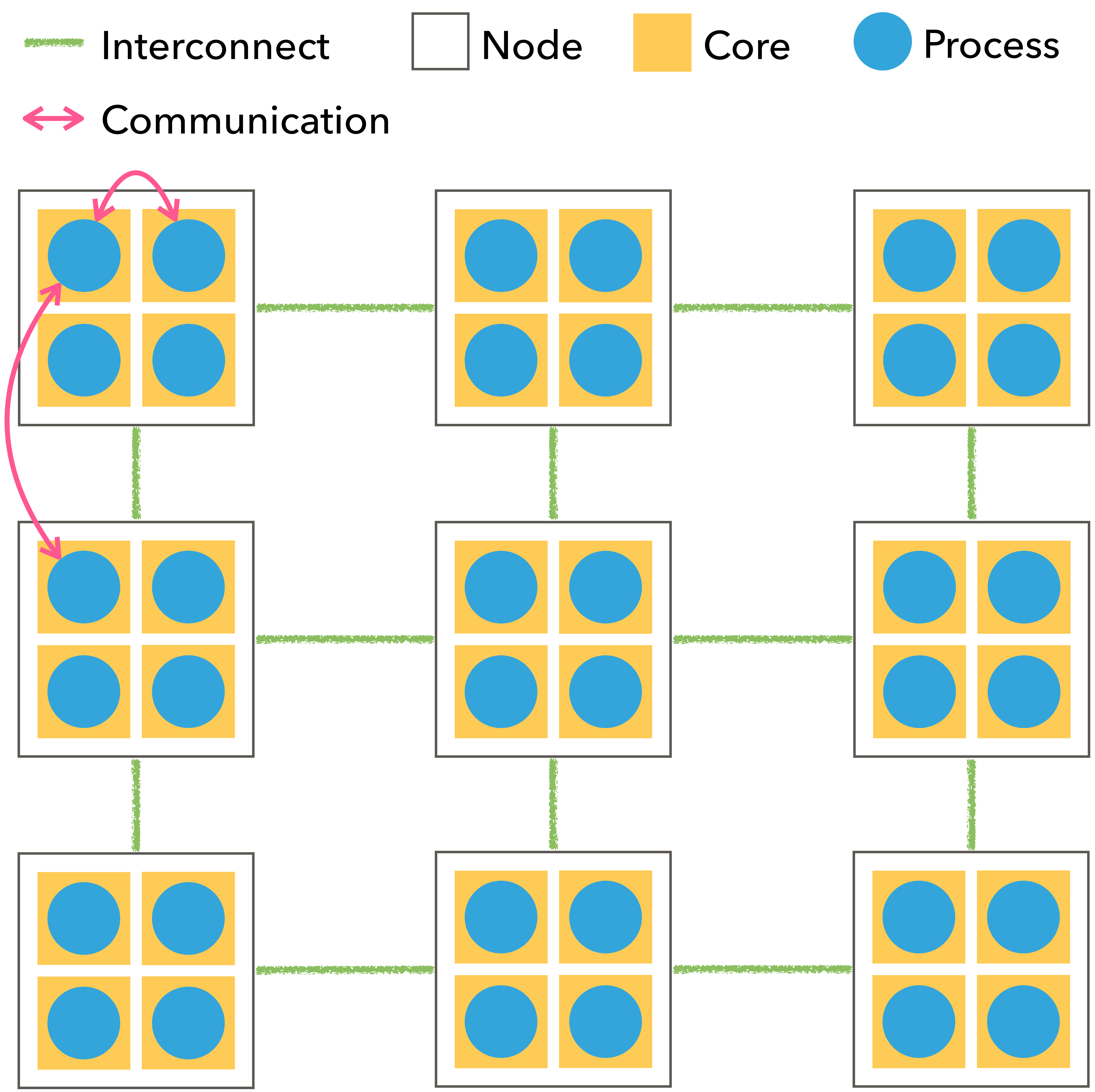
SCALABLE COMMUNICATION ENDPOINTS FOR MPI+THREADS APPLICATIONS

Rohit Zambre,* Aparna Chandramowlishwaran,* Pavan Balaji^
*University of California, Irvine
^Argonne National Laboratory

