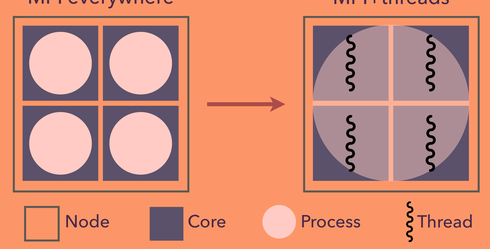
The Coming of Age of Multithreaded High-Performance Communication

Rohit Zambre, University of California-Irvine

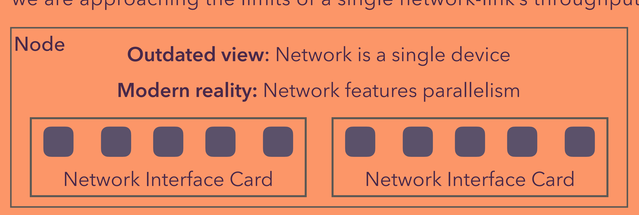
SC20 Everywhere we are | more than hpc.

Premise

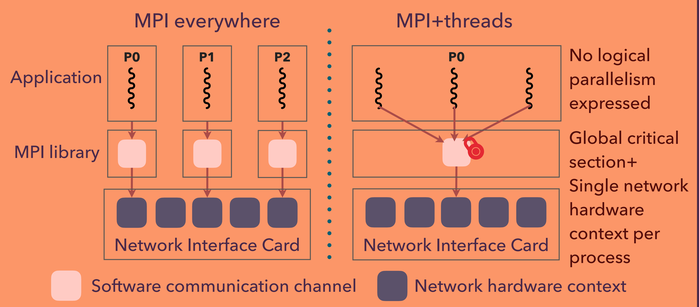
MPI+threads is the new normal on modern supercomputers
- Disproportionate number of cores compared to other resources
- Where MPI everywhere runs out of memory, MPI+threads runs



Increase in network parallelism in high-performance interconnects
- Awareness of this trend not yet commonplace in HPC
- NICs feature multiple network hardware contexts
- Multiple NICs per node to accommodate for higher bandwidth as we are approaching the limits of a single network-link's throughput



MPI+threads communication performance is dismal due to the outdated view of both domain scientists and MPI developers
- Apps typically do not express logical parallelism in communication
- MPI libraries still use conservative approaches to maintain MPI's ordering constraints
- Outdated view does not hurt MPI everywhere since network parallelism is transparently utilized through multiple processes

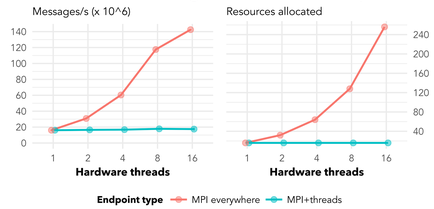


Contributions

Holistic, bottom-up analyses and solutions
- **Low-level analysis** on the capabilities of modern hardware
- **Middleware:** A fast MPI+threads library
- **Application-level outlook** on expressing logical parallelism

Modern hardware capabilities

Guiding question: **What are the differences when threads drive network resources instead of processes?**
- Larger design space in a multithreaded environment than a multiprocess
- State of the art design in MPI libraries represents two extremes



MPI everywhere
- Highest performance
- Lowest resource efficiency

MPI+threads
- Lowest performance
- Highest resource efficiency

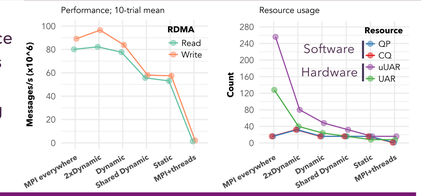
Multithreaded environments allow for arbitrary level of resource sharing
- What is ideal? **Depends** on performance requirements and resource availability

Communication resources:
- **Hardware contexts** (registers): limited; eg. 8K on Mellanox InfiniBand, 160 on Intel Omni-Path
- **Software channels**: limited by memory; not as concerning

Key lessons from a resource-sharing performance analysis
- Threads must use cache-aligned buffers
- Verbs Context most impactful on hardware resource usage

Scalable Communication Endpoints, a resource-sharing model that characterizes performance from maximum resource independence (MPI everywhere) to maximum resource sharing (MPI+threads). **IEEE ICPADS'18**
- **Multiple threads can achieve maximum performance using just a third of the resources**

Performance decreases with increasing resource efficiency

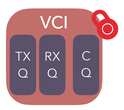


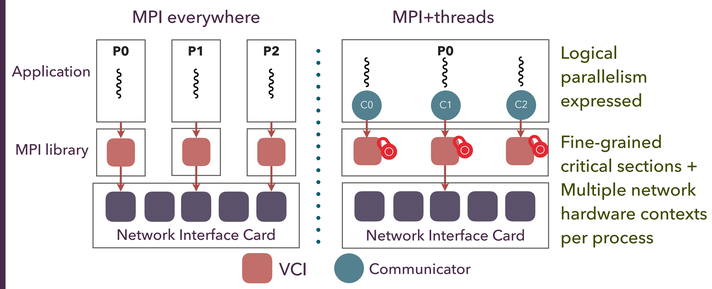
Endpoint type	Performance	Hardware resources	Software resources
MPI everywhere	100%	100%	100%
2xDynamic	100%	31.25%	200%
Dynamic	94%	18.75%	100%
Shared Dynamic	65%	12.5%	100%
Static	64%	6.25%	100%
MPI+threads	3%	6.25%	6.25%

