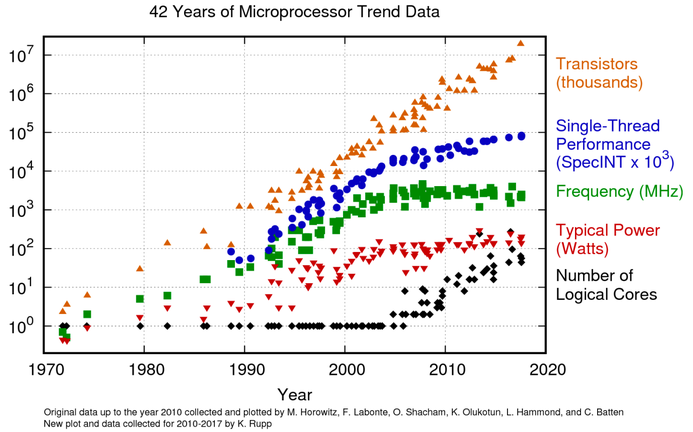
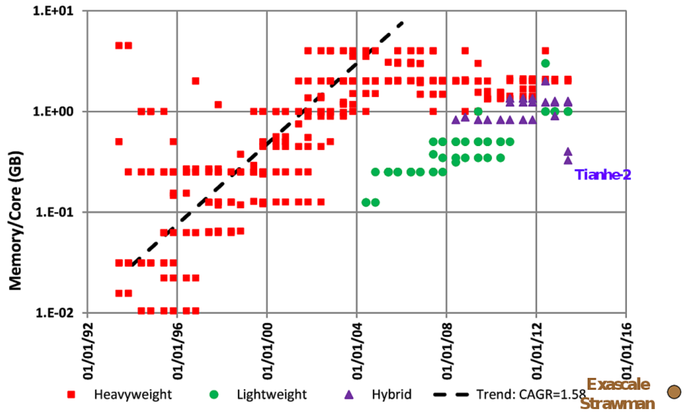
ROHIT ZAMBRE, PH.D. CANDIDATE, UC IRVINE

THE COMING OF AGE OF MULTITHREADED HIGH-PERFORMANCE COMMUNICATION

- *Trend 1*: Disproportionate increase in number of cores per processor.



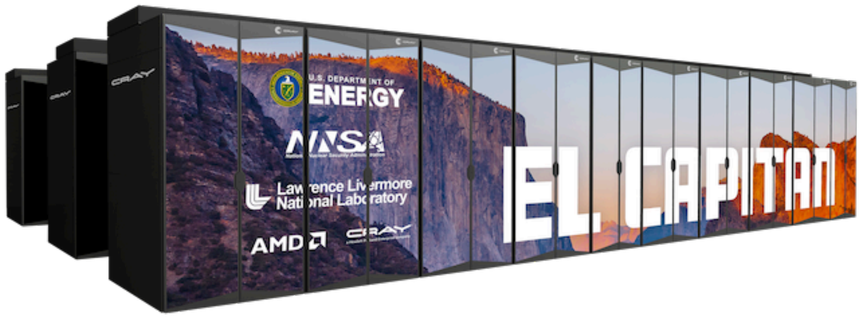


Increasing number of cores

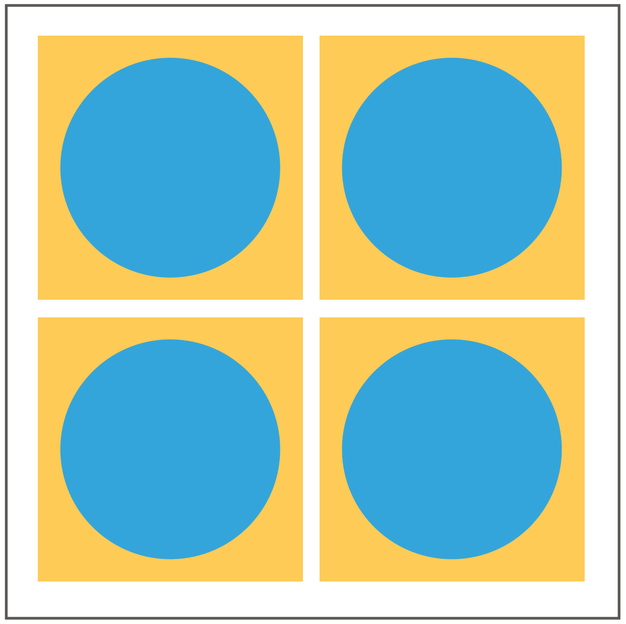
Decreasing memory per core

Fugaku node: 48 cores, 32GB RAM

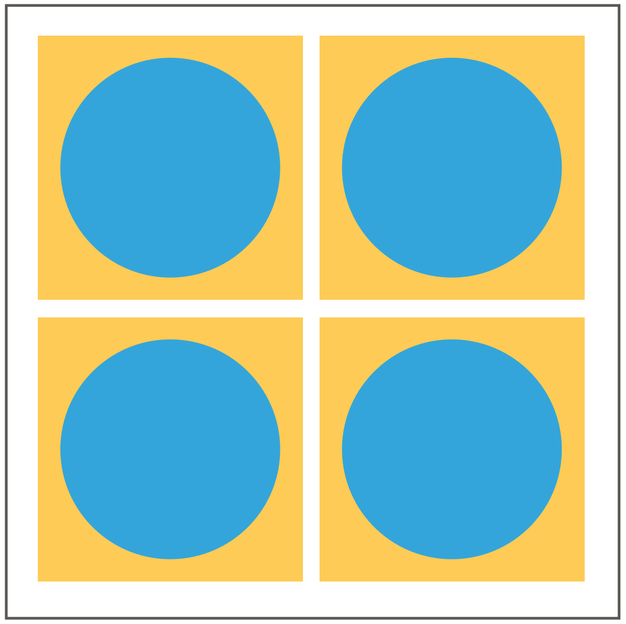
- *Trend 2*: Increase in GPUs per node



MPI everywhere

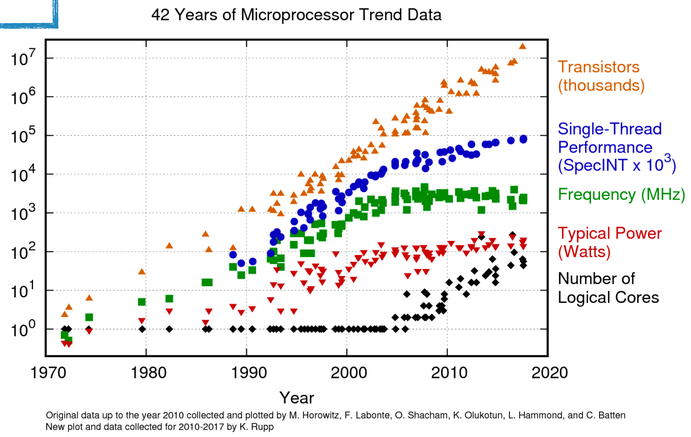


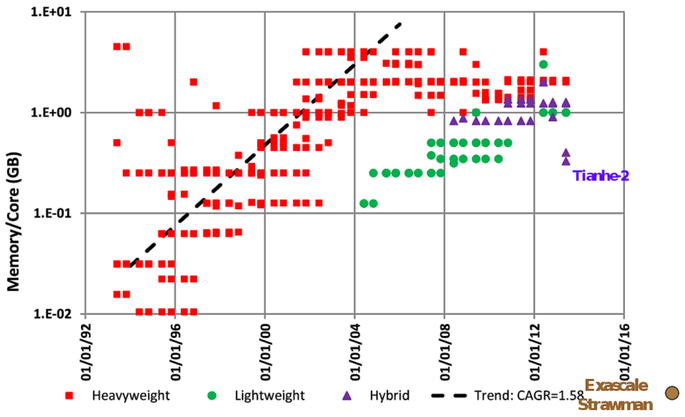
MPI everywhere



- *Model artifact*: high memory requirements that worsen with increase domain-dimensionality and number of ranks.
- *Hardware usage*: resource wastage with static split of limited resources on processor

Trend 1



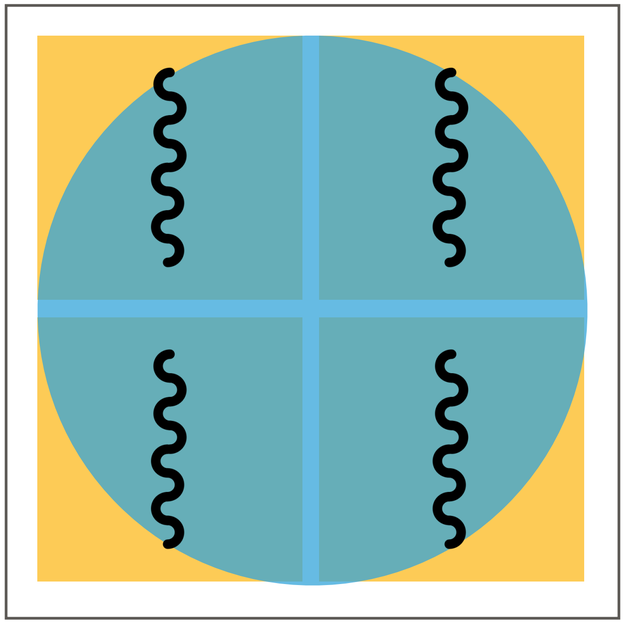


Increasing number of cores

Decreasing memory per core

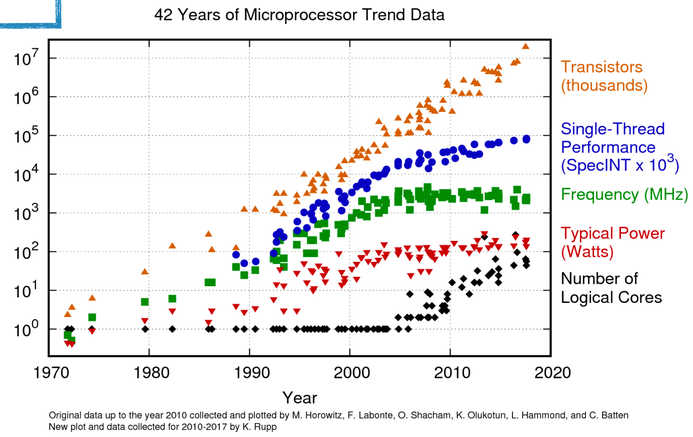
Node Core Process

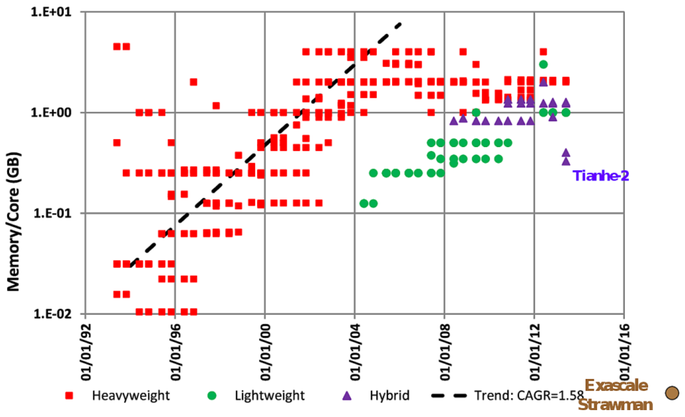
MPI+threads



- *Model artifact*: reduces duplicated data by a factor of number of threads.
- *Hardware usage*: able to use the many cores while sharing all of processor's resources.

Trend 1





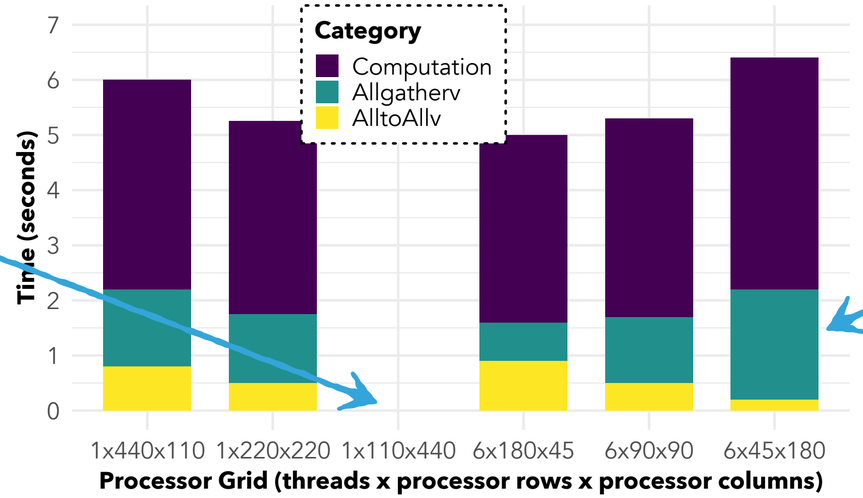
Node Core Process Thread

Increasing number of cores

Decreasing memory per core

MPI everywhere

OOM!

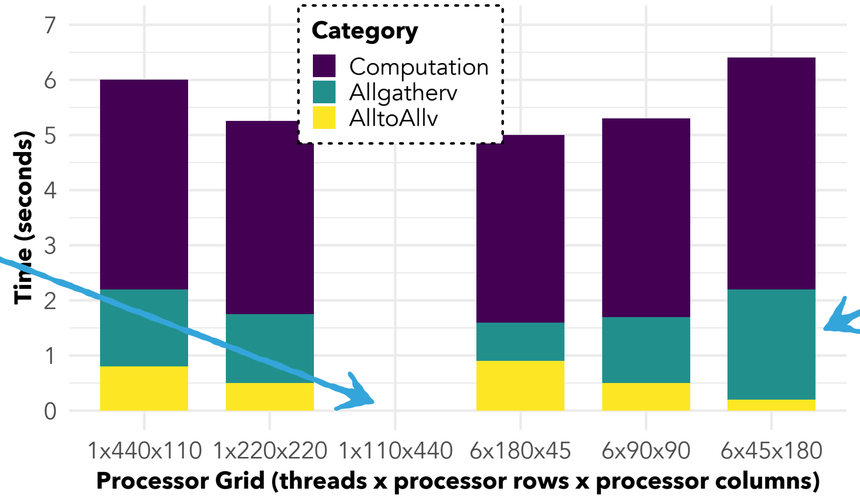


Corresponding MPI+threads runs

Buluc et al. Distributed BFS (https://arxiv.org/abs/1705.04590)

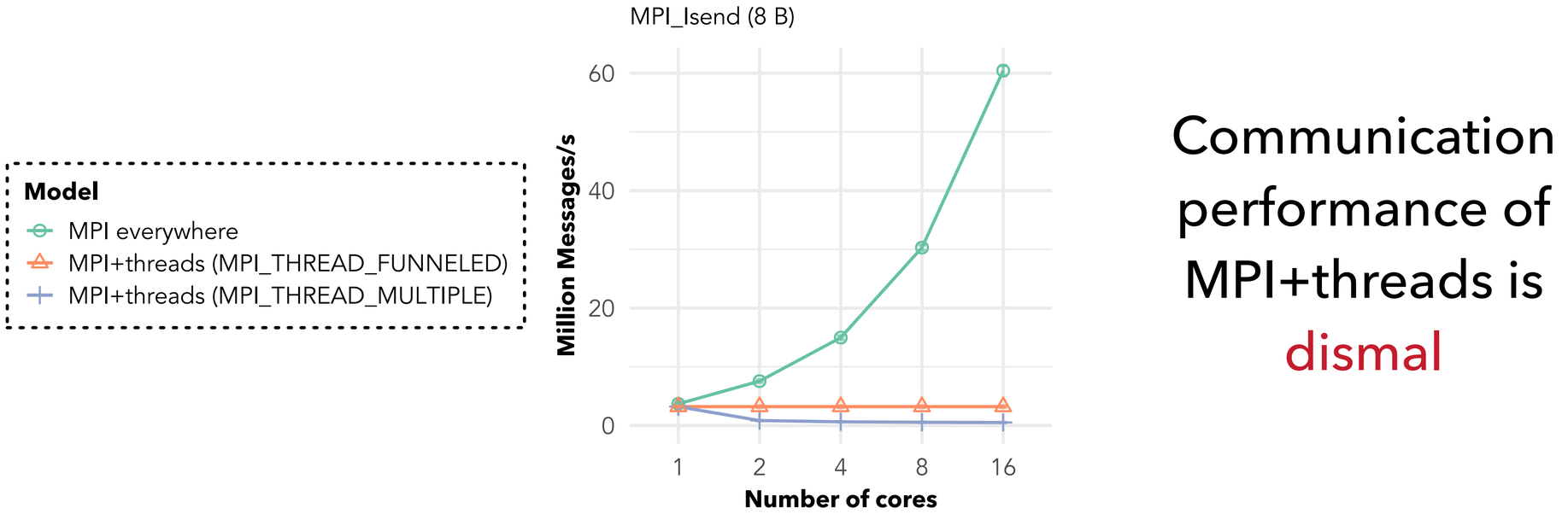
MPI everywhere

OOM!



