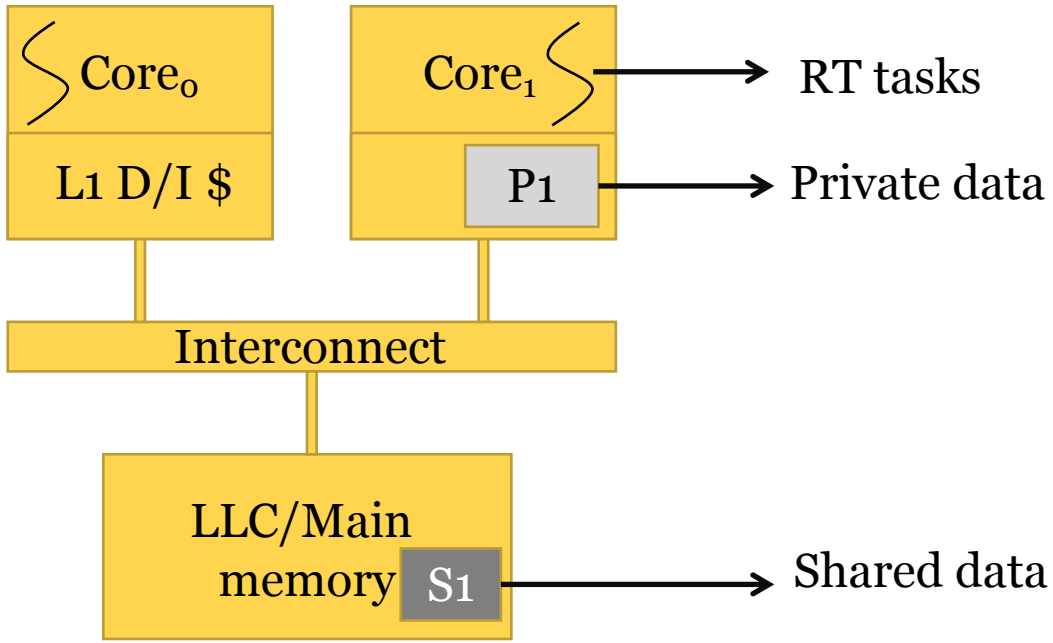
Predictable Cache Coherence for Multi-Core Real-Time Systems

Mohamed Hassan, **Anirudh M. Kaushik** and Hiren Patel

RTAS 2017

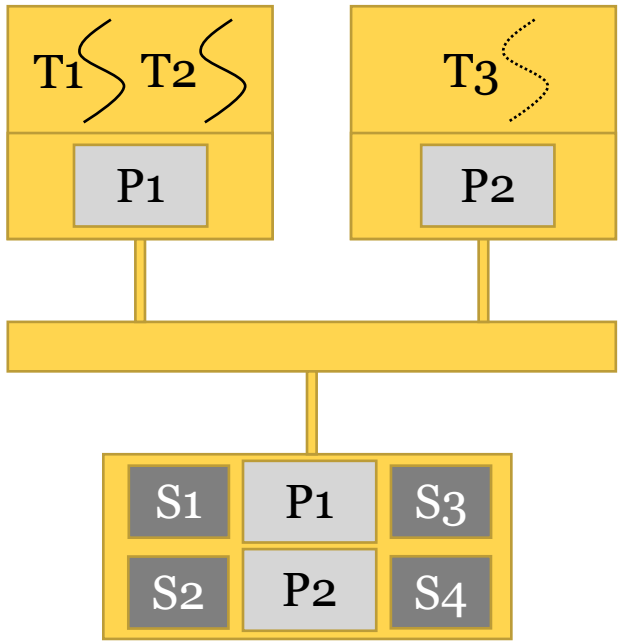
UNIVERSITY OF WATERLOO

Motivation: Data sharing in multi-core real-time systems



Motivation: Data sharing in multi-core real-time systems

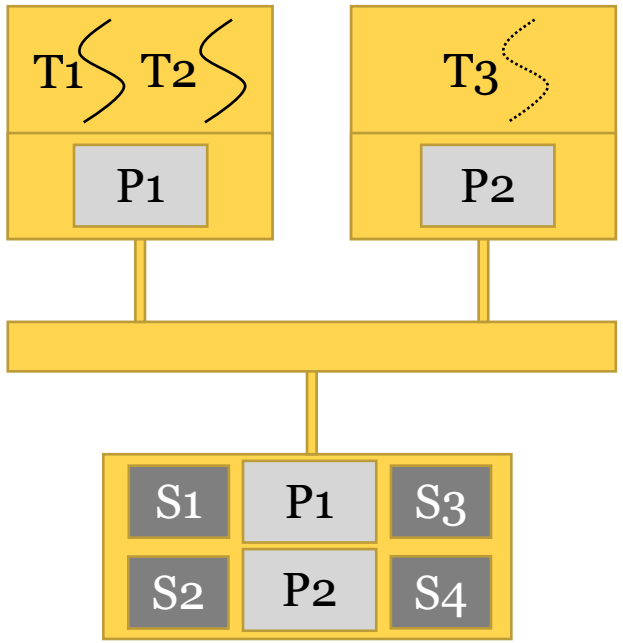
No caching of shared data
[RTSS'09, RTNS'10]



✔ Simpler timing analysis
✖ Hardware support
✖ Long execution time

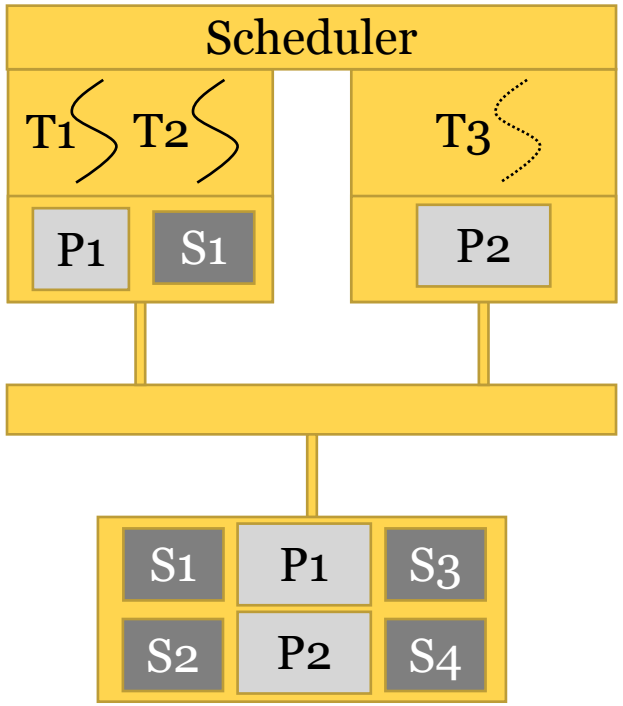
Motivation: Data sharing in multi-core real-time systems

No caching of shared data
[RTSS'09, RTNS'10]



- ✔ Simpler timing analysis
- ✖ Hardware support
- ✖ Long execution time

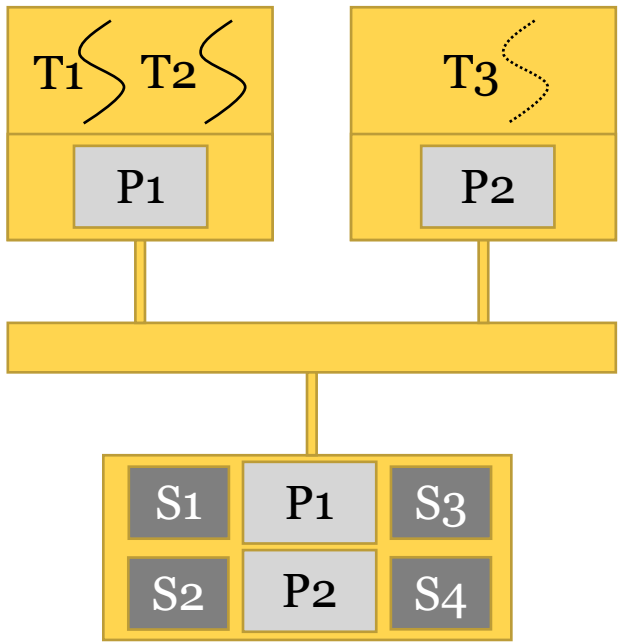
Task scheduling on shared data
[ECRTS'10, RTSS'16]



- ✔ Private cache hits on shared data
- ✔ No hardware support
- ✖ Limited multi-core parallelism
- ✖ Changes to OS scheduler

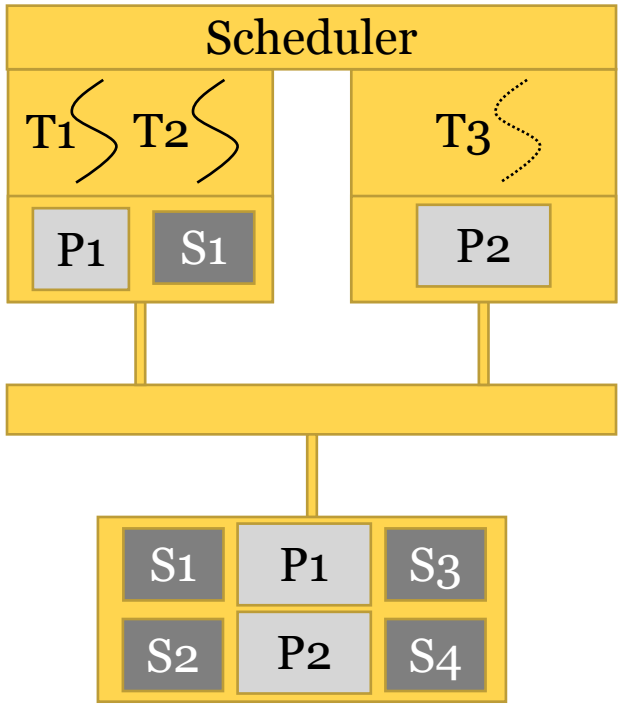
Motivation: Data sharing in multi-core real-time systems

No caching of shared data
[RTSS'09, RTNS'10]



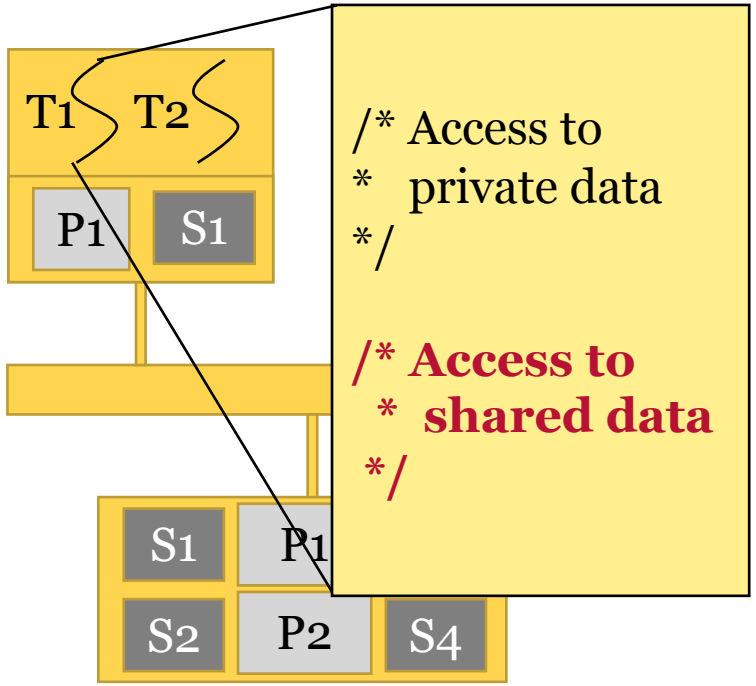
- ✔ Simpler timing analysis
- ✖ Hardware support
- ✖ Long execution time

Task scheduling on shared data
[ECRTS'10, RTSS'16]



- ✔ Private cache hits on shared data
- ✔ No hardware support
- ✖ Limited multi-core parallelism
- ✖ Changes to OS scheduler

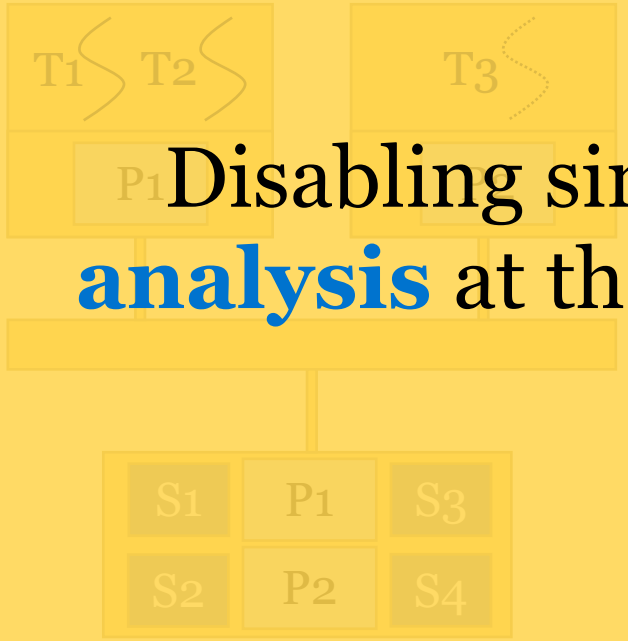
Software cache coherence
[SAMOS'14]



- ✔ Private cache hits on shared data
- ✖ Hardware support
- ✖ Source code changes

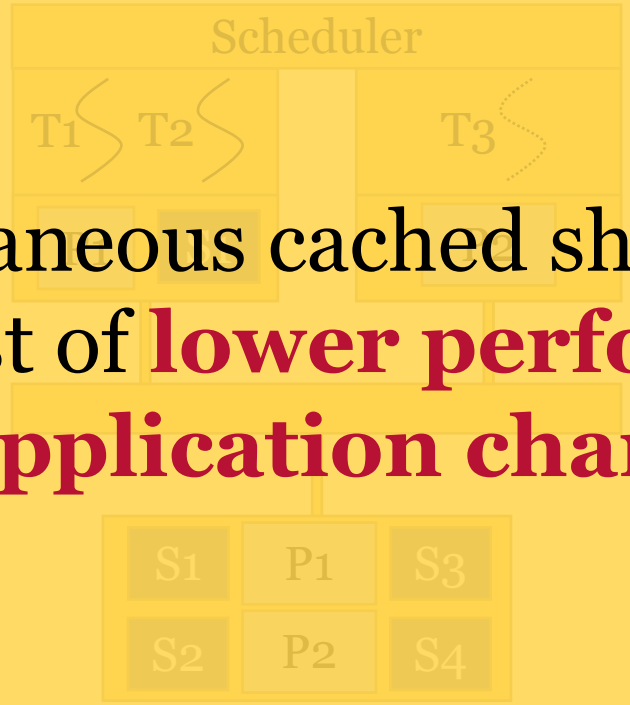
Motivation: Data sharing in multi-core real-time systems

