Memory Consistency Models: They are Broken and Why We Should Care

Sarita Adve

University of Illinois at Urbana-Champaign

sadve@Illinois.edu

Ken Kennedy Lecture

Work with numerous colleagues and students over 30 years

This work is currently supported in part by DARPA, NSF, and by the Applications Driving Architecture (ADA) Research center (JUMP center co-sponsored by SRC and DARPA)

My Story

Wisconsin

Rice

Illinois

Our community

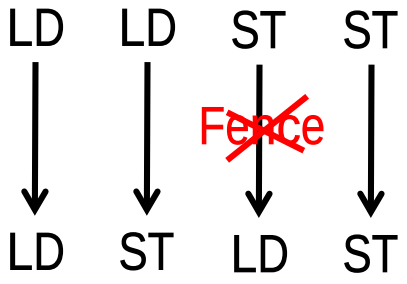
My Story

Wisconsin	Question fundamentals
Rice	Believe in yourself
Illinois	Impact = Change minds. Takes time
Our community	Acknowledge your village. Pay it forward

Wisconsin – Question Fundamentals

[With Mark Hill]

- 1988 to 1989: What is a memory consistency model?
  - Simplest model: sequential consistency (SC) [Lamport79]
    - Memory operations execute one at a time in program order
    - Simple, but inefficient
  - Implementation/performance-centric view
    - Order in which memory operations execute
    - Different vendors w/ different models (orderings)
      - Alpha, Sun, x86, Itanium, IBM, AMD, HP, Cray, …
    - Complex, many ambiguities, …
  - A new memory model virtually everyday



Wisconsin – Question Fundamentals

- 1988 to 1989: What is a memory consistency model?

Memory model = What value can a read return?

Initially X=Y=Flag=0

Thread 1	*Thread 2*
X = 26	if (Flag == 1) {
Y = 90	... = Y ← ~~90~~
...	... = X ← ~~26~~ 0
Flag = 1	...
	}

HW/SW Interface: affects performance, programmability, portability

Wisconsin – Question Fundamentals

- 1990-93: Software-centric view: Data-race-free (DRF) model
  - Sequential consistency for data-race-free programs [Adve, Hill ISCA90]
  - Distinguish data vs. synchronization (race)
    - Data can be optimized ⇒ ↑ performance for DRF programs

Initially X=Y=Flag=0

Thread 1	*Thread 2*
X = 26	if (Flag == 1) {
Y = 90	… = Y
…	… = X
Flag = 1	…
	}

Ack: Jim Goodman, Bart Miller, Rob Netzer, Kourosh Gharachorloo

Wisconsin → Rice

"Two body problem" ⇒
Two body opportunity

Dependence analysis, auto-vectorization, data parallel languages, parallel performance analysis tools, …

Rice – Believe in Yourself

[With Vijay Pai and Partha Ranganathan]

- 1993-99: Performance benefits of relaxed models
  - New out-of-order processors emerging, new speculation techniques
  - No tools to understand performance implications
  - RSIM: Built first publicly available multiprocessor simulator with out-of-order processors [Pai et al. ASPLOS'96, ISCA'97, …]

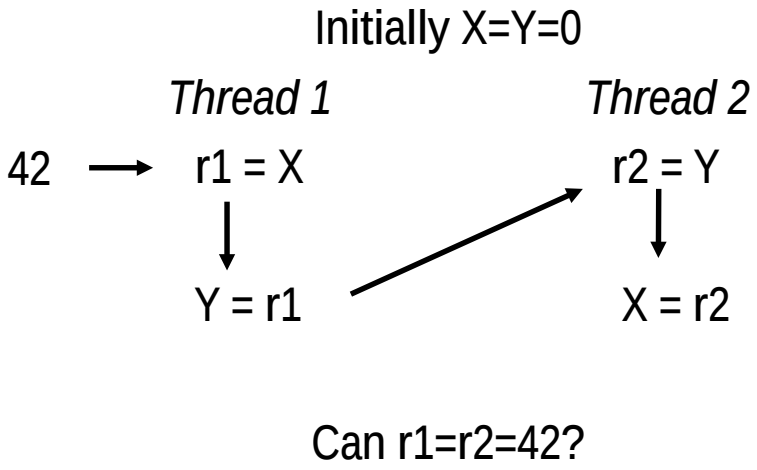
- More confidence in DRF!
  - Called out compiler and PL community
  - Proceedings of IEEE paper caught attention of Bill Pugh

Illinois: Impact = Change Minds. Takes Time.

[with Bill Pugh, Jeremy Manson, Doug Lea, Hans Boehm, et al.]

- 2000-05: Java memory model [Manson, Pugh, Adve POPL'05]
  - DRF model BUT racy programs need semantics

    ⇒ No out-of-thin-air values



Problem: Incredibly hard to formalize a spec that prohibits this result without prohibiting common optimizations

Java memory model = DRF + big mess

Illinois: Impact = Change Minds. Takes Time.

[With Hans Boehm et al.]

- 2005-08: C++ memory model [Boehm, Adve PLDI'08]
  - DRF model BUT need high performance; mismatched hardware
  - Baseline DRF (DRF0) requires synchronization/atomics to be SC
  - Hardware vendors, software developers complained, but no option
  - Compromise: Relaxed atomics (only for experts)

    ⇒ DRF + big mess

Good news: After 20 years, convergence at last!

But: How to debug racy programs, how to avoid out of thin air values, no semantics for relaxed atomics, …

CACM'10: Memory Models: A Case for Rethinking Parallel Languages and Hardware

No Formal Specification for Relaxed Atomics

C++17 "specification" for relaxed atomics

- Races that don't order other accesses
- Implementations should ensure no “out-of-thin-air” values are computed that circularly depend on their own computation

“C++ (relaxed) atomics were the worst idea ever. I just spent days (and days) trying to get something to work. … My example only has 2 addresses and 4 accesses, it shouldn’t be this hard. Can you help?”

