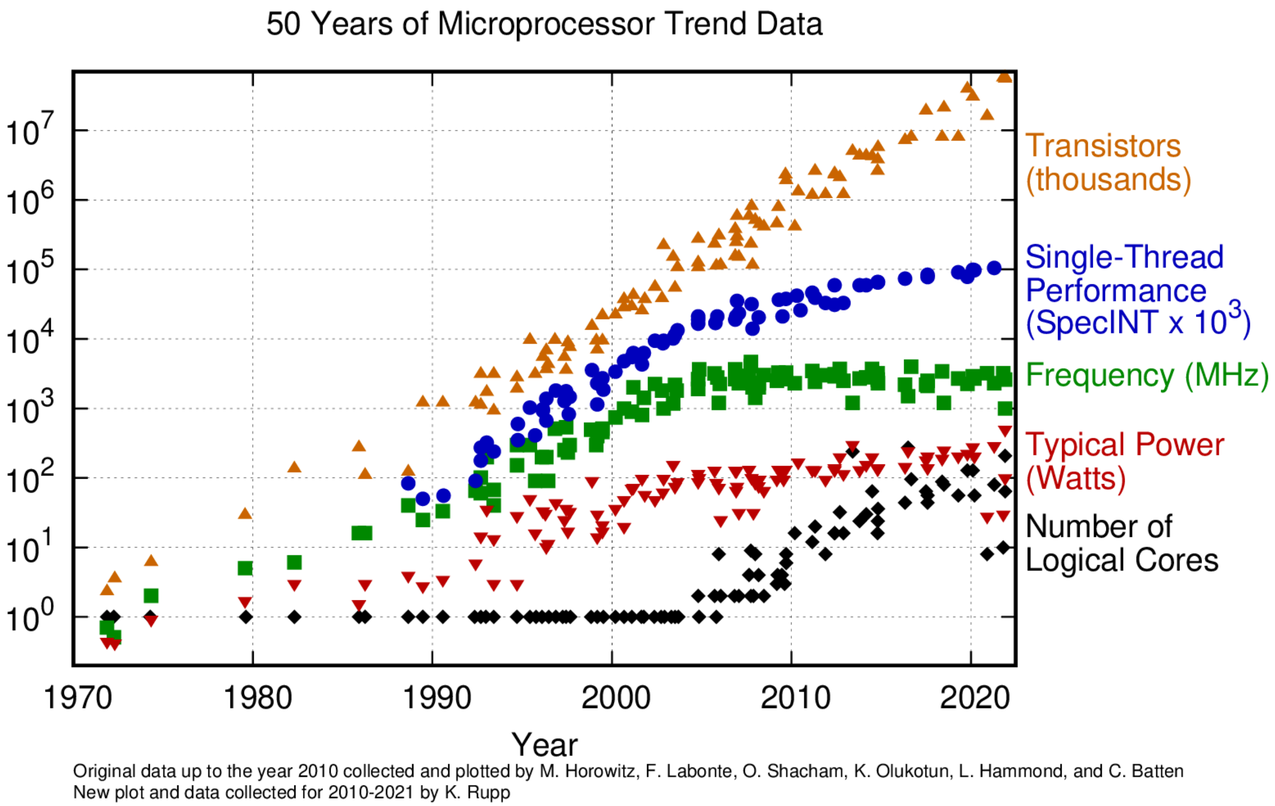
SC22
Dallas, TX | hpc accelerates.

