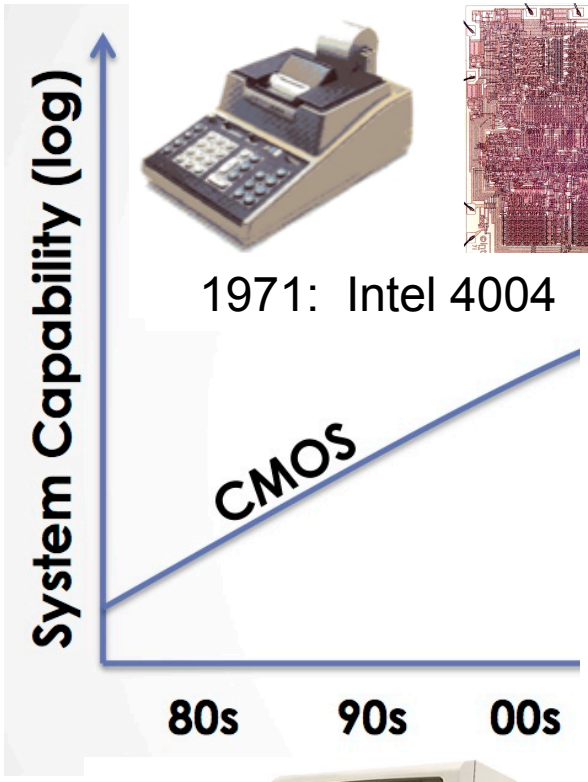
Efficient and Easily Programmable Accelerator Architectures

Tor Aamodt
University of British Columbia

PPL Retreat, 31 May 2013

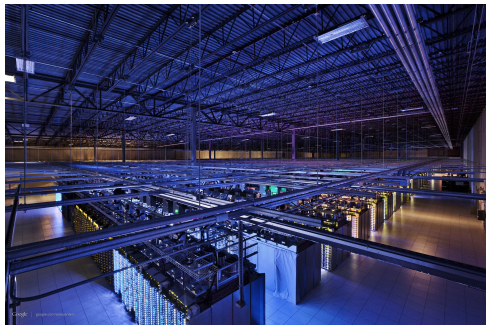
a place of mind
THE UNIVERSITY OF BRITISH COLUMBIA