TRADITIONAL MPI: MPI EVERYWHERE

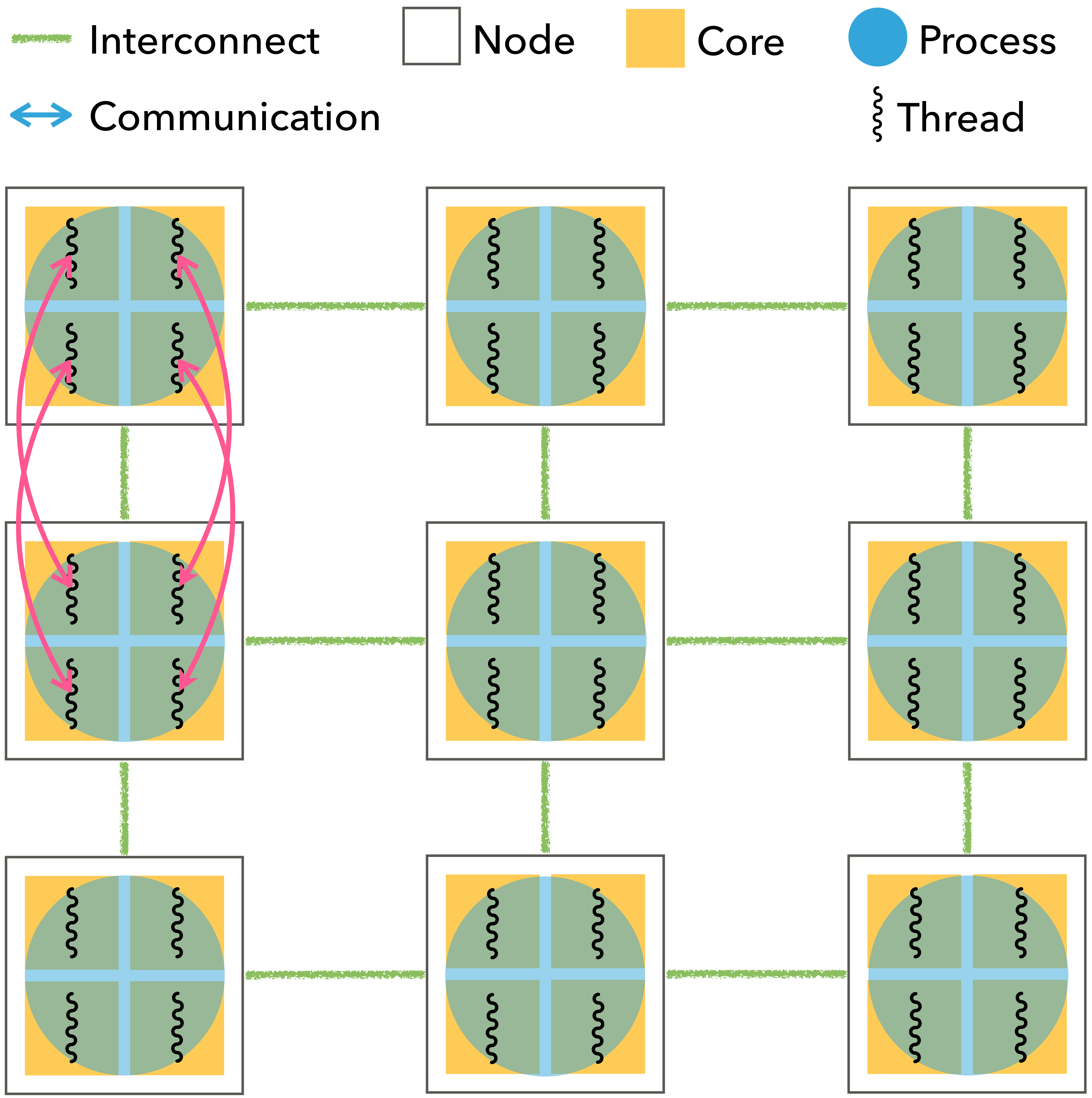
- Application ignores location of processes
- *Problem*: dwindling share of resources per process
    - *Why*: cores scaling a lot faster than other on-node resources