No caching of shared data
[RTSS'09, RTNS'10]



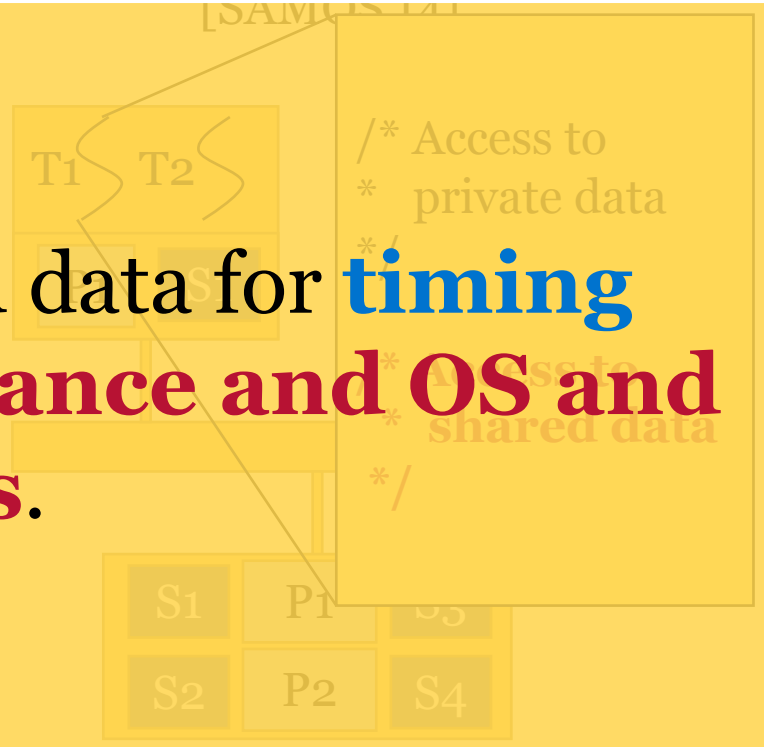
- ✔ Simpler timing analysis
- ✖ Hardware support
- ✖ Long execution time

Task scheduling on shared data
[ECRTS'10, RTSS'16]



- ✔ Private cache hits on shared data
- ✔ No hardware support
- ✖ Limited multi-core parallelism
- ✖ Changes to OS scheduler

Software cache coherence
[SAMOS'14]

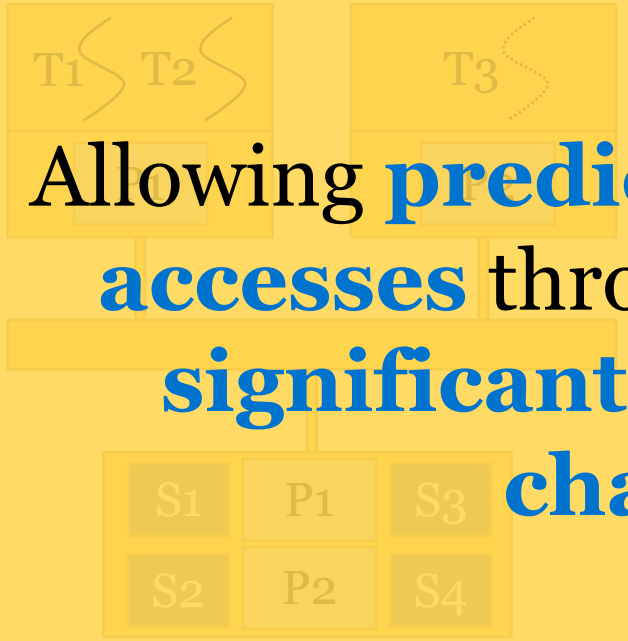


- ✔ Private cache hits on shared data
- ✖ Hardware support
- ✖ Source code changes

Disabling simultaneous cached shared data for **timing analysis** at the cost of **lower performance and OS and application changes**.

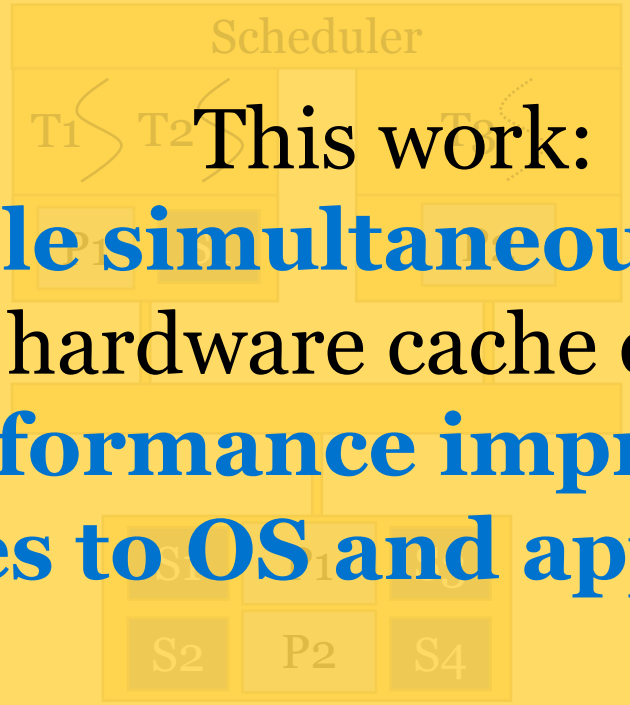
Motivation: Data sharing in multi-core real-time systems

No caching of shared data
[RTSS'09, RTNS'10]



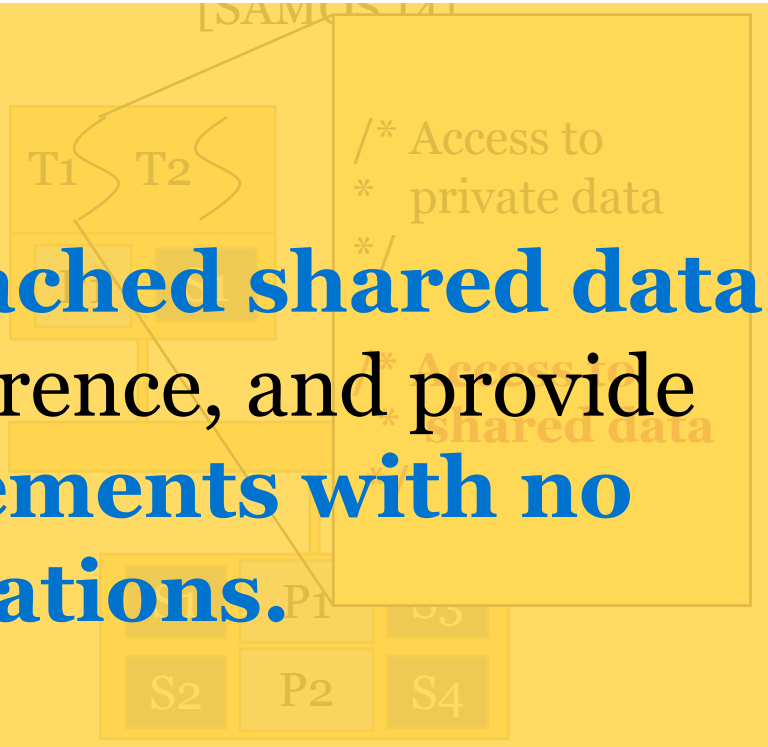
- ✔ Simpler timing analysis
- ✖ Hardware support
- ✖ Long execution time

Task scheduling on shared data
[ECRTS'10, RTSS'16]



- ✔ Private cache hits on shared data
- ✔ No hardware support
- ✖ Limited multi-core parallelism
- ✖ Changes to OS scheduler

Software cache coherence
[SAMOS'14]



- ✔ Private cache hits on shared data
- ✖ Hardware support
- ✖ Source code changes

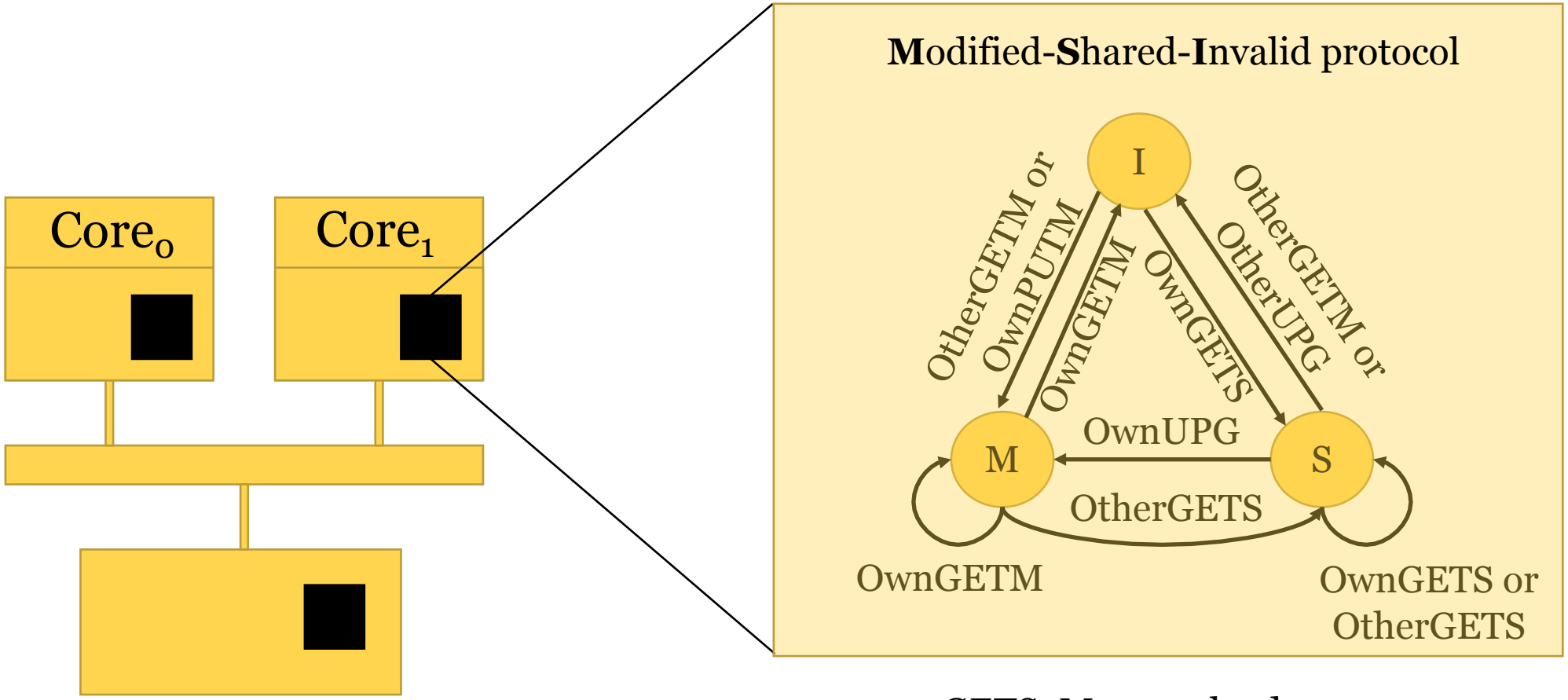
This work:

Allowing **predictable simultaneous cached shared data accesses** through hardware cache coherence, and provide **significant performance improvements with no changes to OS and applications.**

Outline

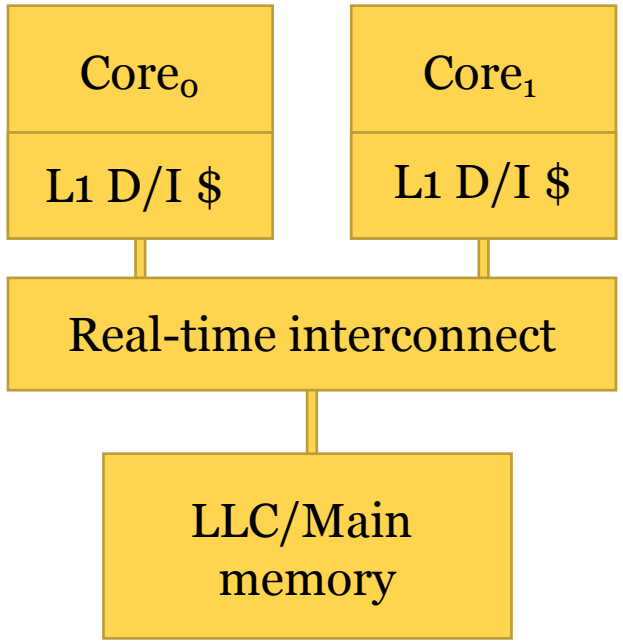
- Overview of hardware cache coherence
- Challenges of applying conventional hardware cache coherence on RT multi-core systems
- Our solution: **P**redictable **M**odified-**S**hared-**I**nvalid (PMSI) cache coherence protocol
- Latency analysis of PMSI
- Results
- Conclusion