Corresponding MPI+threads runs

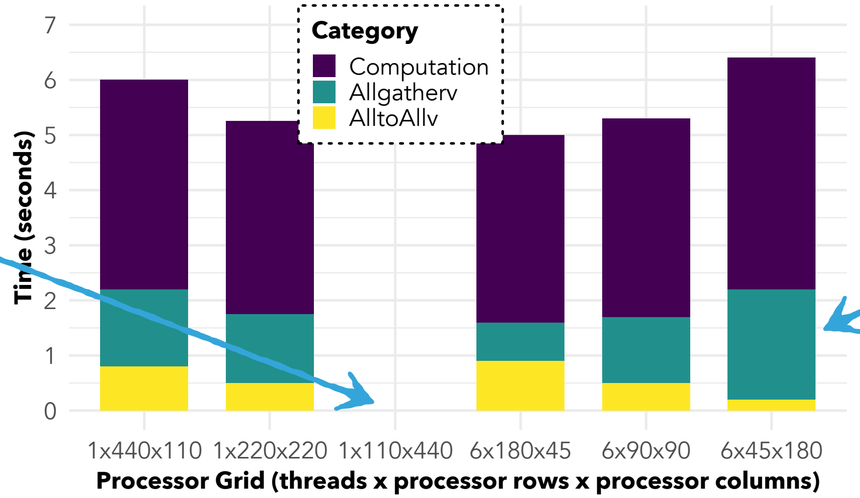
Buluc et al. Distributed BFS (https://arxiv.org/abs/1705.04590)



Communication performance of MPI+threads is dismal

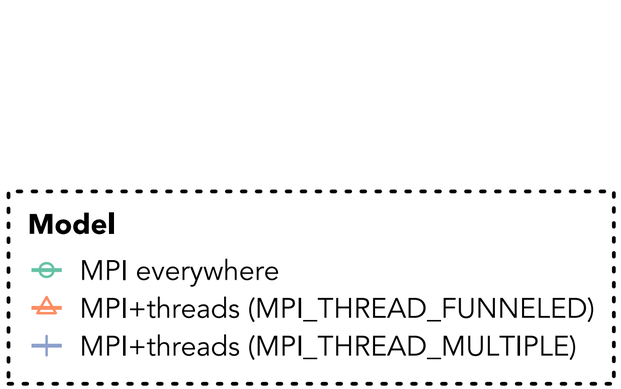
MPI everywhere

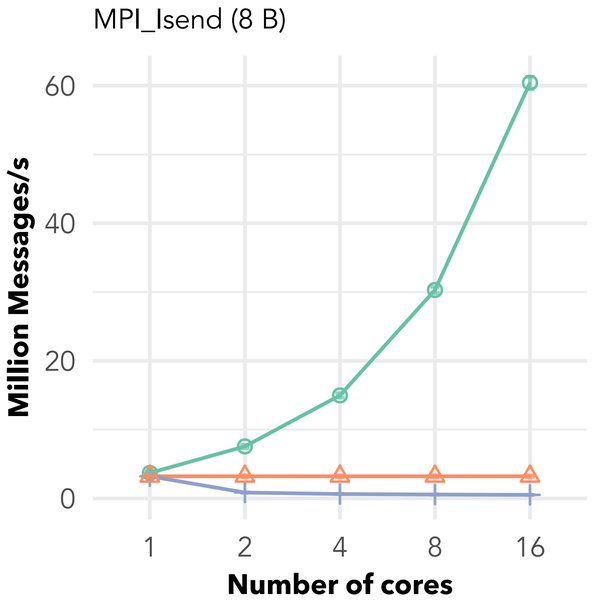
OOM!



Corresponding MPI+threads runs

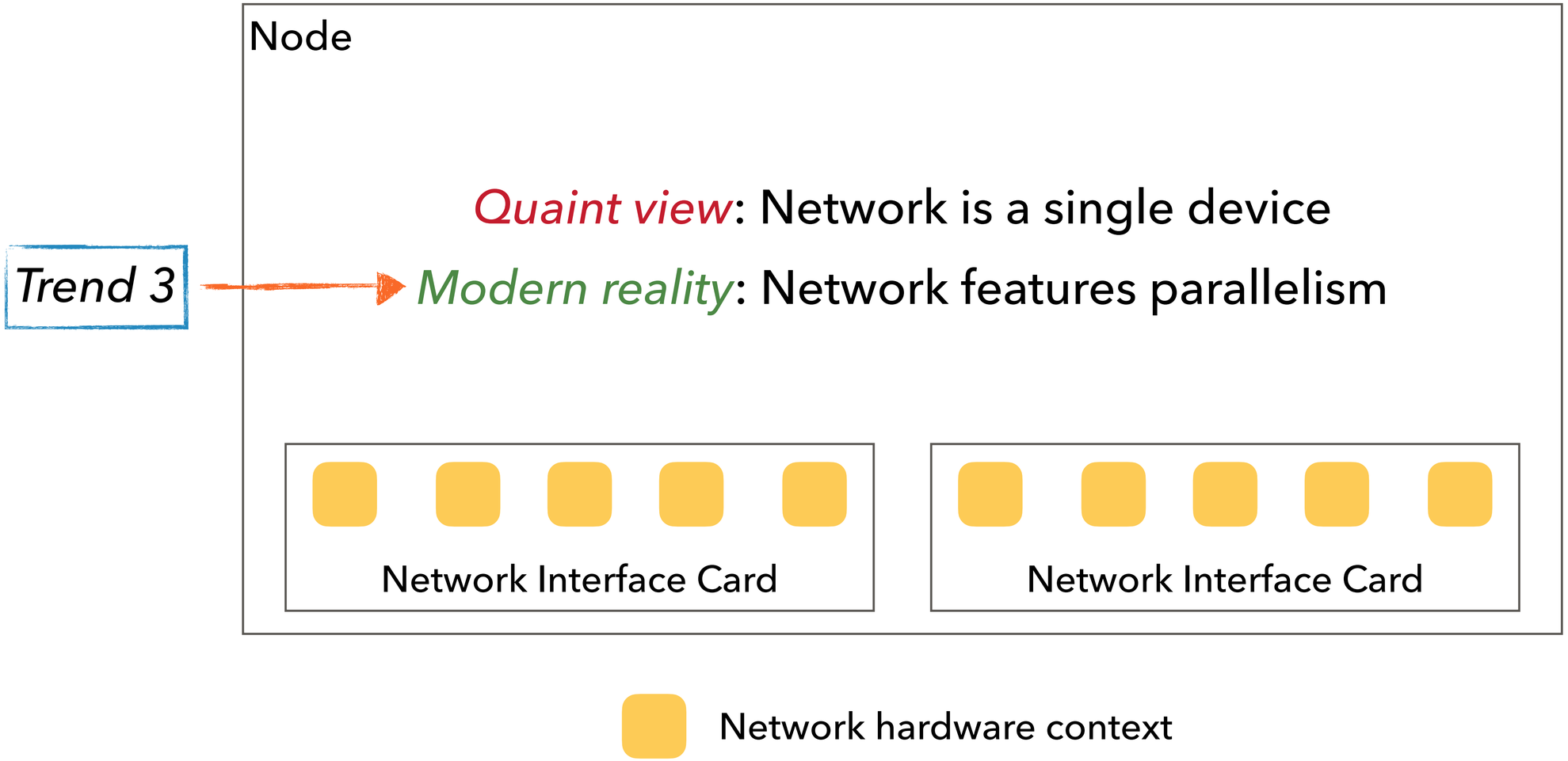
Buluc et al. Distributed BFS (https://arxiv.org/abs/1705.04590)

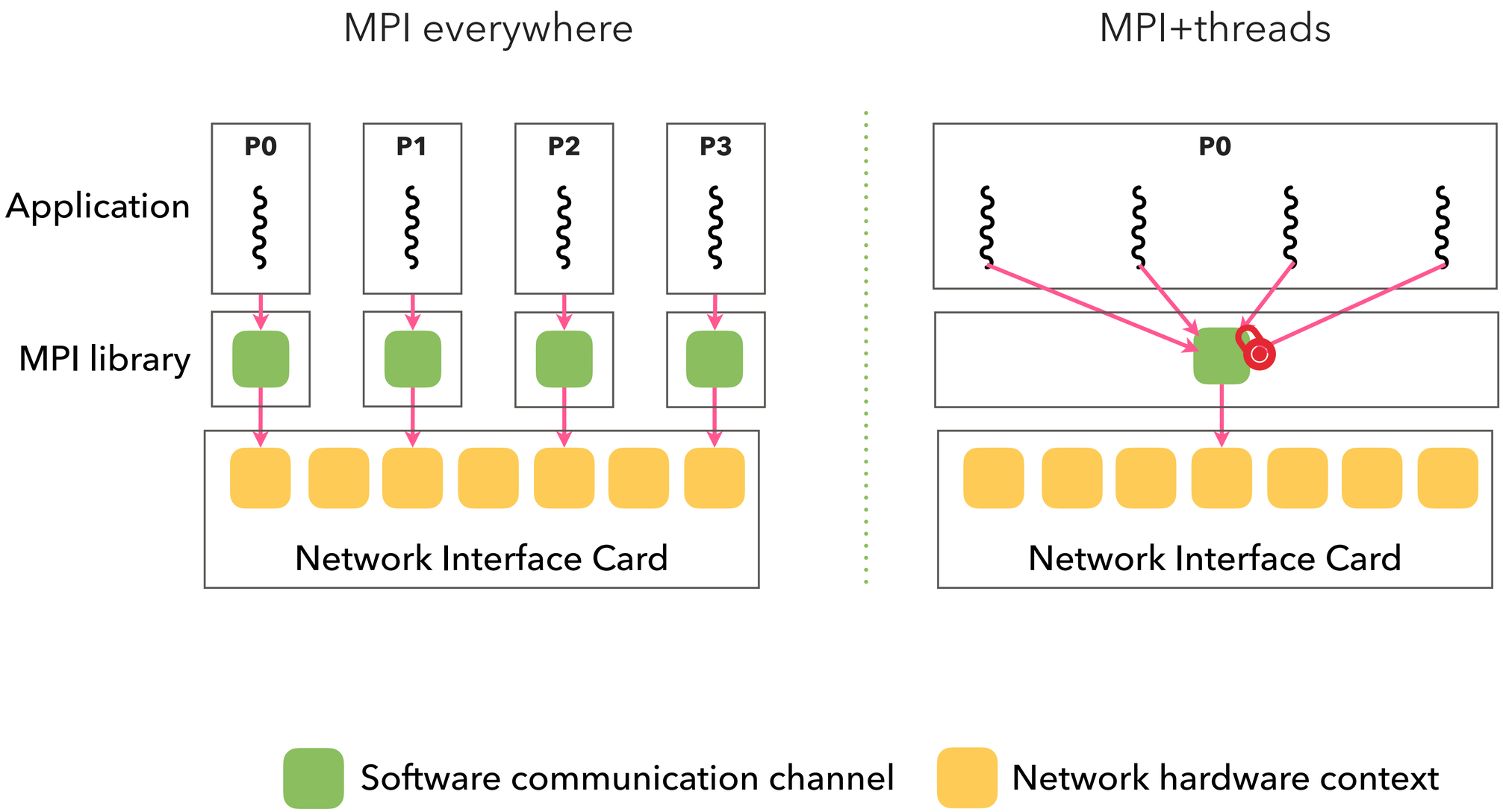


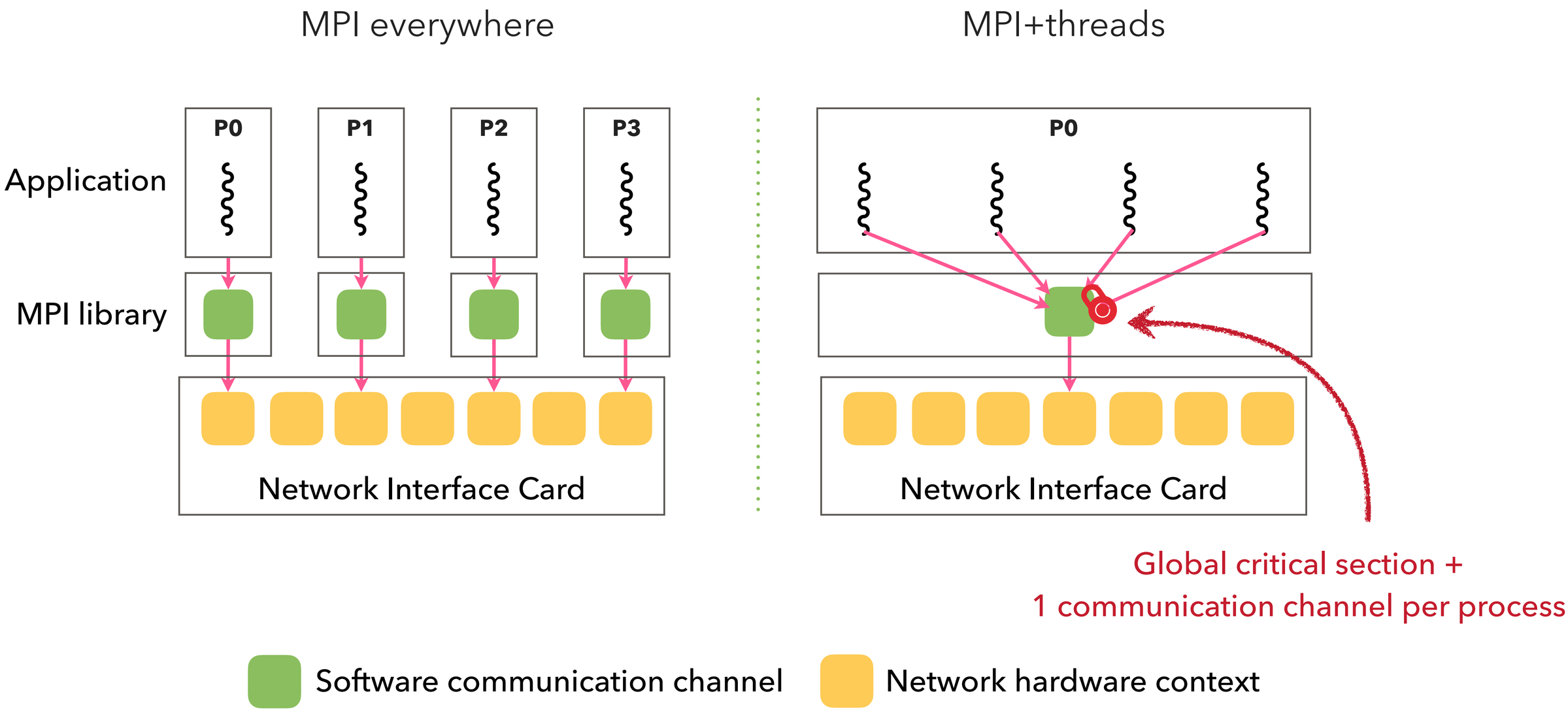


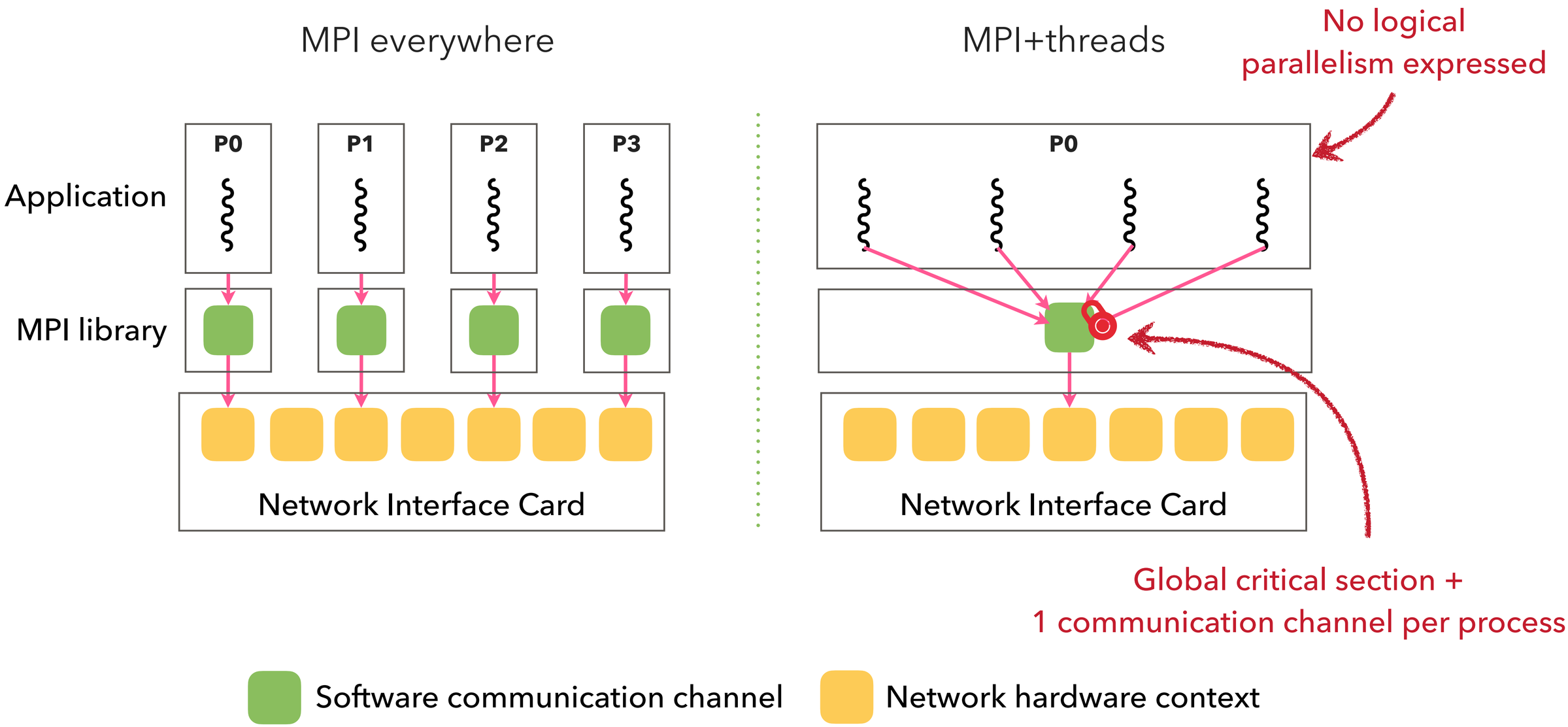
Communication performance of MPI+threads is dismal