Conversation with the MPI community

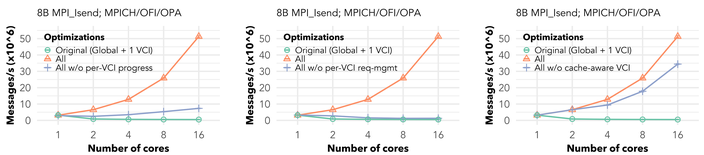
Capstone question: **How can domain scientists utilize network parallelism?**
- Explicit approach through user-visible MPI Endpoints (proposed as an extension)
- Implicit approach using the existing MPI objects (communicators, windows)
- **Do we really need user-visible endpoints?**

Parallel communication streams in the MPI library
- Virtual communication interfaces (VCIs) for network parallelism
- Fine-grained critical sections for parallel access from threads
- Map user-expressed parallelism to pool of VCIs



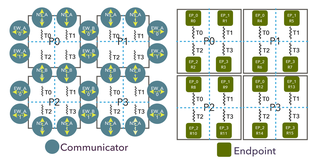


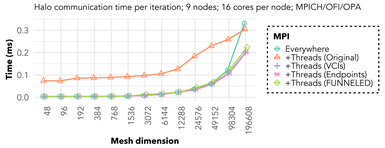
Optimizations to architectures of MPI libraries to extract multi-VCI performance:



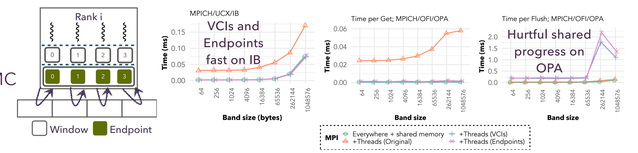
How I Learned to Stop Worrying about User-Visible Endpoints and Love MPI, ACM ICS'20
- Performance difference between implicitly using VCIs and explicitly using Endpoints depends on communication pattern

Category 1: Direct use of parallel communication streams
- VCIs as good as user-visible Endpoints
- Example: Point-to-point neighborhood (eg. stencil applications)
- Exposing parallelism with MPI-3.1 can be clumsy for point-to-point operations

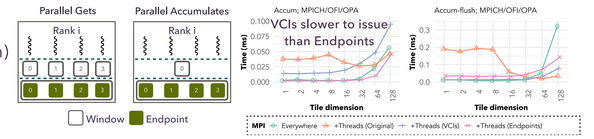




Category 2: Requiring shared progress
- VCIs and user-visible endpoints perform poorly
- Example: Energy Band Memory Server of OpenMC
- Shared progress between threads required for correctness



Category 3: Limiting MPI-3.1 semantics
- User-visible endpoints perform better than VCIs
- Example: Block-sparse matrix multiplication (NWChem)
- Semantics of MPI-3.1 prevent user from explicitly exposing parallelism for Accumulates (possible for implicit parallelism with hints)

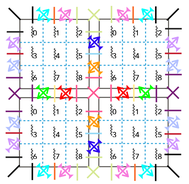


Implicit versus Explicit approaches: A design comparison (ongoing survey)

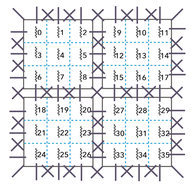
Concerns with Implicit approach: (1) Complex to express parallelism in real applications, unlike MPI Endpoints; **(2)** Adds the onus of completing the intra-node portion of a collective on the user, unlike MPI Endpoints

Concerns with Explicit approach: The term "Endpoints" is misleading and MPI users mistakenly assume that it is their job to map to network resources

If not MPI Endpoints, how about MPI Parapoints? Decouple roles of user-visible endpoints in expressing logical parallelism and mapping to network resources

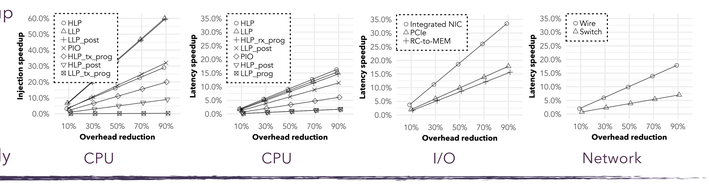


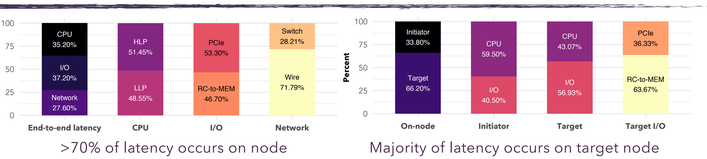
Halo with communicators (represented by lines)



Halo with MPI Endpoints (addressable threads)

Simulated optimizations: how much speedup if we optimize component X by Y%?
- NIC integrated into SoC most impactful on latency
- Optimizing Programmed I/O most impactful for injection
- Substantial network optimizations unlikely





Initiation dominates injection overhead

>70% of latency occurs on node

Majority of latency occurs on target node

Key insights from breakdown on Arm-based server