- Email from employee at major research lab

Last Decade: Back to Fundamentals

[With Vikram Adve, Byn Choi, Rakesh Komuravelli, Matt Sinclair, Hyojin Sung]

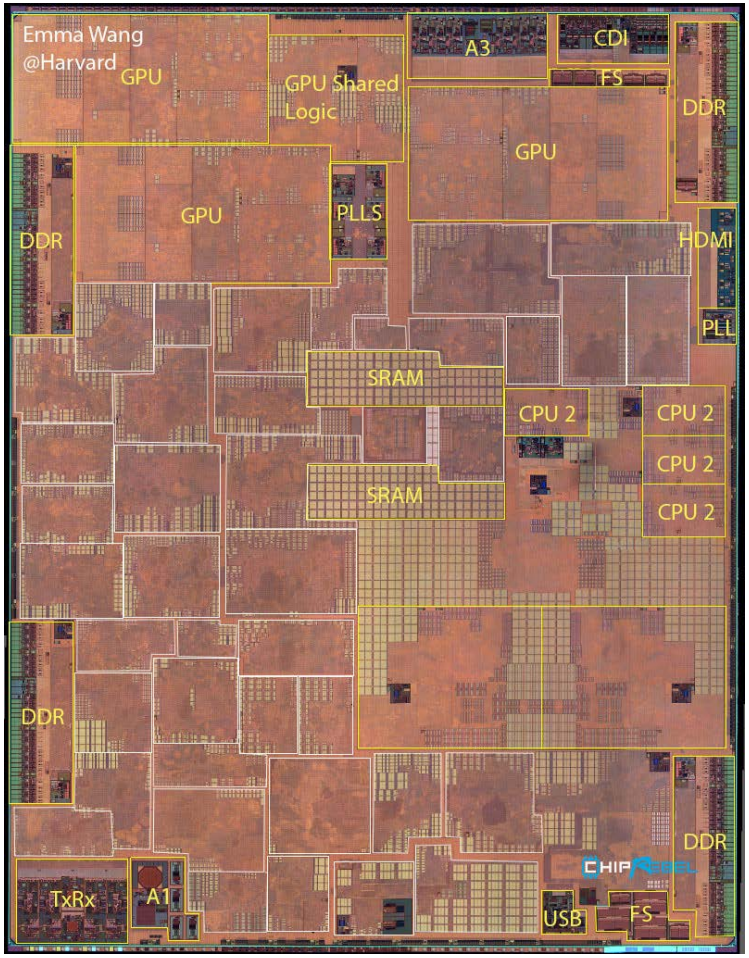
- 2008-14: Software-centric view for coherence: DeNovo protocol
  - More performance-, energy-, and complexity-efficient than MESI
    - Began with DPJ's disciplined parallelism
    - Identified fundamental, minimal coherence mechanisms
    - Loosened s/w constraints, but still minimal, efficient hardware

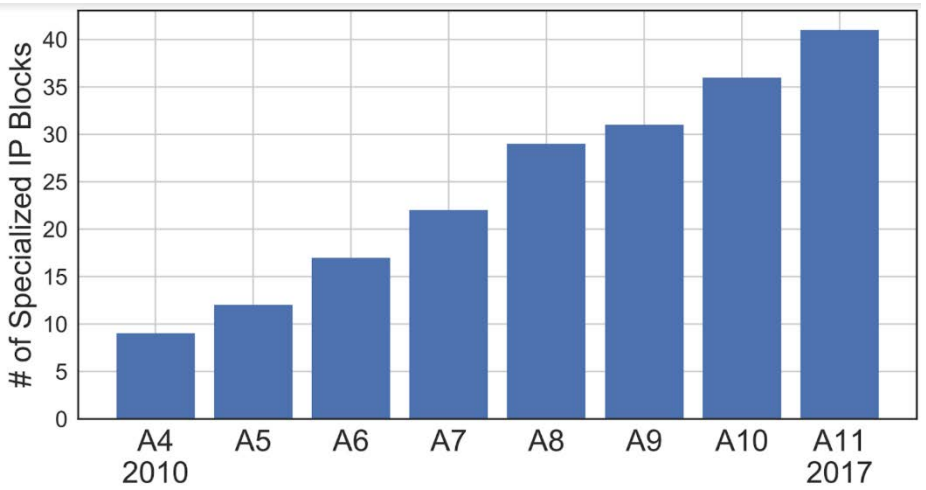
  Ack: Marc Snir, UPCRC

- Meanwhile: the end of Dennard and Moore's laws
  - Architecture enters golden age
  - *Déjà vu for coherence and consistency*