HYBRID MPI: MPI+THREADS

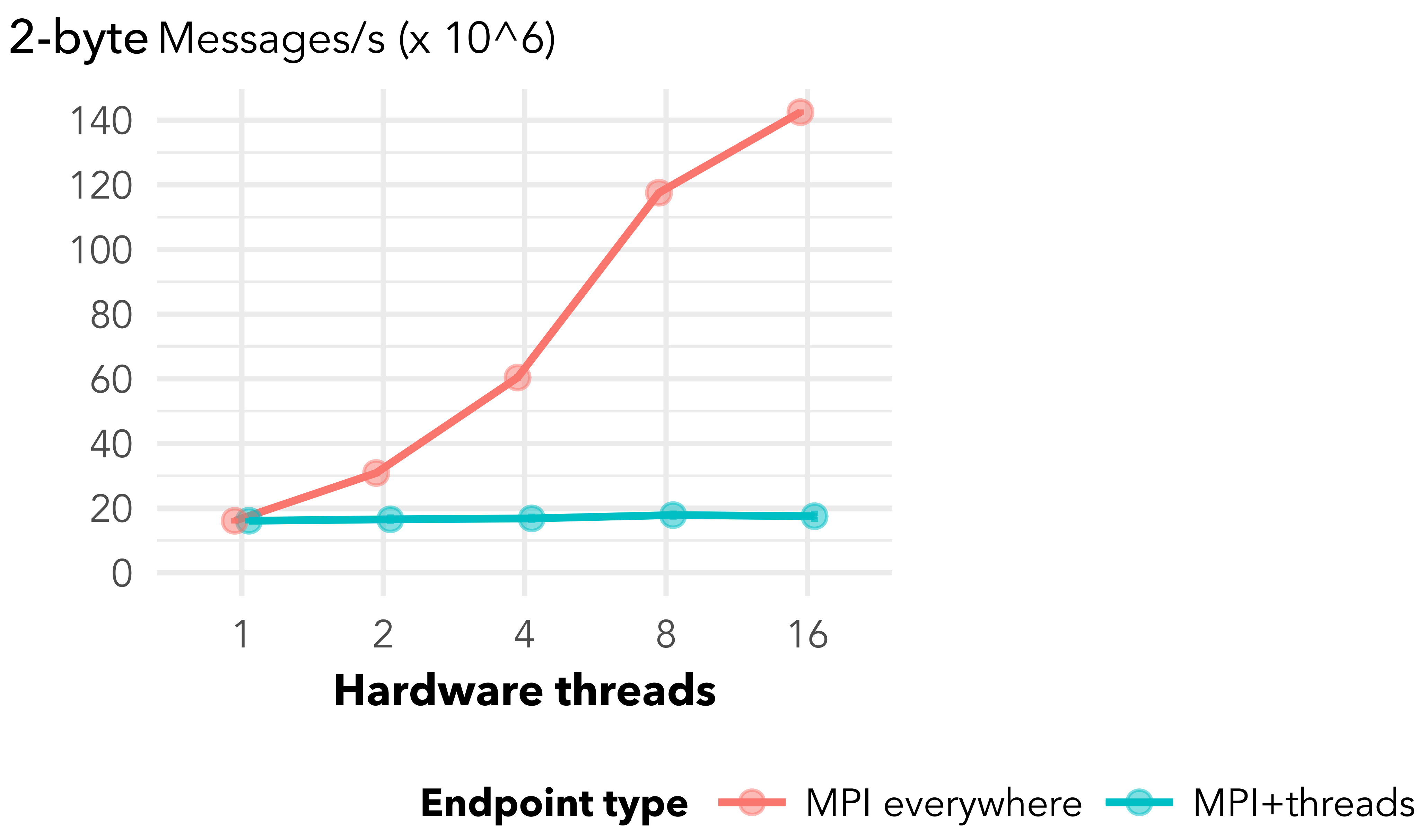
e.g. MPI+OpenMP

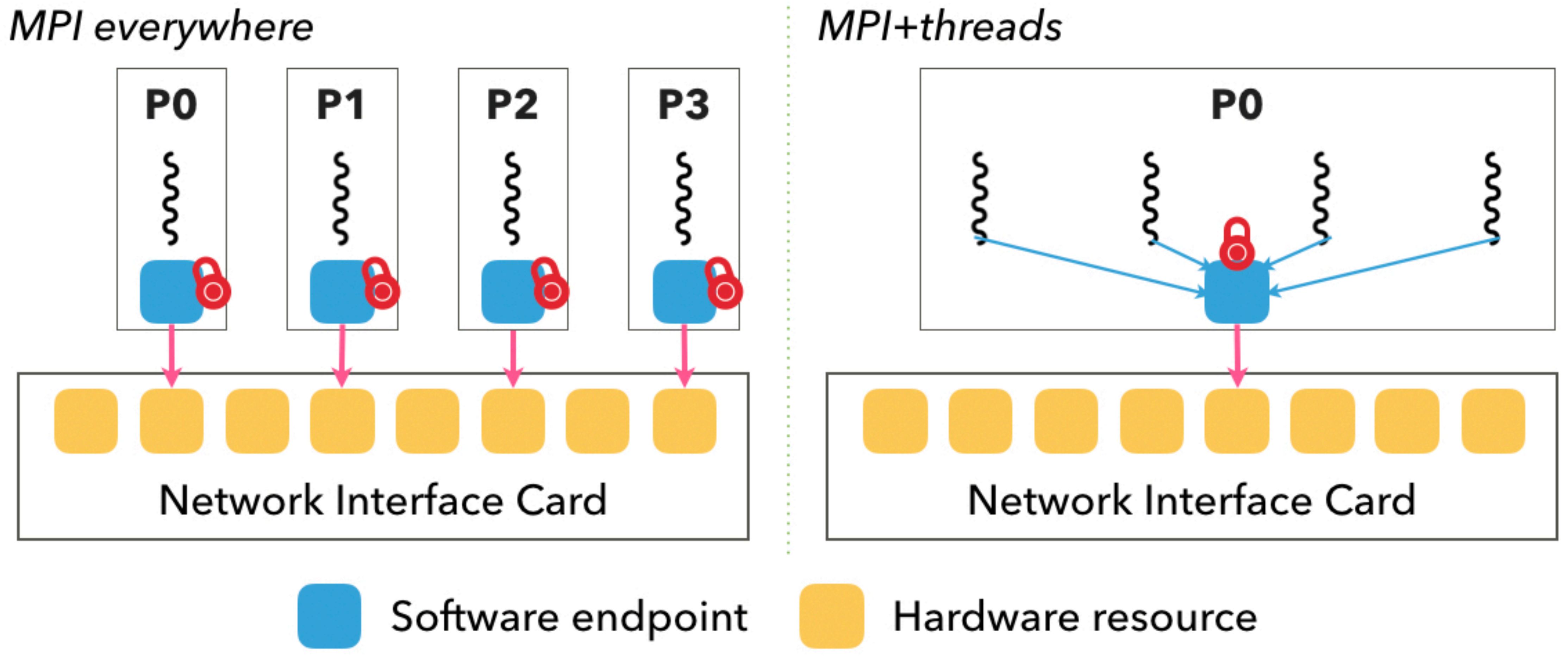
- 1 multi-threaded process per node
- Like processes, each thread offers a program counter and stack
- Unlike processes, each thread can access all on-node resources