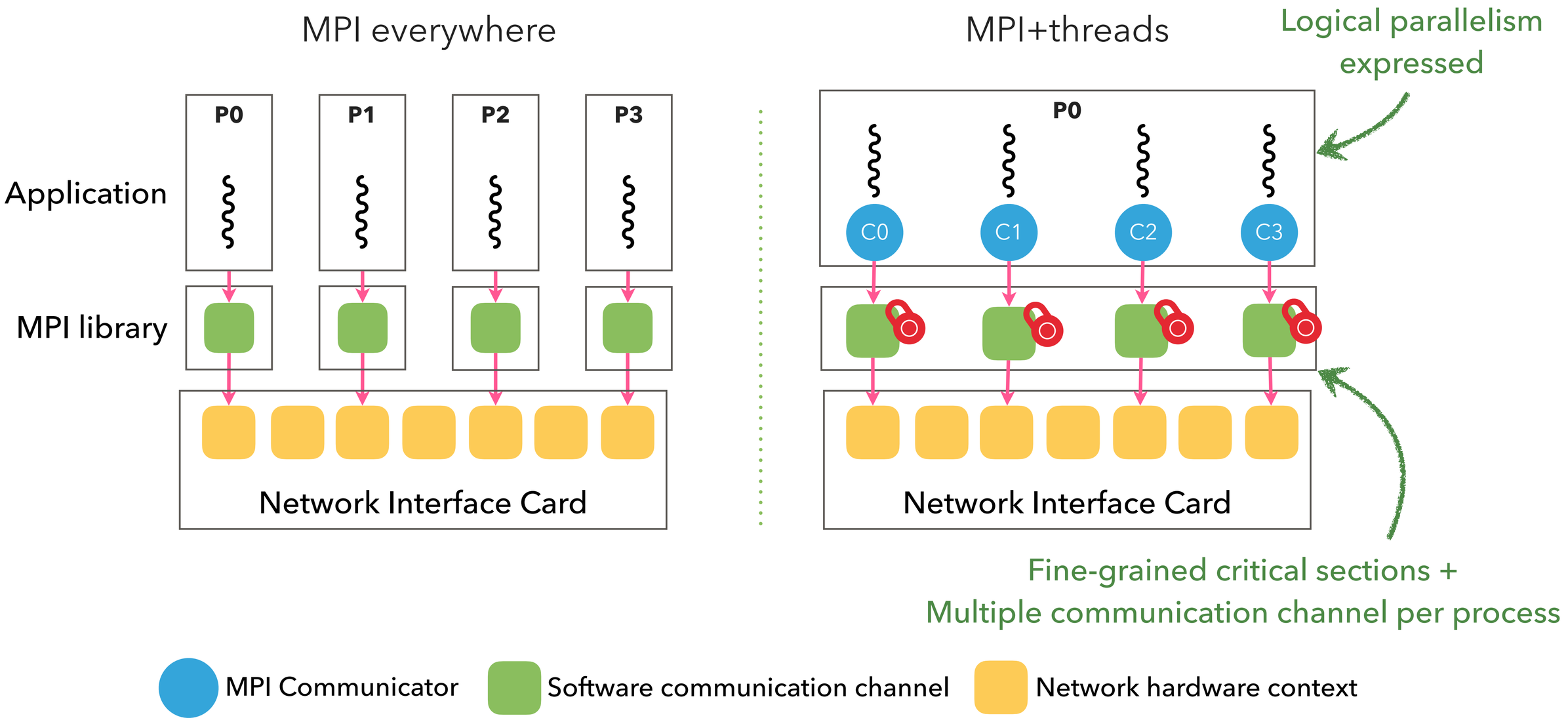
Problematic even for *Trend 2*

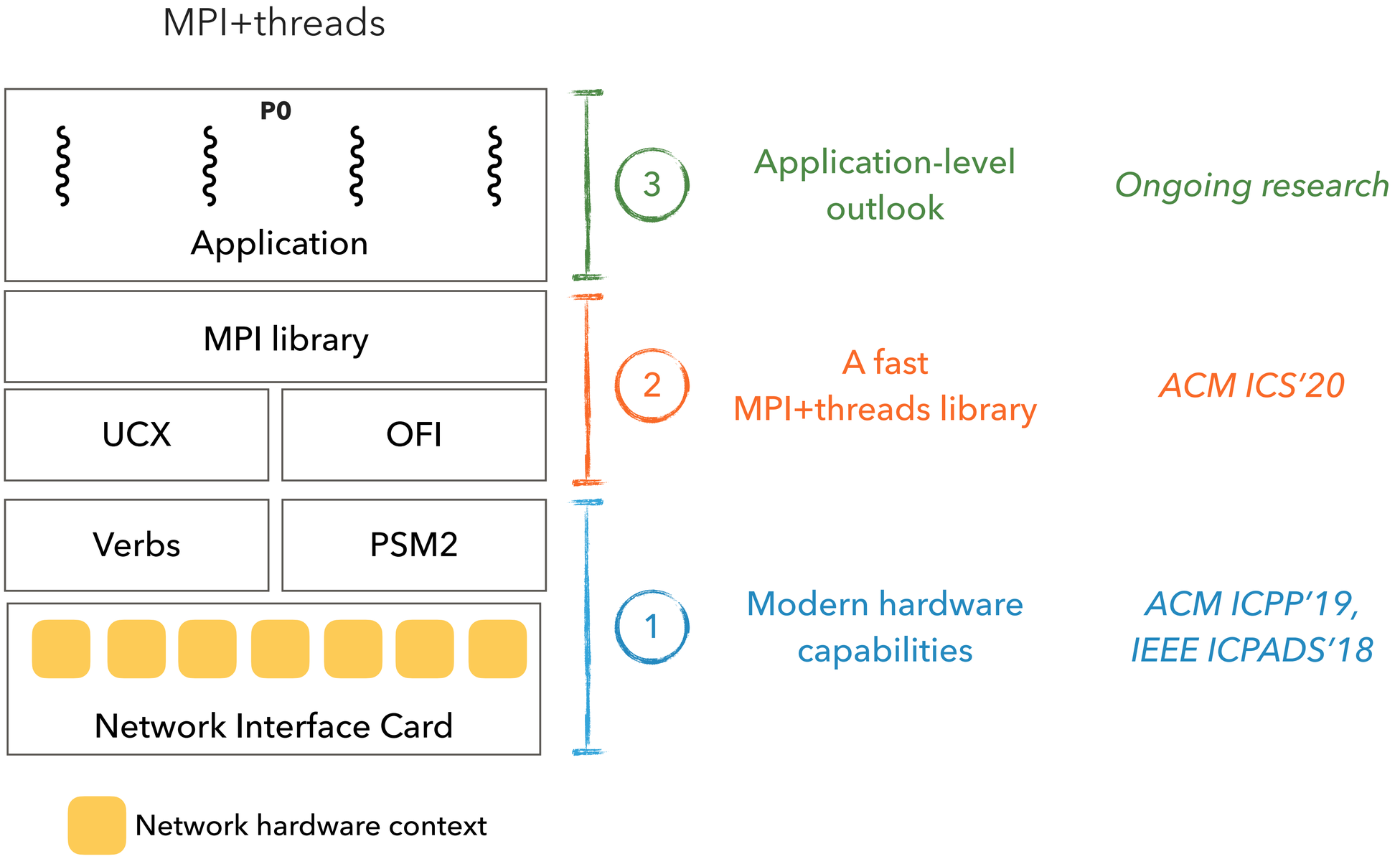

Node
Quaint view: Network is a single device
Trend 3
Modern reality: Network features parallelism
Network Interface Card
Network Interface Card
Network hardware context

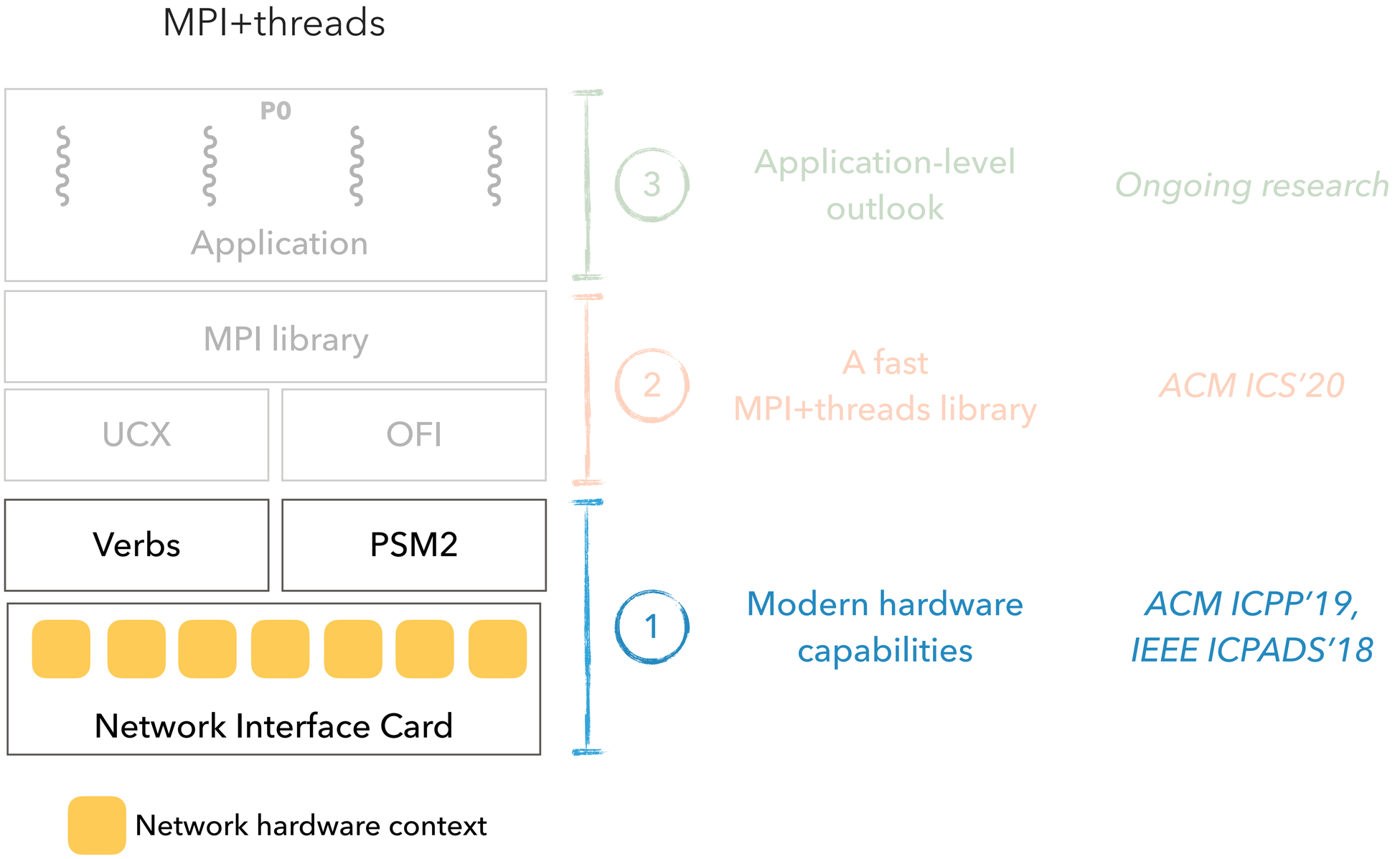

MPI everywhere
MPI+threads
Application
MPI library
P0
P1
P2
P3
Network Interface Card
P0
Network Interface Card
Software communication channel
Network hardware context


MPI everywhere
MPI+threads
Application
MPI library
P0
P1
P2
P3
Network Interface Card
P0
Network Interface Card
Global critical section +
1 communication channel per process
Software communication channel
Network hardware context


MPI everywhere
MPI+threads
No logical parallelism expressed
Application
MPI library
P0
P1
P2
P3
P0
Network Interface Card
Network Interface Card
Global critical section +
1 communication channel per process
Software communication channel
Network hardware context

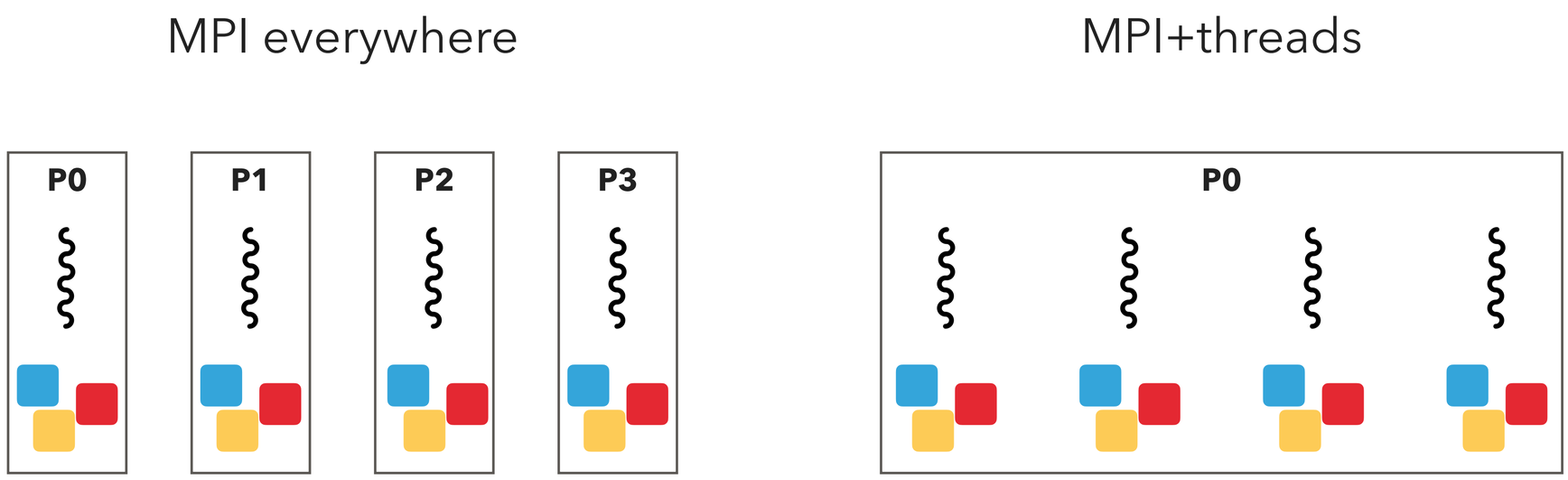

MPI everywhere
MPI+threads
Logical parallelism expressed
Application
MPI library
P0
P1
P2
P3
C0
C1
C2
C3
Network Interface Card
Network Interface Card
Fine-grained critical sections +
Multiple communication channel per process
MPI Communicator
Software communication channel
Network hardware context


MPI+threads
P0
Application
MPI library
UCX
OFI
Verbs
PSM2
Network Interface Card
Network hardware context
3
Application-level outlook
Ongoing research
2
A fast MPI+threads library
ACM ICS'20
1
Modern hardware capabilities
ACM ICPP'19, IEEE ICPADS'18


MPI+threads
P0
Application
MPI library
UCX
OFI
Verbs
PSM2
Network Interface Card
Network hardware context
3
Application-level outlook
Ongoing research
2
A fast MPI+threads library
ACM ICS'20
1
Modern hardware capabilities
ACM ICPP'19, IEEE ICPADS'18