- *Next phase with Matt Sinclair and John Alsop, current group*

The Golden Age of Specialization & Heterogeneity



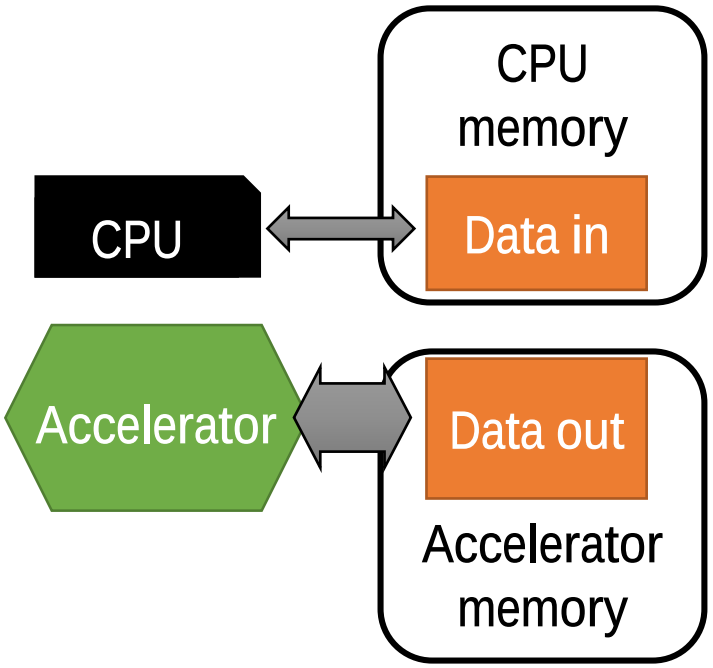


Explosion of accelerators in SoCs

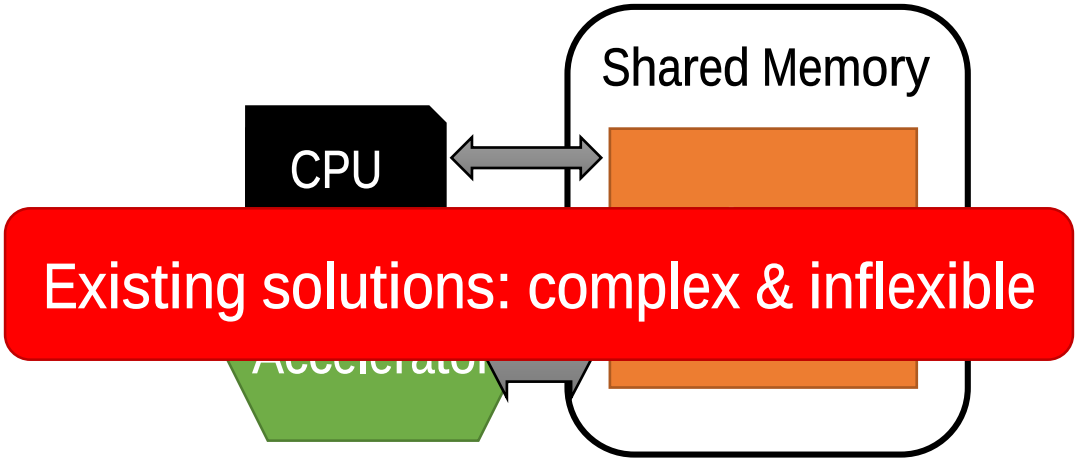
Source: Brooks, Wei group, http://vlsiarch.eecs.harvard.edu/accelerators/die-photo-analysis

Specialization Requires Better Memory Systems

Traditional heterogeneity



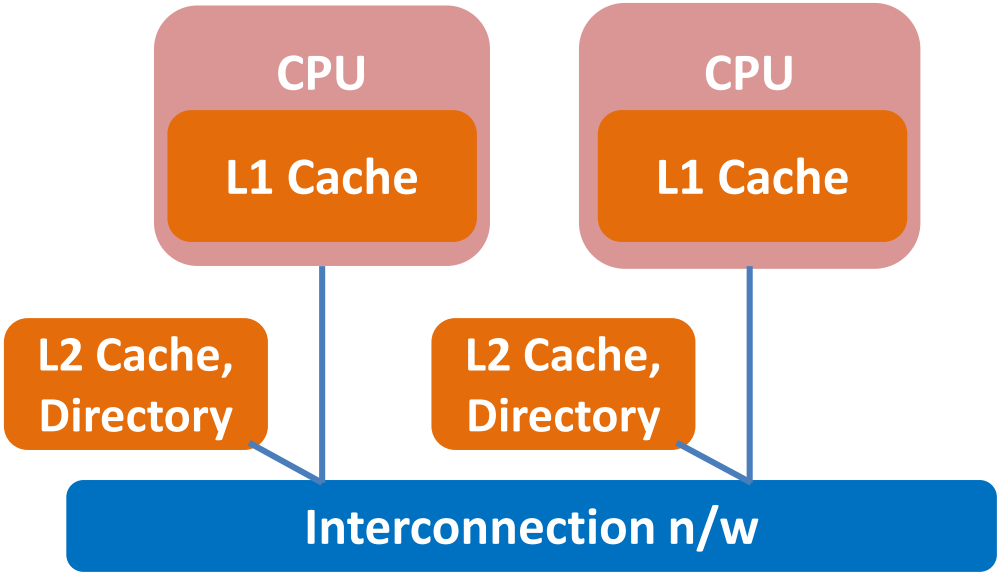
Coherent shared memory



- ✗ Wasteful data movement
- ✗ No fine-grain synch
- ✗ No irregular access patterns

- ✓ Implicit data reuse
- ✓ Fine-grain synchronization
- ✓ Irregular access

CPU Coherence: MSI



- Single writer, multiple reader
  - On write miss, get ownership + invalidate all sharers
  - On read miss, add to sharer list

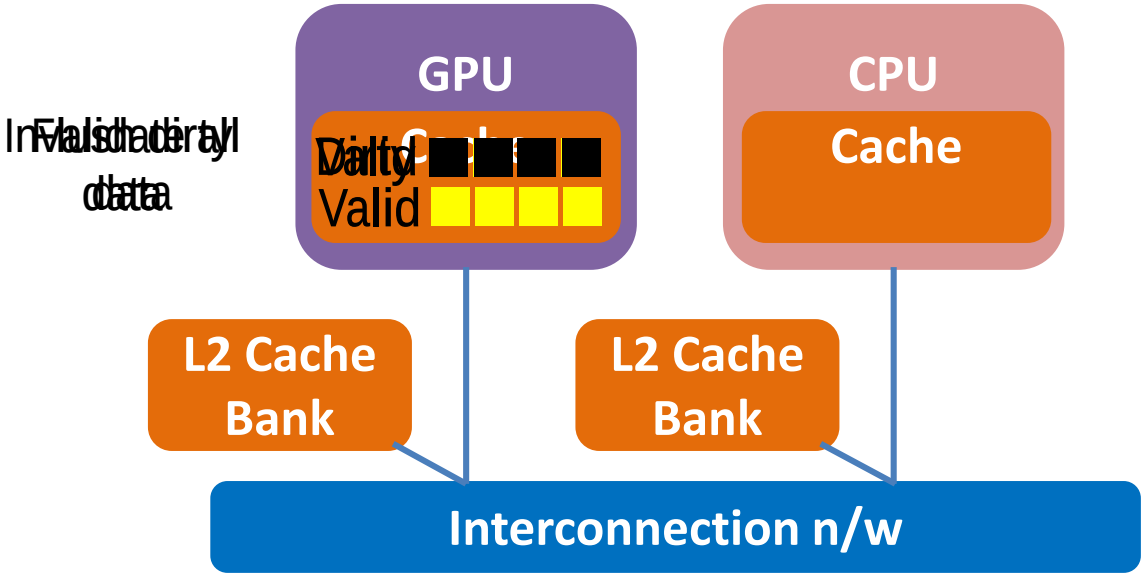
⇒ Directory to store sharer list

⇒ Many transient states

⇒ Excessive traffic, indirection

Complex + inefficient

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)

Simple, but inefficient at synchronization

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

heterogeneous HRF

- With ~~data~~ race-free (~~DRF~~) memory model
  - No ~~data~~ races; synchs must be explicitly distinguished
    heterogeneous and their scopes
  - At all synch points
    global
    - Flush all dirty data: Unnecessary writethroughs
    - Invalidate all data: Can't reuse data across synch points

Global
  - Synchronization accesses must go to last level cache (LLC)
  - No overhead for locally scoped synchs

- But higher programming complexity

Modern GPU Coherence & Consistency

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

[Sinclair et al. Micro'15]