Decreasing cost per unit computation



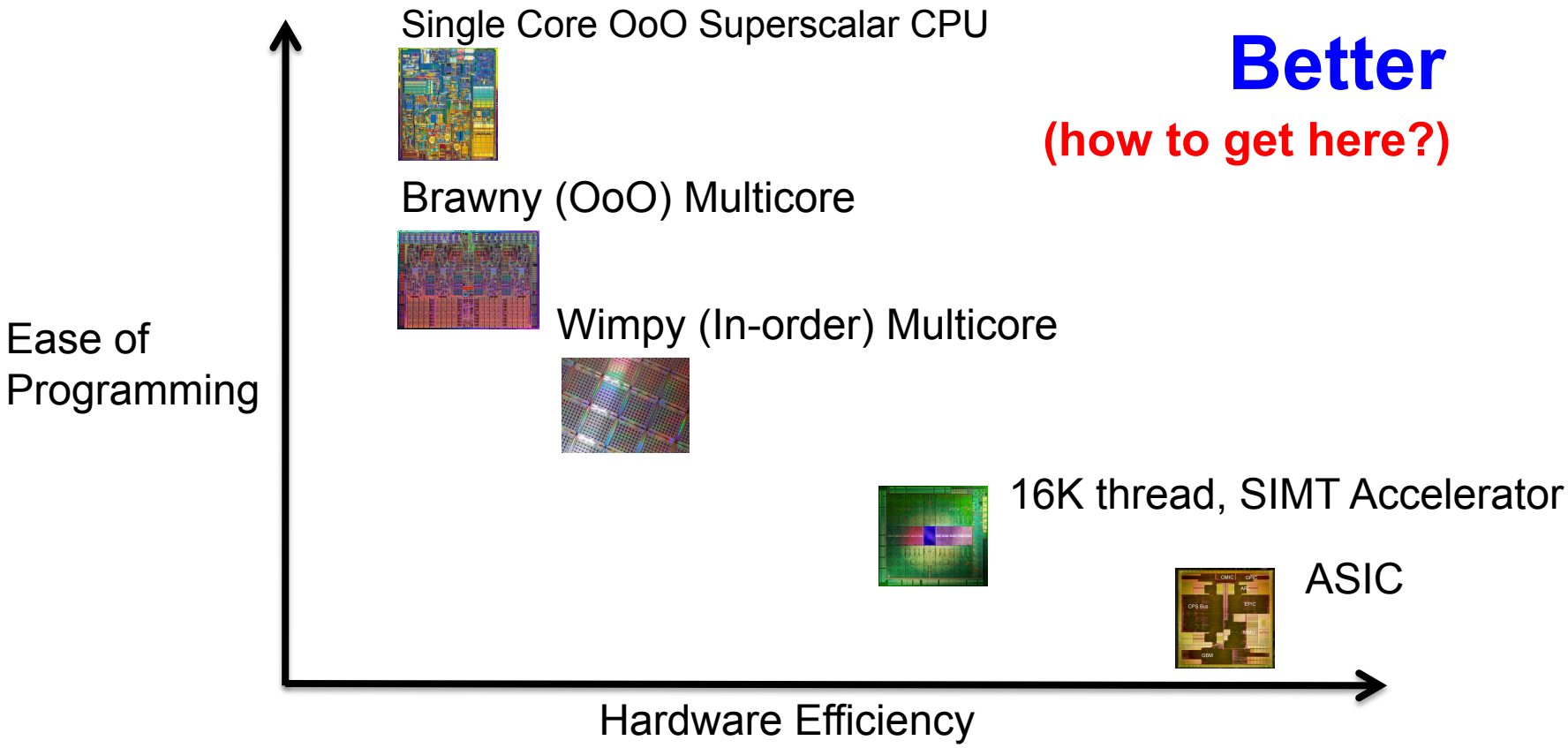
1971: Intel 4004

2007: iPhone

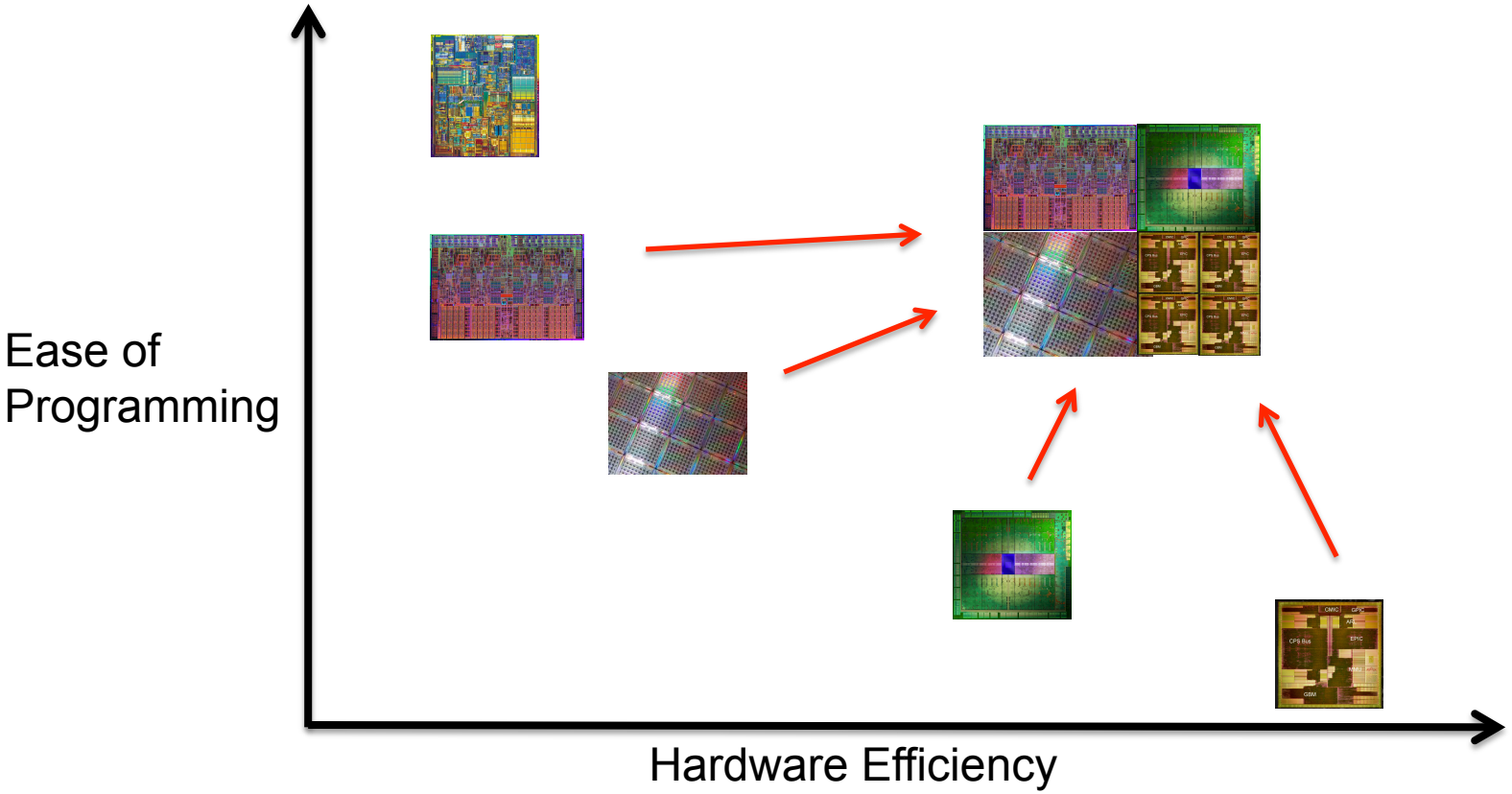


2012: Google Datacenter

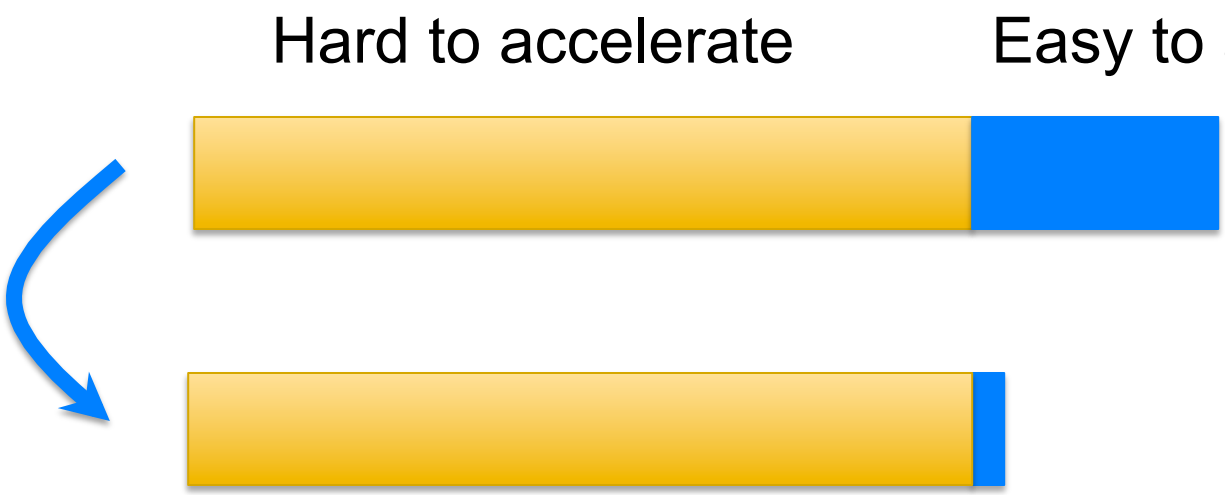
1981: IBM 5150


Single Core OoO Superscalar CPU
Better
(how to get here?)
Brawny (OoO) Multicore
Wimpy (In-order) Multicore
16K thread, SIMT Accelerator
ASIC
Ease of Programming
Hardware Efficiency

Heterogeneity helps…



Review: Amdahl's Law

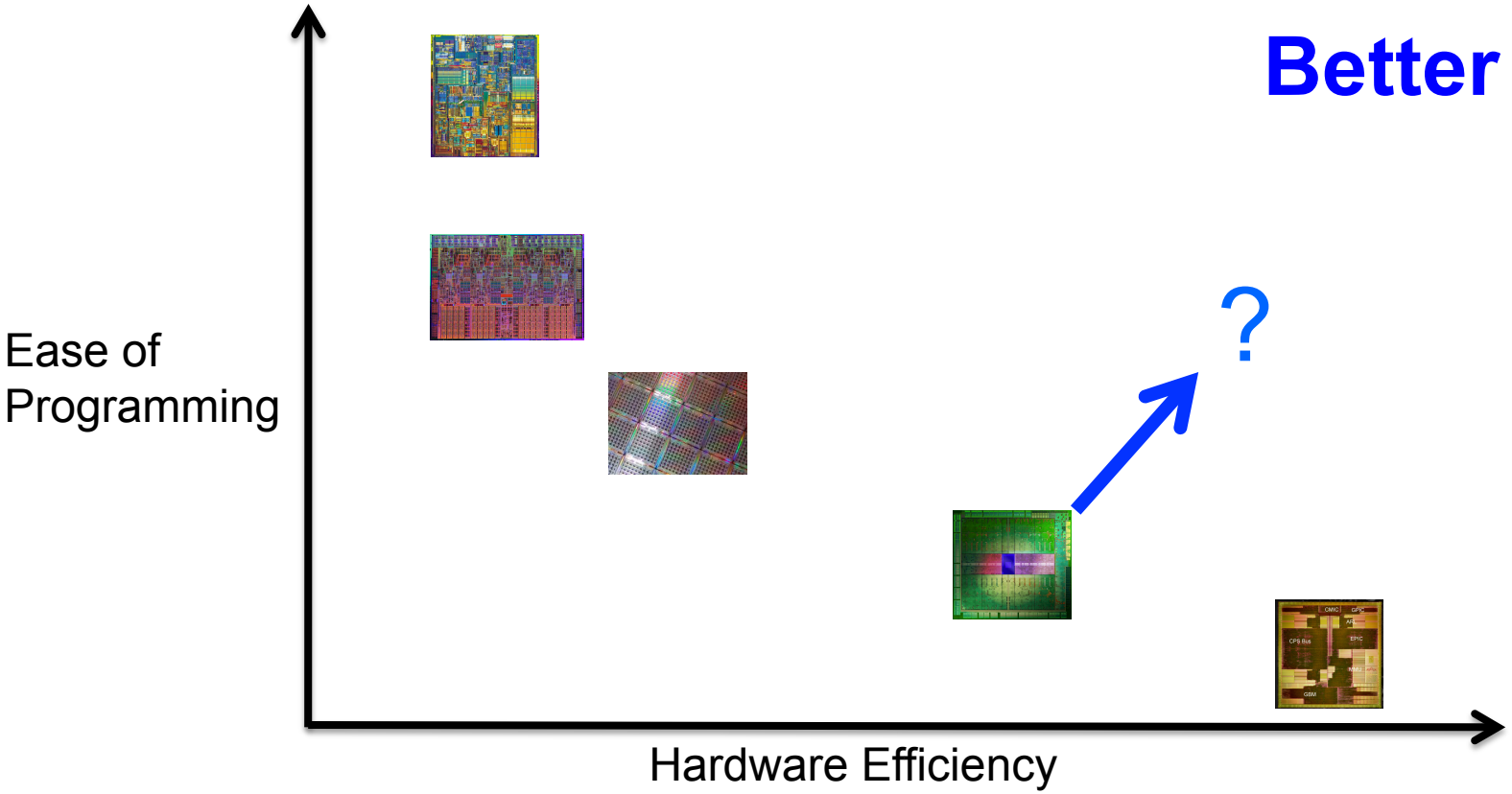


$$\text{Improvement}_{\text{overall}} = \frac{1}{\text{Fraction}_{\text{hard}} + \frac{1 - \text{Fraction}_{\text{hard}}}{\text{Improvement}_{\text{easy}}}}$$

What defines division between hard and easy?

$Fraction_{hard}$ = f(problem, prog. model, SW budget)

Goal:


Better
?
Ease of Programming
Hardware Efficiency

Increase Accelerator Efficiency (x-axis)

- Control Flow
- Memory

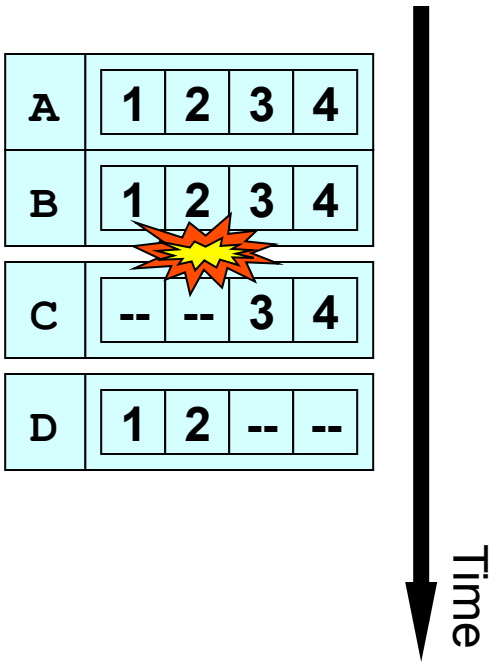
Improve Accelerator Programmability (y-axis)

- Easier coding
- Fewer bugs
- Easier debugging

SIMT Execution (MIMD on SIMD)