THE PROBLEM WITH TODAY'S MPI IMPLEMENTATIONS

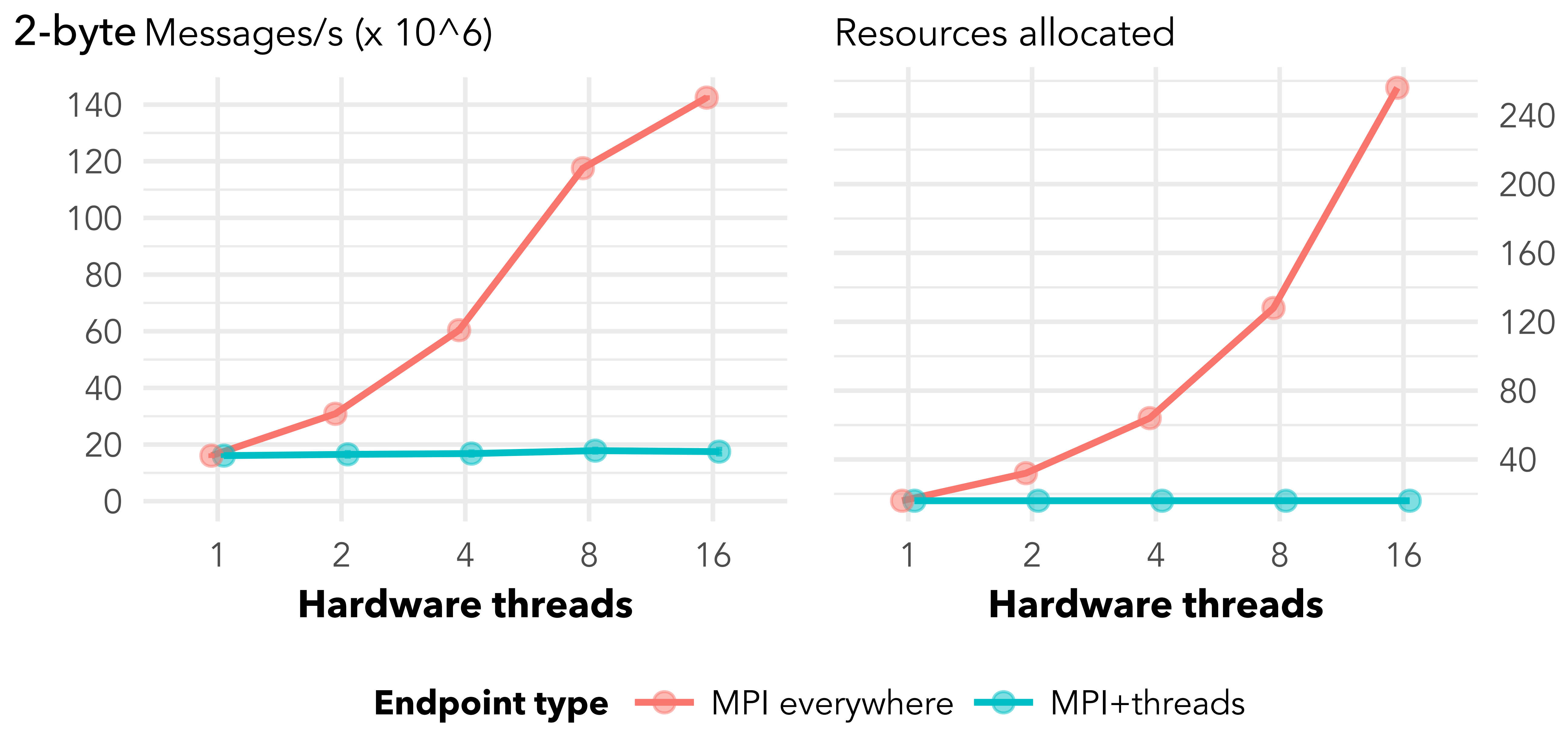
- The communication performance of MPI+threads is 9x worse than MPI everywhere
  - *Why*: the MPI library uses only 1 endpoint per process
- *Naive solution*: emulate endpoint configuration of MPI everywhere