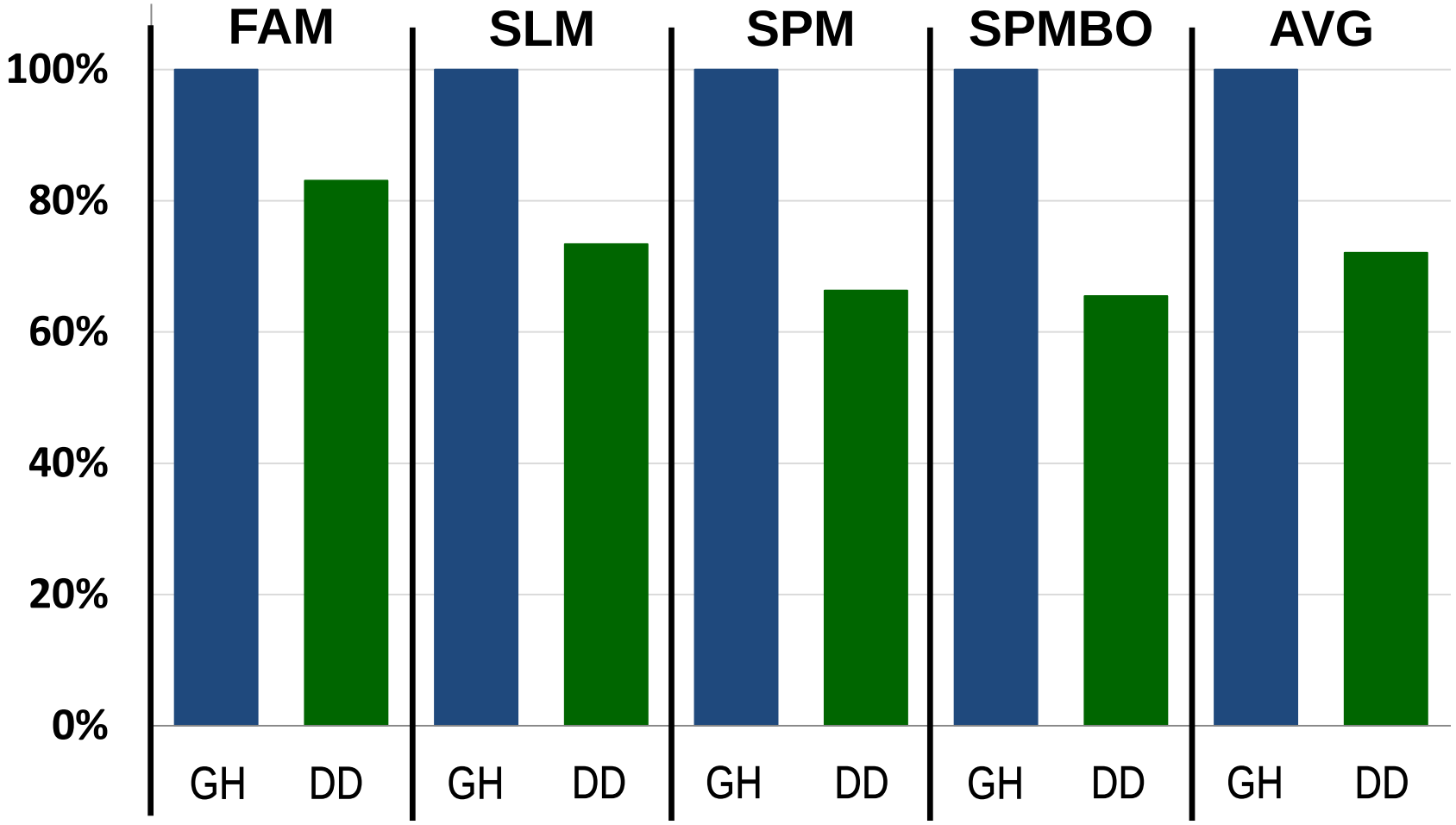
A Classification of Coherence Protocols

- 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

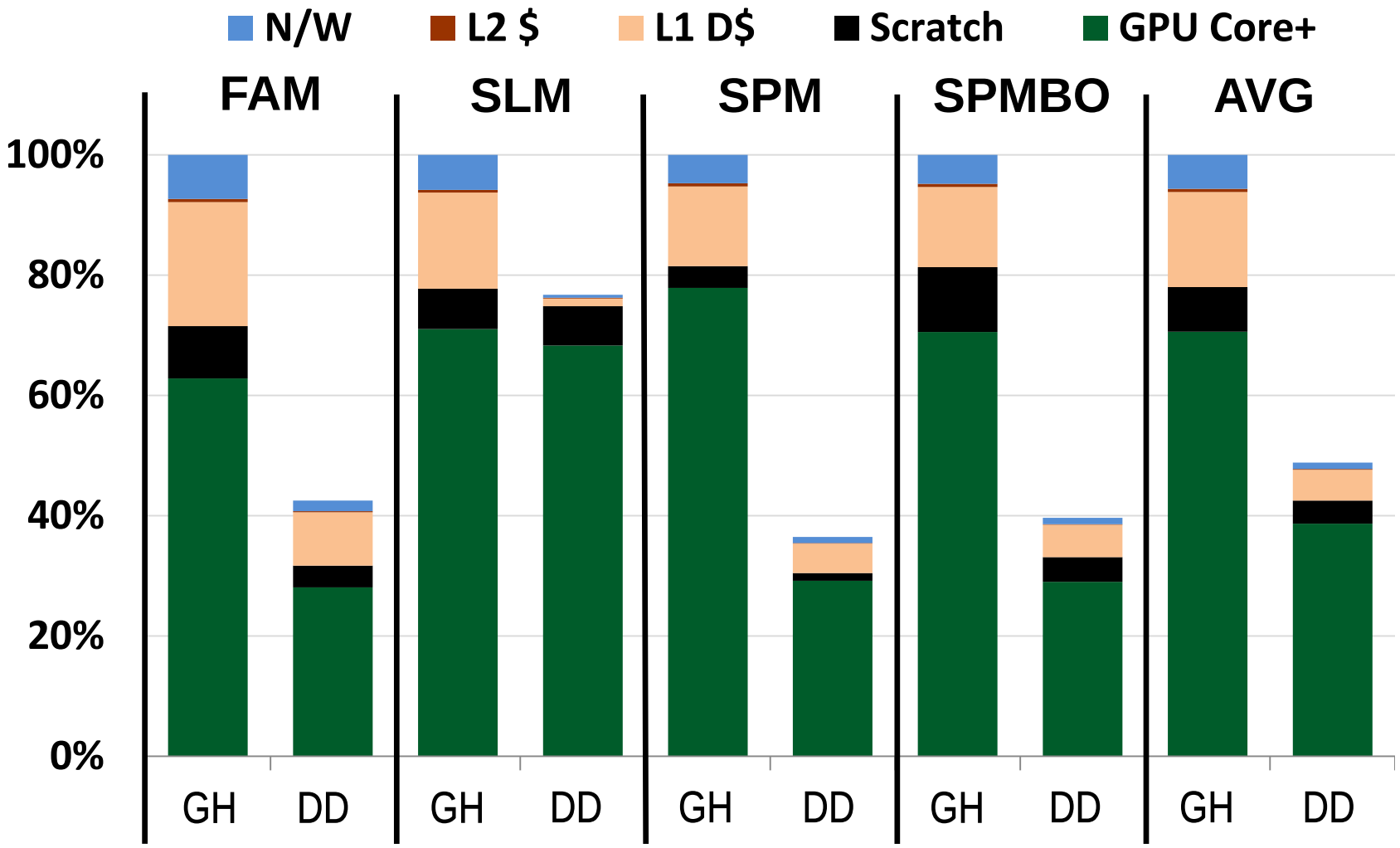
- Reader-initiated invalidations
  - No invalidation or ack traffic, directories, transient states
- Obtaining ownership for written data
  - Reuse owned data across synchs (not flushed at synch points)
  - Complexity, performance, energy