Overview of hardware cache coherence

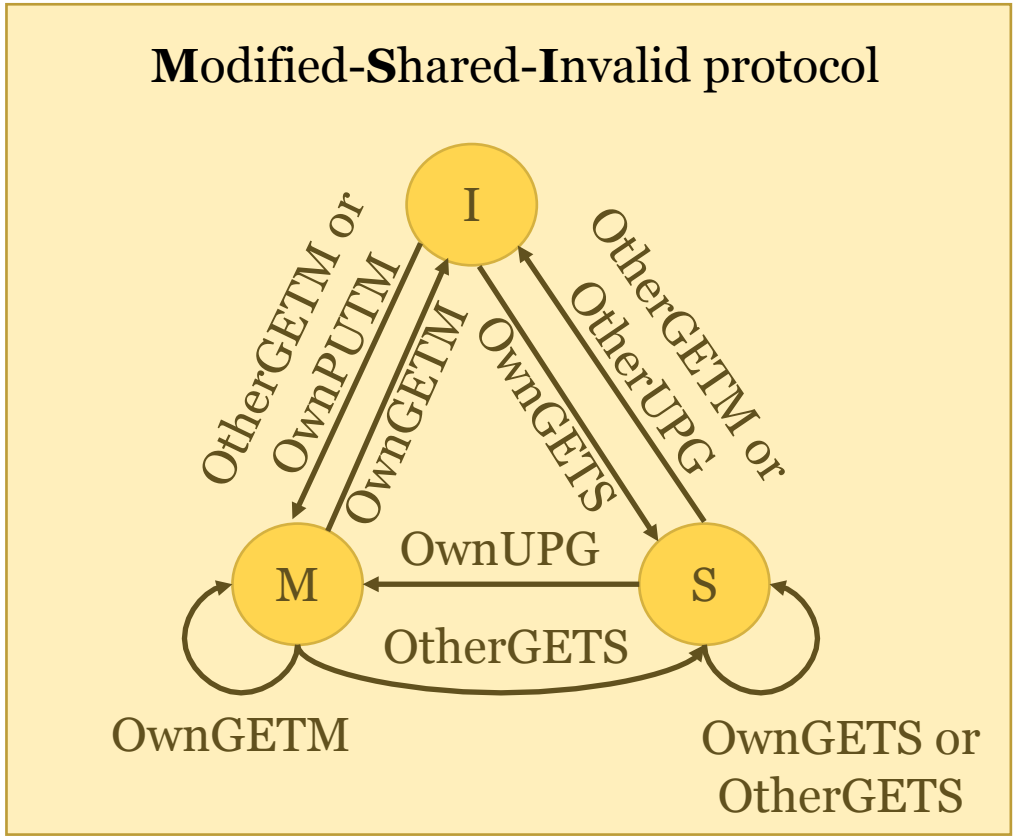


GETS: Memory load
GETM/UPG: Memory store
PUTM: Replacement

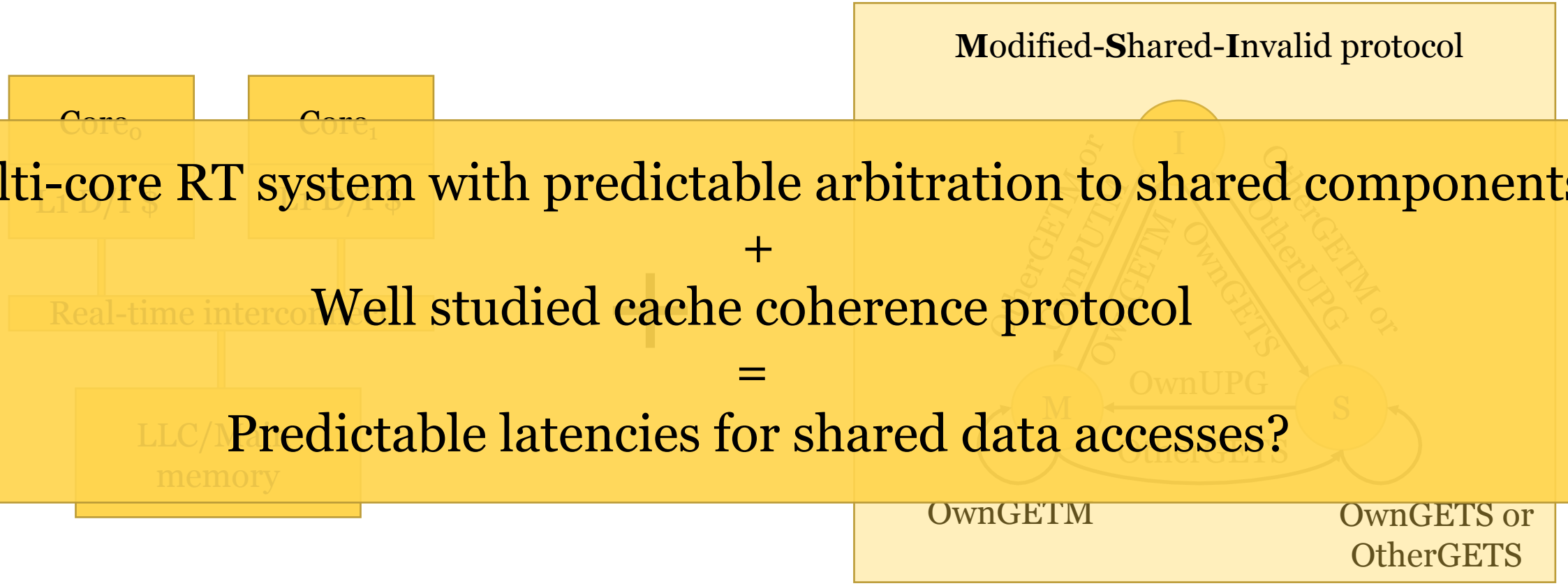
Applying hardware cache coherence to multi-core RT systems



+



Applying hardware cache coherence to multi-core RT systems



Multi-core RT system with predictable arbitration to shared components

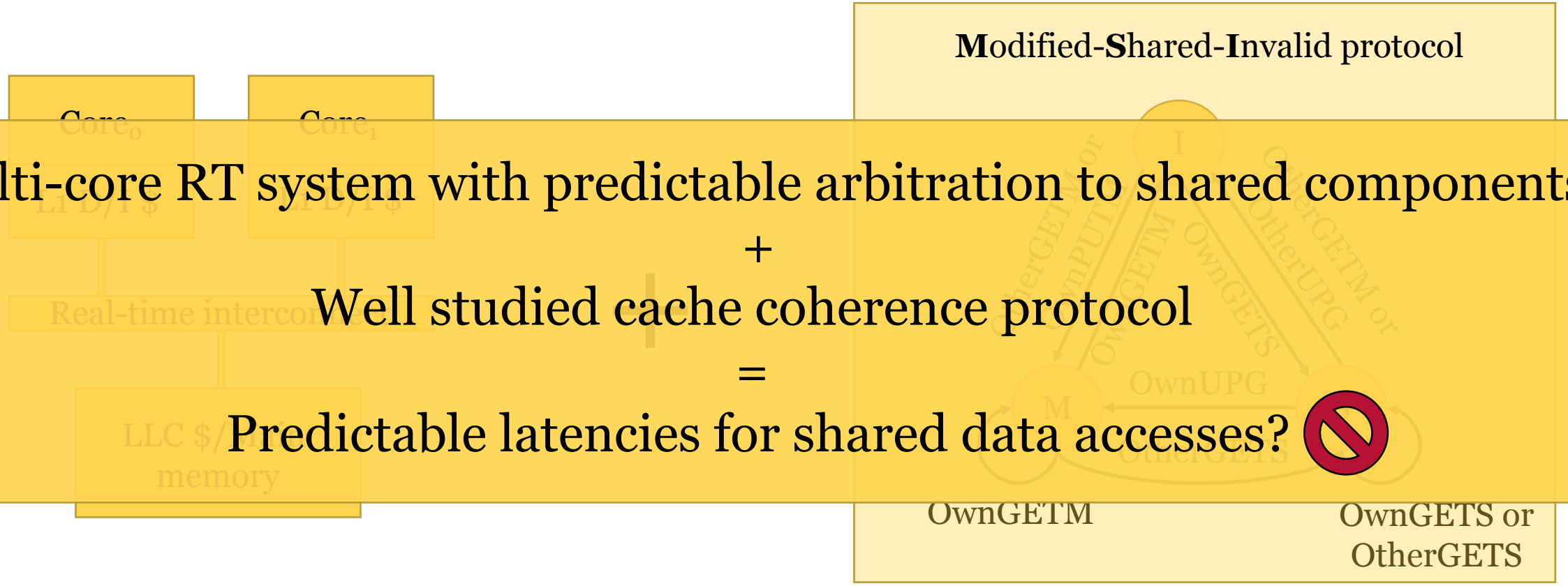
+

Well studied cache coherence protocol

=

Predictable latencies for shared data accesses?

Applying hardware cache coherence to multi-core RT systems



Multi-core RT system with predictable arbitration to shared components

+

Well studied cache coherence protocol

=

Predictable latencies for shared data accesses?

Sources of unpredictable memory access latencies due to hardware cache coherence

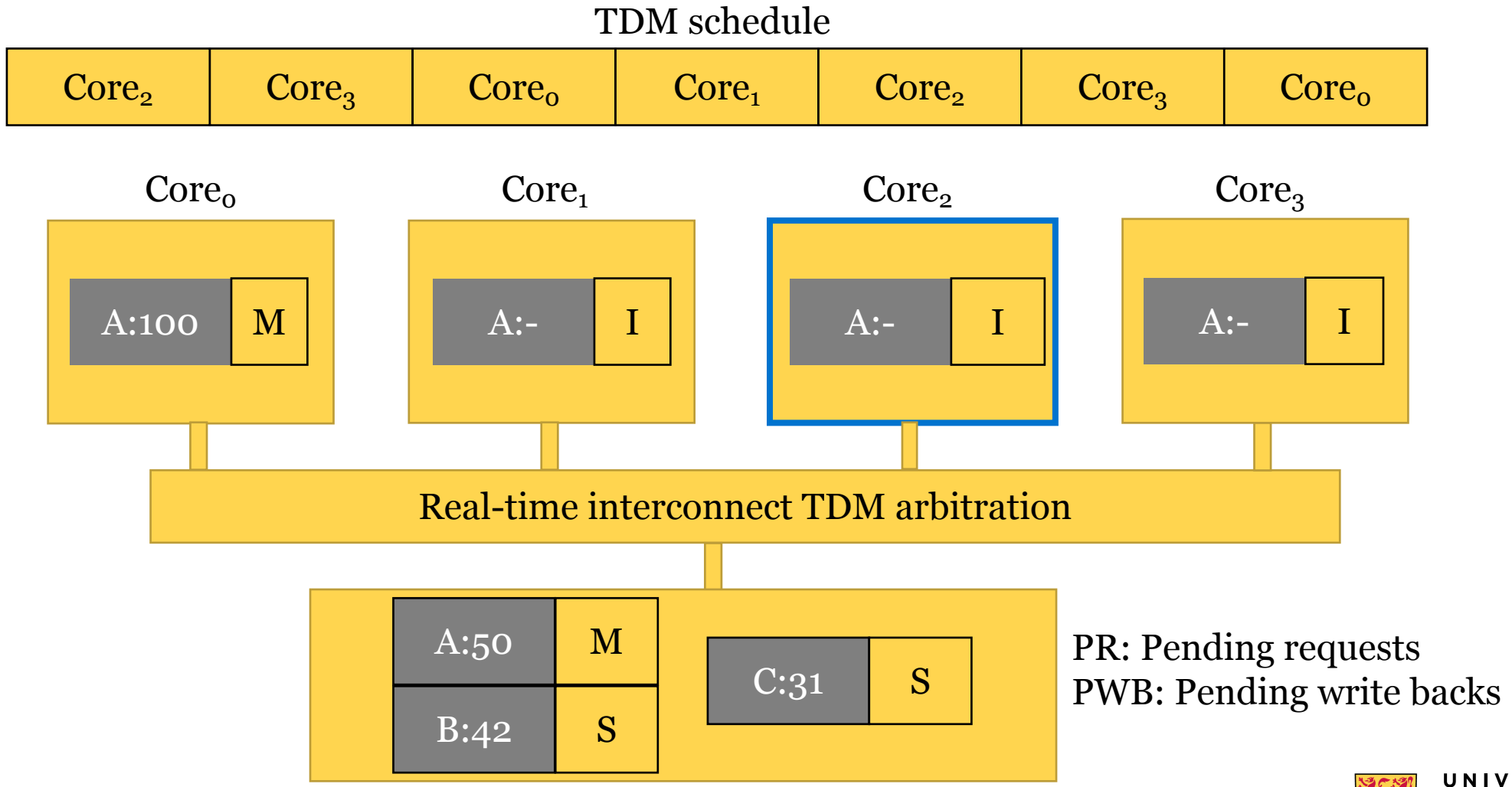
1. Inter-core coherence interference on same cache line

2. Inter-core coherence interference on different cache lines

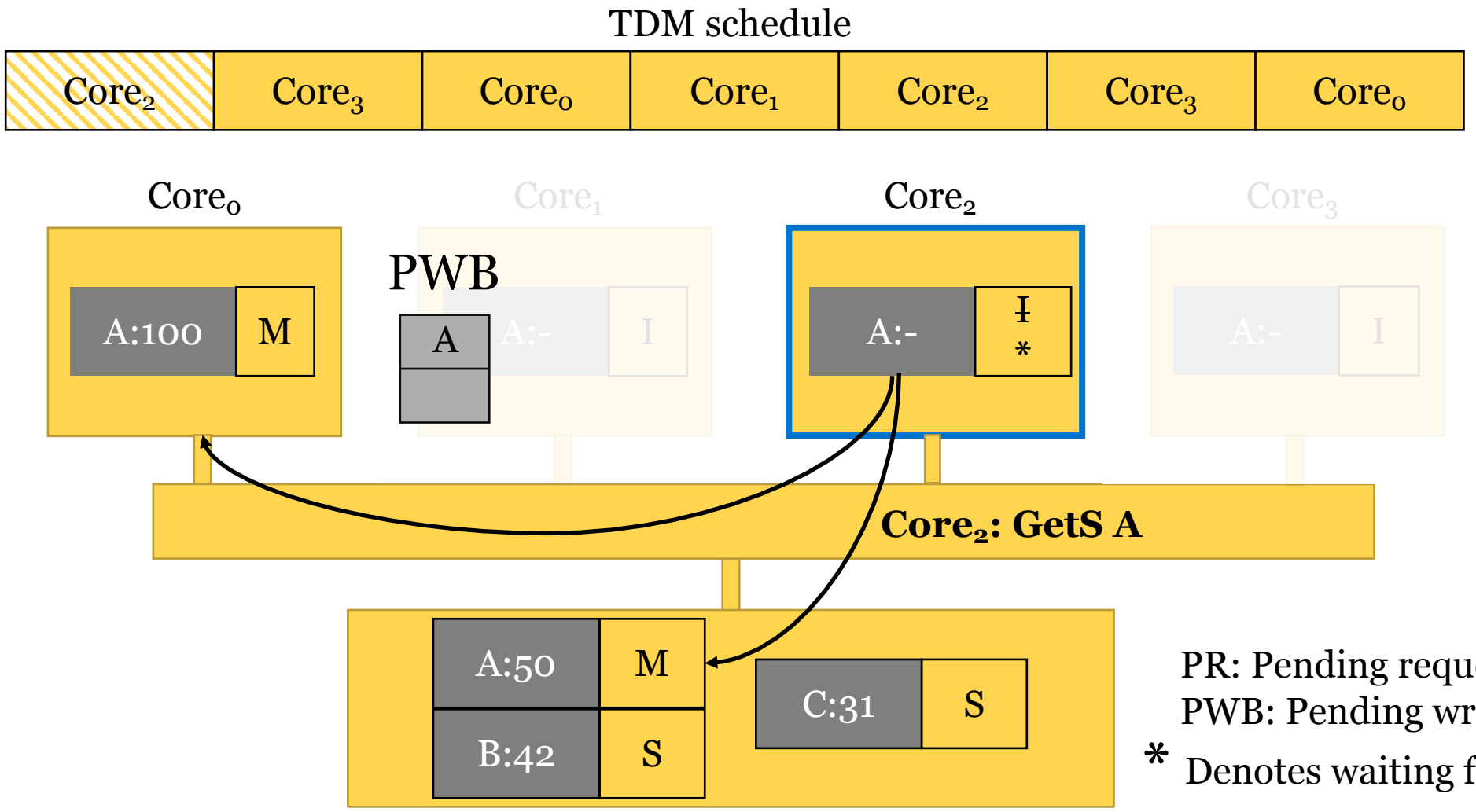
3. Inter-core coherence interference due to write hits