LIMIT STUDY FOR MULTITHREADED COMMUNICATION

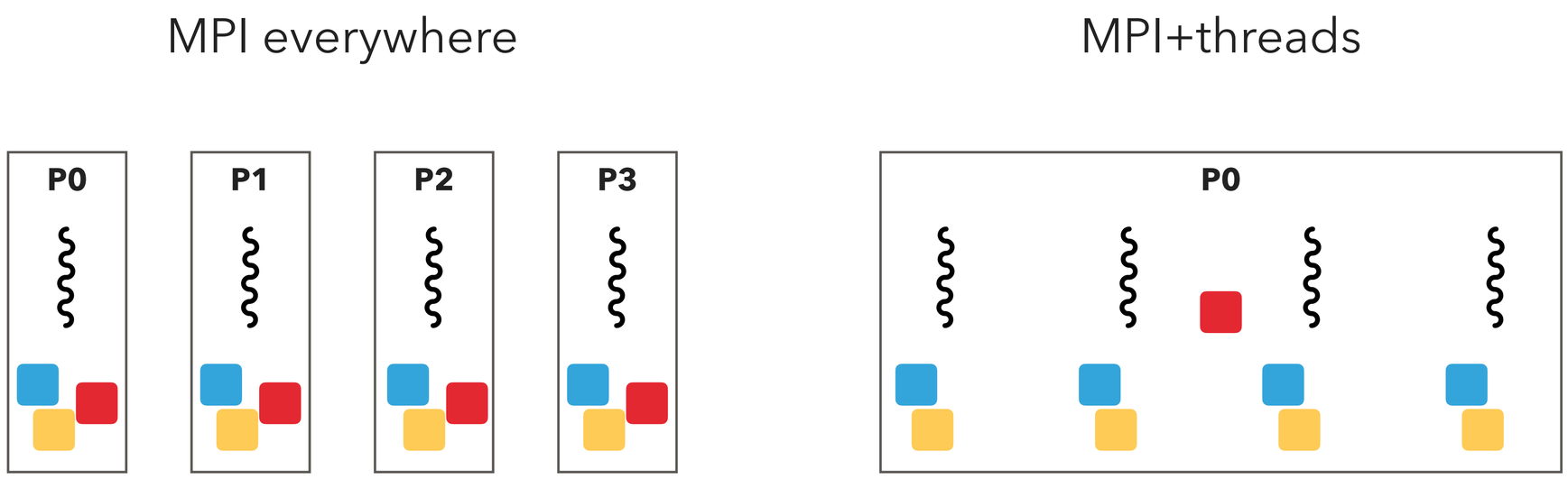
- Multithreaded environments feature larger design space



Communication resources: TX Queues, RX Queues, Completion Queues, Network registers

LIMIT STUDY FOR MULTITHREADED COMMUNICATION

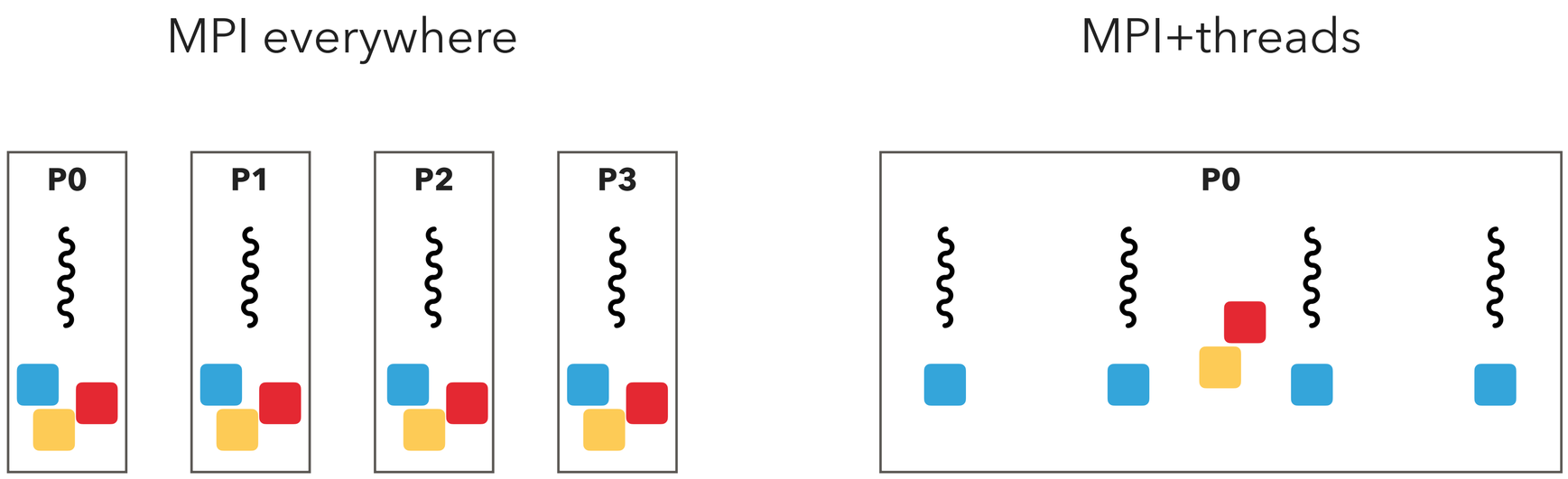
- Multithreaded environments feature larger design space



Communication resources: TX Queues, RX Queues, Completion Queues, Network registers

LIMIT STUDY FOR MULTITHREADED COMMUNICATION

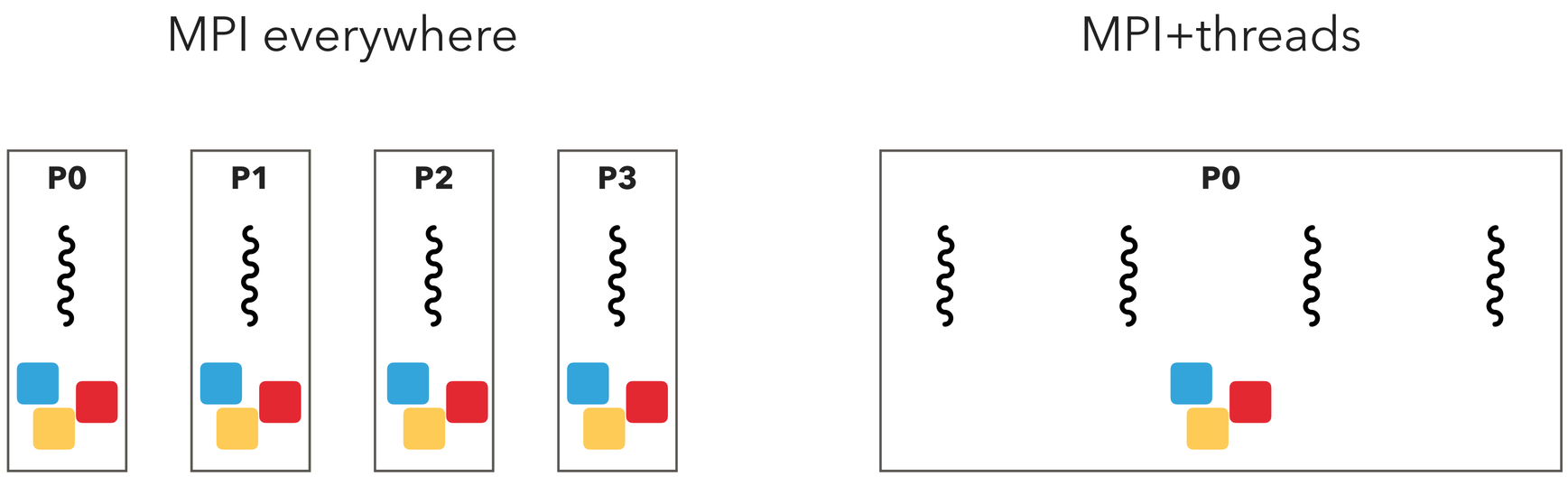
- Multithreaded environments feature larger design space



Communication resources: TX Queues, RX Queues, Completion Queues, Network registers

LIMIT STUDY FOR MULTITHREADED COMMUNICATION

- Multithreaded environments feature larger design space



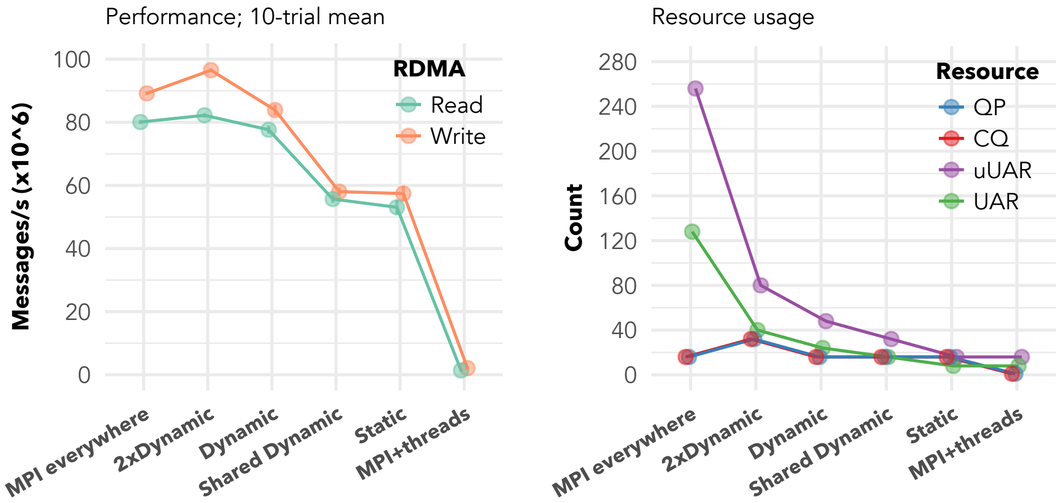
Communication resources: TX Queues, RX Queues, Completion Queues, Network registers

LIMIT STUDY FOR MULTITHREADED COMMUNICATION

- Multithreaded environments feature larger design space
- State of the art configurations at extreme ends of sharing levels
- *What level of sharing is ideal for multithreaded environments?*
  - *It depends* on performance requirements and availability of resources
- Key lessons from resource-sharing analysis on Mellanox IB using Verbs
  - NIC has a parallel TLB → Threads must use cache-aligned buffers
  - The Verbs Context most impactful on hardware resource usage

LIMIT STUDY FOR MULTITHREADED COMMUNICATION

Global arrays matrix multiplication kernel



Performance decreases with increasing resource efficiency

Scalable Communication Endpoints, ICPADS'18

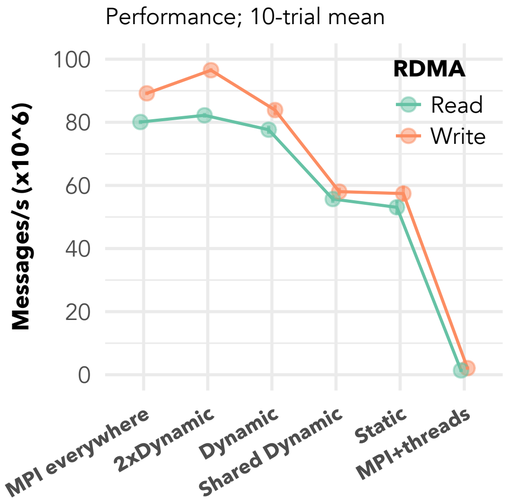
State of the art

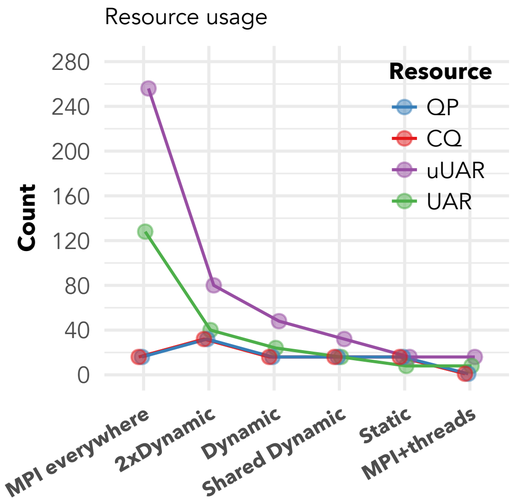
State of the art

	Performance (Higher is better)	Hardware resources (Lower is better)	Software resources (Lower is better)
MPI everywhere	100%	100%	100%
2xDynamic	100%	31.25%	200%
Dynamic	94%	18.75%	100%
Shared Dynamic	65%	12.5%	100%
Static	64%	6.25%	100%
MPI+threads	3%	6.25%	6.25%

LIMIT STUDY FOR MULTITHREADED COMMUNICATION

Global arrays matrix multiplication kernel





Performance decreases with increasing resource efficiency

Multiple threads can achieve same performance as multiple processes with lesser resources

Scalable Communication Endpoints, ICPADS'18

State of the art

State of the art

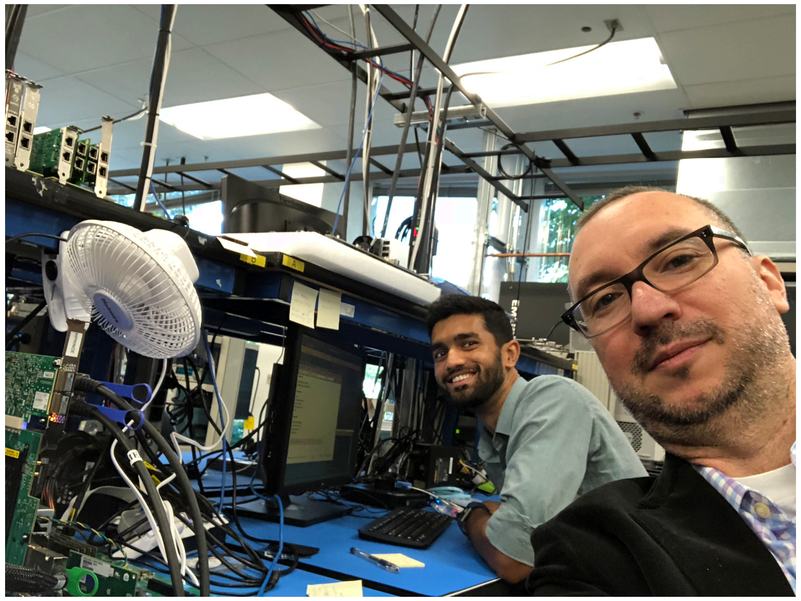
	Performance (Higher is better)	Hardware resources (Lower is better)	Software resources (Lower is better)
MPI everywhere	100%	100%	100%
2xDynamic	100%	31.25%	200%
Dynamic	94%	18.75%	100%
Shared Dynamic	65%	12.5%	100%
Static	64%	6.25%	100%
MPI+threads	3%	6.25%	6.25%

HOW DOES COMMUNICATION PERFORMANCE OF NON-TRADITIONAL PLAYERS FARE?

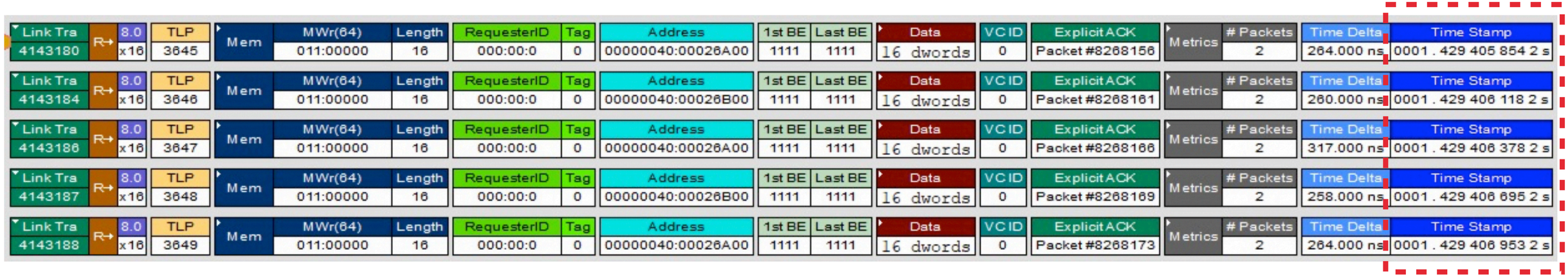
Traditional players

New players

VALIDATING MODELS WITH PCIE ANALYZERS



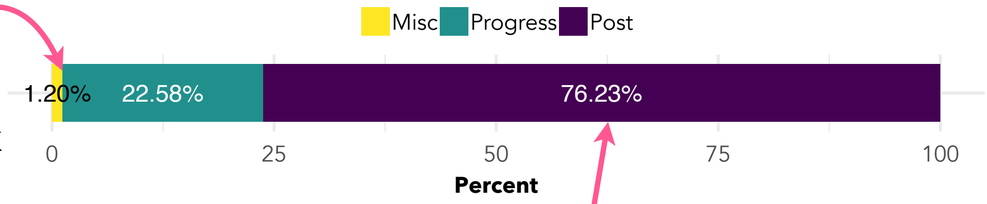
SYSTEM MEASUREMENTS USING PCIE TRACES



Link Tra		8.0	TLP	Mem	MWr(64)	Length	RequesterID	Tag	Address	1st BE	Last BE	Data	VC ID	ExplicitACK	Metrics	# Packets	Time Delta	Time Stamp
4143180	R→	x16	3645		011:00000	16	000:00:0	0	00000040:00026A00	1111	1111	16 dwords	0	Packet #8268156		2	264.000 ns	0001 . 429 405 854 2 s
4143184	R→	x16	3646		011:00000	16	000:00:0	0	00000040:00026B00	1111	1111	16 dwords	0	Packet #8268161		2	260.000 ns	0001 . 429 406 118 2 s
4143186	R→	x16	3647		011:00000	16	000:00:0	0	00000040:00026A00	1111	1111	16 dwords	0	Packet #8268166		2	317.000 ns	0001 . 429 406 378 2 s
4143187	R→	x16	3648		011:00000	16	000:00:0	0	00000040:00026B00	1111	1111	16 dwords	0	Packet #8268169		2	258.000 ns	0001 . 429 406 695 2 s
4143188	R→	x16	3649		011:00000	16	000:00:0	0	00000040:00026A00	1111	1111	16 dwords	0	Packet #8268173		2	264.000 ns	0001 . 429 406 953 2 s