Lessons Learned on MPI+Threads Communication

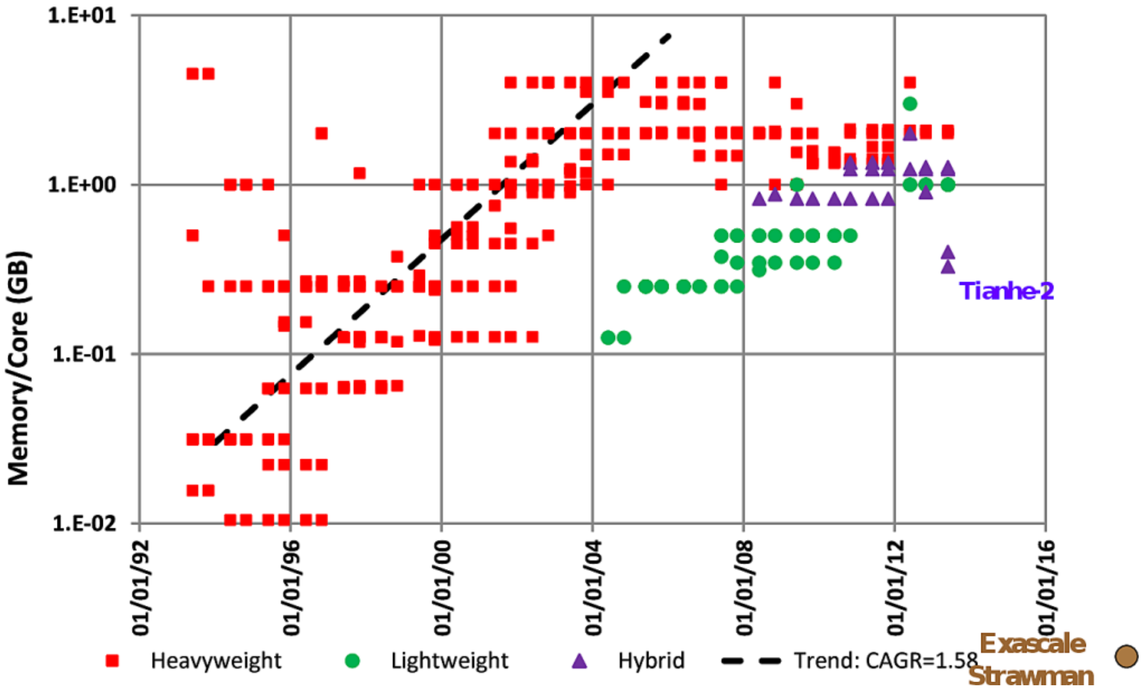
Rohit Zambre, and Aparna Chandramowlishwaran

University of California, Irvine

Why does MPI+Threads matter?