(Levinthal SIGGRAPH'84)

```
foo[] = {4,8,12,16};
A: n = foo[tid.x];
B: if (n > 10)
C:     …;
   else
D:     …;
E: …
```

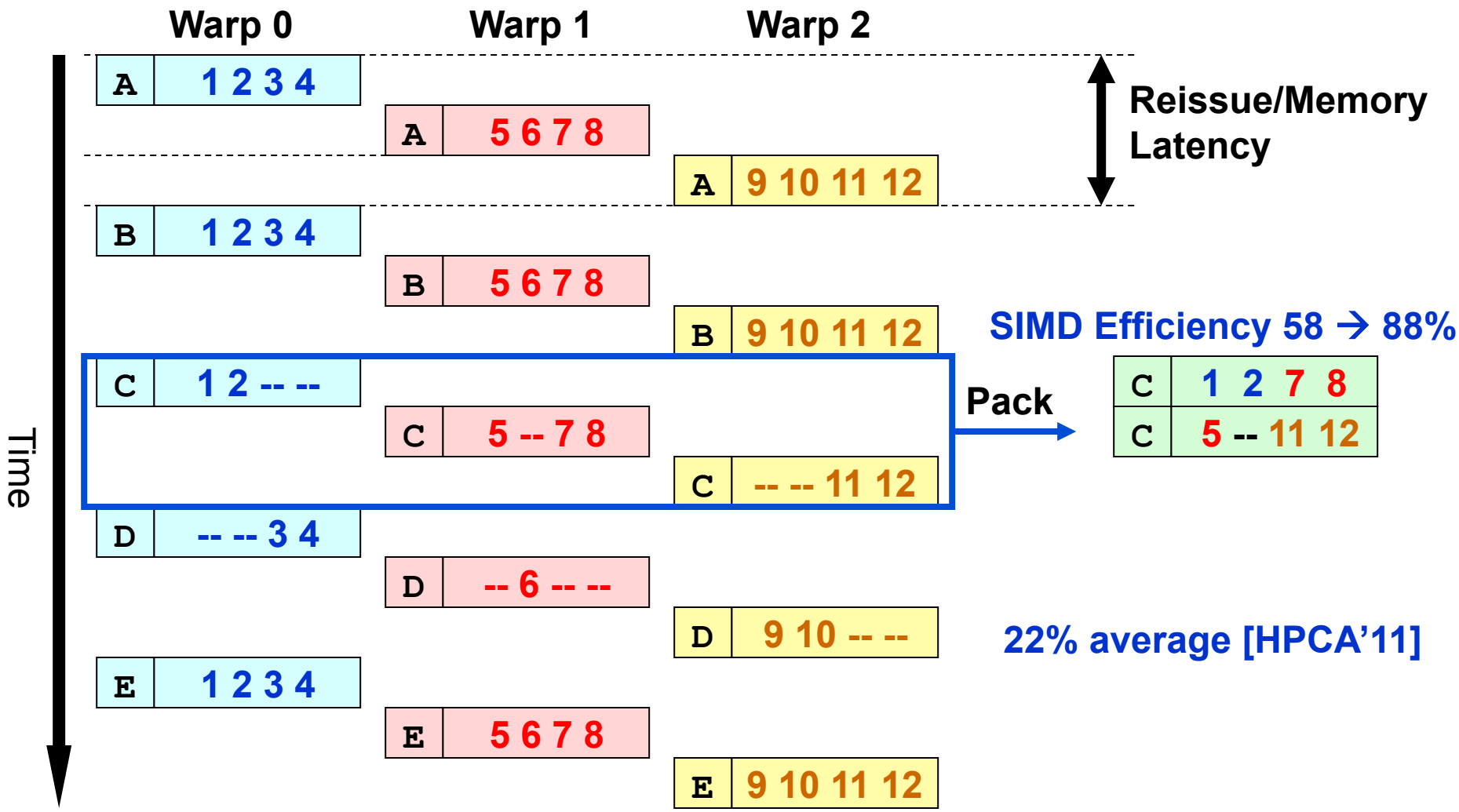


PC	Active Mask
E	1111
D	1100
C	0011

Branch Divergence

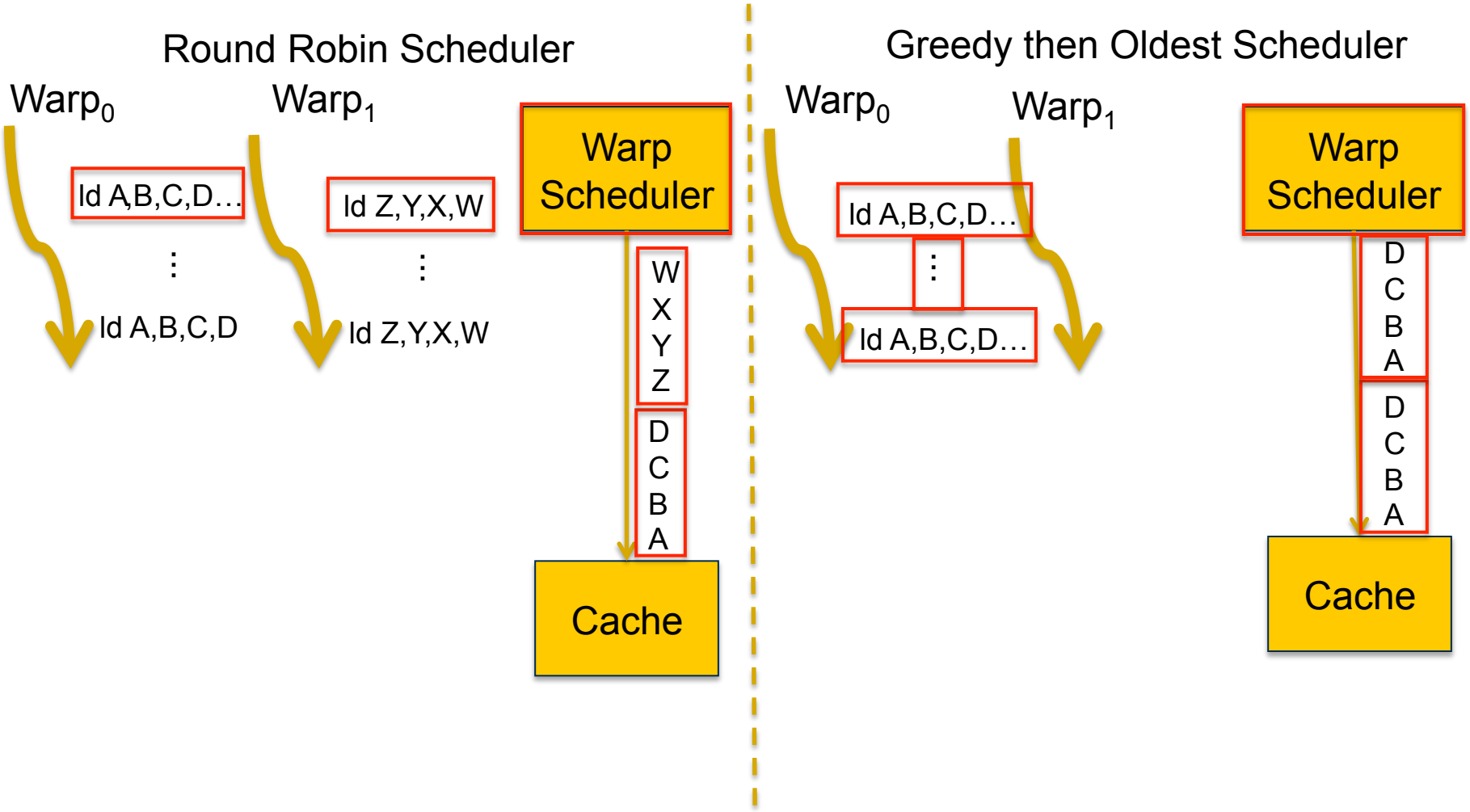
Dynamic Warp Formation

(Fung: MICRO 2007, HPCA 2011)