4. **Intra-core coherence interference**

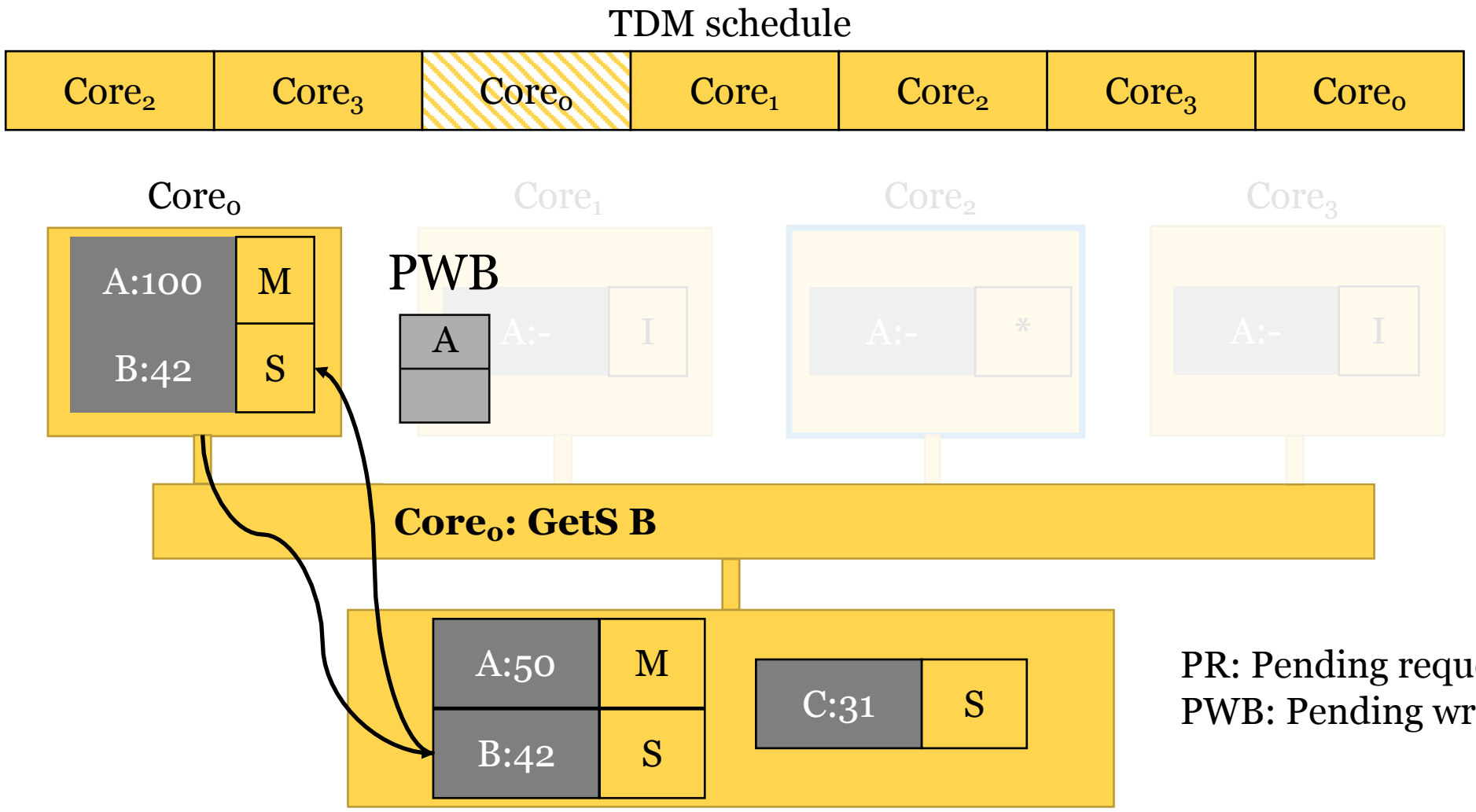
System model



Intra-core coherence interference



Intra-core coherence interference

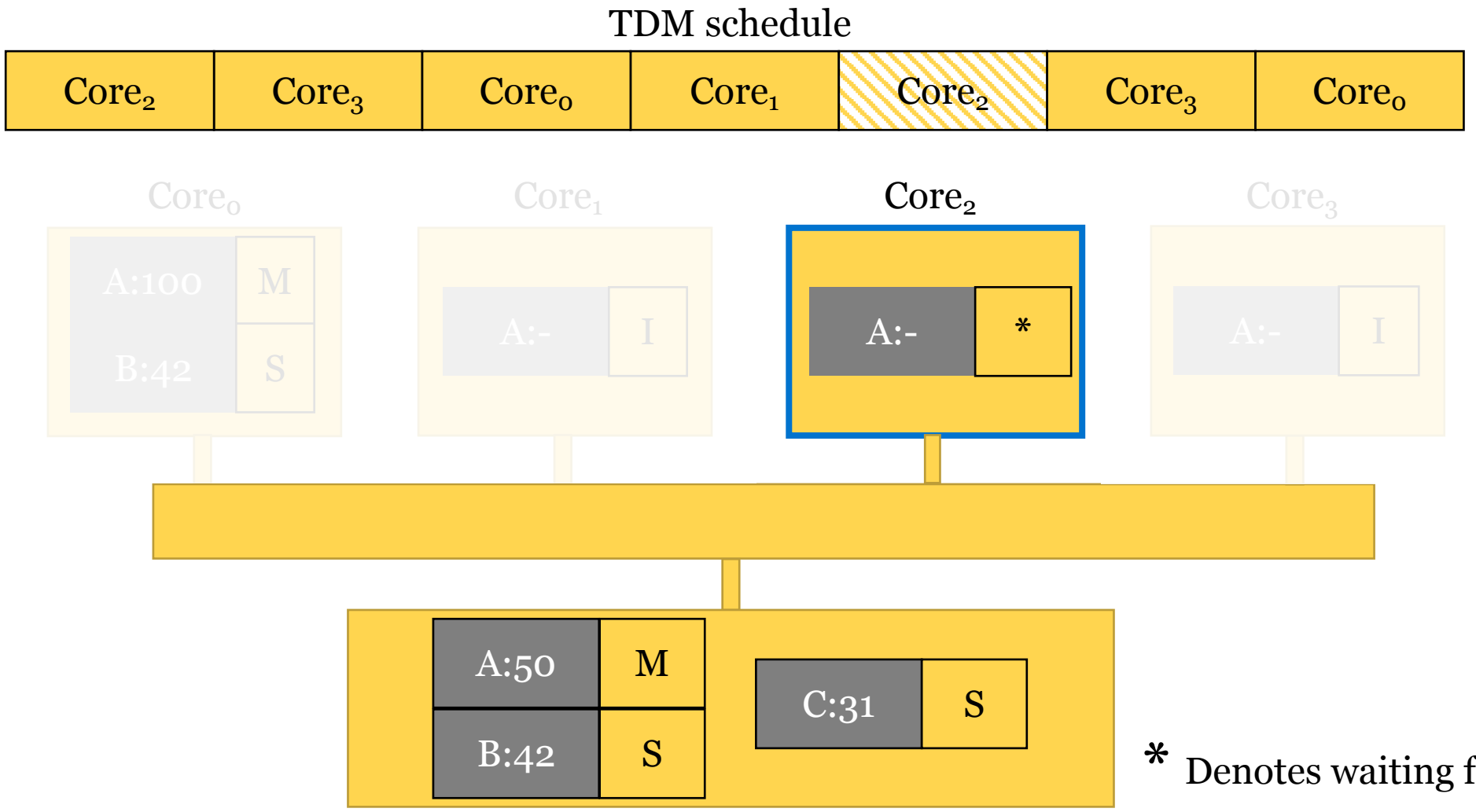


Core	Event
$Core_2$	Read A
$Core_0$	Read B

PR: Pending requests
PWB: Pending write backs

UNIVERSITY OF WATERLOO

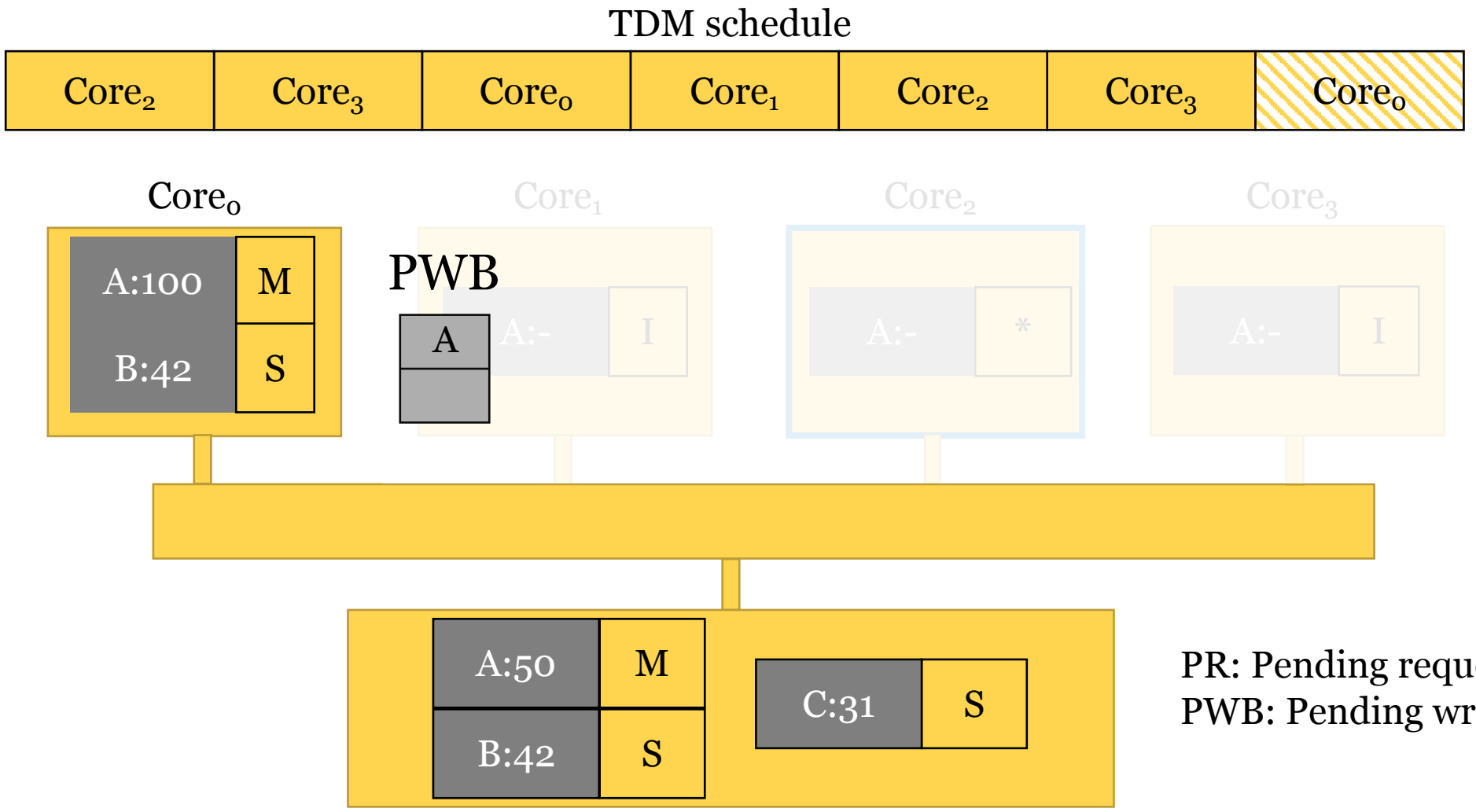
Intra-core coherence interference



Core	Event
$Core_2$	Read A
$Core_0$	Read B
$Core_2$	Waiting for data

* Denotes waiting for data

Intra-core coherence interference

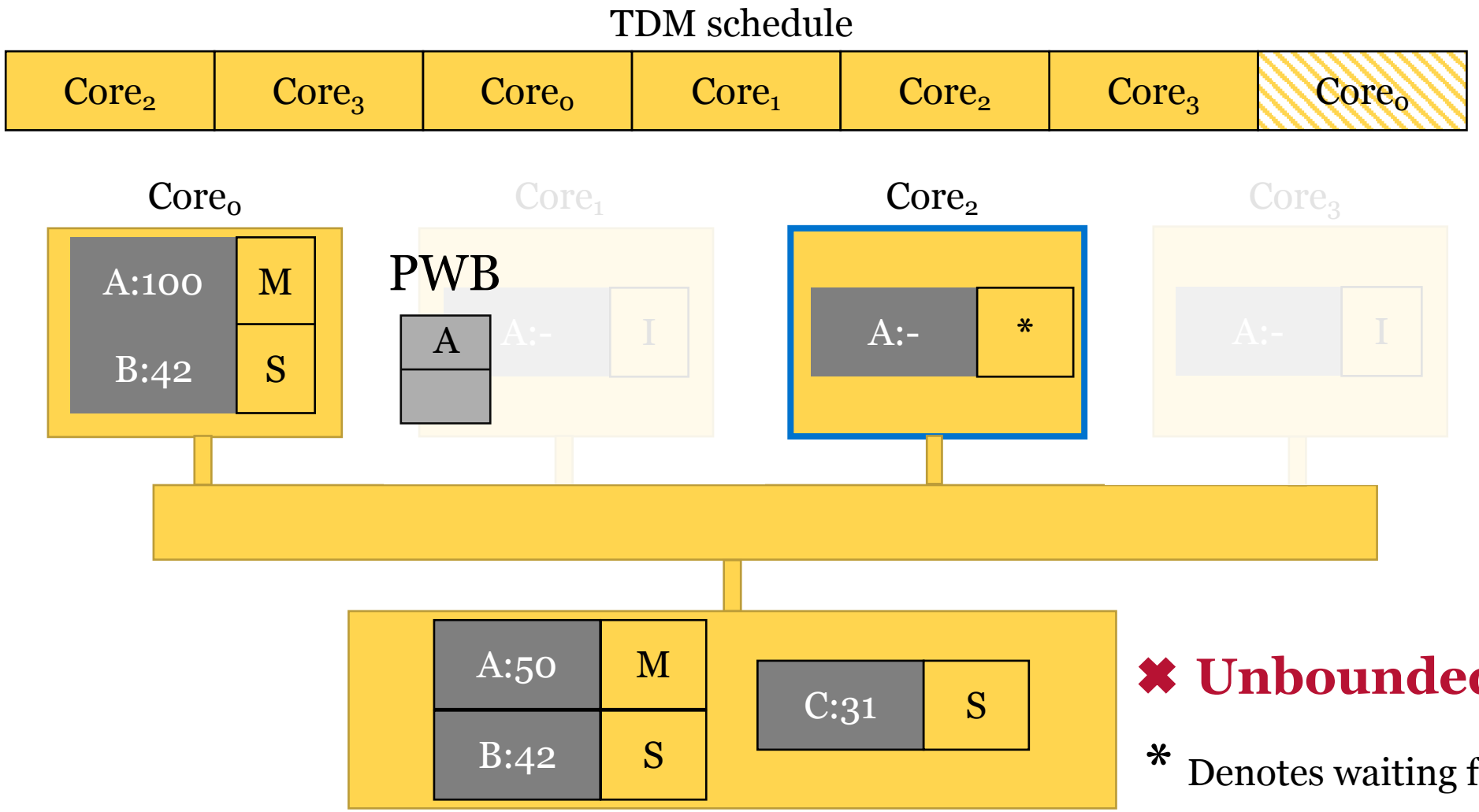


Core	Event
$Core_2$	Read A
$Core_0$	Read B
$Core_2$	Waiting for data
$Core_0$	Read C