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2021 by K. Rupp

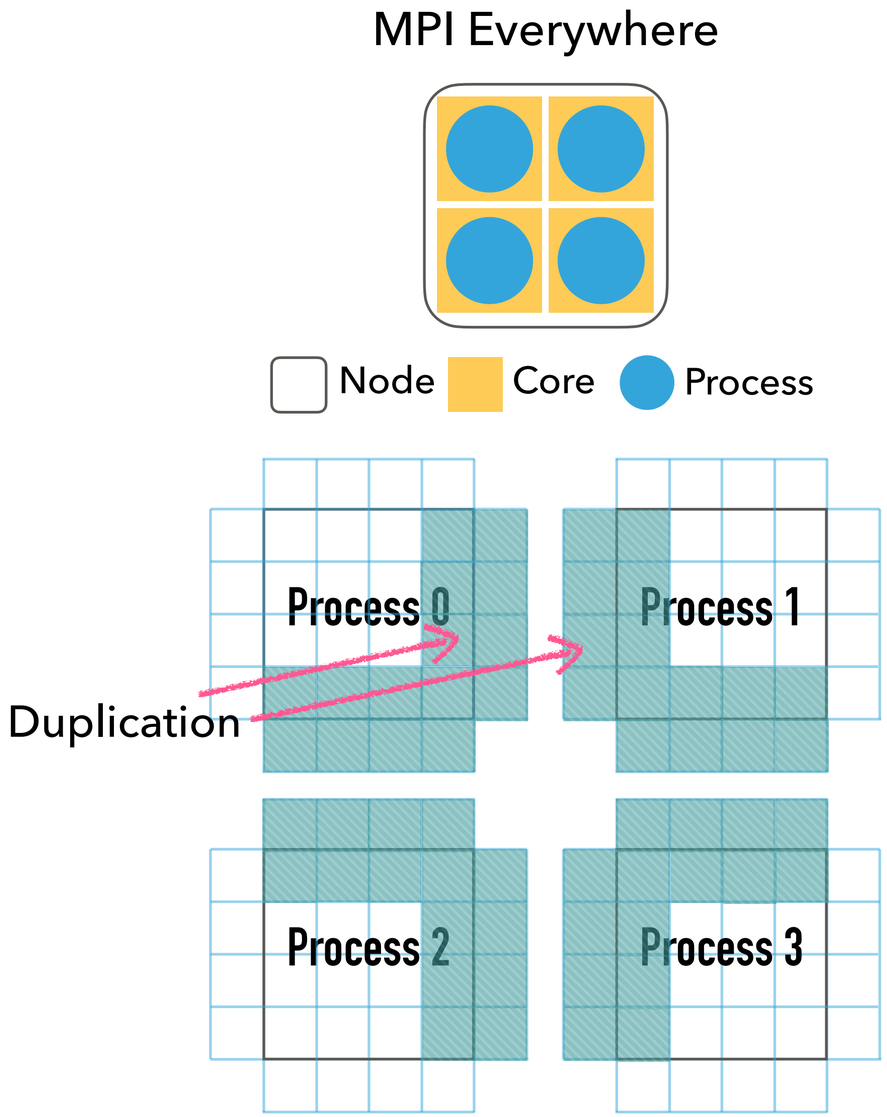


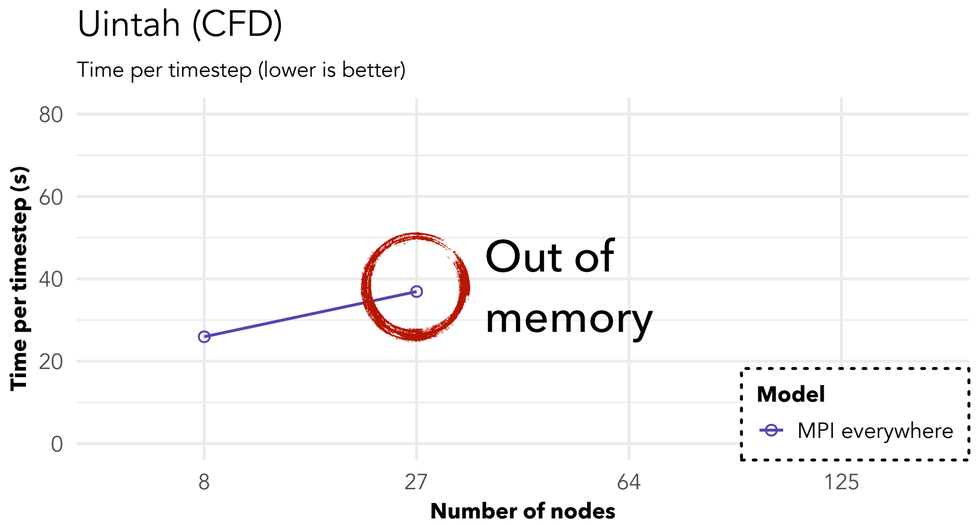
Increasing number of cores

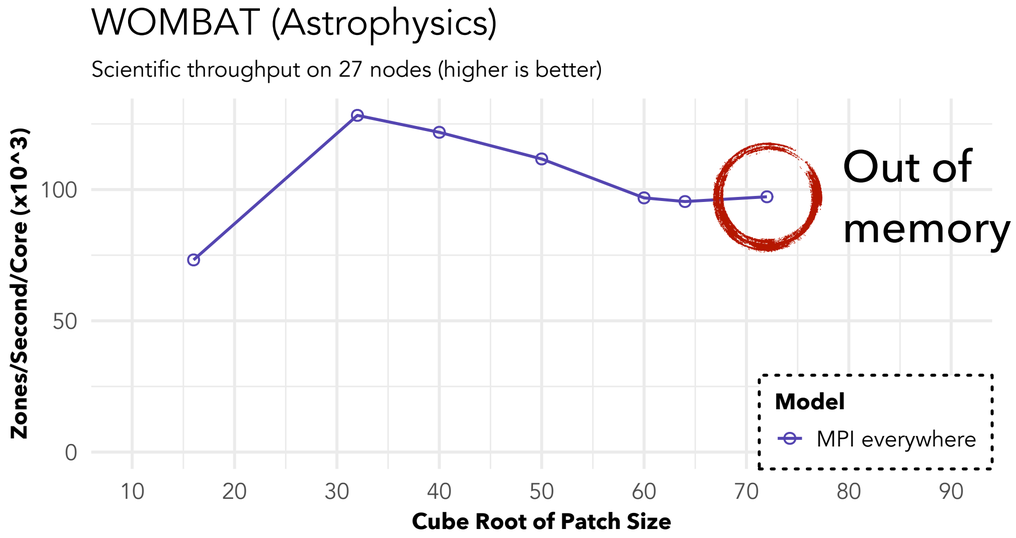
Decreasing memory per core