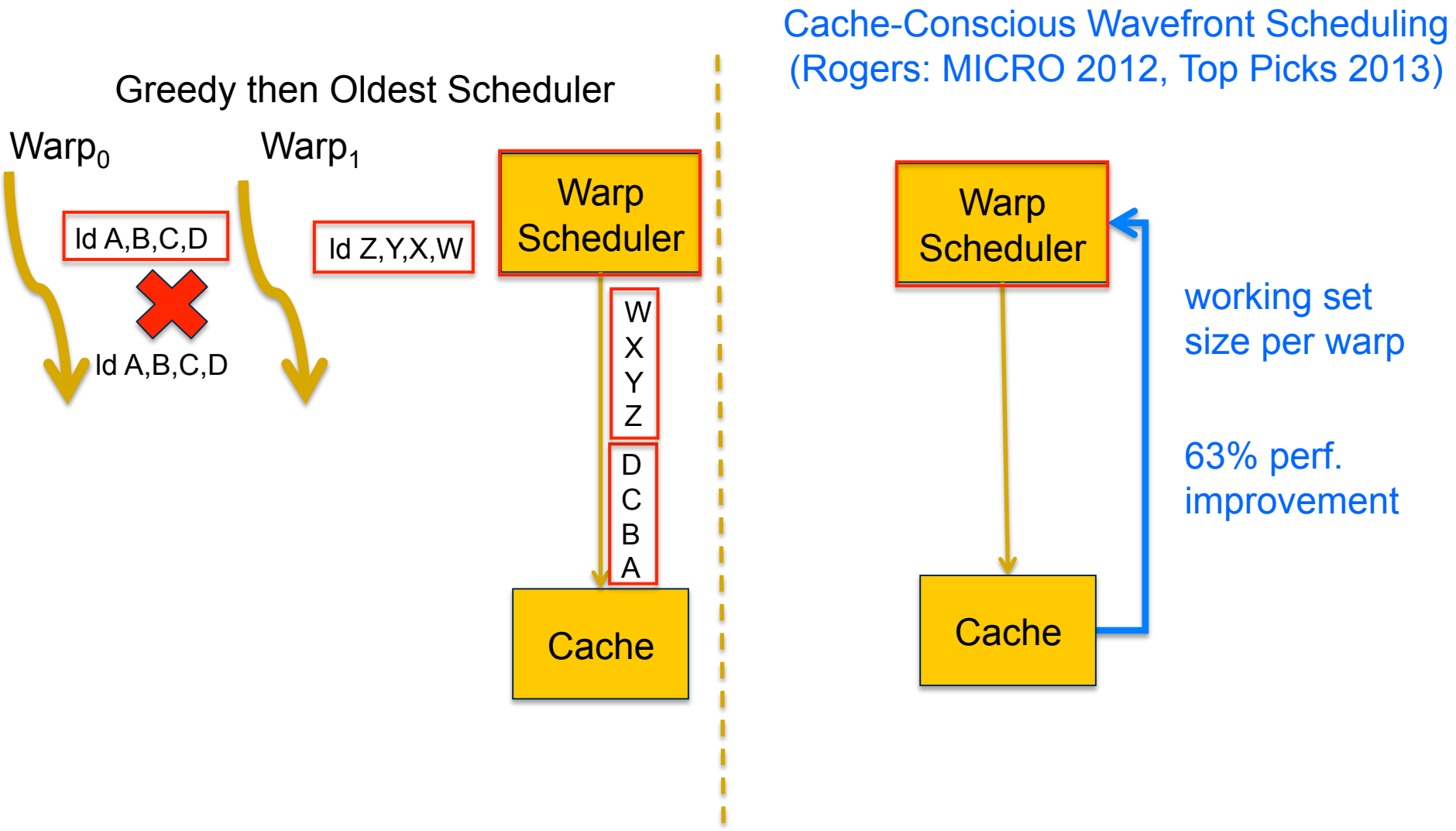


Memory

Scheduler affects access pattern



Use scheduler to *shape* access pattern



Easier coding

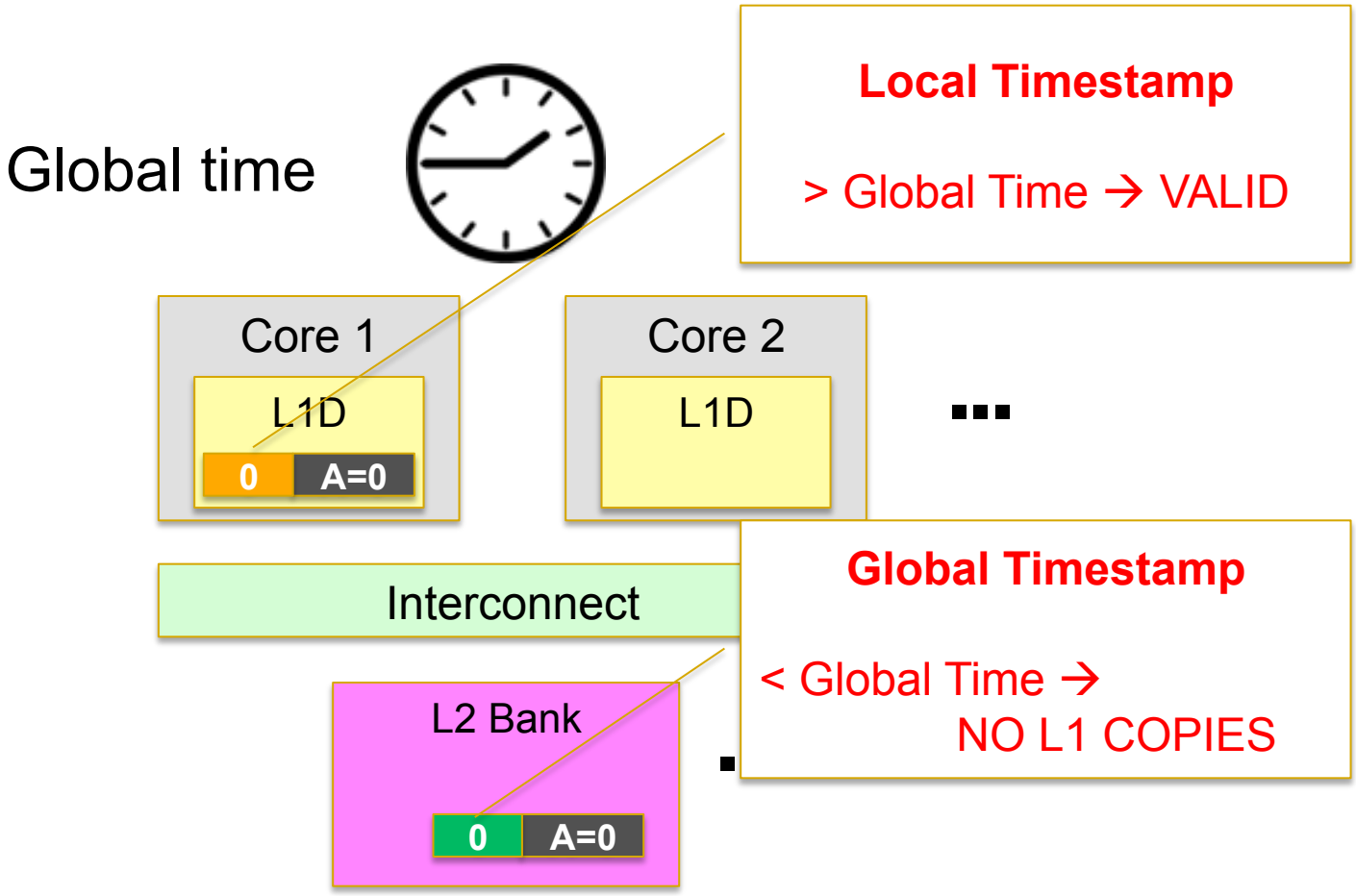
Accelerator Coherence Challenges

- Challenges of introducing coherence messages on a GPU
  1. Traffic: transferring messages
  2. Storage: tracking message
  3. Complexity: managing races between messages

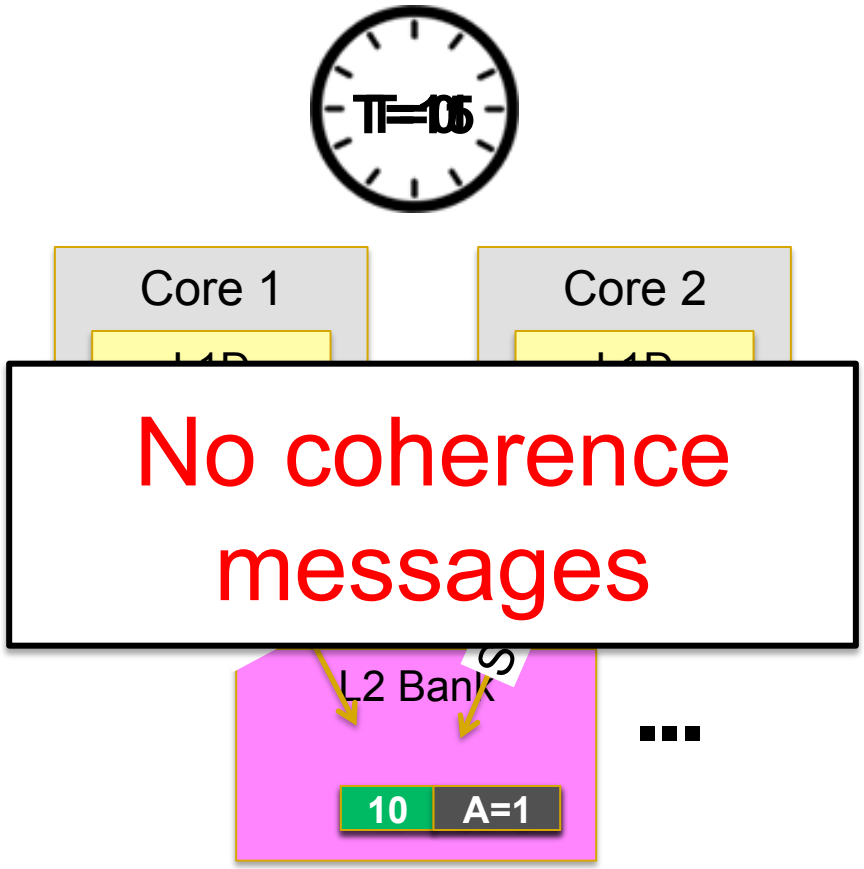
- GPU cache coherence without coherence messages?
  - YES – using global time