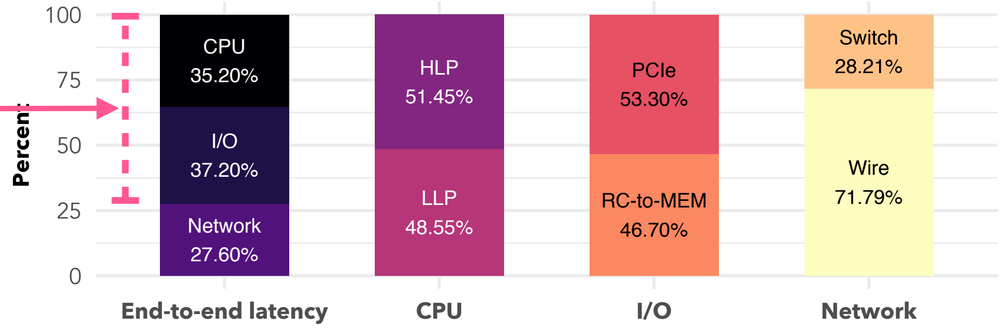
Time of event = Timestamp of packet after event -
Timestamp of packet before event

Progress is semantic bottleneck

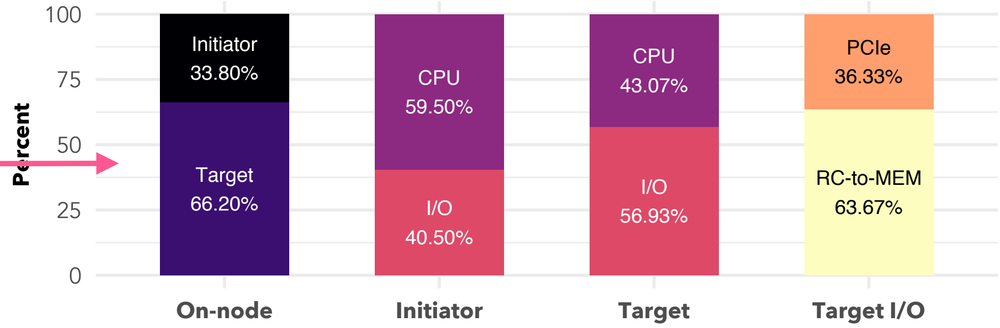


Post is performance bottleneck

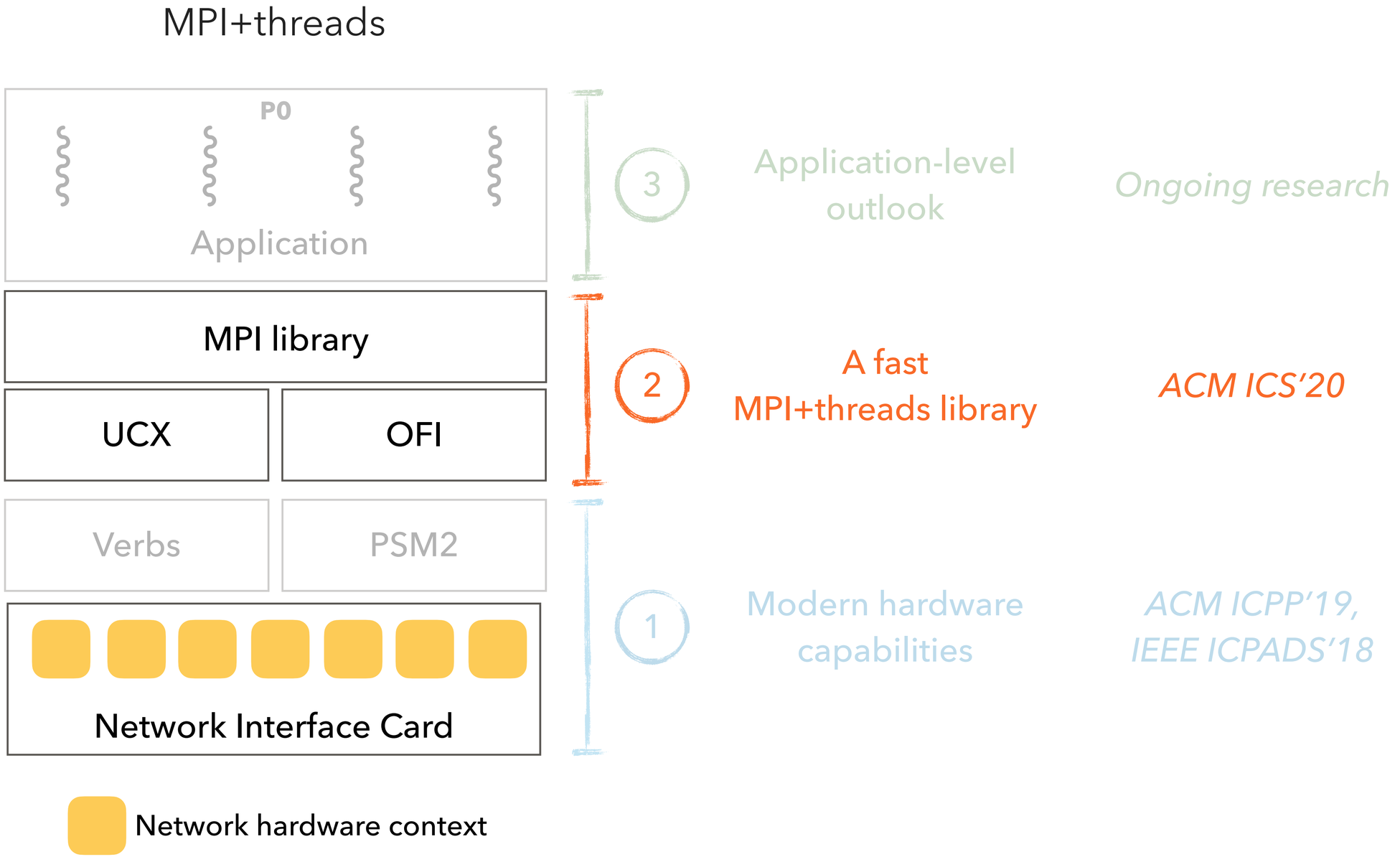
On-node time dominates latency



Target dominates on-node time

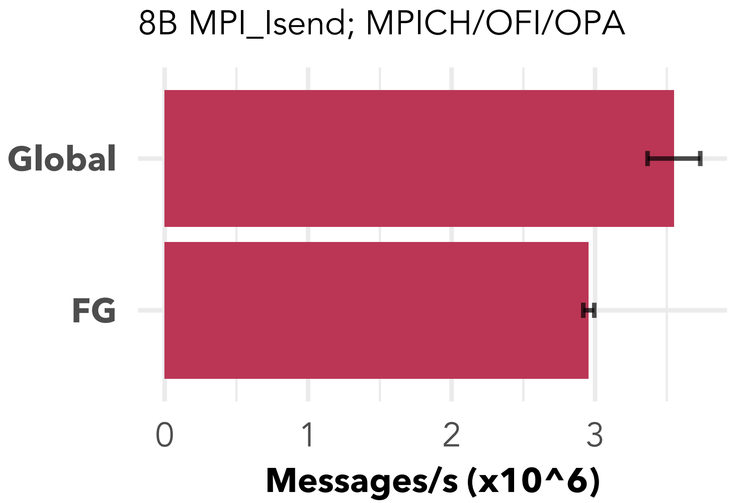


- Breakdown helps researchers guide their optimization efforts
- Detailed measurement methodology
- First work of this kind on an Arm-based server

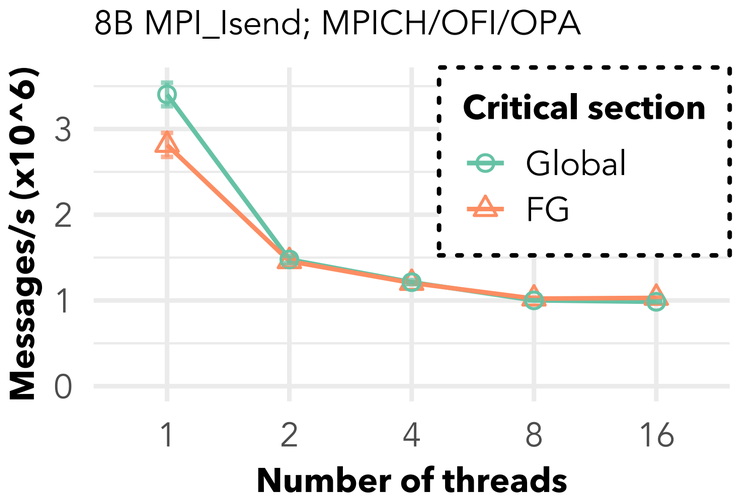

MPI+threads
P0
Application
MPI library
UCX
OFI
Verbs
PSM2
Network Interface Card
Network hardware context
3
Application-level outlook
Ongoing research
2
A fast MPI+threads library
ACM ICS'20
1
Modern hardware capabilities
ACM ICPP'19, IEEE ICPADS'18

DESERIALIZING ACCESS TO THE MPI LIBRARY

- State of the art: global critical section
- Adopt fine-grained critical sections (Balaji et al., Amer et al.)
  - Higher parallelism
  - More lock acquisitions
  - Atomics for counters



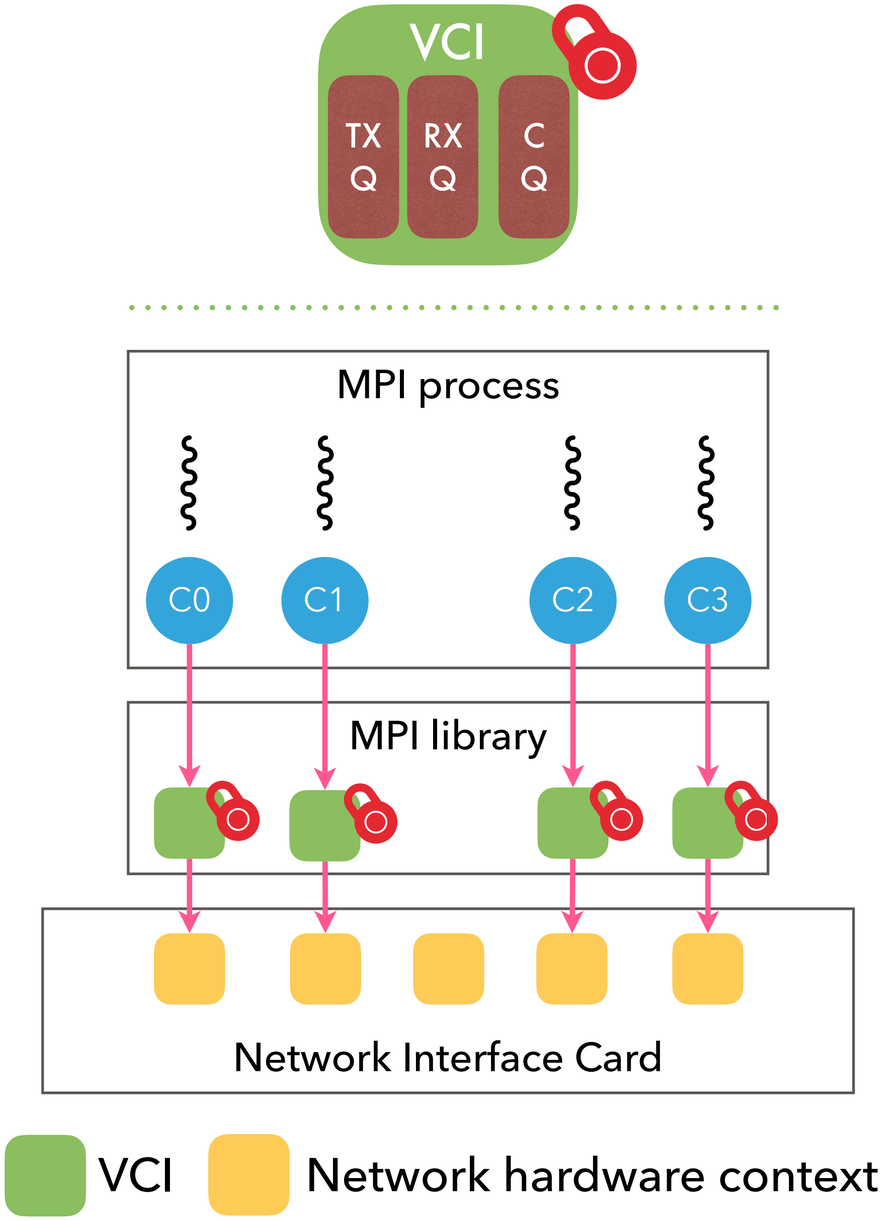
Overheads of FG in the single thread case

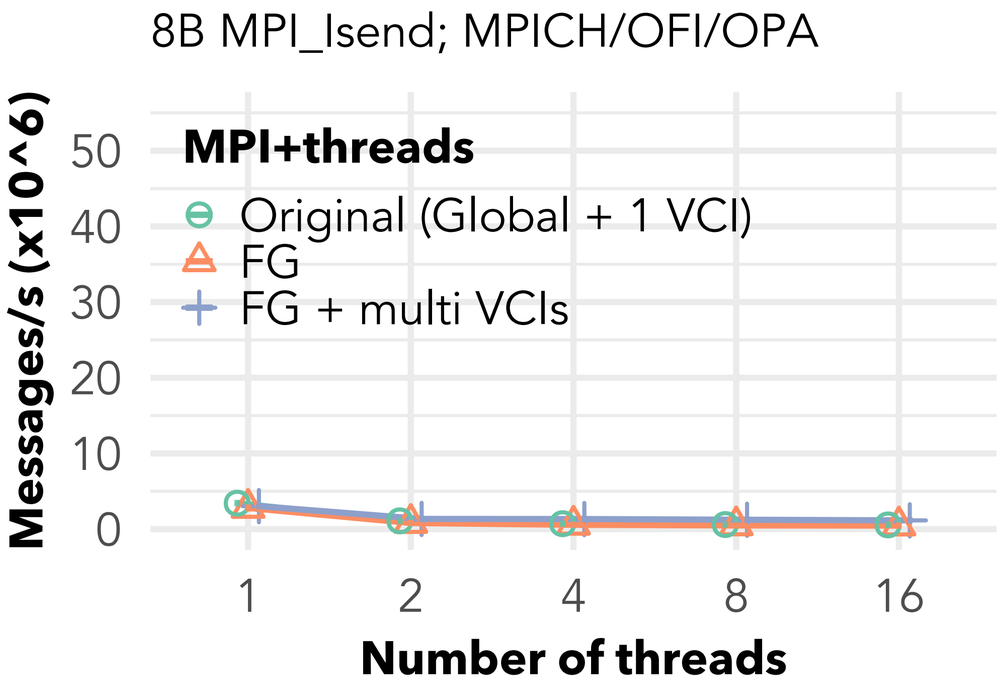


FG outperforms Global at higher thread count

PARALLEL COMMUNICATION STREAMS

- Virtual Communication Interfaces (VCIs)
  - Independent set of communication resources with FIFO order
  - Each VCI protected by its own lock
  - Maps to a network hardware context
- VCI pool
  - Eg: Allocate a VCI to a communicator/window
  - Fallback mechanisms

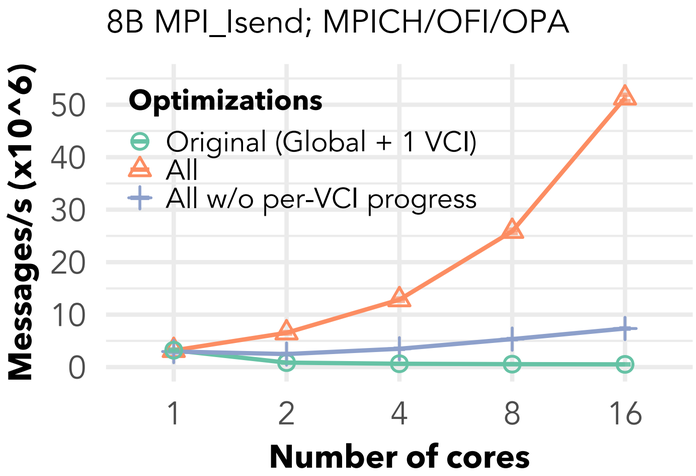




Fine-grained critical sections + multiple VCIs alone give practically no benefit

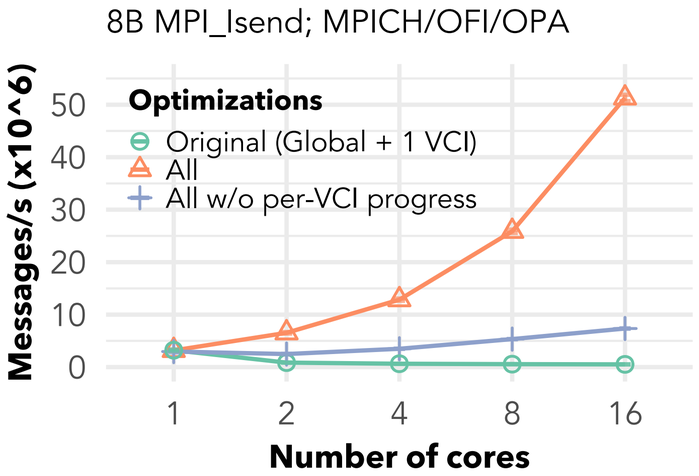
Per-VCI progress

- *Global progress*: progress all VCIs
  - High contention on VCIs' locks
- *Pure per-VCI progress*: progress only VCI of operation
  - Deadlock in codes that require shared progress → thread A needs to progress operations of thread B issued on VCI B
- *Hybrid per-VCI progress*