Image: https://github.com/karlrupp/microprocessor-trend-data

Why does MPI+Threads matter?

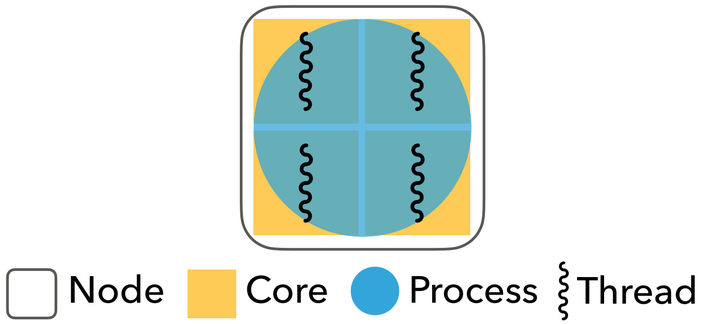




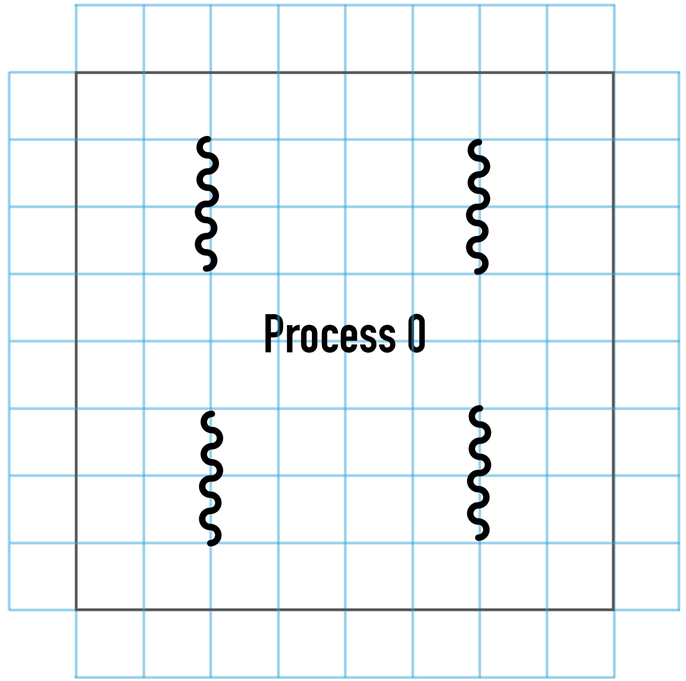


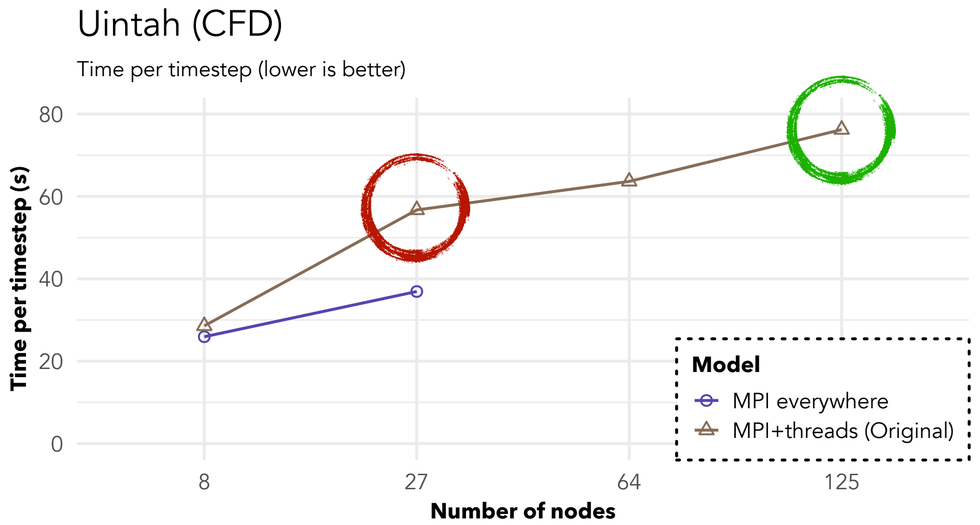
Why does MPI+Threads matter?