Temporal Coherence (Singh: HPCA 2013)

Related: Library Cache Coherence



Temporal Coherence Example



Complexity

Non-Coherent L1

	Load	L1 WThru	L1 Atomic	L1 Replacement	Data	Data Done	WBorAtomic	WBorAtomic Done
I	o i pr+ a k /I S	i pw+ ds k /I1	i pw+ ds a k /I1					
S	h k	i pw+ ds f k /I1	i pw+ ds a f k /I1	f /I				
I_S		pw+ ds f k /I1	pw+ ds a f k /I1	z		pr- u h s o /S		
I I	pr+ a k	pw+ ds k	pw+ ds a k	z	pr- h o	pr- h s o /I	pw- h o	pw- h s o /I

Non-Coherent L2

	L1 GETS	WB Data	L2 Atomic	L2 Replacement	L2 Replacement clean	Mem Data
NP	q lpB i s a j /ISS	q lpB d i x as j /IM	q lpB d i x as j /IMA			
SS	lpR ds set j	f lpW d de mr set j	f lpW d ds a mr set j	f lpE c r /NP	f lpE r /NP	
ISS	lpR s j	z	z	z	z	m c s o /SS
IM	z	z	z	z	z	m mt ee s o /SS
IMA	z	z	z	z	z	m mt e a s o /SS

TC-Weak L1

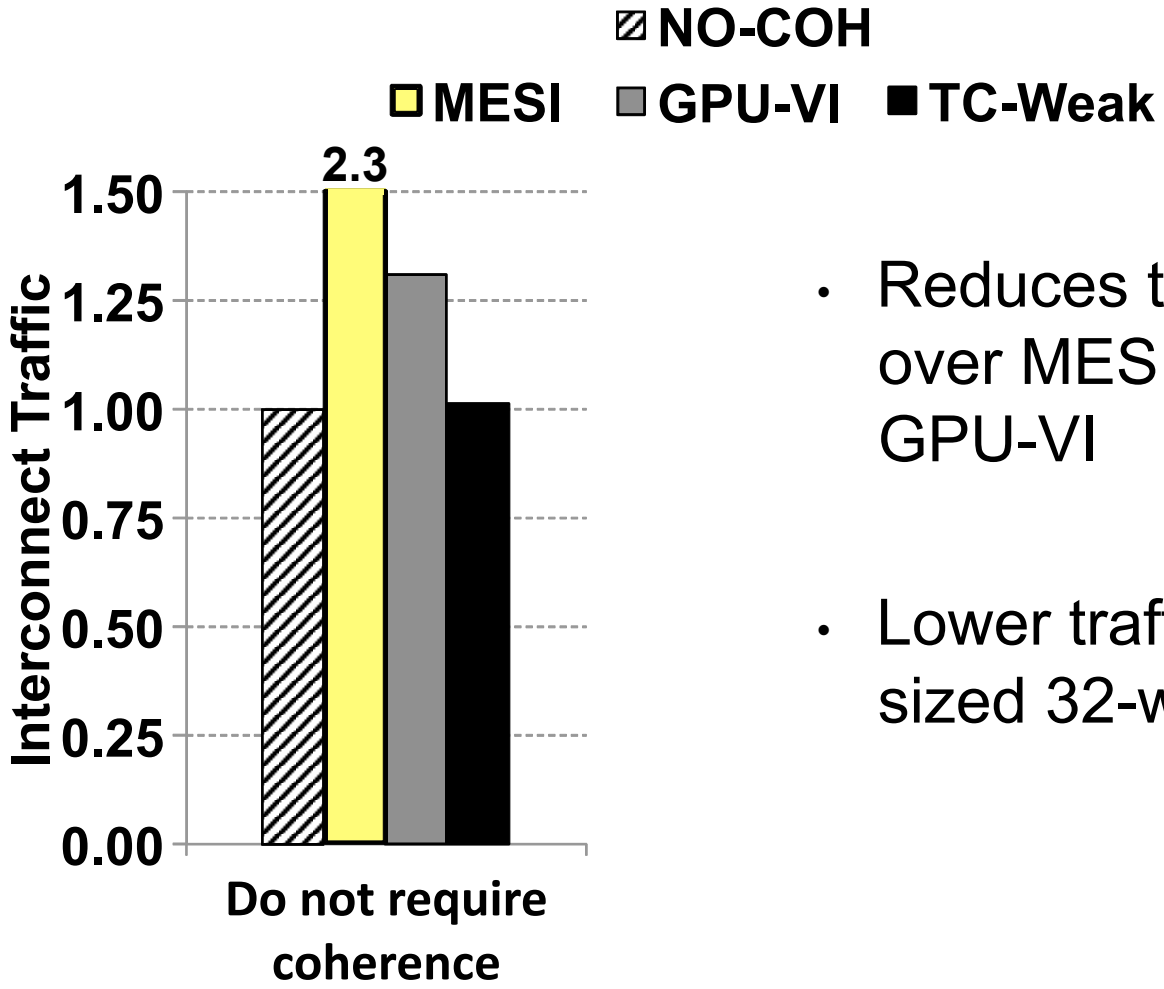
	Load	L1 WThru	L1 Atomic	L1 Replacement	Data	Data Done	DataG	DataG Done	Ack	Ack Done	AckG	AckG Done	DataAtomic	DataAtomic Done
I	o i pr+ a k /I S	i pw+ d2t k /I1	i pw+ d2t a k /I1	f										
S	h k	ist pw+ u k /SM	ist pw+ d2t a k /I I	f /I										
I S		pw+ d2t k /I1	pw+ d2t a k /I1	z		pr- dd t h s o /S								
I I	pr+ a k	pw+ d2t k	pw+ d2t a k	f	pr- h o	pr- h s o /I	pw- h gt o	pw- h gt s o /I	pw- h o	pw- h s o /I	pw- h gt o	pw- h gt s o /I	pw- h gt o	pw- h gt s o /I
SM	h k	pw+ u k	pw+ d2t a k /I1	z			pw- h dd t gt o	pw- h dd t gt s o /S	pw- h t o	pw- h t s o /S	pw- h t gt o	pw- h t gt s o /S		

TC-Weak L2

	L1 GETS	L1 Upgrade T	L1 Upgrade NT	L1 Write	L2 Atomic	L2 Expire	L2 Replacement	L2 Replacement clean	Mem Data	Mem Data SS
NP	q i s a j /IS		q d il x as j /IM	q d il x as j /IM	q d il x as j /IMA					
E	t ds set j /S		d wa ti mr set j /S	d wa ti mr set j /S	d ds a ti mr set j /S		c r /NP	r /NP		
S	t ds set j /SS	d wa ti mr set j	d ds g ti mr set j	d wa g ti mr set j	d ds a ti mr set j		ir it c r /MI	ir it r /MI		
SS	t ds set j	d wa g ti mr set j /S	d ds g ti mr set j /S	d wa g ti mr set j /S	d ds a ti mr set j /S		ir it c r /MI	ir it r /MI		
IS	s j		z	z	z		z	z	m tt dst s o /S	m tt dst s o /SS
IM	z		z	z	z		z	z	m tn mt wagt s o /S	
IMA	z		z	z	z		z	z	m tn mt dst a s o /S	
MI	q is s a j /IS		q d ils x as j /IM	q d ils x as j /IM	q d ils x as j /IMA	s ot /NP				