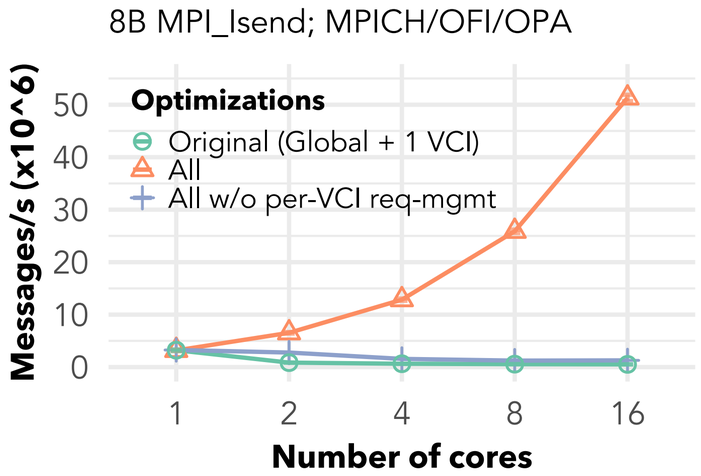
Per-VCI progress

- *Global progress*: progress all VCIs
  - High contention on VCIs' locks
- *Pure per-VCI progress*: progress only VCI of operation
  - Deadlock in codes that require shared progress
- *Hybrid per-VCI progress*



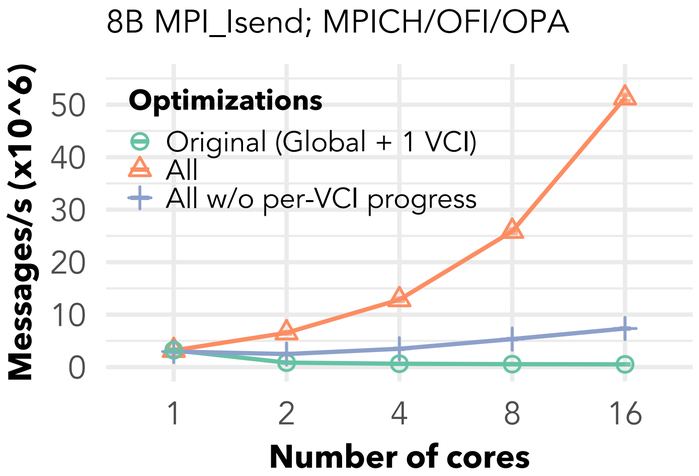
Per-VCI Request management

- *Request class lock*: high contention
  - *Per-VCI request cache*
- *Global lightweight request*: contended atomics for refcounting
  - *Per-VCI lightweight request*



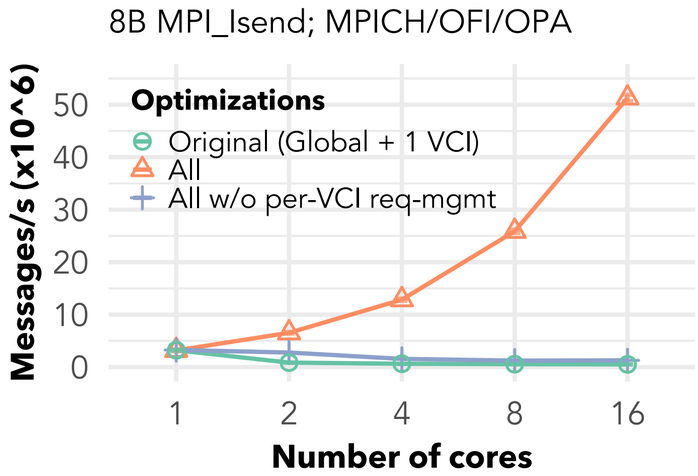
Per-VCI progress

- *Global progress*: progress all VCIs
  - High contention on VCIs' locks
- *Pure per-VCI progress*: progress only VCI of operation
  - Deadlock in codes that require shared progress
- *Hybrid per-VCI progress*



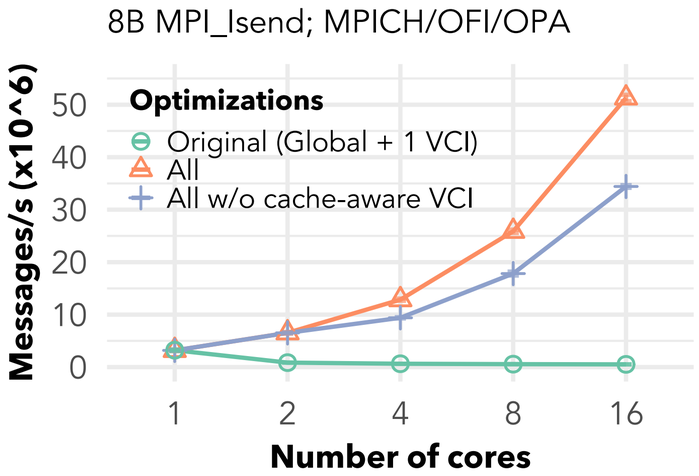
Per-VCI Request management

- *Request class lock*: high contention
  - *Per-VCI request cache*
- *Global lightweight request*: contended atomics for refcounting
  - *Per-VCI lightweight request*



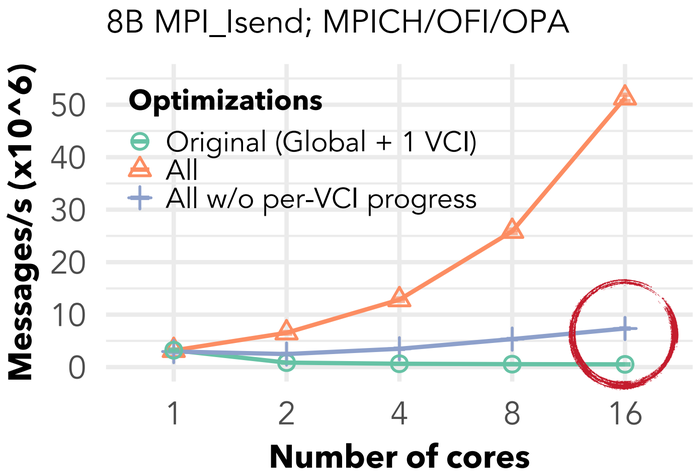
Per-VCI cache-line awareness

- *False-sharing*: locks of consecutive VCIs
- *Per-VCI cache alignment*



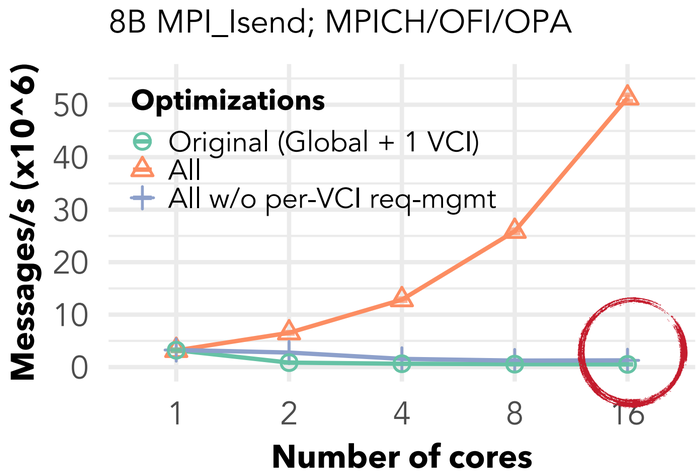
Per-VCI progress

- *Global progress*: progress all VCIs
  - High contention on VCIs' locks
- *Pure per-VCI progress*: progress only VCI of operation
  - Deadlock in codes that require shared progress
- *Hybrid per-VCI progress*



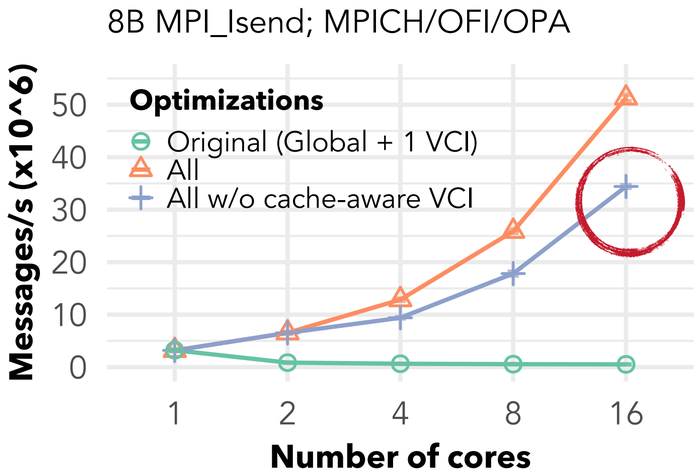
Per-VCI Request management

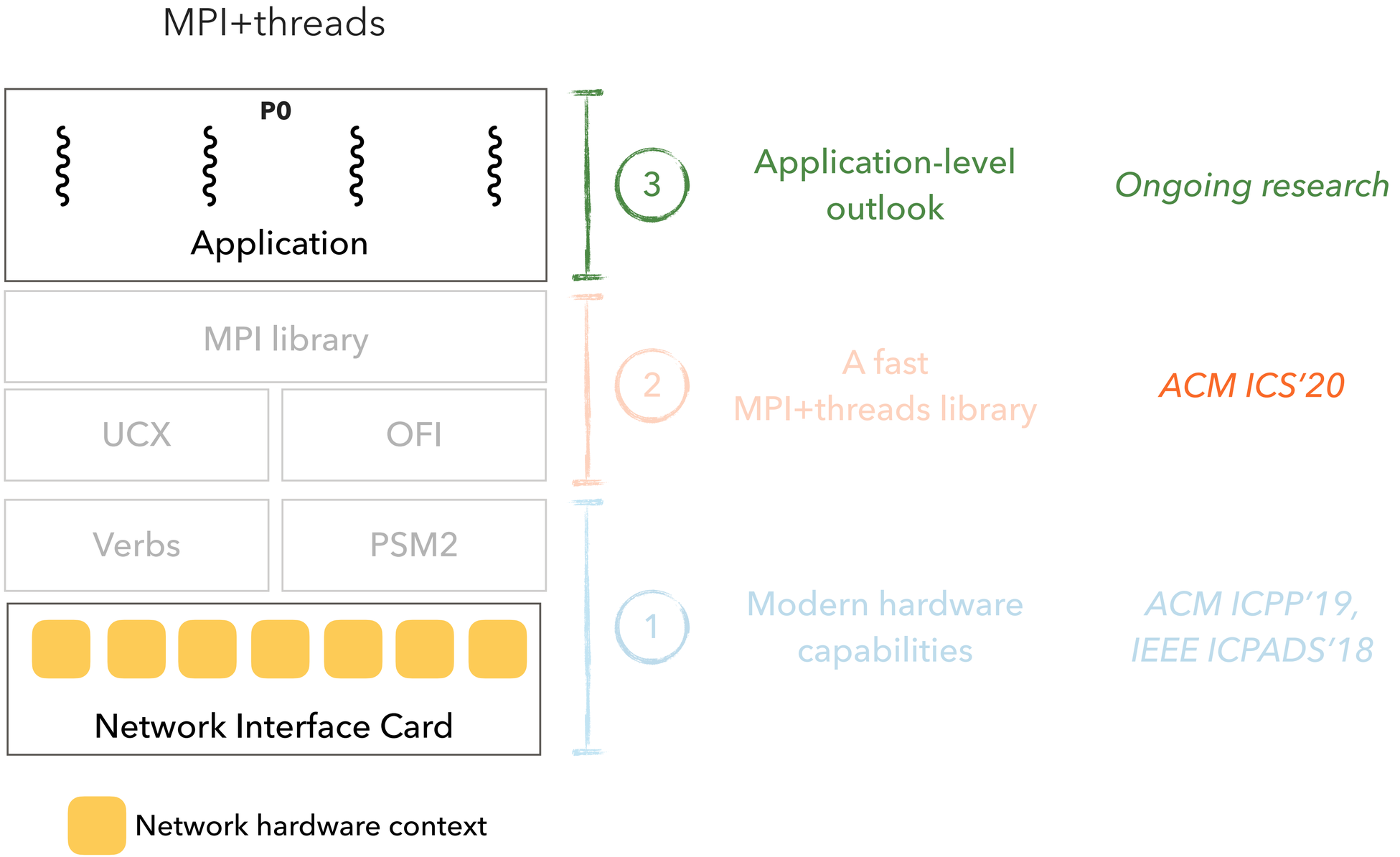
- *Request class lock*: high contention
  - *Per-VCI request cache*
- *Global lightweight request*: contended atomics for refcounting
  - *Per-VCI lightweight request*



Per-VCI cache-line awareness

- *False-sharing*: locks of consecutive VCIs
- *Per-VCI cache alignment*

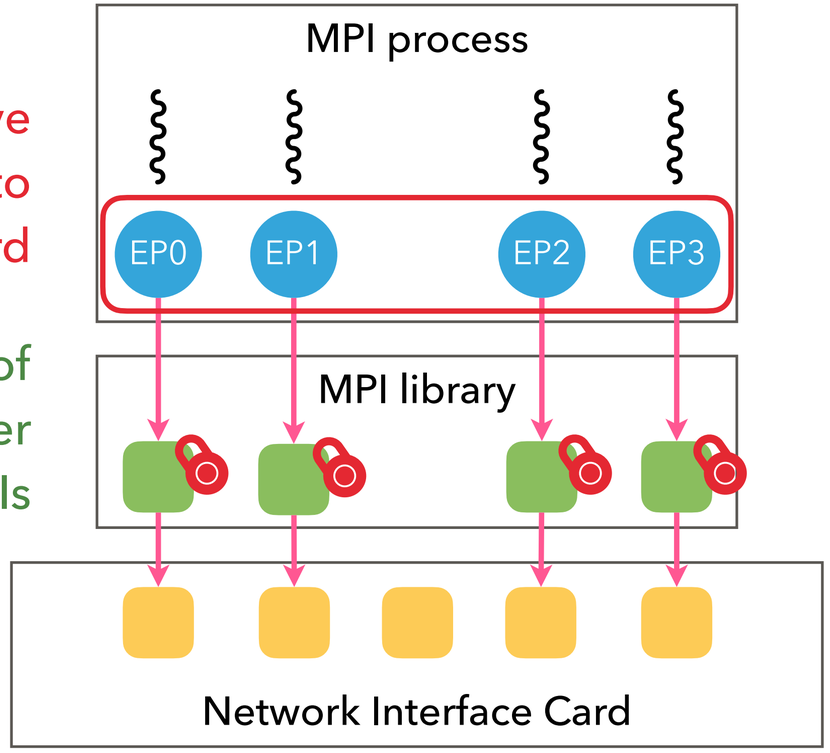



MPI+threads
P0
Application
MPI library
UCX
OFI
Verbs
PSM2
Network Interface Card
Network hardware context
3
Application-level outlook
Ongoing research
2
A fast MPI+threads library
ACM ICS'20
1
Modern hardware capabilities
ACM ICPP'19, IEEE ICPADS'18

Explicit mapping through user-visible endpoints

Intrusive extension to standard

Flexibility of control over VCIs

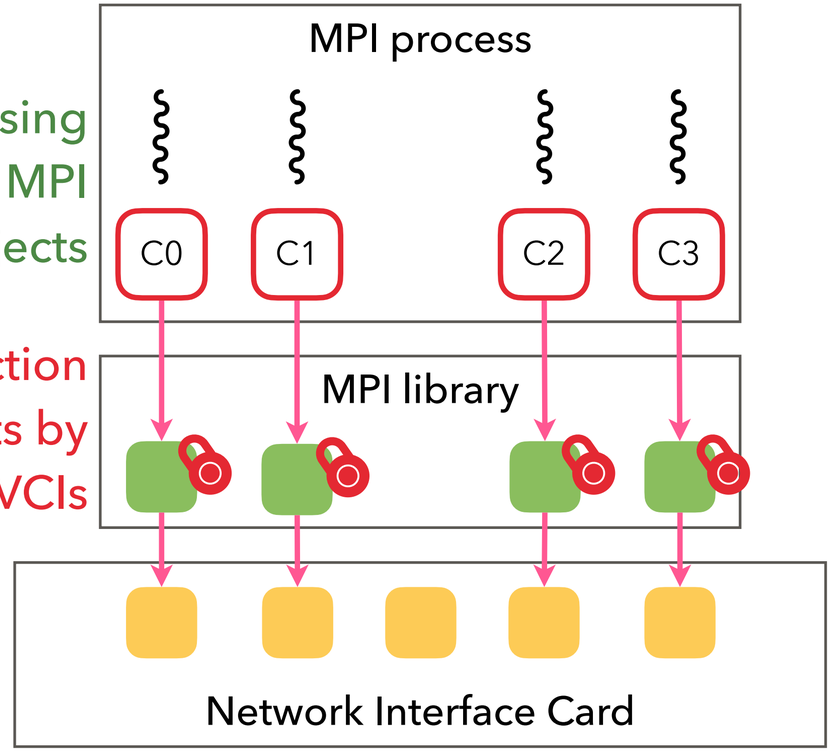


Implicit mapping through VCIs

(using MPI-3.1, or MPI-3.1 with hints)