PR: Pending requests
PWB: Pending write backs

UNIVERSITY OF WATERLOO

Intra-core coherence interference

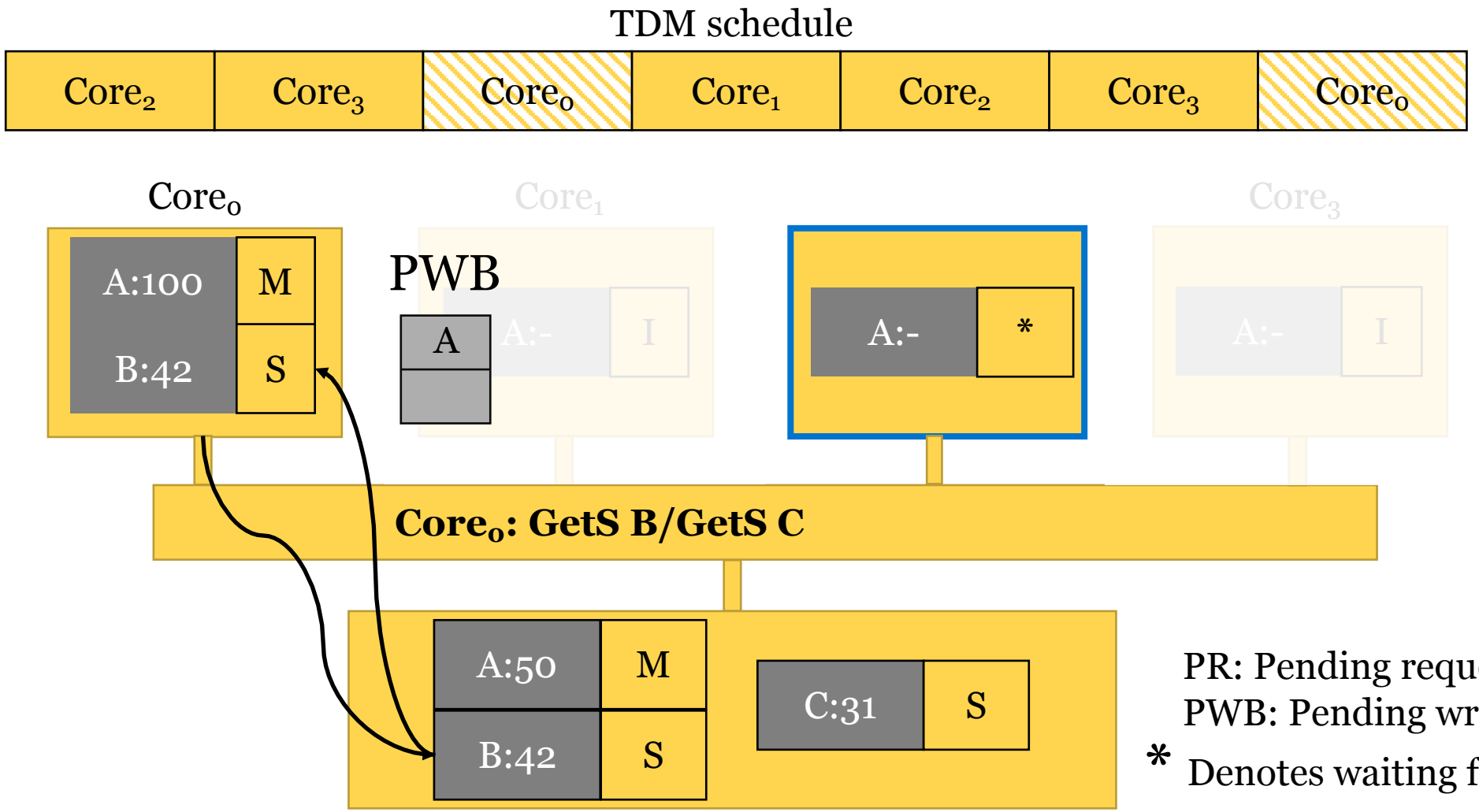


Core	Event
$Core_2$	Read A
$Core_0$	Read B
$Core_2$	Waiting for data
$Core_0$	Read C

✖ **Unbounded latency for $Core_2$**

* Denotes waiting for data

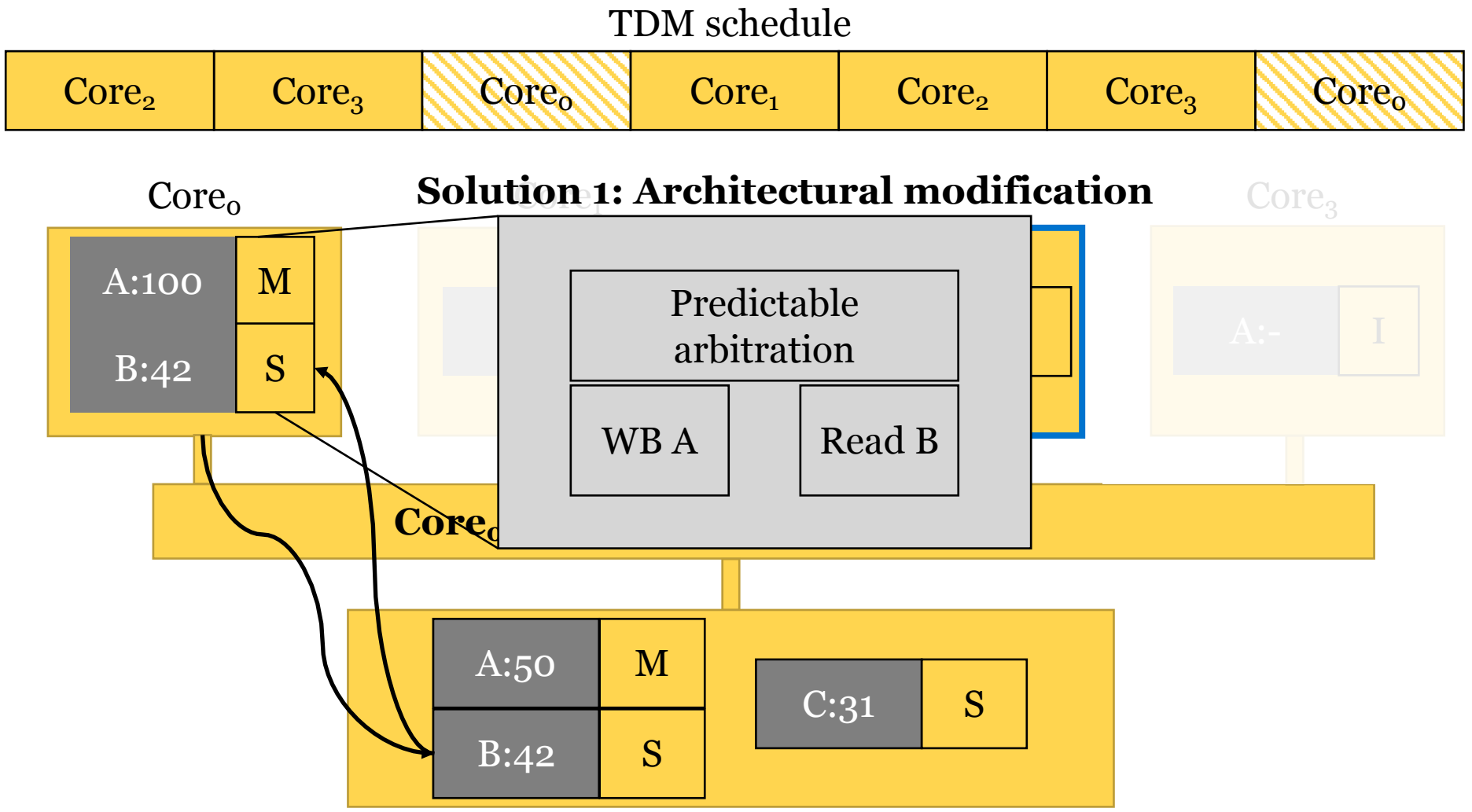
Intra-core coherence interference



Core	Event
$Core_2$	Read A
$Core_0$	Read B
$Core_2$	Waiting for data
$Core_0$	Read C

PR: Pending requests
PWB: Pending write backs
* Denotes waiting for data

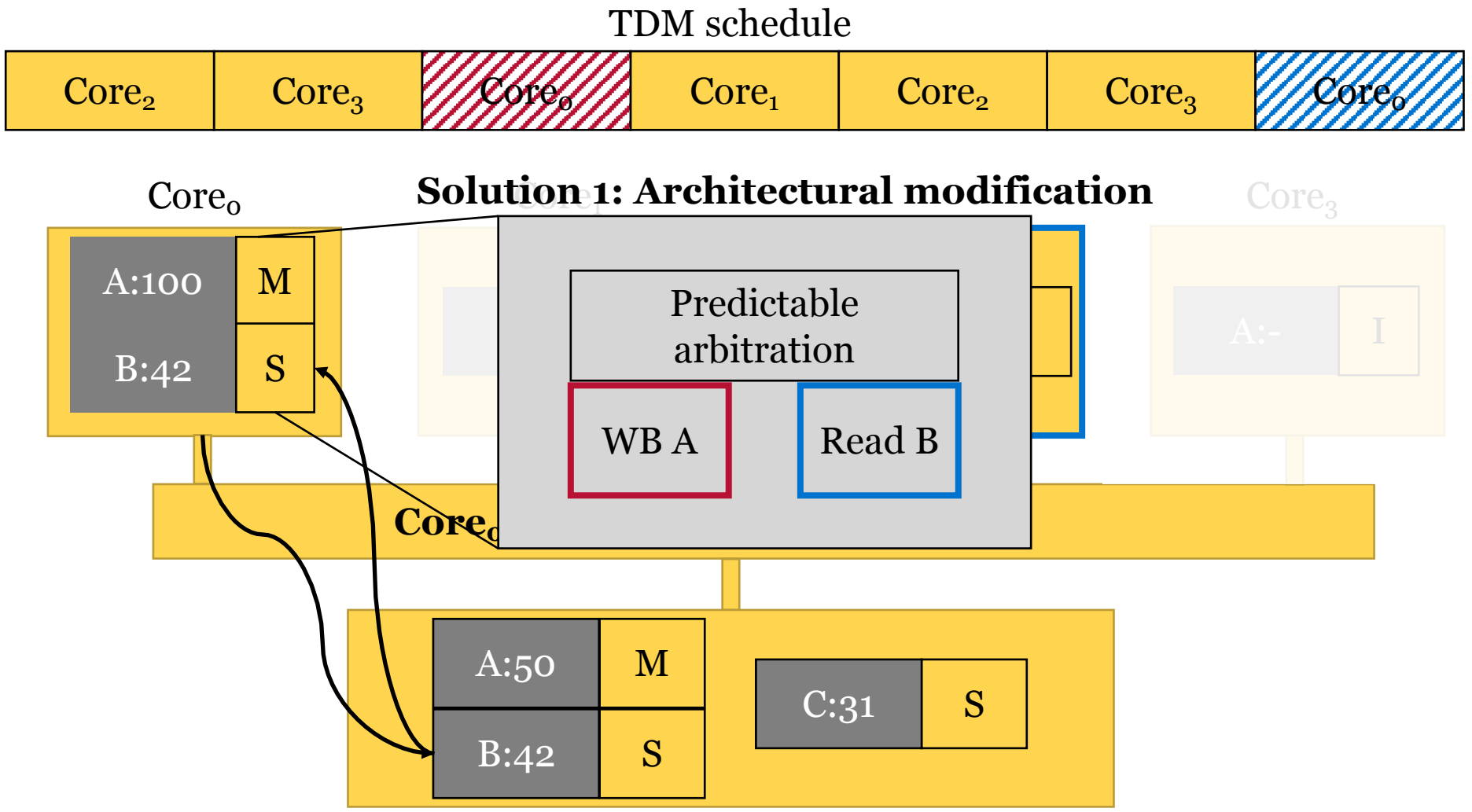
Intra-core coherence interference



Core	Event
$Core_2$	Read A
$Core_0$	Read B
$Core_2$	Waiting for data
$Core_0$	Read C