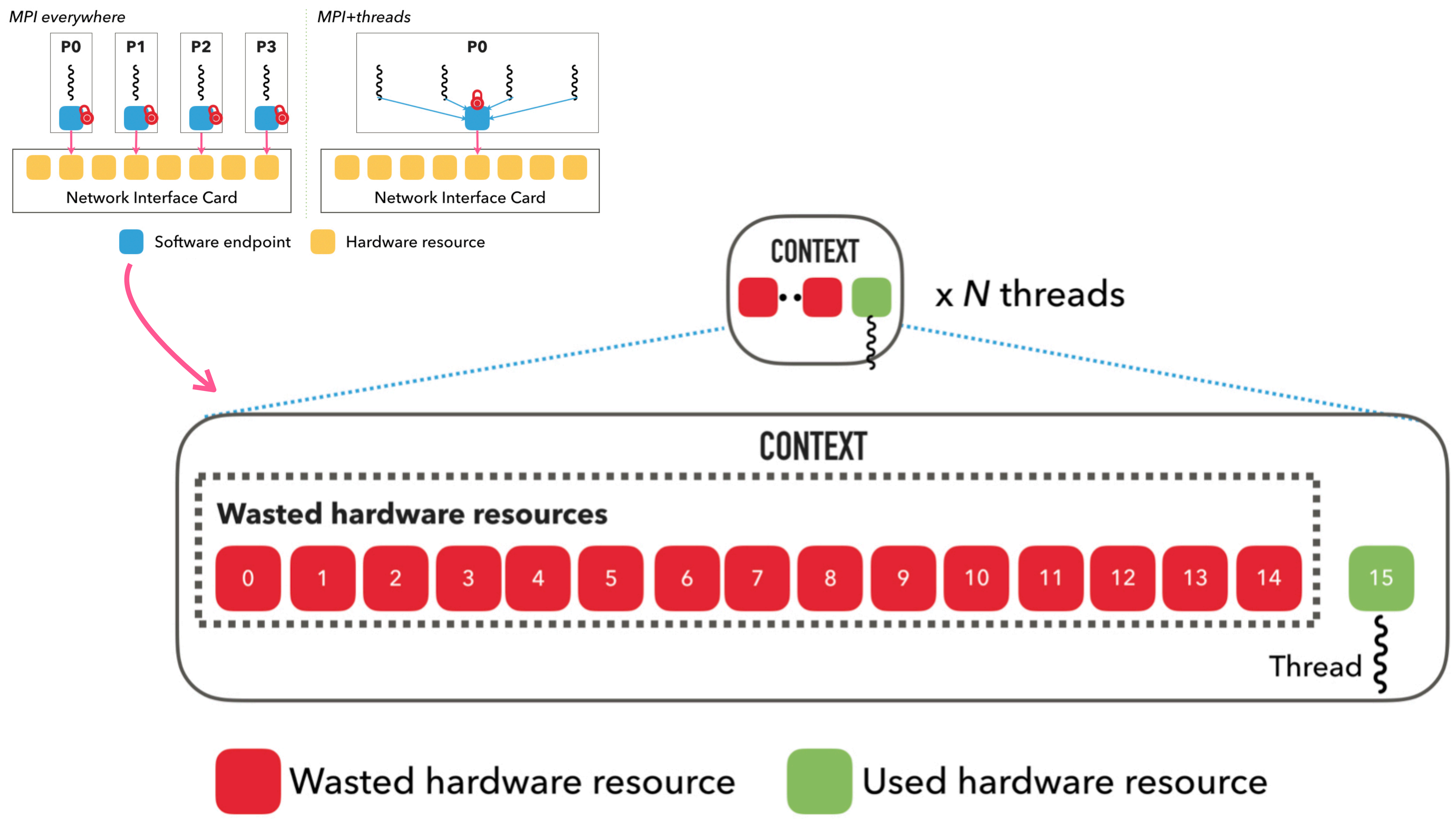




THE PROBLEM WITH TODAY'S MPI IMPLEMENTATIONS

- MPI+threads performs up to 9x worse than MPI everywhere
- MPI everywhere allocates 16x more resources than MPI+Threads and wastes 93.75% of them
  - *Why*: only 1 every 16 hardware resources used