Using existing MPI objects

Abstraction costs by hiding VCIs



MPI Communicator

MPI Endpoint

Virtual Communication Interface

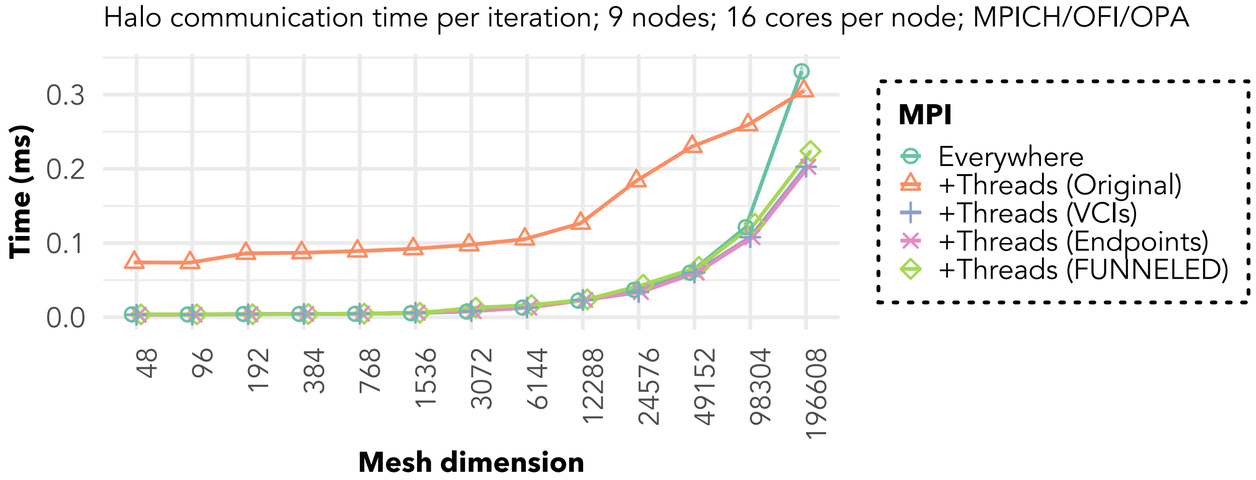
Network hardware context

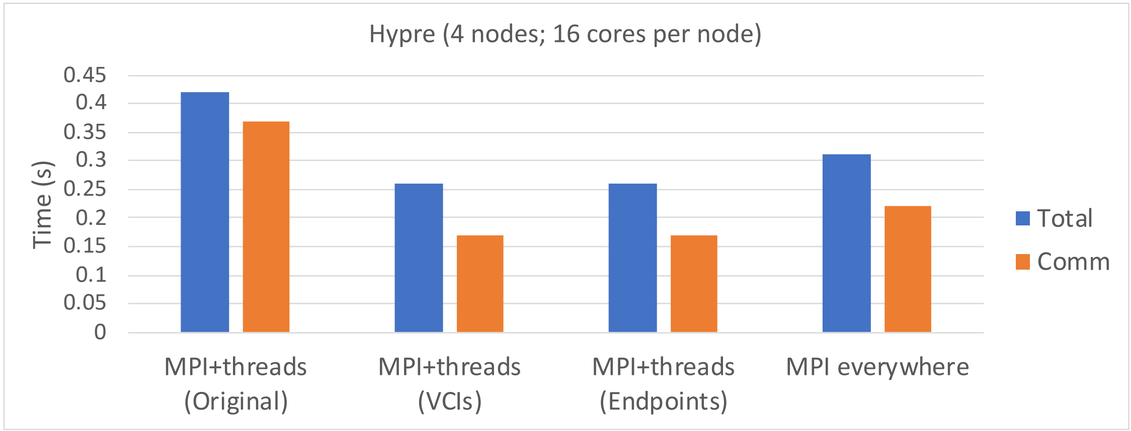
APPLICATION CATEGORIES FOR PERFORMANCE

- Category 1
  - Direct use of parallel communication streams
  - VCIs as good as user-visible endpoints and MPI everywhere
- Category 2
  - Require shared progress
  - Both VCIs and user-visible endpoints perform poorly
- Category 3
  - Abstraction through MPI-3.1 prevents user from expressing parallelism
  - User-visible endpoints perform better than VCIs

CATEGORY 1: STENCIL APPLICATIONS

MPI+threads with network parallelism utilized is 16% faster than MPI everywhere and 38% faster than MPI+threads without network parallelism





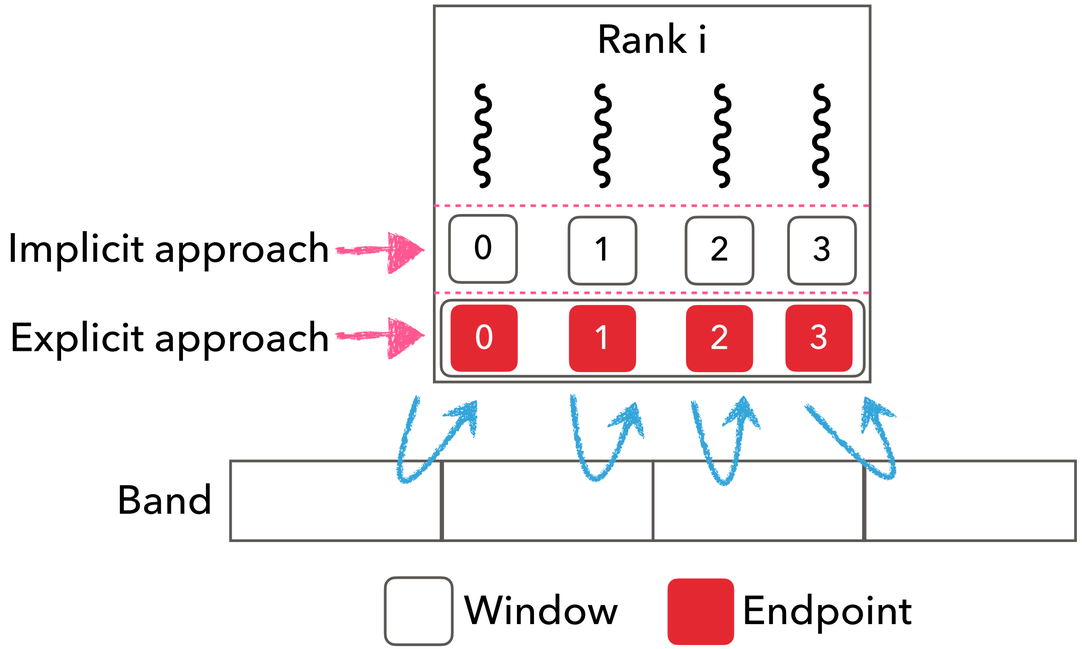
Takeaway: For basic communication, VCIs and endpoints perform similarly and as well as MPI everywhere

APPLICATION CATEGORIES FOR PERFORMANCE

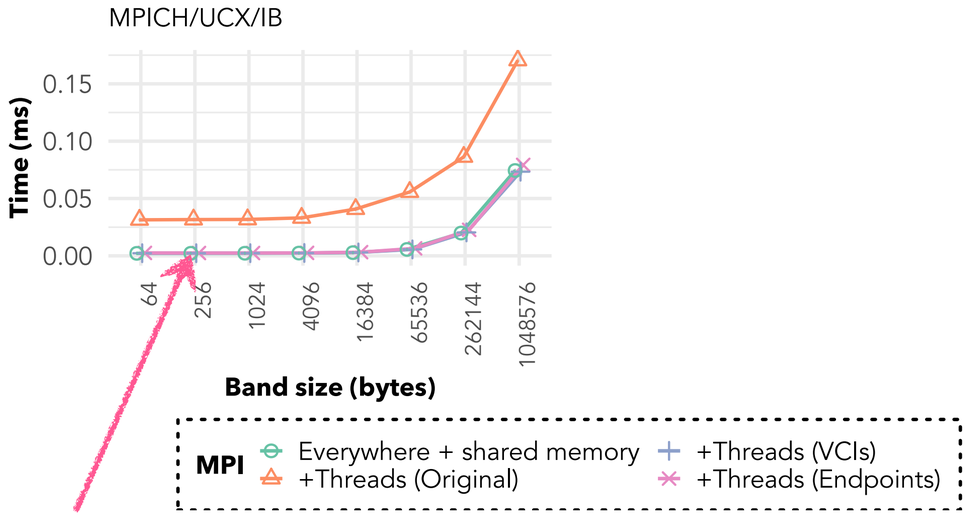
- Category 1
  - Direct use of parallel communication streams
  - VCIs as good as user-visible endpoints and MPI everywhere
- Category 2
  - Require shared progress
  - Both VCIs and user-visible endpoints perform poorly
- Category 3
  - Abstraction through MPI-3.1 prevents user from expressing parallelism
  - User-visible endpoints perform better than VCIs

CATEGORY 2: OPENMC

- OpenMC: distributed Monte-Carlo neutron-transport code
  - Band data equally distributed between nodes
  - Particles distributed between nodes for simulation
  - Each node fetches (MPI_Get) a band of data, processes its particles, and iterates



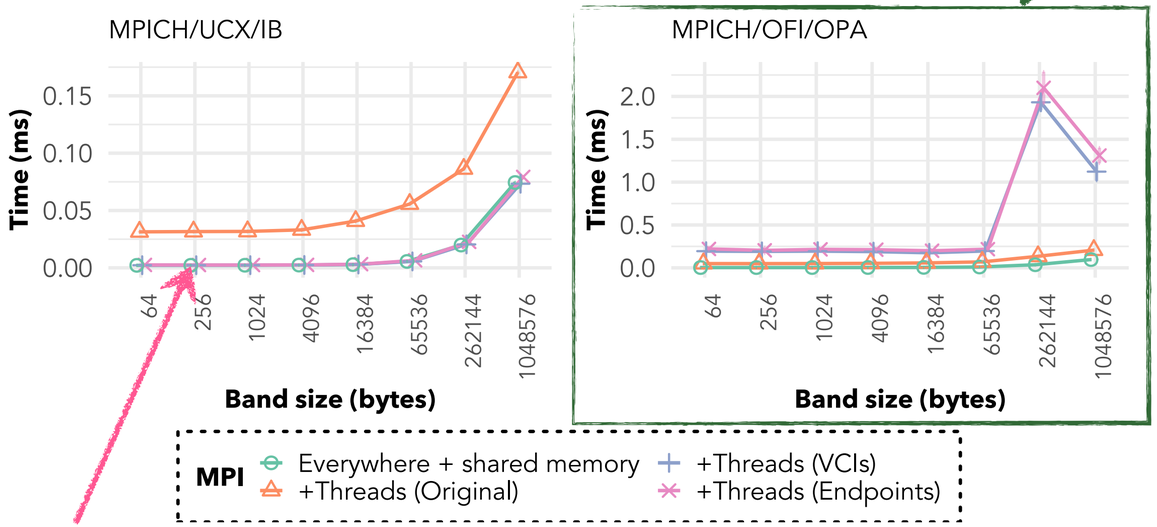
CATEGORY 2: OPENMC

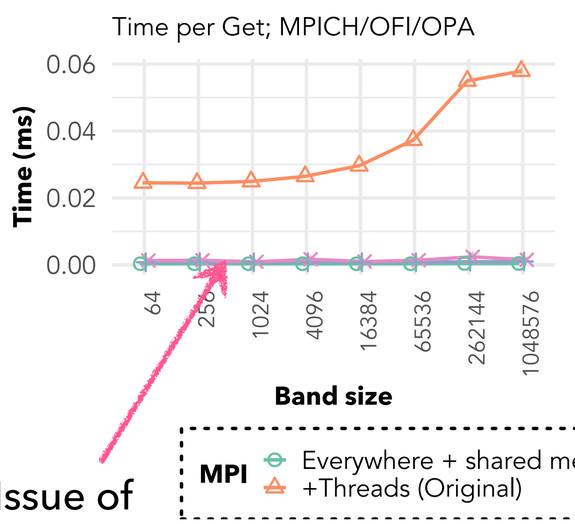


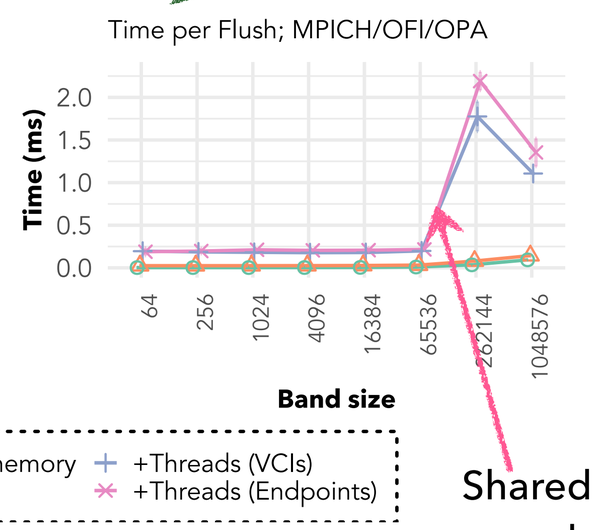
VCIs as good as endpoints and MPI everywhere when shared progress not required

CATEGORY 2: OPENMC

Shared progress: thread A progresses VCI of thread B. Required for correctness.







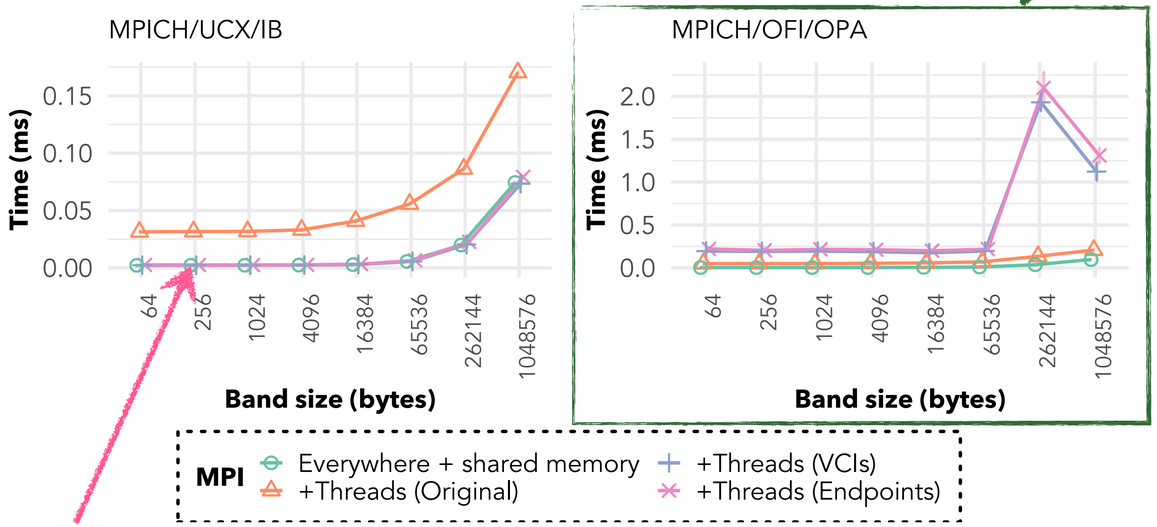
VCIs as good as endpoints and MPI everywhere when shared progress not required

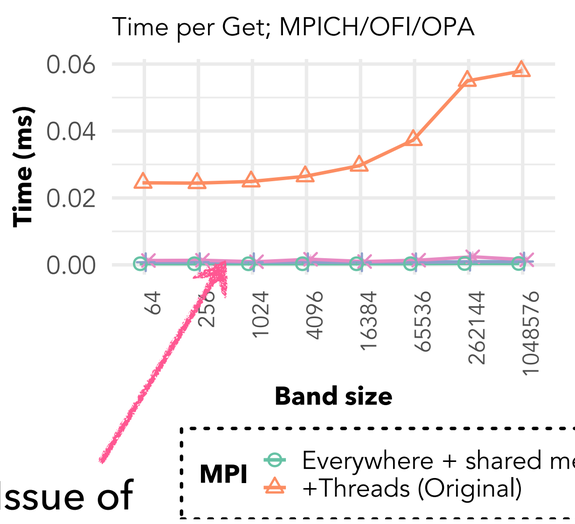
Issue of operations is fast

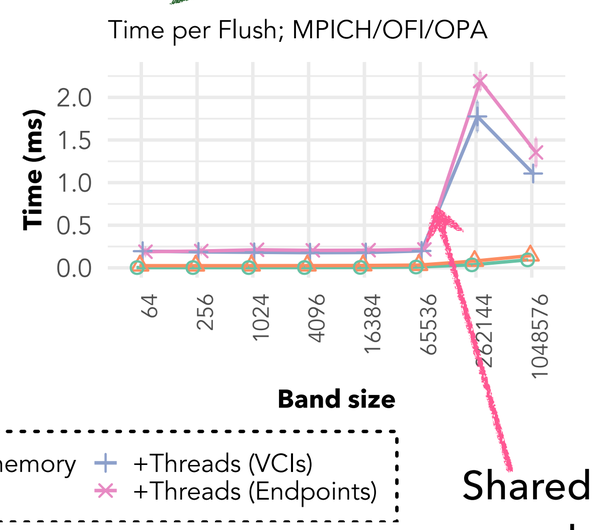
Shared progress hurts completion of operations

CATEGORY 2: OPENMC

Shared progress: thread A progresses VCI of thread B. Required for correctness.