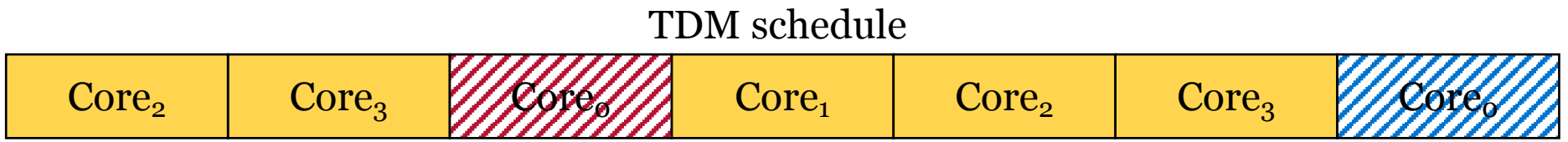
UNIVERSITY OF WATERLOO

Intra-core coherence interference

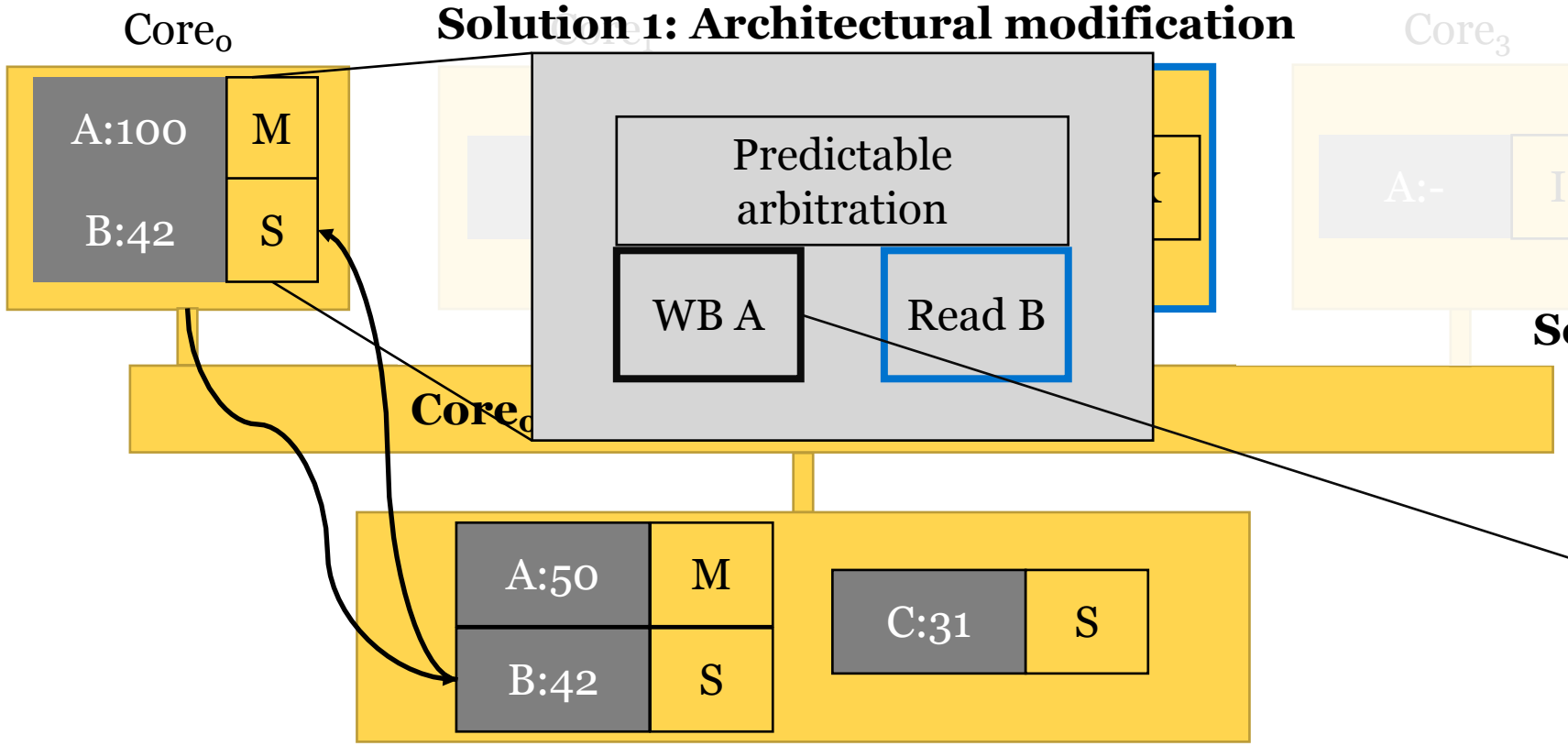


Core	Event
$Core_2$	Read A
$Core_0$	Read B
$Core_2$	Waiting for data
$Core_0$	Read C

Intra-core coherence interference



Core	Event
$Core_2$	Read A
$Core_0$	Read B
$Core_2$	Waiting for data
$Core_0$	Read C



Solution 2: Coherence protocol modifications

Need to maintain state to indicate $Core_0$'s pending WB of A

Sources of unpredictable memory access latencies due to cache coherence

1. Inter-core coherence interference on same cache line
   - ✓ Solution: Architectural modification + Coherence modifications
2. Inter-core coherence interference due to write hits
   - ✓ Solution: Architectural modification + Coherence modifications
3. Inter-core coherence interference due to write hits
   - ✓ Solution: Architectural modification to the core + Coherence modifications
4. Intra-core coherence interference
   - ✓ Solution: Architectural modification to the core + Coherence modifications

Architectural modification + Coherence modifications =

Predictable **M**odified-**S**hared-**I**nvalidate cache coherence protocol (PMSI)

- ✓ Predictable latencies for simultaneous shared data access
- ✓ No application/OS changes
- ✓ Significant performance benefits

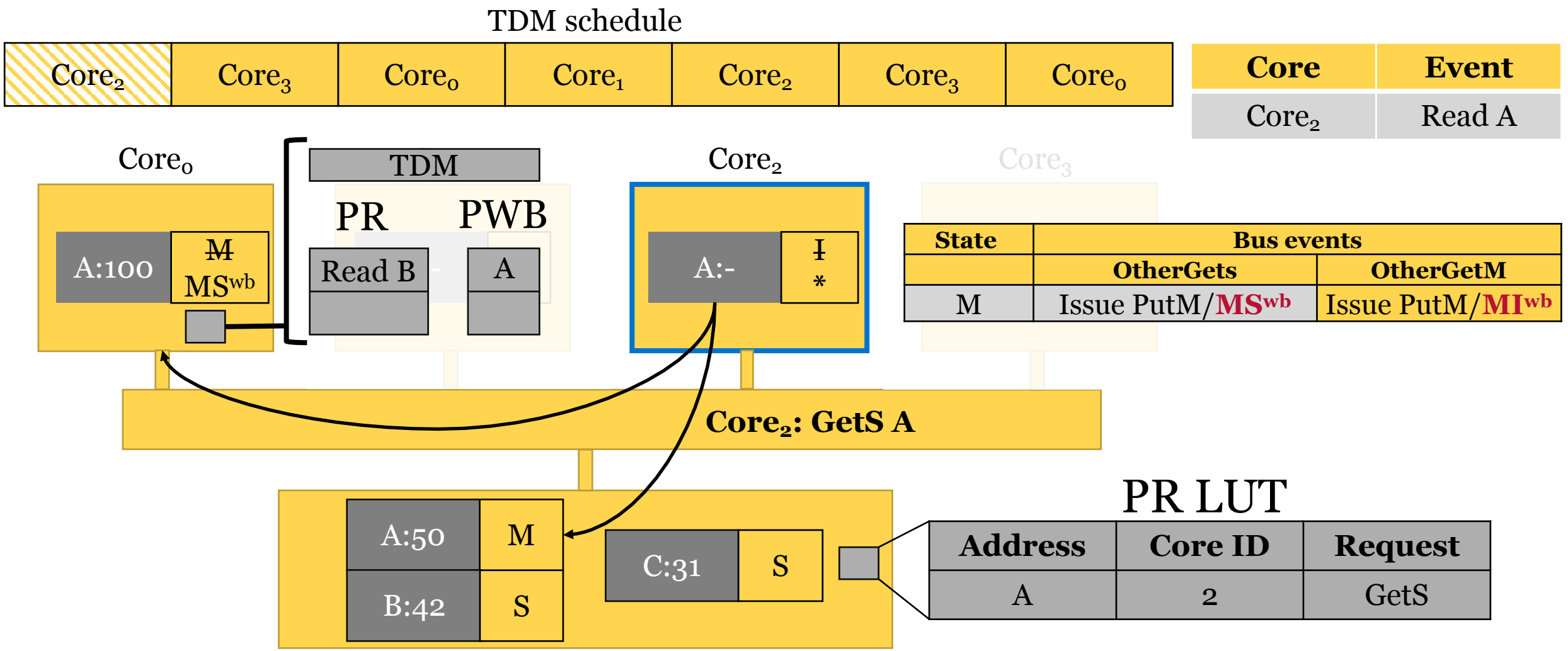
PMSI cache coherence protocol: Protocol modifications

State	Core events			Bus events						
	Load	Store	Replacement	OwnData	OwnUpg	OwnPutM	OtherGetS	OtherGetM	Other Upg	Other PutM
I	Issue GetS/IS^d	Issue GetM/IM^d	X	X	X	X	N/A	N/A	N/A	N/A
S	Hit	Issue Upg/**SM^w**	X	N/A	X	X	N/A	I	I	X
M	Hit	Hit	Issue PutM/**MI^{wb}**	X	X	X	Issue PutM/**MS^{wb}**	Issue PutM/**MI^{wb}**	X	X
IS^d	X	X	X	Read/S	X	X	N/A	IS^dI	IS^dI	N/A
IM^d	X	X	X	Write/S	X	X	IM^dS	IM^dI	X	N/A
IM^dI	X	X	X	Write/**MI^{wb}**	X	X	N/A	X	X	N/A
IS^dI	X	X	X	Read/I	X	X	N/A	X	X	N/A
IM^dS	X	X	X	Write/**MS^{wb}**	X	X	N/A	X	X	N/A
SM^w	**X**	**X**	**X**	**N/A**	**Store/ M**	**X**	**N/A**	**I**	**I**	**X**
MI^{wb}	**Hit**	**Hit**	**N/A**	**X**	**X**	**Send data/I**	**N/A**	**N/A**	**X**	**X**
MS^{wb}	**Hit**	**Hit**	**MI^{wb}**	**X**	**X**	**Send data/S**	**N/A**	**MI^{wb}**	**X**	**X**

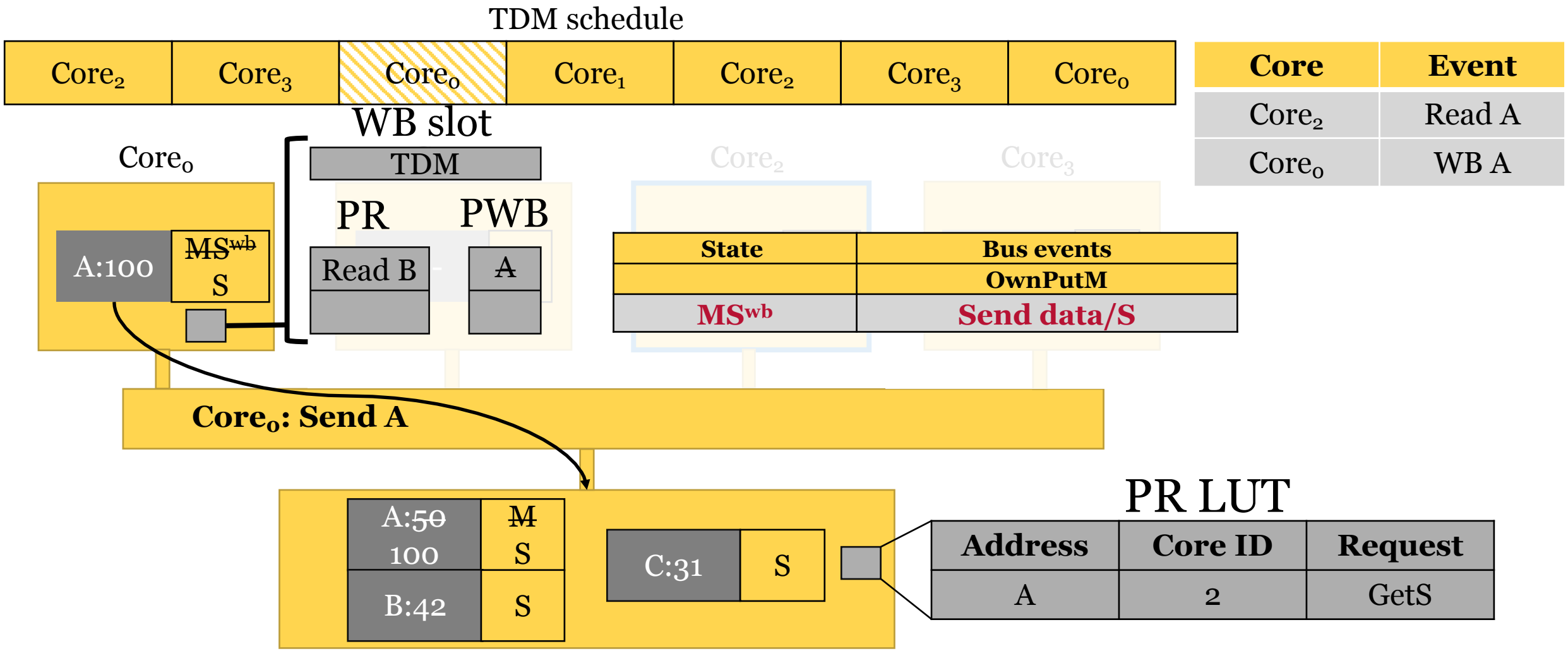
PMSI cache coherence protocol: Protocol modifications

State	Core events			Bus events						
	Load	Store	Replacement	OwnData	OwnUpg	OwnPutM	OtherGetS	OtherGetM	Other Upg	Other PutM
I	Issue GetS/IS^d	Issue GetM/IM^d	X	X	X	X	N/A	N/A	N/A	N/A
S	Hit	Issue Upg/**SM^w**	X	N/A	X	X	N/A	I	I	X
M	Hit	Hit	Issue PutM/**MI^{wb}**	X	X	X	Issue PutM/**MS^{wb}**	Issue PutM/**MI^{wb}**	X	X
IS^d	X	X	X	Read/S	X	X	N/A	IS^dI	IS^dI	N/A
IM^d	X	X	X	Write/S	X	X	IM^dS	IM^dI	X	N/A
IM^dI	X	X	X	Write/**MI^{wb}**	X	X	N/A	X	X	N/A
IS^dI	X	X	X	Read/I	X	X	N/A	X	X	N/A
IM^dS	X	X	X	Write/**MS^{wb}**	X	X	N/A	X	X	N/A
SM^w	**X**	**X**	**X**	**N/A**	**Store/ M**	**X**	**N/A**	**I**	**I**	**X**
MI^{wb}	**Hit**	**Hit**	**N/A**	**X**	**X**	**Send data/I**	**N/A**	**N/A**	**X**	**X**
MS^{wb}	**Hit**	**Hit**	**MI^{wb}**	**X**	**X**	**Send data/S**	**N/A**	**MI^{wb}**	**X**	**X**