Global Synch – Execution Time



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

Global Synch – Energy



DeNovo shows 51% lower energy than GPU with global synch

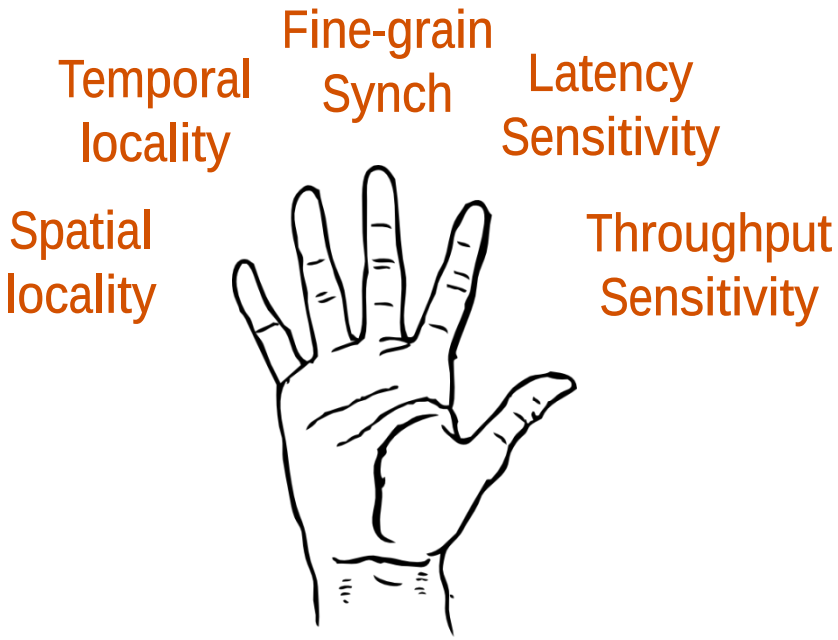
Modern GPU Coherence & Consistency

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

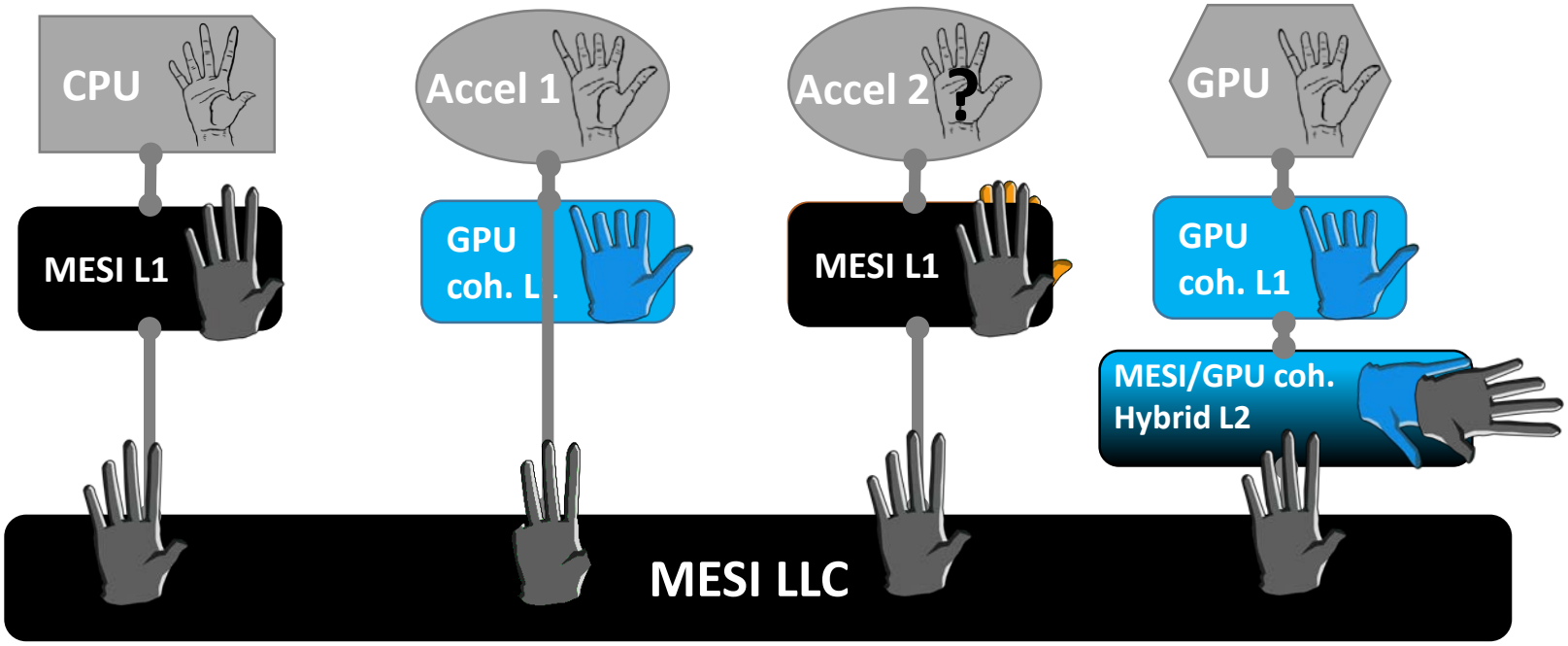
Heterogeneous Devices have Diverse Memory Demands