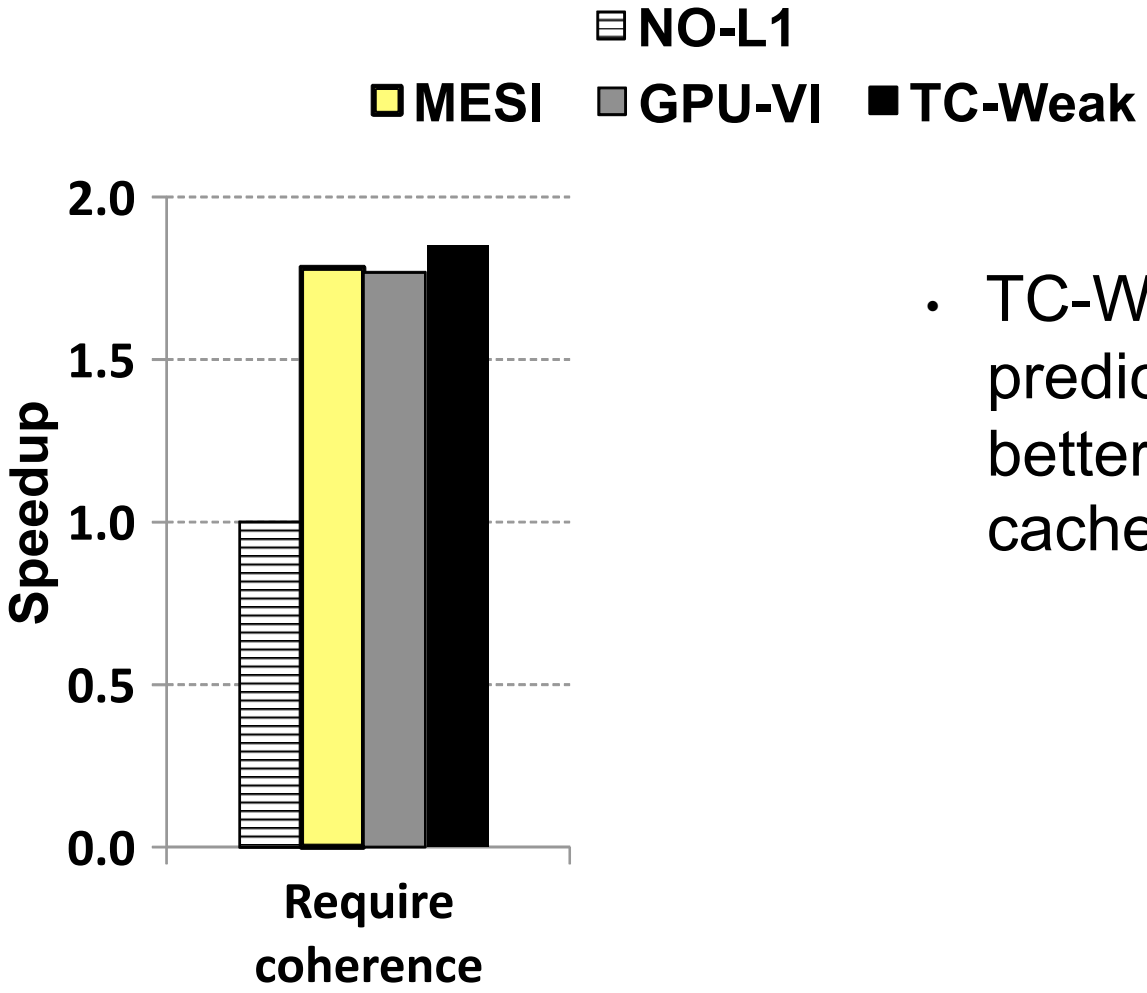
Unblock Cancel	Exclusive Unblock
k /SS	mu k /MT
k /MT	mu k /MT
	mu k /MT
k /MT	

Interconnect Traffic



- Reduces traffic by 53% over MESI and 23% over GPU-VI
- Lower traffic than 16x-sized 32-way directory

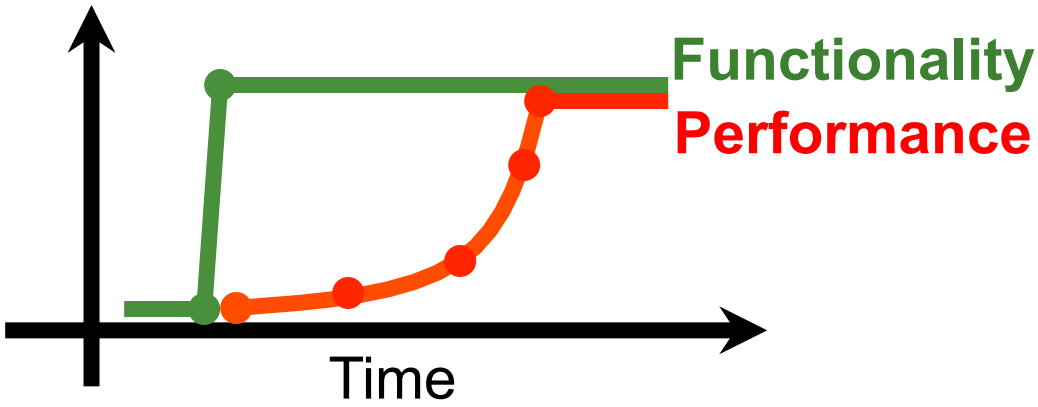
Performance



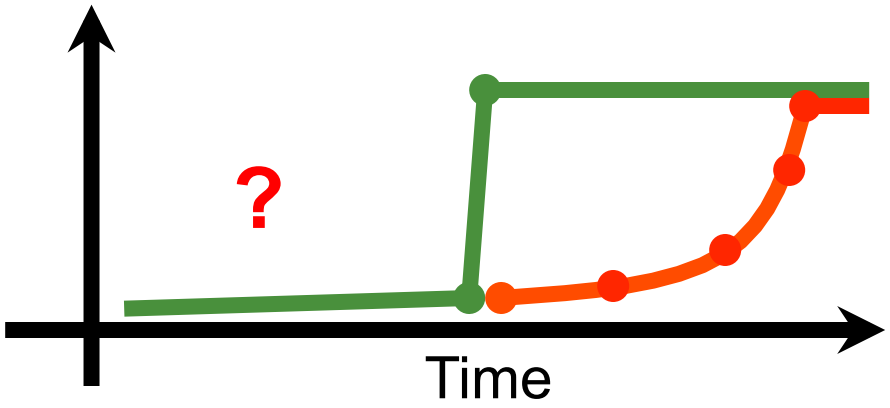
- TC-Weak with simple predictor performs 85% better than disabling L1 caches

Fewer bugs

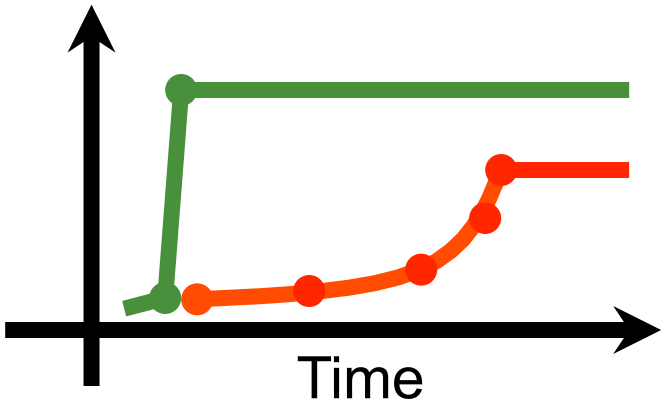
- Lifetime of Accelerator Application Development



Fine-Grained Locking



Transactional Memory



Are TM and GPUs Incompatible?

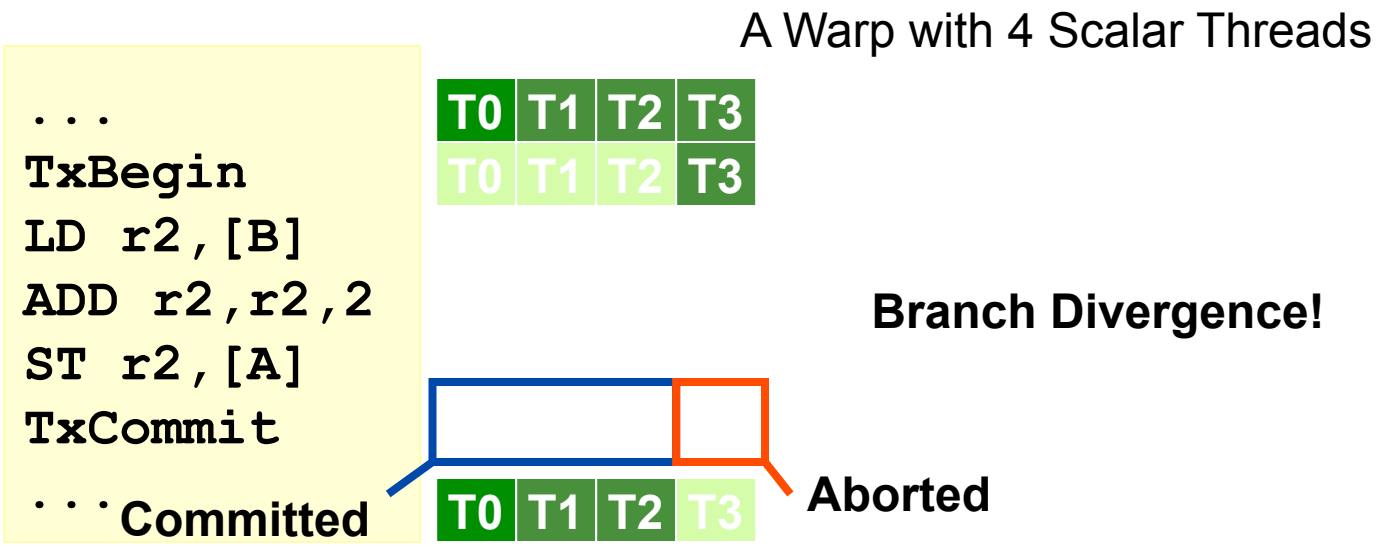
GPU uarch very different from multicore CPU.