MPI+THREADS ALLOWS RESOURCE SHARING

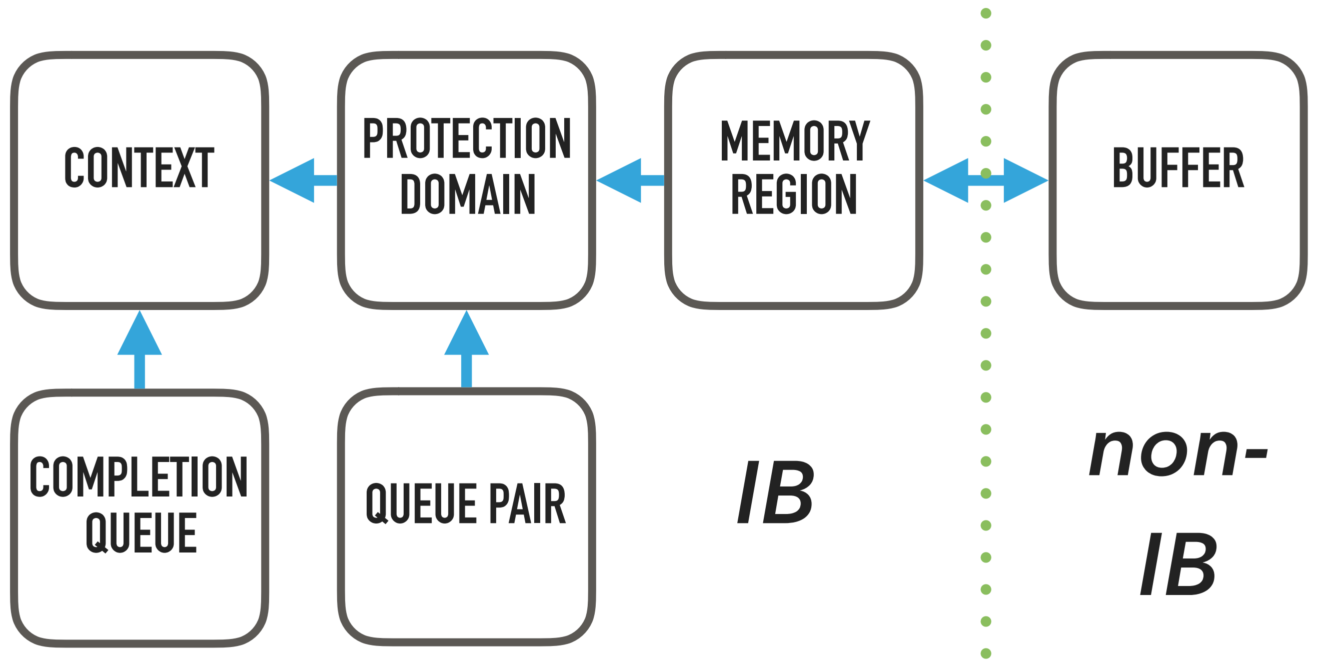
- But what level of sharing is ideal?
- Depends on performance requirements and resource availability.
- There exists a tradeoff space between performance and sharing resources
  - *Scalable Communication Endpoints*: a resource sharing model that concretely categorizes the tradeoff space ranging from fully independent to fully shared paths.

COMMUNICATION RESOURCES IN INFINIBAND

- We work with the InfiniBand communication stack
  - InfiniBand, the preferred interconnect on the TOP500 and for AI and HPC applications.
- **Transmit Queue**: Verbs' Queue Pair (impacts memory usage)
- **Completion Queue**: Verbs' Completion Queue (impacts memory usage)
- **Hardware resources**: micro User Access Region (uUARs) within UAR pages in Mellanox InfiniBand (impacts usage of limited hardware resources)

