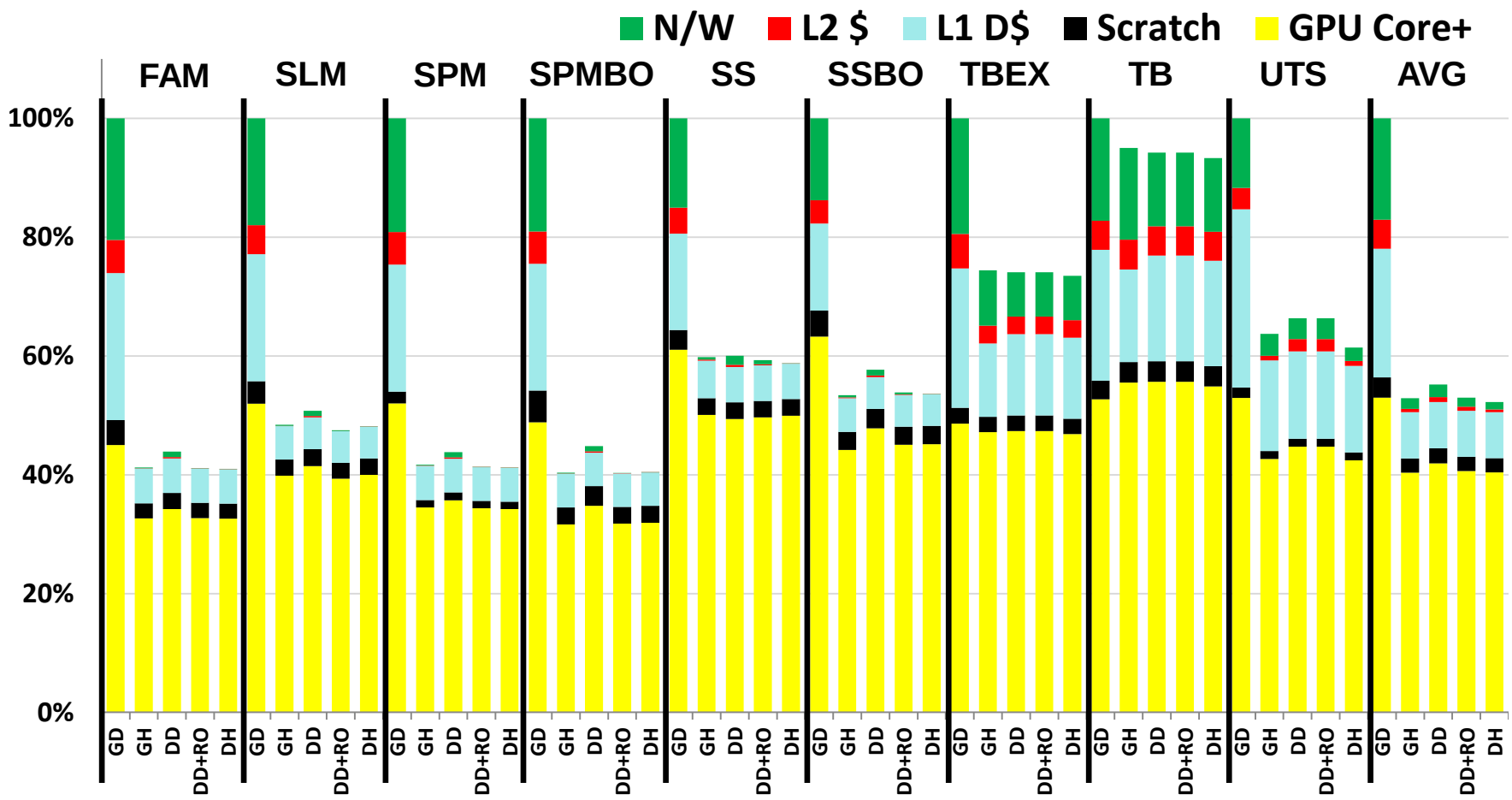
GPU+HRF is much better than GPU+DRF with local synch [ASPLOS '14]

DeNovo+DRF comparable to GPU+HRF, but simpler consistency model

DeNovo-RO+DRF reduces gap by not invalidating read-only data

DeNovo+HRF is best, if consistency complexity acceptable

Local Synch – Energy



Energy trends similar to execution time

Conclusions

- **Emerging heterogeneous apps use fine-grained synch**
  - **GPU coherence + DRF: inefficient, but simple memory model**
  - **GPU coherence + HRF: efficient, but complex memory model**

Do GPU models (HRF) need to be more complex than CPU models (DRF)?

  - **DeNovo + DRF: efficient AND simple memory model**

complex consistency models!