KILO TM [Fung MICRO'11, Top Picks'12]

- Hardware TM for GPUs
- Half performance of fine grained locking
- Chip area overhead of 0.5%

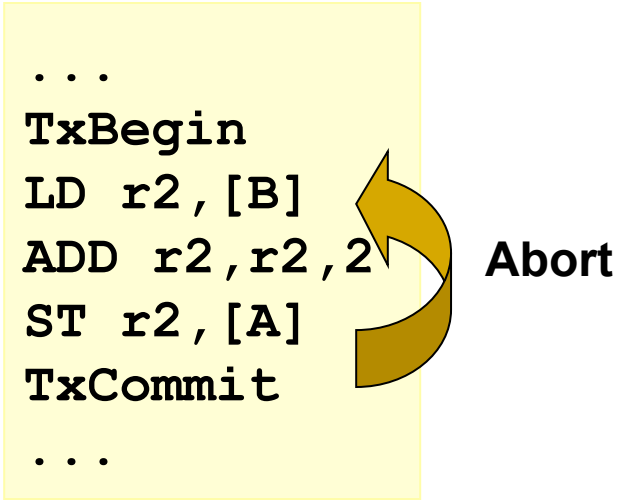
Hardware TM for GPUs Challenge #1: SIMD Hardware

- On GPUs, scalar threads in a warp/wavefront execute in lockstep

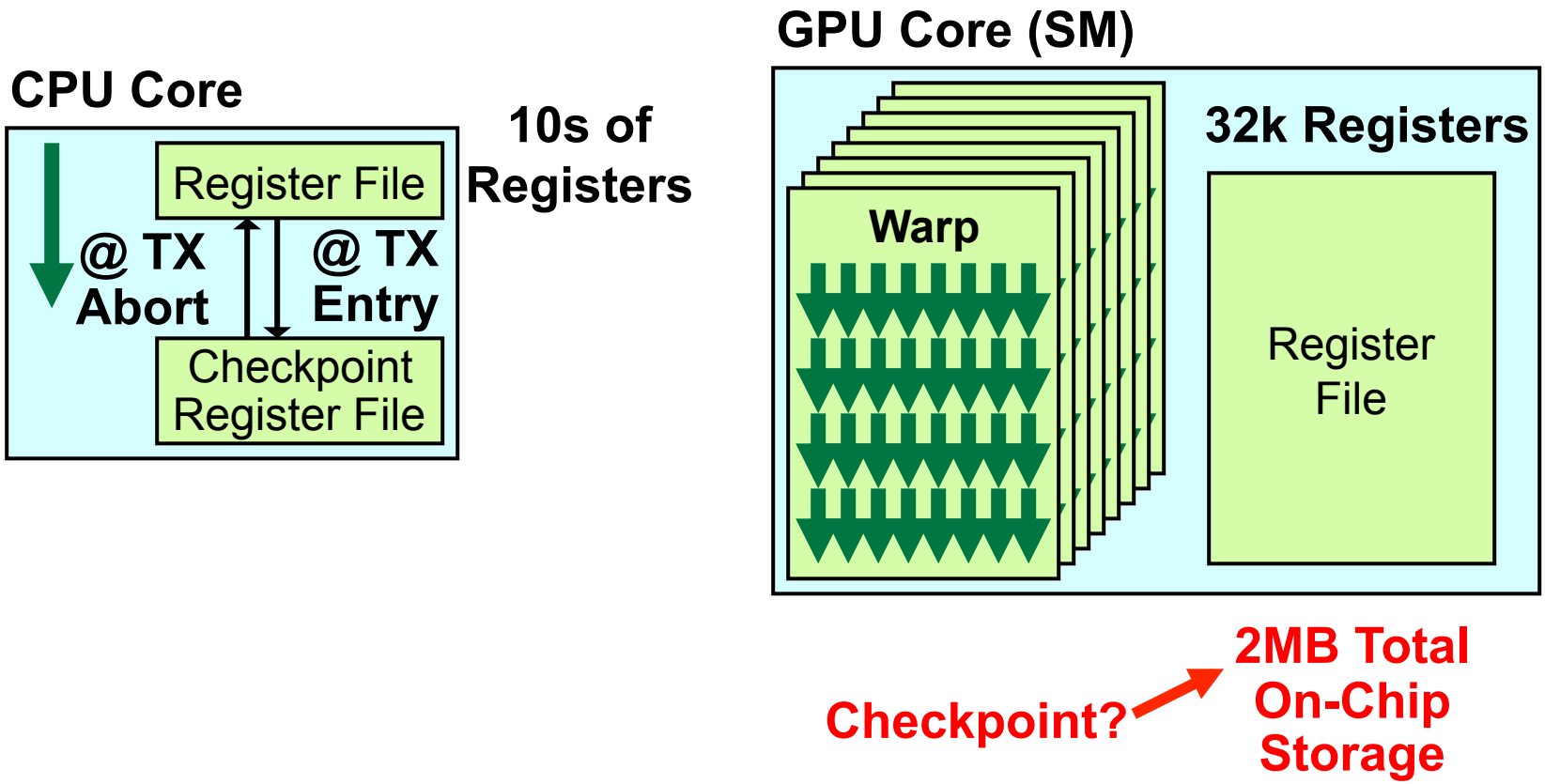


KILO TM – Solution to Challenge #1: SIMD Hardware

- Transaction Abort
  - Like a Loop
  - Extend SIMT Stack

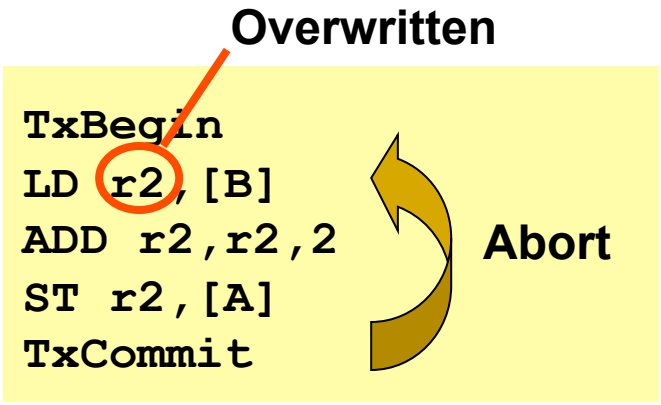


Hardware TM for GPUs
Challenge #2: Transaction Rollback



KILO TM – Solution to Challenge #2: Transaction Rollback

- SW Register Checkpoint
  - Most TX: Reg overwritten first appearance (idempotent)
  - TX in Barnes Hut: Checkpoint 2 registers