VCIs as good as endpoints and MPI everywhere when shared progress not required

Issue of operations is fast

Shared progress hurts completion of operations

Takeaway: Both VCIs and Endpoints perform equally and as well as MPI everywhere in issuing non-atomic RMA operations

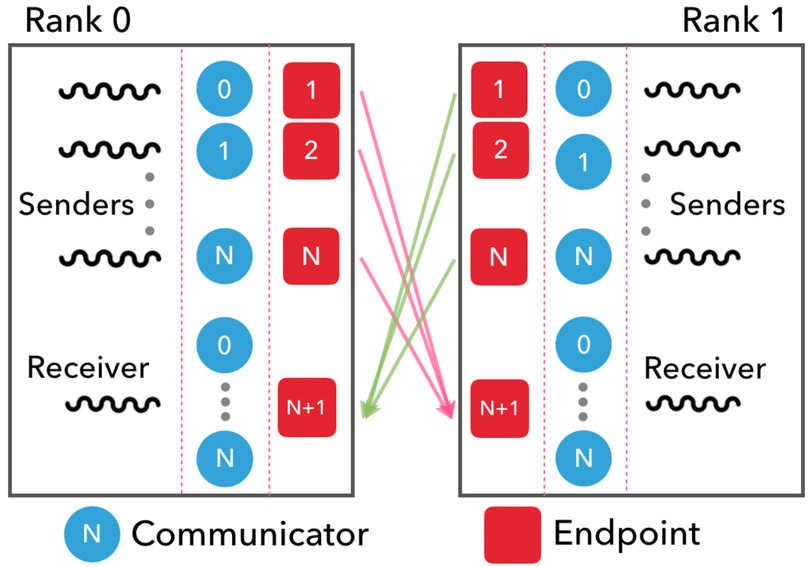
Warning: Independent communication with VCIs and user-visible endpoints fundamentally opposes shared progress

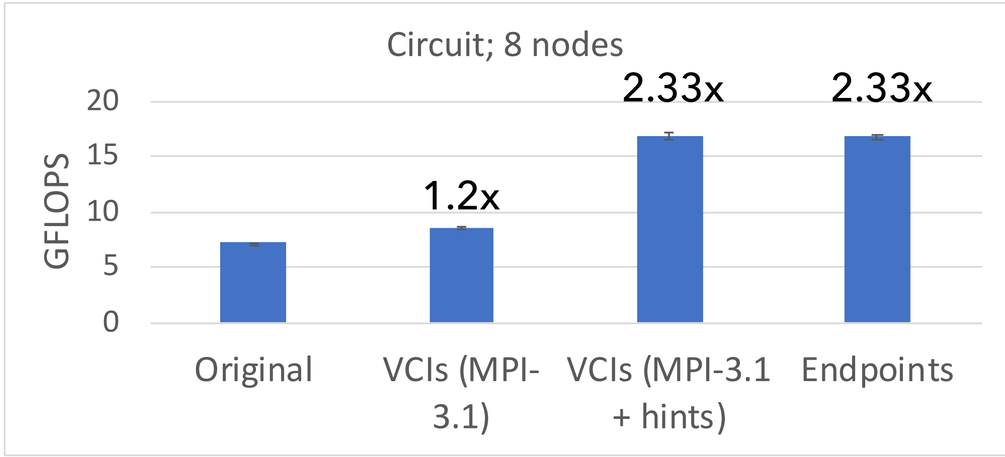
APPLICATION CATEGORIES FOR PERFORMANCE

- Category 1
  - Direct use of parallel communication streams
  - VCIs as good as user-visible endpoints and MPI everywhere
- Category 2
  - Require shared progress
  - Both VCIs and user-visible endpoints perform poorly
- Category 3
  - Abstraction through MPI-3.1 prevents user from expressing parallelism
  - User-visible endpoints could perform better than VCIs

CATEGORY 3: LEGION RUNTIME

- Example: communication pattern in Legion's runtime.
- MPI-3.1: Contention between receiver thread and sender threads with communicators.
- MPI-3.1 with hints: No contention between sender and receiver threads with tags.
- No contention with endpoints.





SUMMARY OF PERFORMANCE DIFFERENCES

	Implicit		Explicit
	MPI-3.1	MPI-3.1 with hints	Endpoints
Category 1: Dedicated point-to-point communication	Maximum network parallelism possible	Maximum network parallelism possible	Maximum network parallelism possible
Category 2: Requiring shared progress	Detrimental	Detrimental	Detrimental
Category 3: Limiting MPI semantics	Fraction of possible network parallelism	Nearly as good as Endpoints	Maximum network parallelism possible

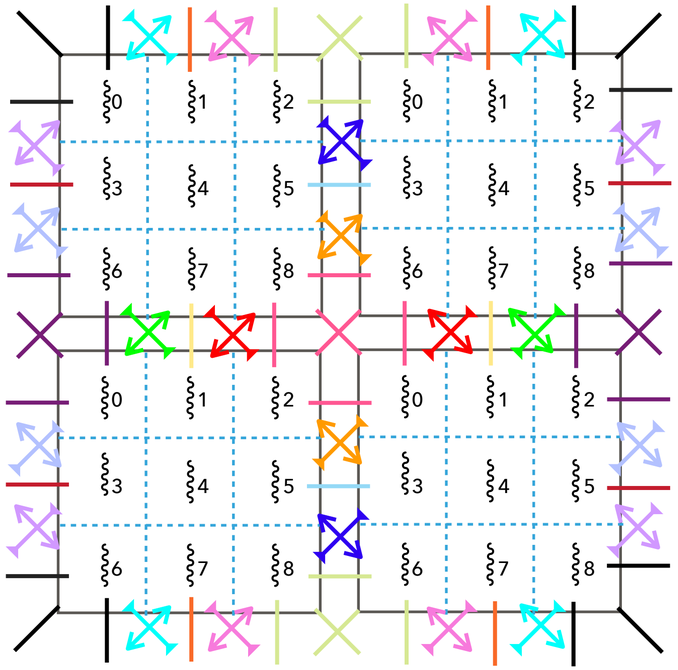
PRODUCTIVITY OF THE APPROACHES

"People ignore design that ignores people." — Frank Chimero

PRODUCTIVITY OF THE APPROACHES: LISTENING TO DOMAIN SCIENTISTS

- "It's a nightmare to express parallelism with communicators" - developers from Hypre-Uintah (ICCS'20), Smilei (CPC'18), and Pencil (SC'20)

MPI-3.1



OR

MPI-3.1 + hints

Encode sender and receiver thread ID into Tag

All 3 applications already doing this

Similar effort

OR

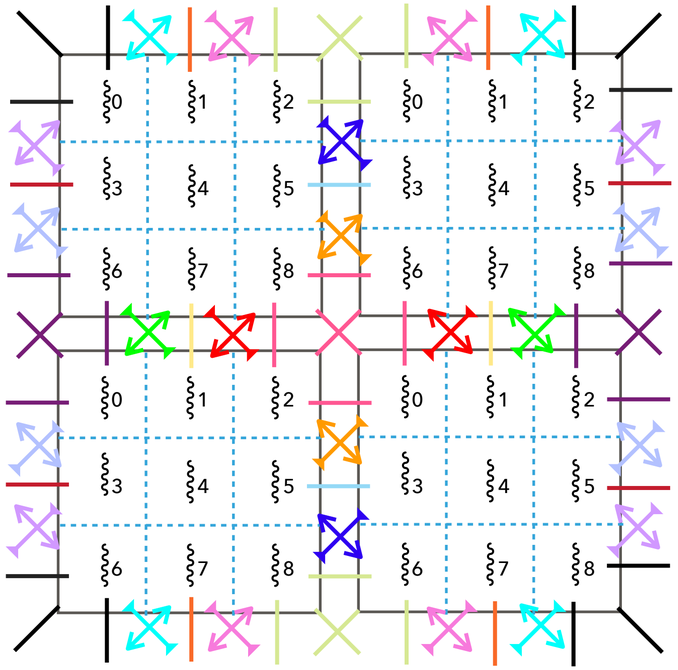
Endpoints

Using directly addressable endpoints

PRODUCTIVITY OF THE APPROACHES: LISTENING TO DOMAIN SCIENTISTS

- "It's a nightmare to express parallelism with communicators" - developers from Hypre-Uintah (ICCS'20), Smilei (CPC'18), and Pencil (SC'20)

MPI-3.1



OR

MPI-3.1 ★ + hints

Encode sender and receiver thread ID into Tag

All 3 applications already doing this

Similar effort

OR

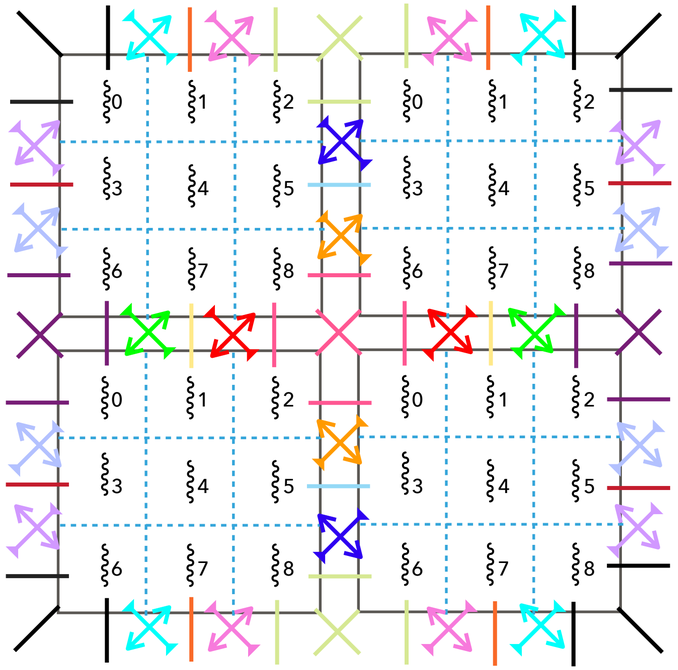
Endpoints

Using directly addressable endpoints

PRODUCTIVITY OF THE APPROACHES: LISTENING TO DOMAIN SCIENTISTS

- "It's a nightmare to express parallelism with communicators" - developers from Hypre-Uintah (ICCS'20), Smilei (CPC'18), and Pencil (SC'20)

MPI-3.1



OR

MPI-3.1 + hints

Encode sender and receiver thread ID into Tag

All 3 applications already doing this

Similar effort

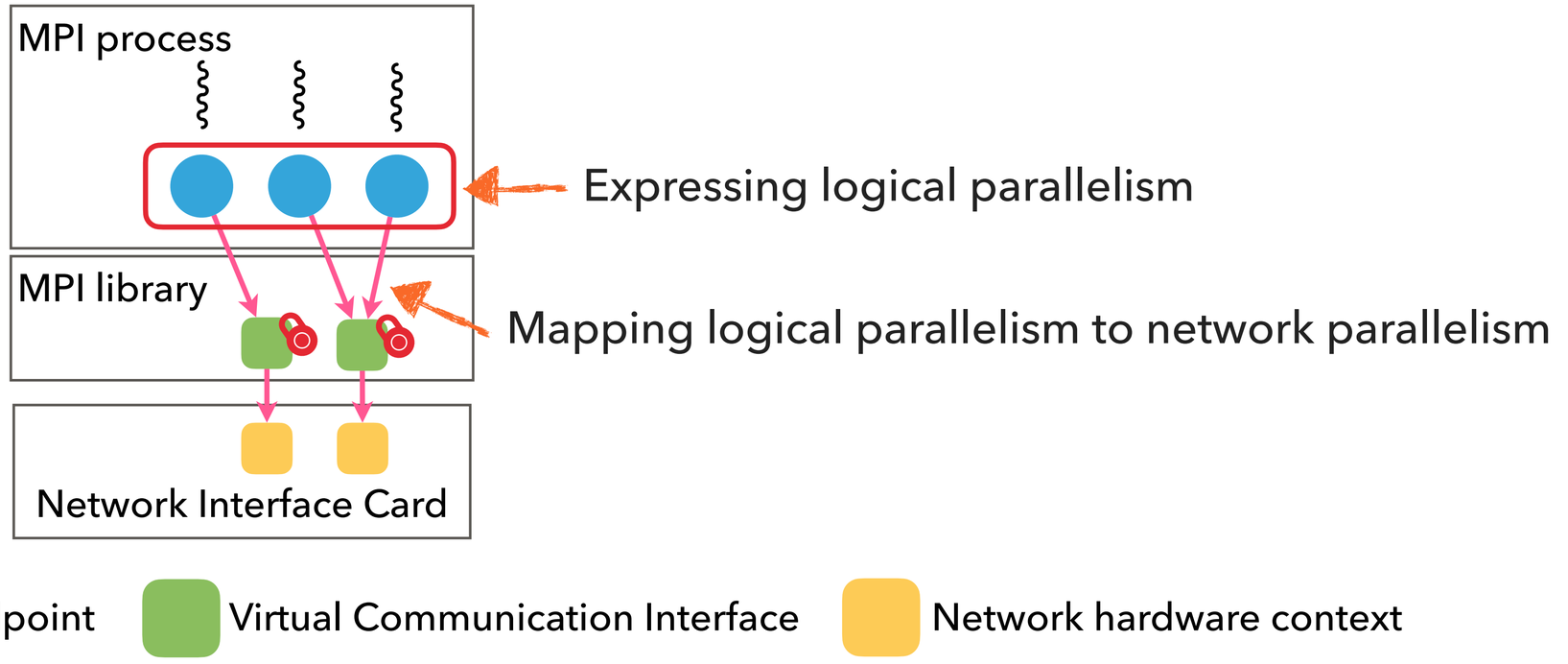
OR

Endpoints

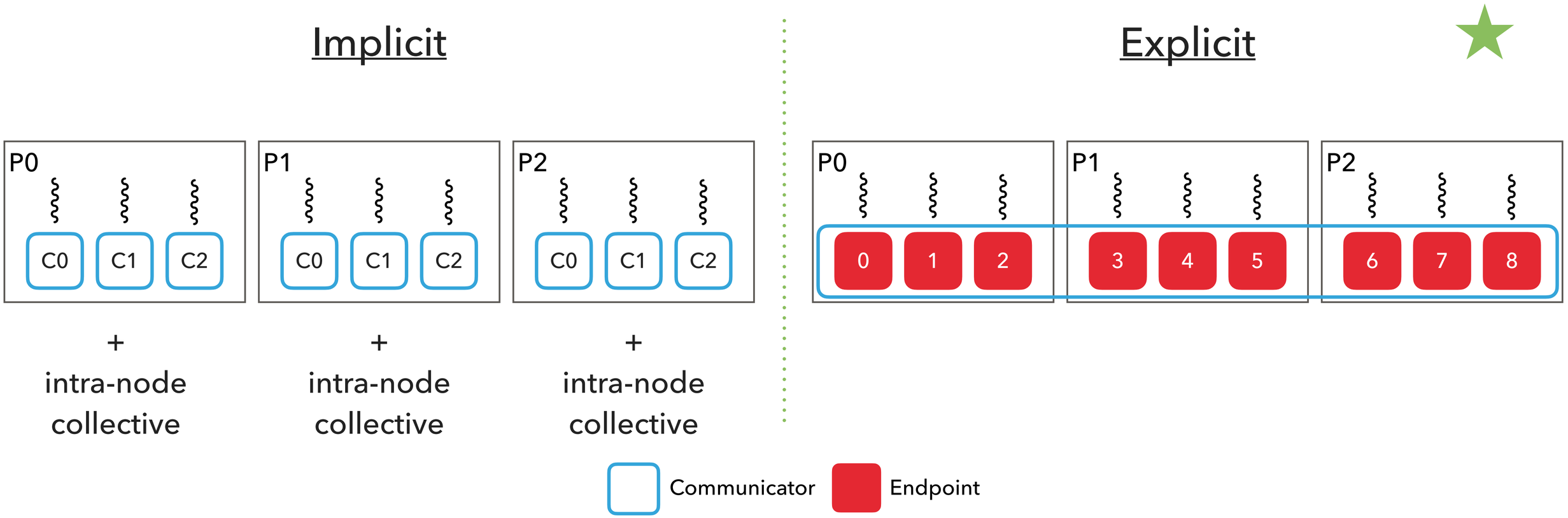
Using directly addressable endpoints

PRODUCTIVITY OF THE APPROACHES: LISTENING TO DOMAIN SCIENTISTS

- “MPI Endpoints is MPI library’s way of getting its job done by the MPI user. I don’t know how many endpoints to create for a given interconnect.” - developer from WOMBAT (ApJS’17)



COMPARISON OF THE APPROACHES: COLLECTIVES



SCHOOLS OF THOUGHT IN THE MPI COMMUNITY: DESIGN PERSPECTIVE

	Implicit		Explicit
	MPI-3.1	MPI-3.1 with hints	Endpoints
Point-to-point	Communicators	Communicators, Tags	Endpoints
RMA	Windows	Window(s)	Endpoints
Collectives	Communicators + user-driven intra-node collective		Endpoints
	Many options		One option

RULE OF THUMB FOR UX: MORE OPTIONS, MORE PROBLEMS

Scott Belsky

CONCLUDING REMARKS

- MPI+threads is critical for modern processors
  - Network parallelism is largely underutilized
  - Users must proactively express logical parallelism
- User-visible endpoints not critical to express logical parallelism
  - Objects in the existing MPI standard allow for parallelism
  - Explicit approach is a better means to express logical parallelism
- This work serves as a stepping stone into communication in heterogenous environments.

National lab collaborators:

Industry research collaborators:

University collaborators:

THANK YOU!

Email questions to rzambre@uci.edu