Typical CPU workloads: fine-grain synch, latency sensitive

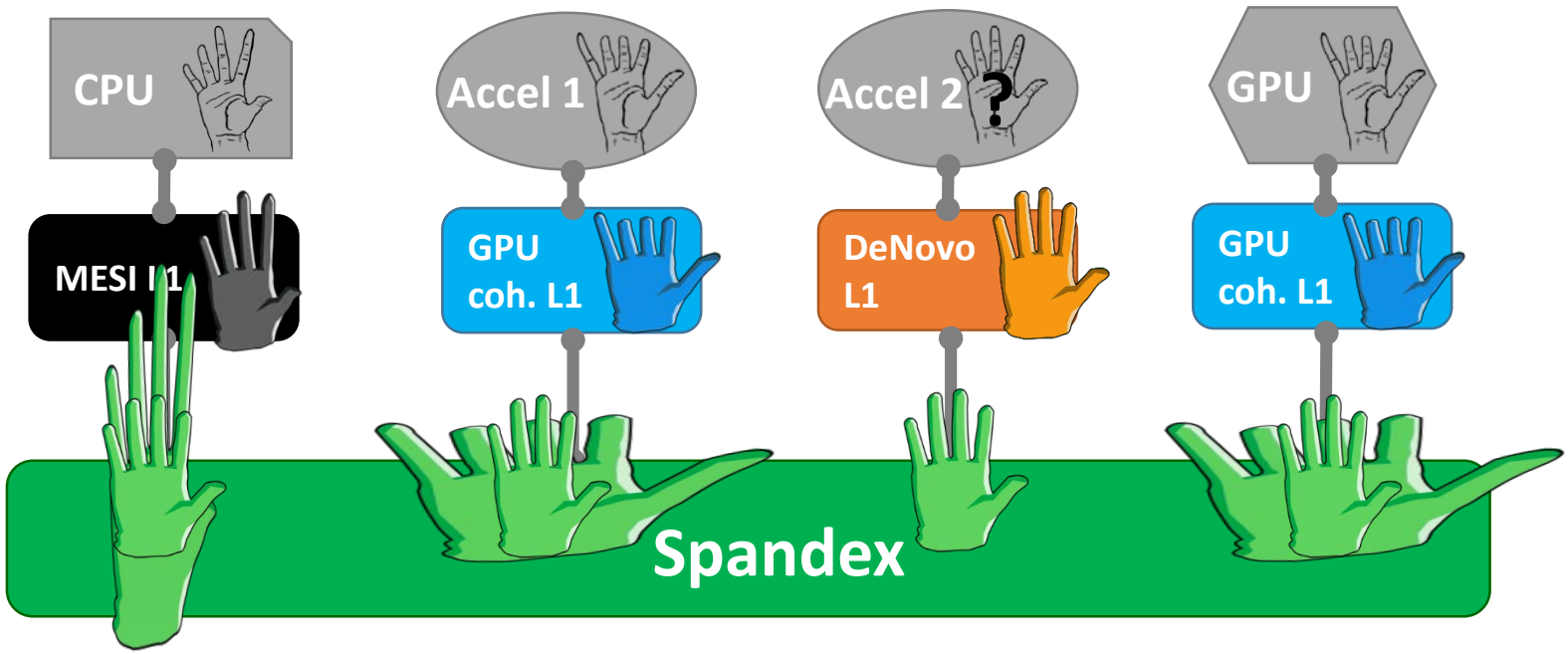
Typical GPU workloads: spatial locality, throughput sensitive

Existing Solutions: Inflexible and Inefficient



Examples: ARM CHI, IBM CAPI, AMD APU

Spandex: Flexible Heterogeneous Coherence Interface



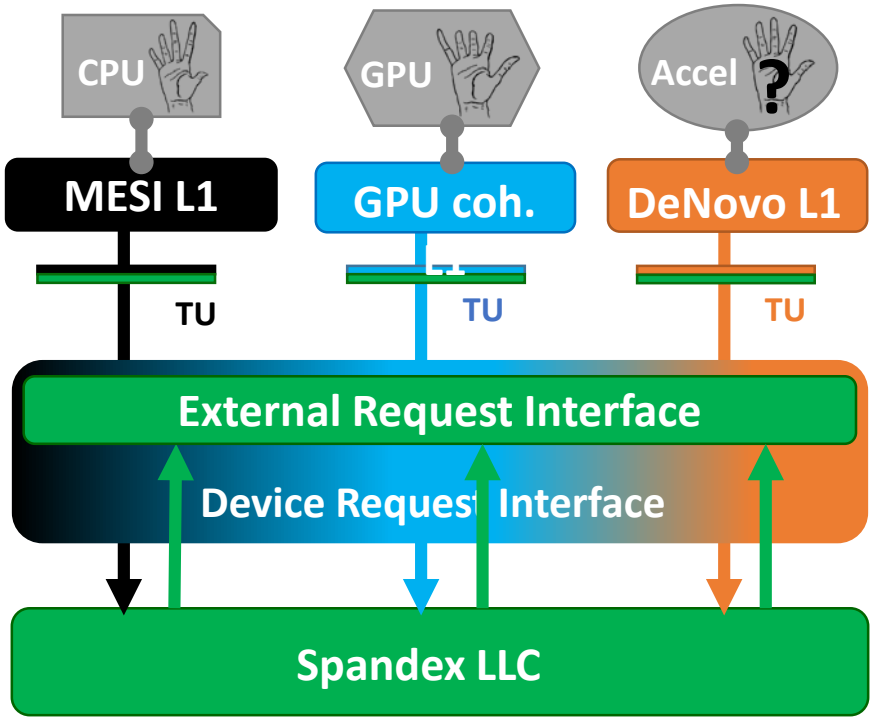
Adapts to exploit individual device's workload attributes

Better performance, lower complexity

⇒ Fits like a glove for each device!