RESOURCE-SHARING ANALYSIS

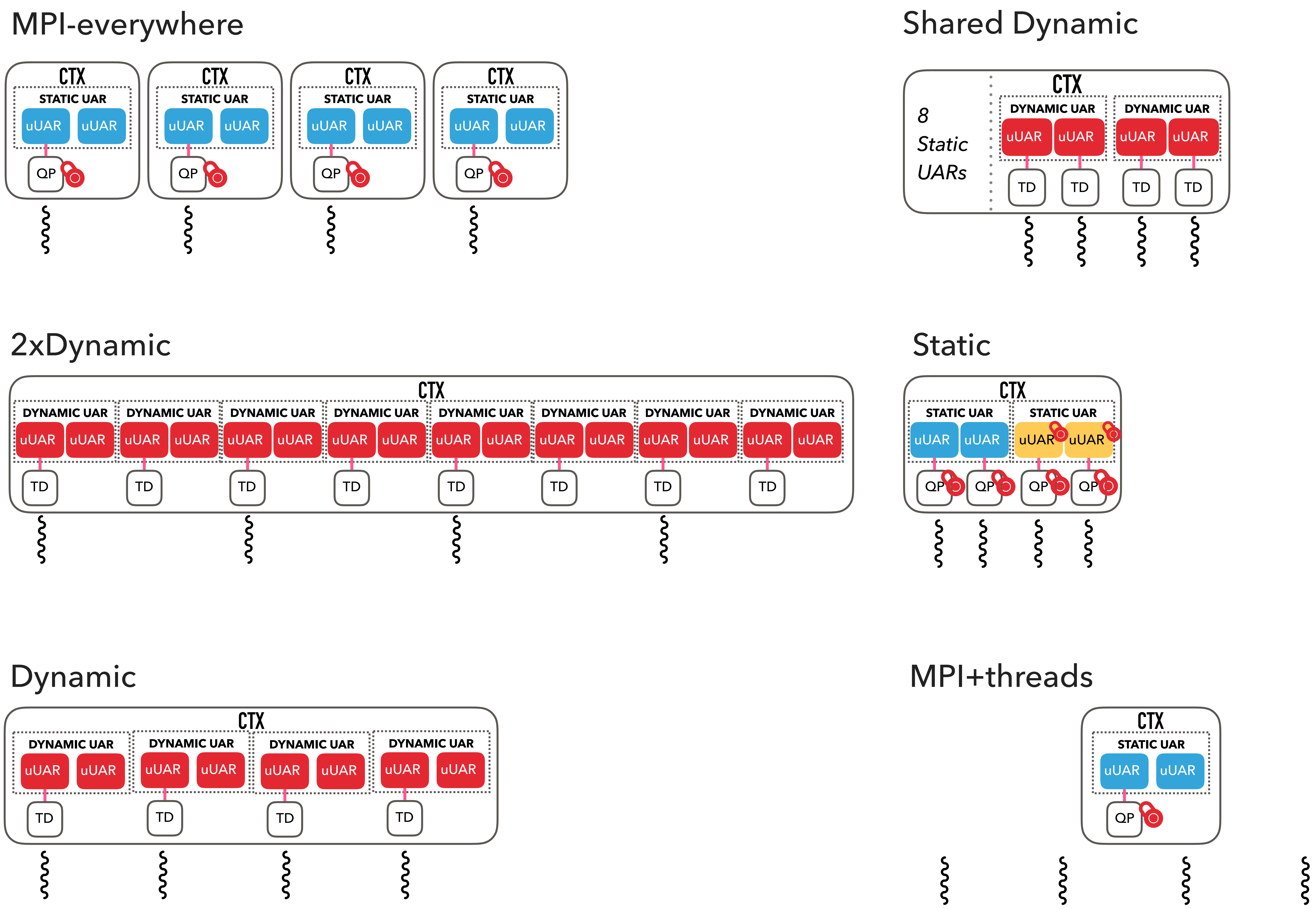
- Analyze the impact of sharing resources on performance and resource usage
  - The Verbs API exposes 6 levels of sharing



- Lessons learned
  - Each thread must have its own cache-aligned buffer
  - Can use Protection Domain and Memory Region at will
  - Sharing the Context is the most critical for hardware resource usage
  - Only QP- and CQ-sharing impact memory usage