Intra-core coherence interference

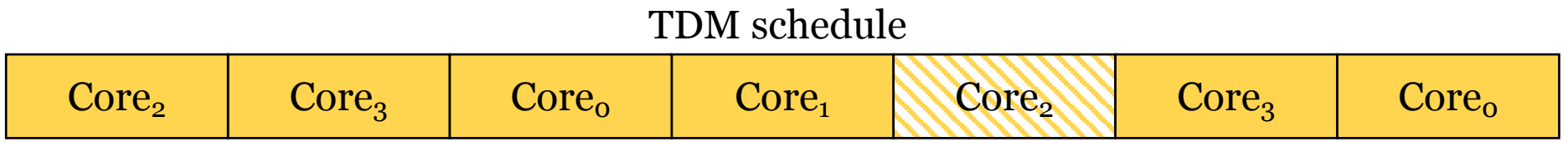


Intra-core coherence interference

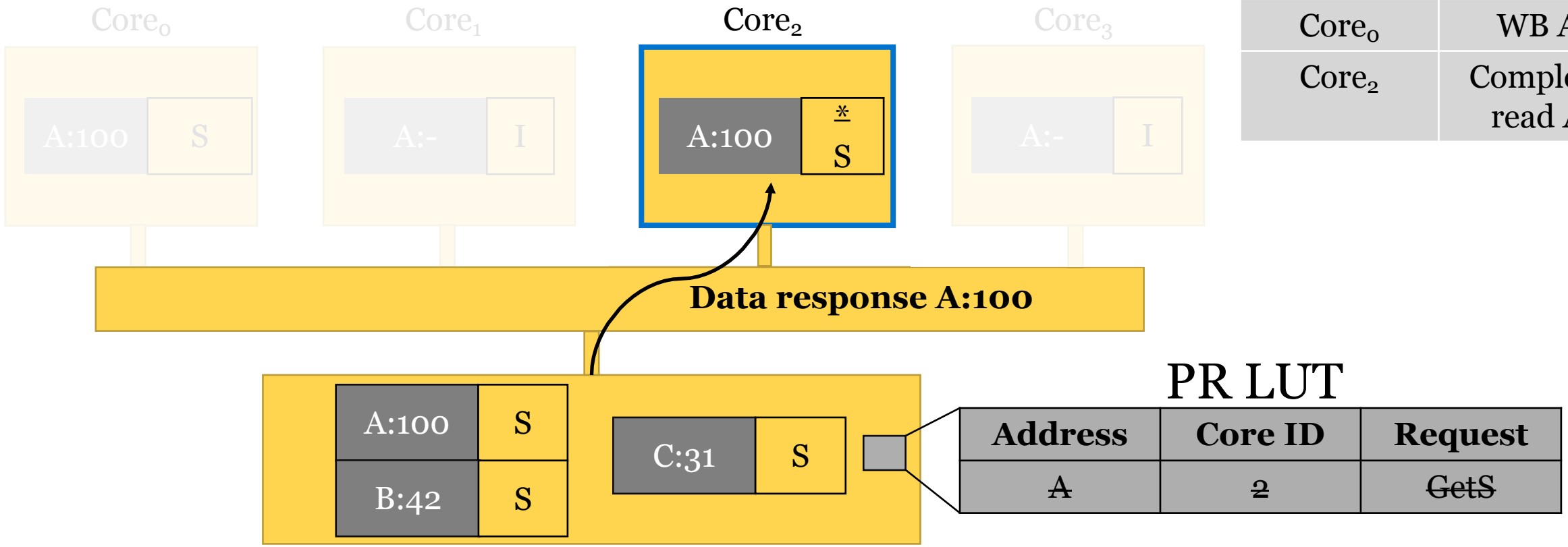


Core	Event
$Core_2$	Read A
$Core_0$	WB A

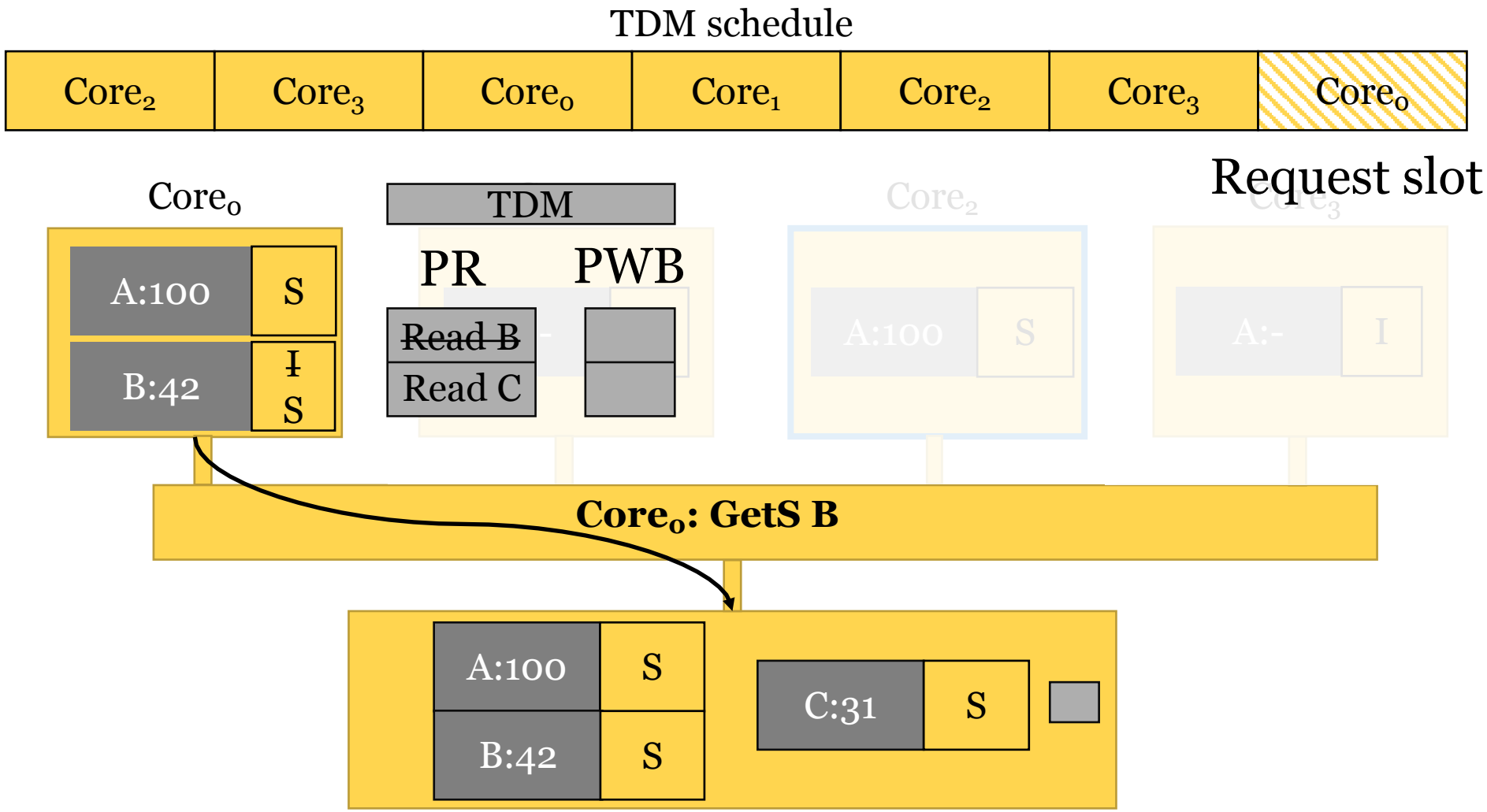
Intra-core coherence interference



Core	Event
$Core_2$	Read A
$Core_0$	WB A
$Core_2$	Complete read A



Intra-core coherence interference

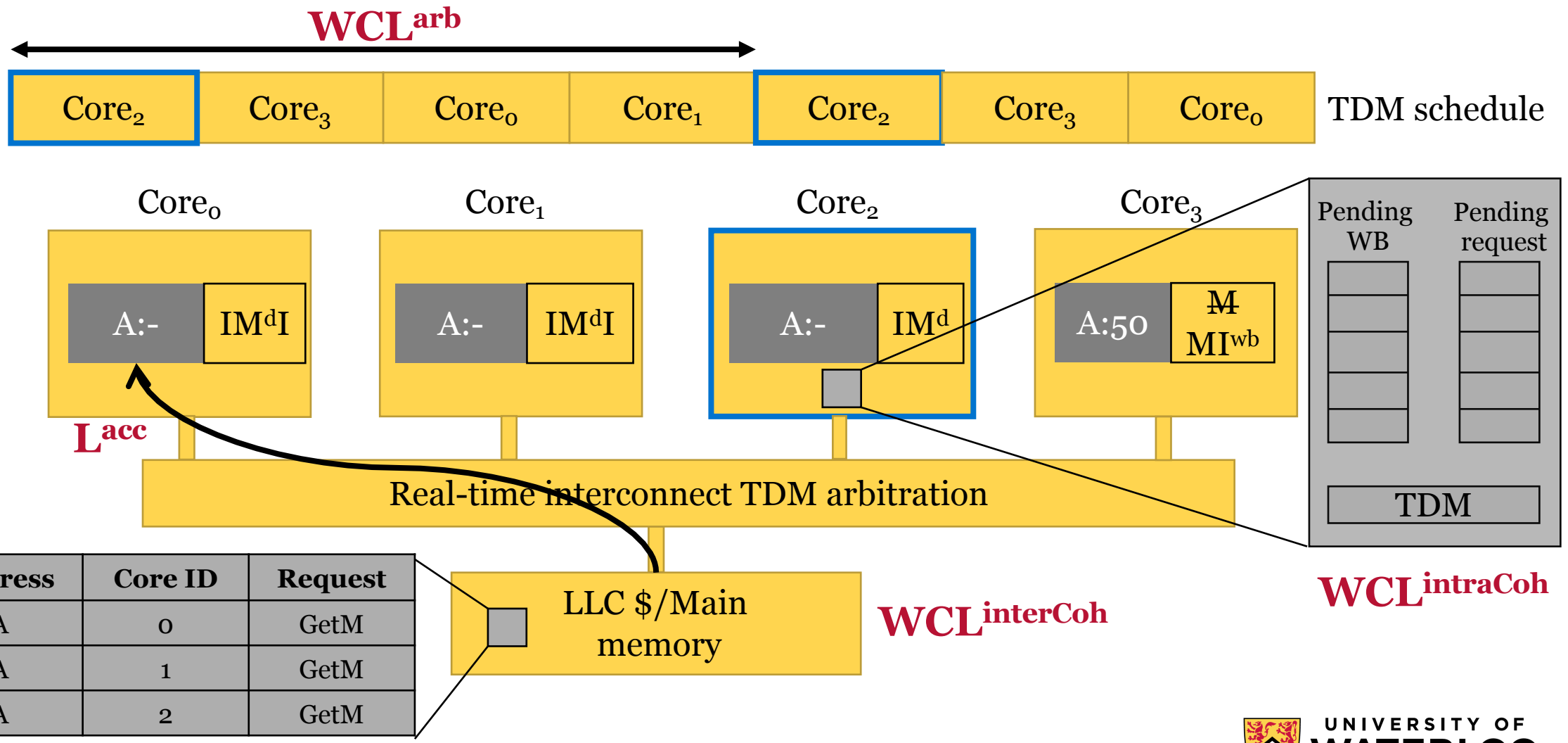


Core	Event
$Core_2$	Read A
$Core_0$	WB A
$Core_2$	Complete read A
$Core_0$	Read B