Hardware TM for GPUs
Challenge #3: Conflict Detection

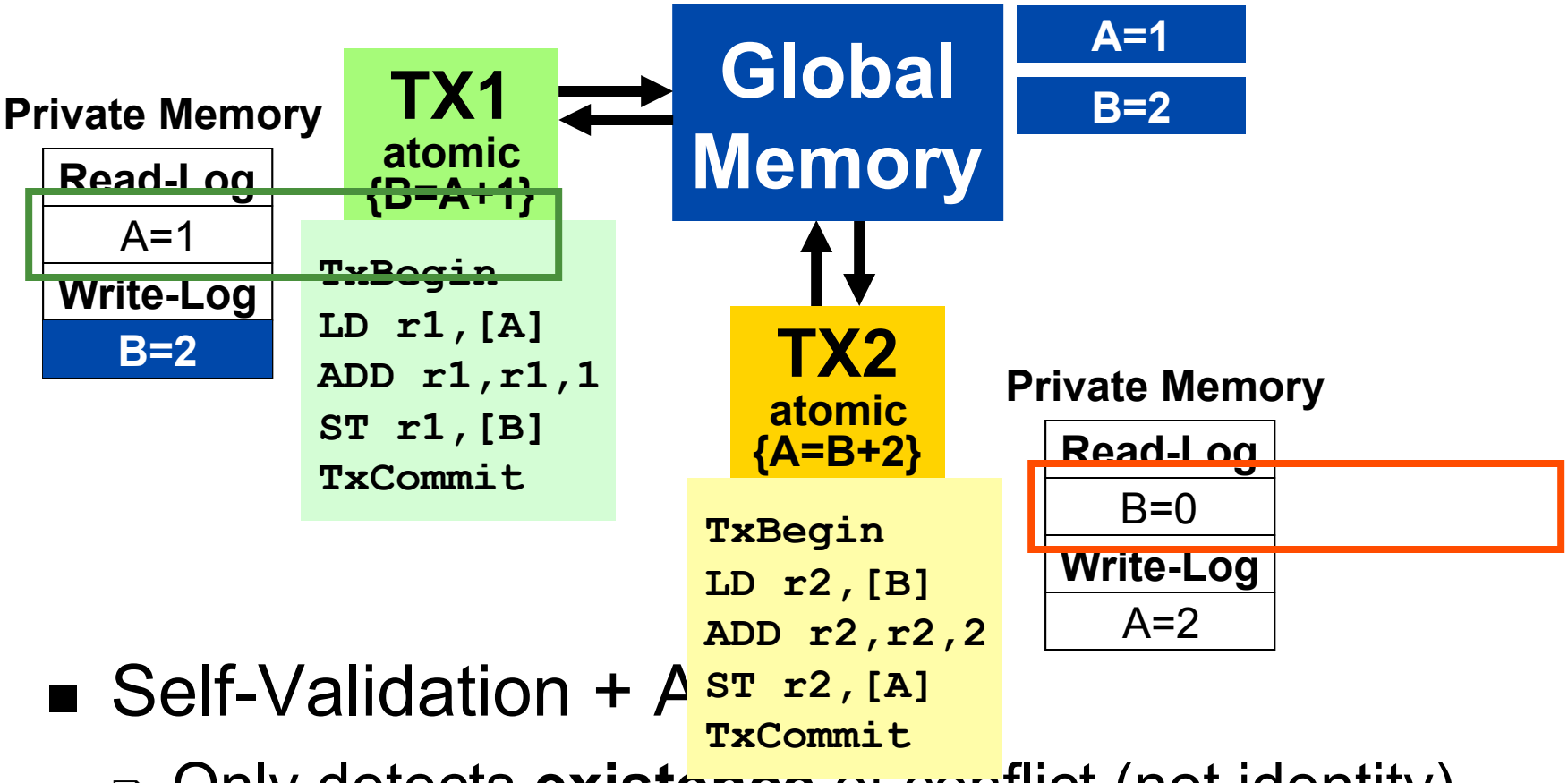
Existing HTMs use Cache Coherence Protocol

- Not Available on (current) GPUs
- No Private Data Cache per Thread

Signatures?

- 1024-bit / Thread
- **3.8MB / 30k Threads**

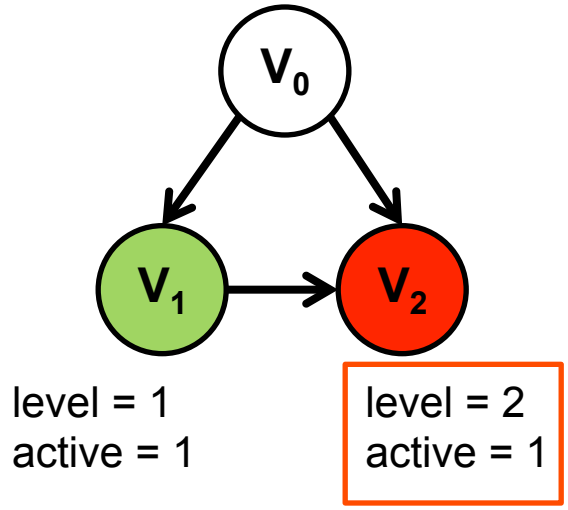
KILO TM:
Value-Based Conflict Detection



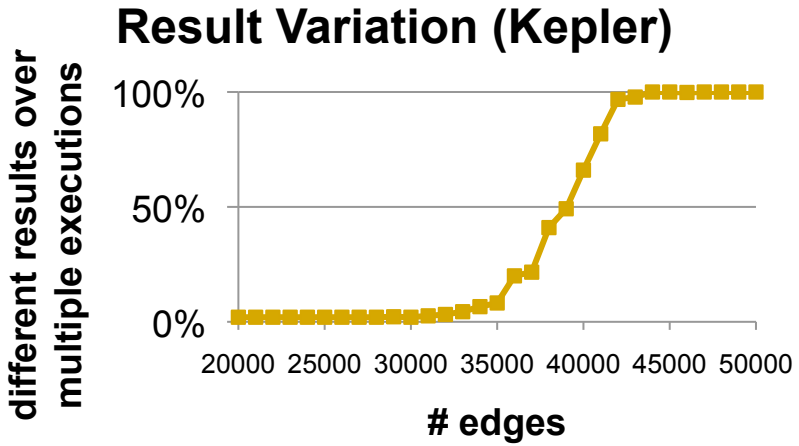
- Self-Validation + A
  - Only detects **existence** of conflict (not identity)

Easier debugging

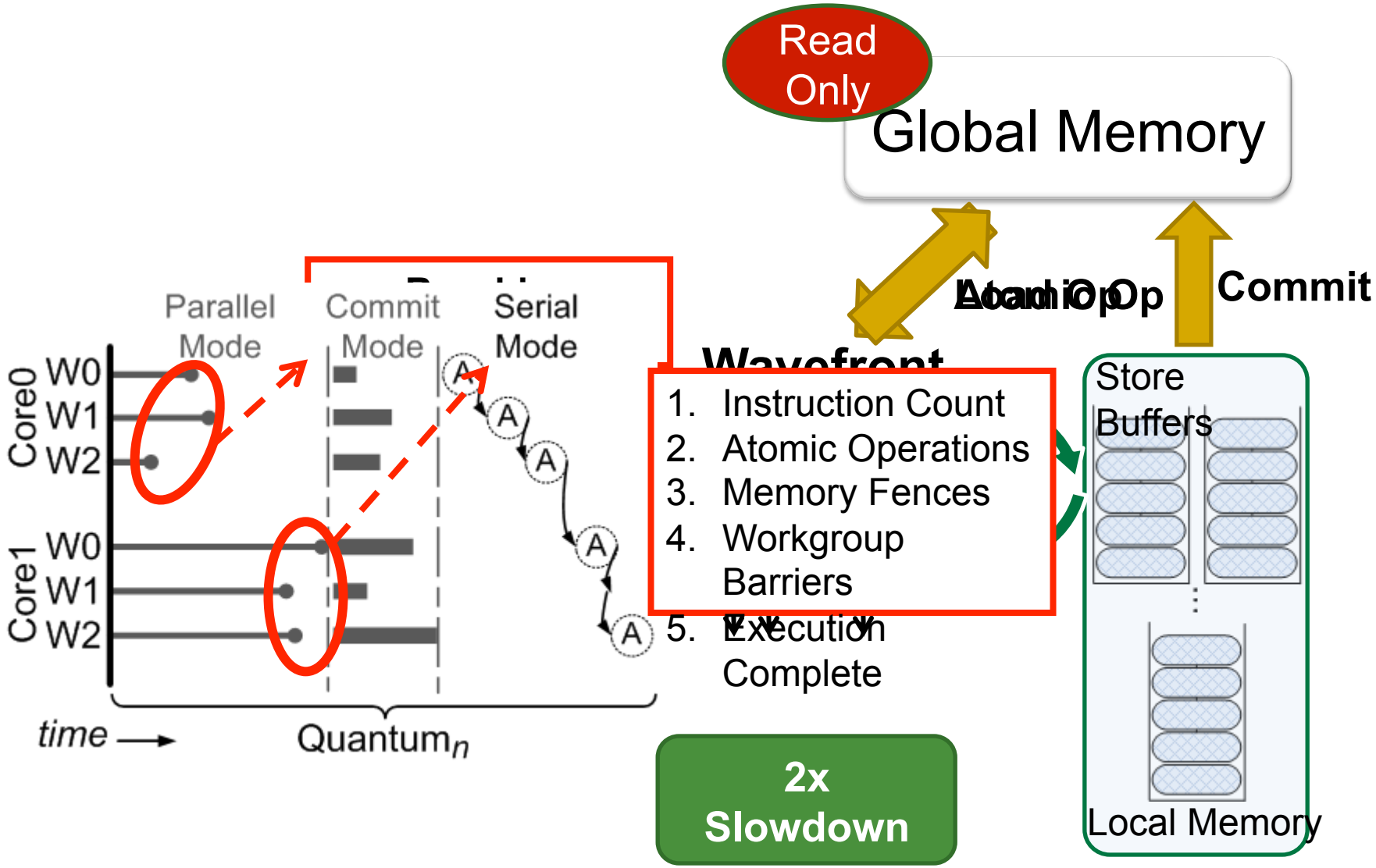
```
0  __global__ void BFS_step_kernel(...) {
1    if( active[tid] ) {
2      active[tid] = false;
3      visited[tid] = true;
4      foreach (int id = neighbour_nodes){
5        if( visited[id] == false ){
6          level[id] = level[tid] + 1;
7          active[id] = true;
8          …
9  } } } }
```



BFS algorithm
Published in HiPC 2007



GPUDet (Jooybar: ASPLOS 2013)



Summary

- Start from efficient architecture and try to improve programmability
  - Get efficiency and keep programmers happy

reasonably

Thanks!
Questions?