[Alsop et al. ISCA'18]

Spandex Key Components

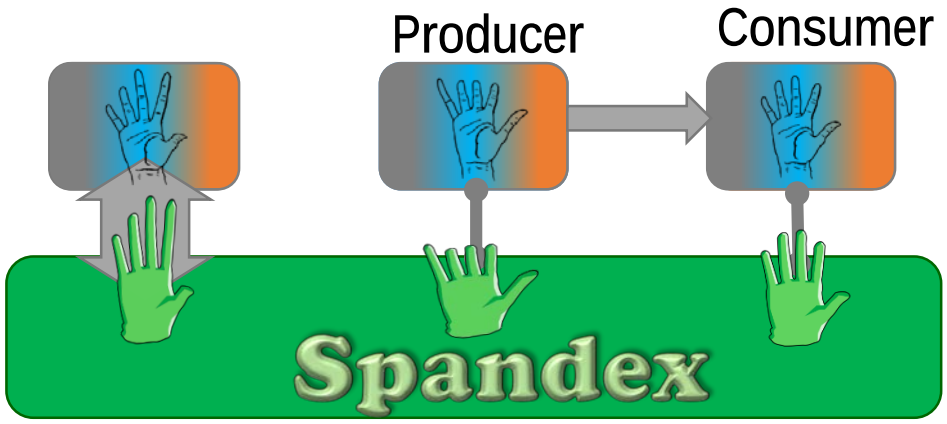


Key Components

- Flexible device request interface
- DeNovo-based LLC
- External request interface

Device may need translation unit

Spandex So Far and Next Steps



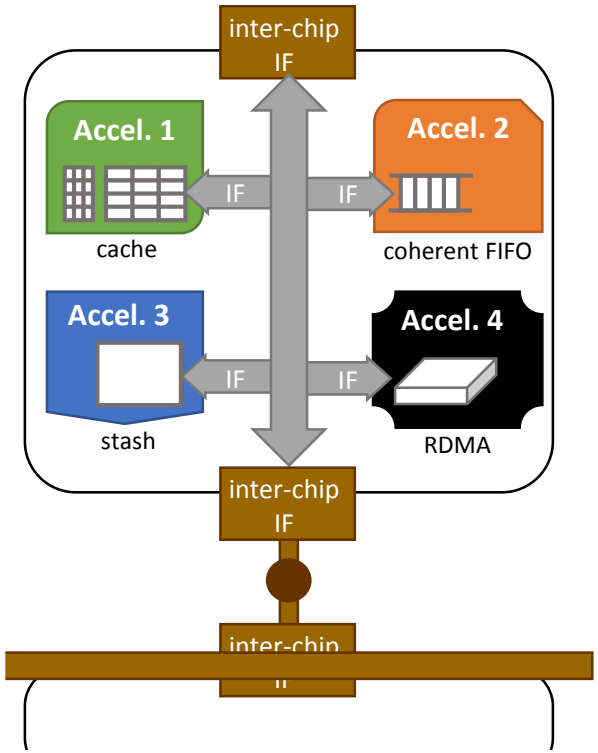
⇒ Simple, Flexible, Efficient

Next steps: Dynamic coherence specialization

Exploit SW or HW hints about data access patterns

- Dynamic Spandex request selection
- Producer-consumer forwarding
- Extended granularity flexibility

Accelerator Communication Architecture



Specialized coherence a la Spandex

Handle specialized memories in global address space
- Scratchpad, FIFOs, …, compute-in-memory, HBM,
- Stash: globally addressable scratchpads [ISCA'15]

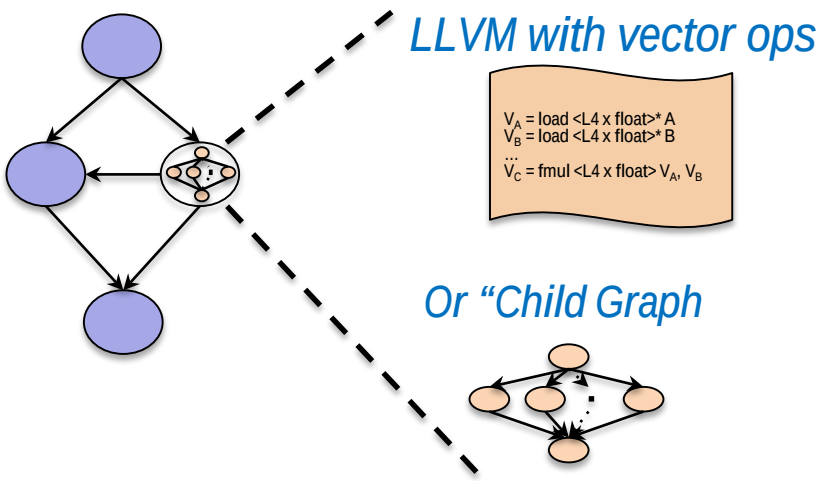
Relaxed atomics
- DRFrlx [Sinclair et al. ISCA'17]
  - SC-centric semantics for good code patterns
- How to formalize other patterns?

Handle approximations & solution quality, security

Hardware-Software Interface

Heterogeneous Parallel Virtural Machine (HPVM) [PPoPP'18]

– Virtual ISA, compiler IR (LLVM for heretogeneous systems)



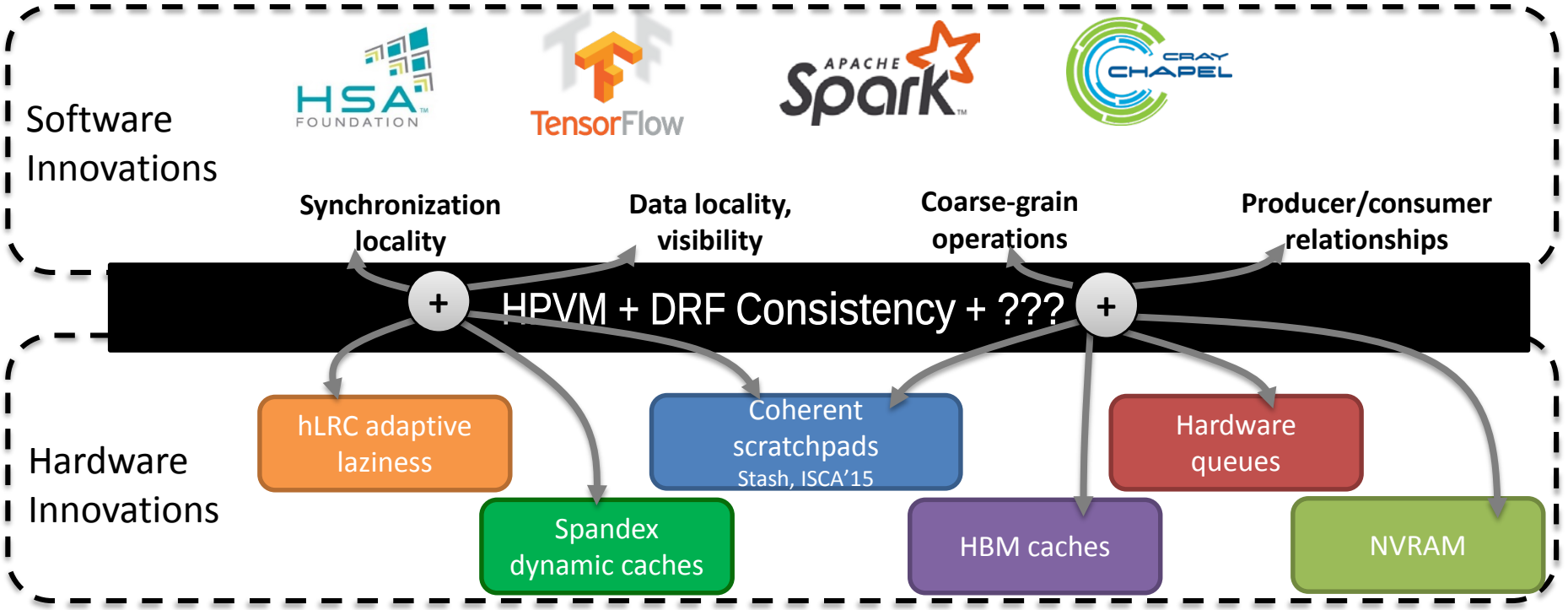
Targets: CPUs, vector extensions, GPUs, FPGAs, domain specific accelerators