UNIVERSITY OF WATERLOO

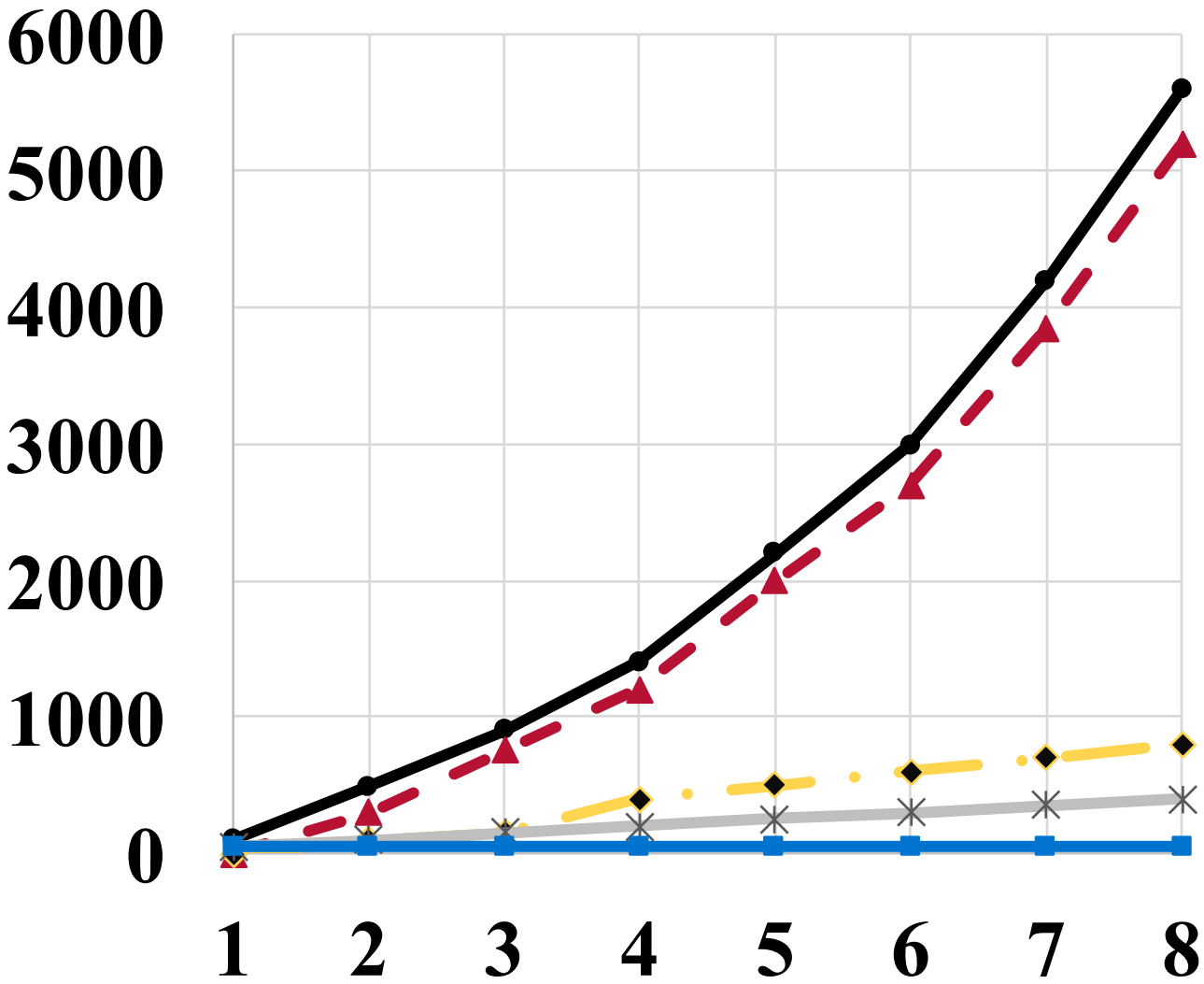
Latency analysis of PMSI

N: Number of cores



Address	Core ID	Request
A	0	GetM
A	1	GetM
A	2	GetM

Latency analysis of PMSI



WCL of memory request

=

L^{acc} +

$WCL^{arb}(\alpha N)$ +

$WCL^{intraCoh}(\alpha N)$ +

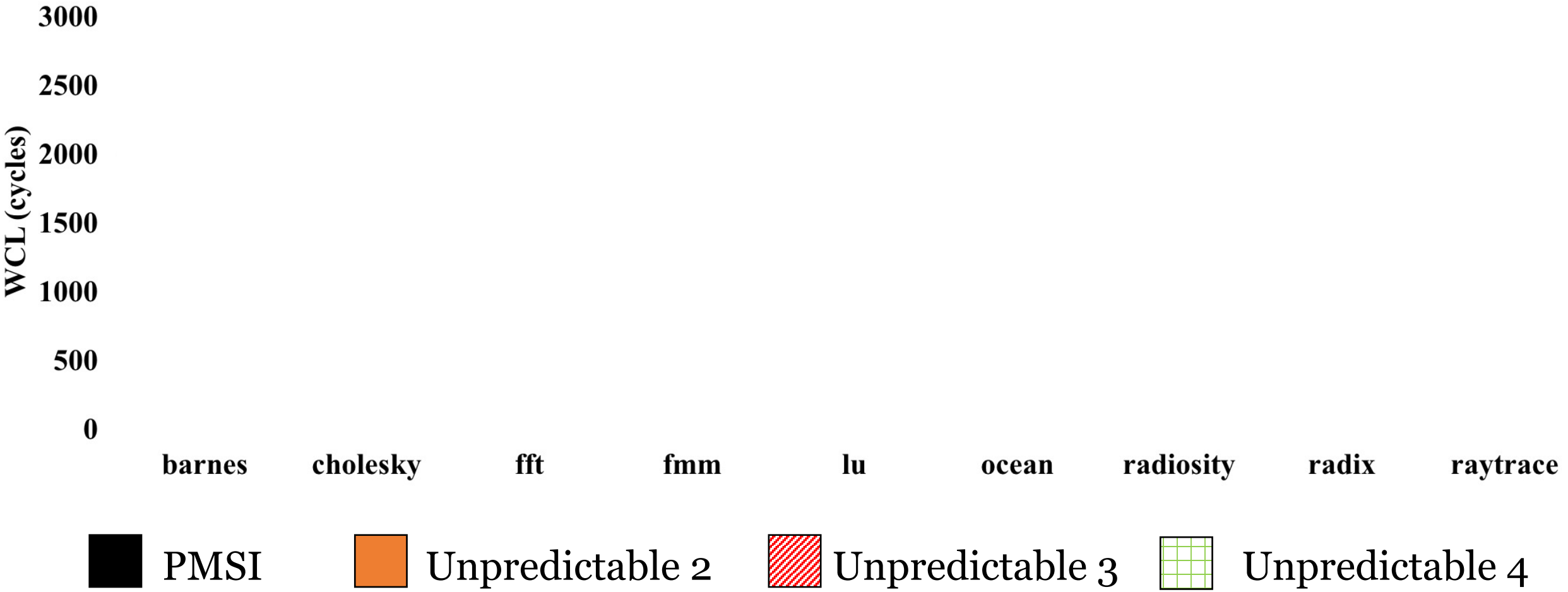
$WCL^{interCoh}(\alpha N^2)$

UNIVERSITY OF WATERLOO

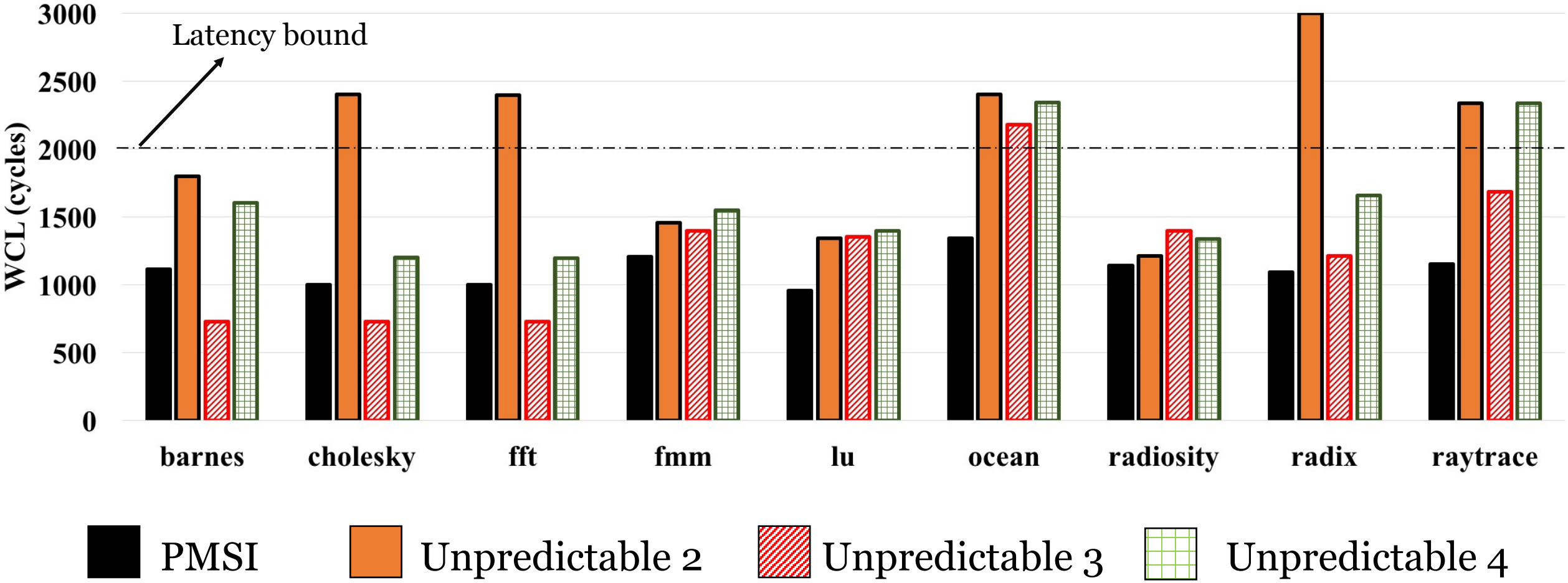
Results

- Cycle accurate Gem5 simulator
- 2, 4, 8 cores, 16KB direct mapped private L1-D/I caches
- In-order cores, 2GHz operating frequency
- TDM bus arbitration with slot width = 50 cycles
- Benchmarks: SPLASH-2, synthetic benchmarks to stress worst-case scenarios
- Simulation framework available at https://git.uwaterloo.ca/caesr-pub/pmsi

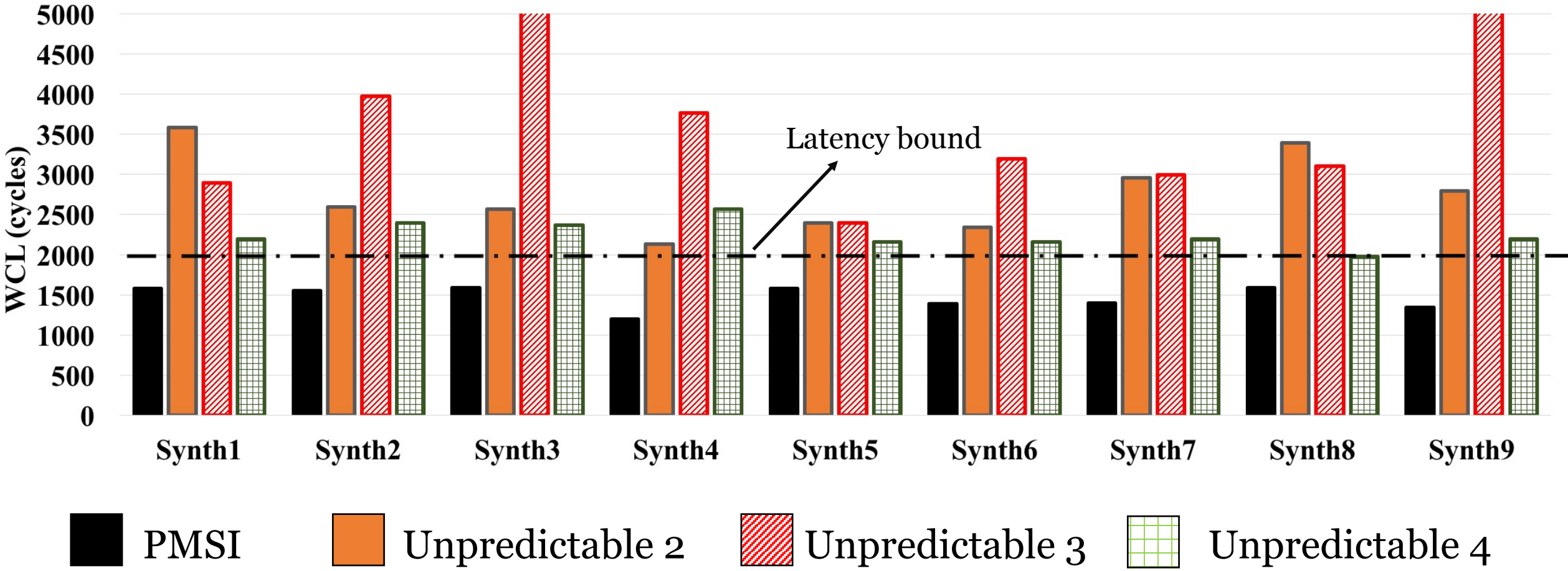
Results: Observed worst case latency per request



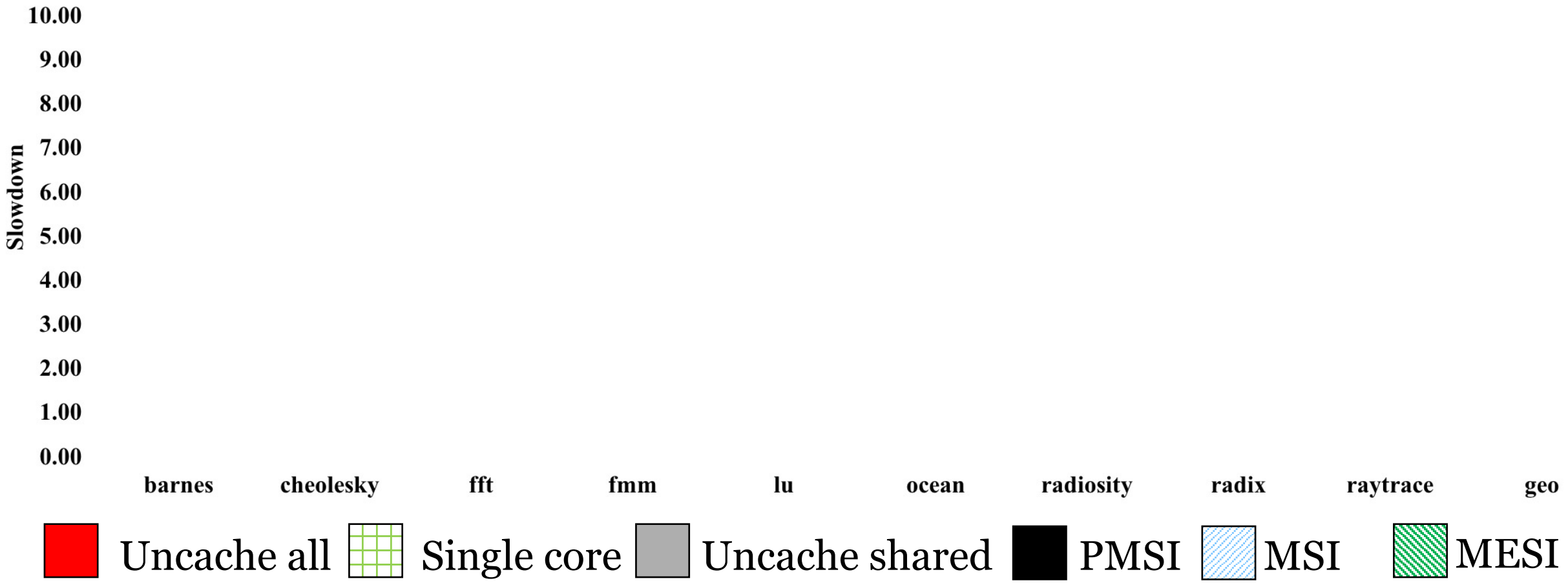
Results: Observed worst case latency per request



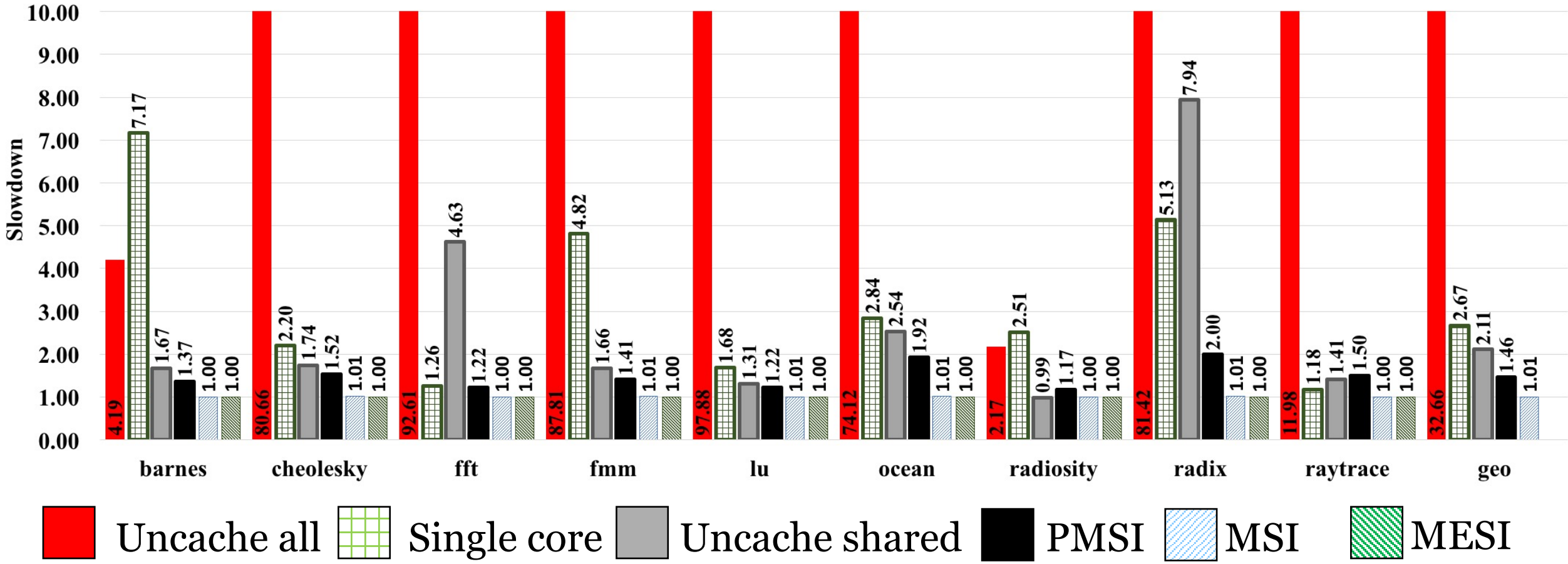
Results: Observed worst case latency per request



Results: Slowdown relative to MESI cache coherence



Results: Slowdown relative to MESI cache coherence



Conclusion

- Enumerate sources of unpredictability when applying conventional hardware cache coherence on multi-core real-time systems
- Propose PMSI cache coherence that allows **predictable simultaneous shared data accesses with no changes to application/OS**
- Hardware overhead of PMSI is ~ **128 bytes**
- WCL of memory access with PMSI α square of number of cores (α N^2)
- **PMSI outperforms prior techniques for handling shared data accesses (upto 4x improvement over uncache-shared)**, and suffers on **average 46% slowdown compared to conventional MESI cache coherence protocol**

UNIVERSITY OF WATERLOO

THANK YOU & QUESTIONS?