SCALABLE ENDPOINTS

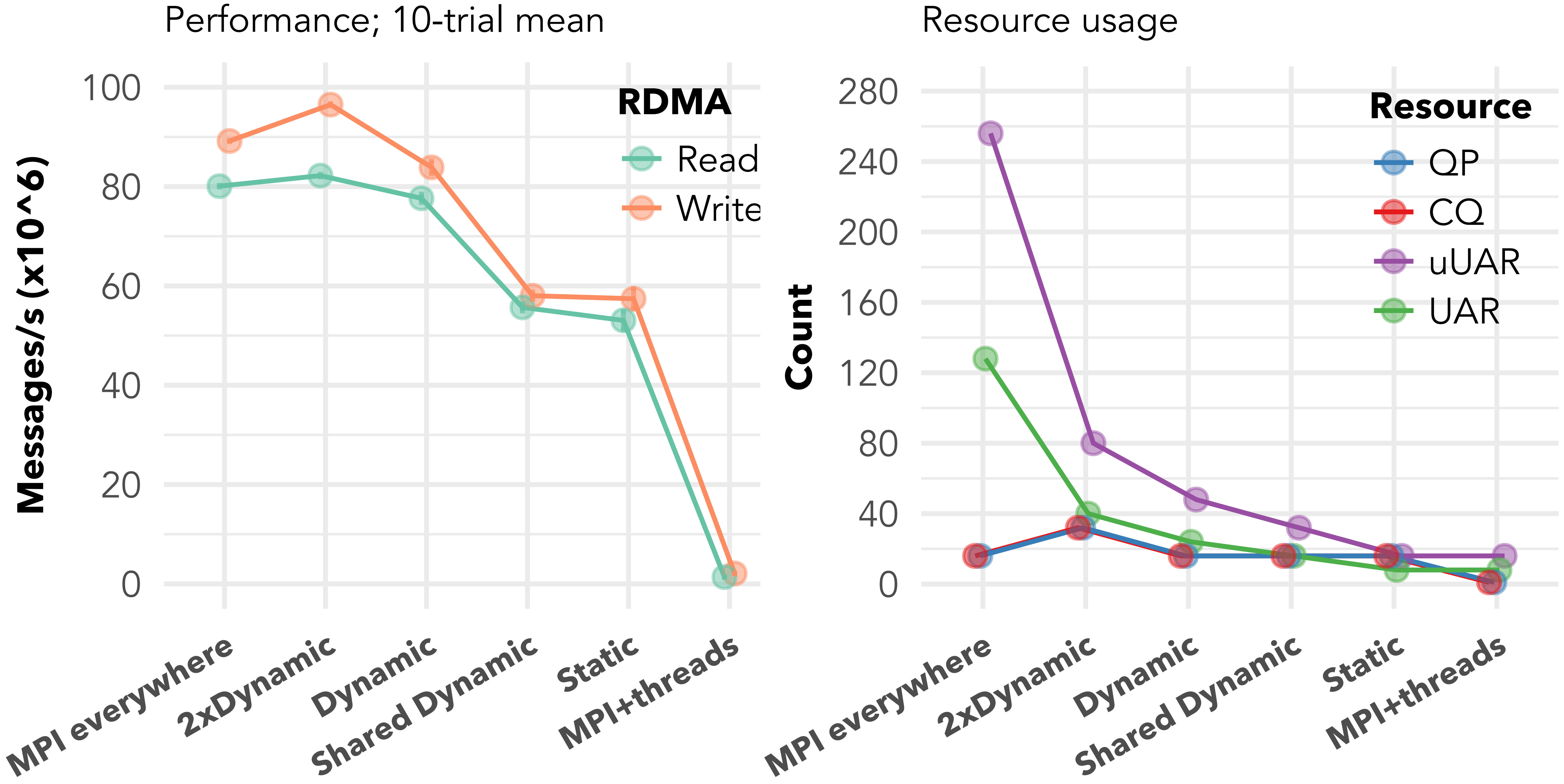
- Categorize the sharing space into 6 categories from maximum independence to maximum sharing



GLOBAL ARRAY EVALUATION

	Performance	Hardware resources	Memory resources
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%
	Higher is better	Lower is better	Lower is better

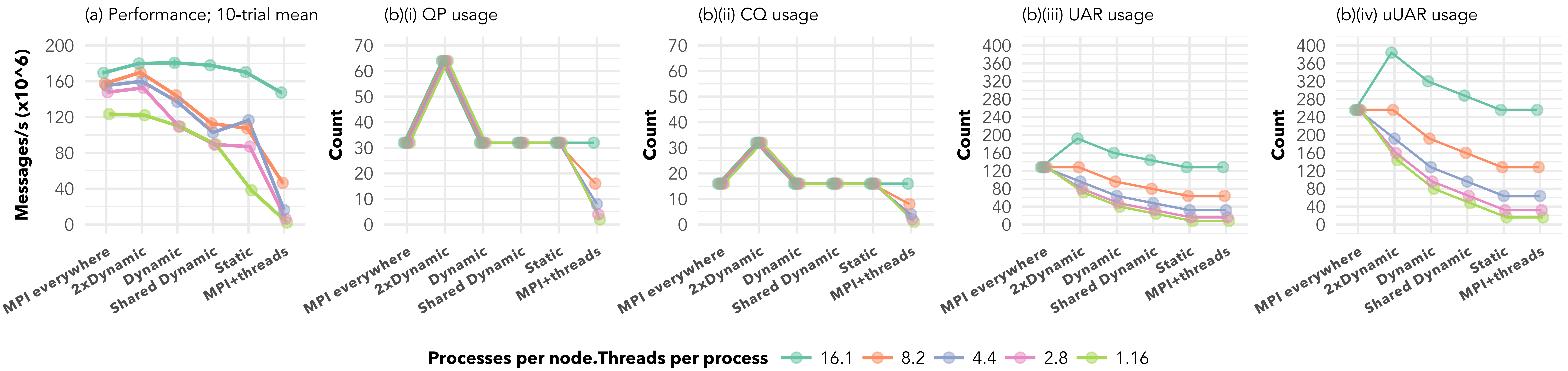
16 threads



Performance decreases with increasing resource efficiency

STENCIL EVALUATION

- Similar trend of decreasing performance with increasing resource sharing
- More processes means more messages, hence higher message-rate
- Details in https://github.com/rzambre/research-docs/blob/master/papers/icpads_18_preprint.pdf



CONCLUSION

- A tradeoff space between performance and communication resource usage exists.
- We design scalable endpoints, a resource-sharing model that concretely categorizes the tradeoff space into 6 categories.
- MPI+Threads with 2xDynamic, for example, achieves 100% of the performance of MPI everywhere while using only 31.25% as many resources.