Model: Hierarchical dataflow graph with side effects

Runtime maps to accelerators

Another talk!

Looking Forward…



Our Community: Paying it Forward

It takes a village to make a successful researcher

Paying it forward …

Our Community: Paying it Forward

SIGARCH EC

Joel Emer

Babak Falsafi,

Natalie Enright Jerger,

Scott Mahlke,

Partha Ranganathan,

Karin Strauss,

David Wood

Natalie Enright Jerger,

Kim Hazelwood,

Margaret Martonosi,

Kathryn McKinley

Highlight: Diversity and Inclusion

A community effort to emulate

Our Community: Paying it Forward

Sl
Jo
Ba
Na
Sc
Pa
Ka
Da

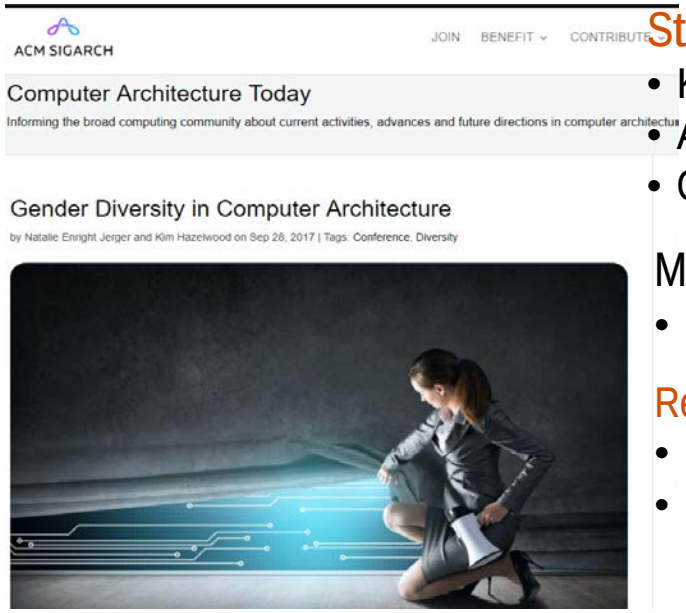
Janie Irwin

Natalie Enright Jerger,
Kim Hazelwood,
Margaret Martonosi,
Kathryn McKinley

Highlight: Diversity and Inclusion

A community effort to emulate

Key Events Last Year in Architecture Community

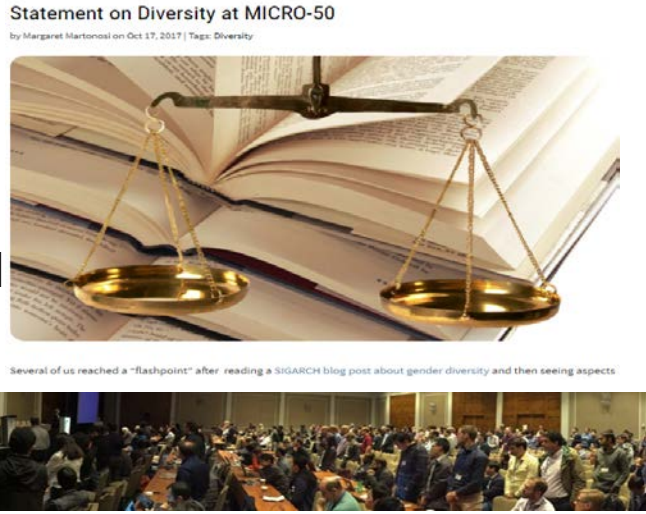


Study shows poor gender ratios

- Keynotes, PC chairs, Awards
- All conferences must improve
- One stands out

Micro50: Legends of Micro panel

- All white, all male



Reading of Diversity Statement

- Call to action
- Clear public support for change

Personal accounts of harassment

SIGARCH works for diversity

But study is wakeup call

Diversity in conference governance

- Institution, academic lineage, ...

Key Events Last Year in Architecture Community

SIGARCH CARES to Report on Discrimination and Harassment

by Sarita Adve, SIGARCH Chair on Mar 1, 2018 | Tags: ACM SIGARCH, Discrimination, Diversity, Harassment

SIGARCH CARES:
To help report harassment
Chairs: Martonosi, McKinley

SIGMICRO and SIGARCH Join Hands on CARES

by Sarita Adve, Michael Gschwind, Margaret Martonosi, Kathryn McKinley on Mar 24, 2018 | Tags: Discrimination, Diversity, Harassment

SIGMICRO joins CARES

Welcome to the Women in Computer Architecture (WICARCH) community

by Natalie Enright Jerger on May 7, 2018 | Tags: Diversity

WICArch is SIGARCH subcommittee
Web portal w/ directory, profiles
Slack mentoring channel
Graduating women brochure
Strategizes diversity efforts
Chair: Enright Jerger

Get data
Raise awareness, fix problems
CARES, WiCarch, Bias busting worksop, Conference mentoring, ...

Thank You!

Thank You!

Wisconsin	Question fundamentals
Rice	Believe in yourself
Illinois	Impact = Change minds. Takes time
Our community	Acknowledge your village. Pay it forward