MPI+Threads

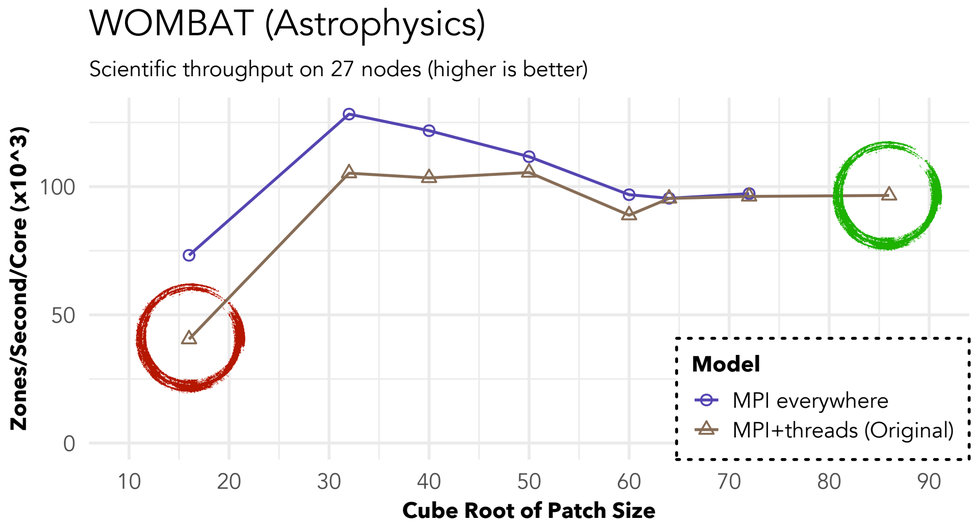


Reduced data duplication

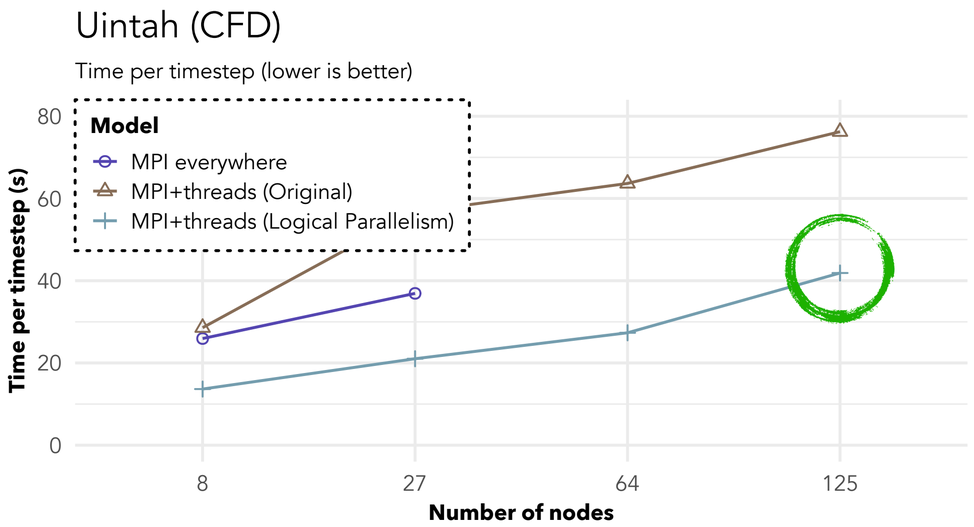


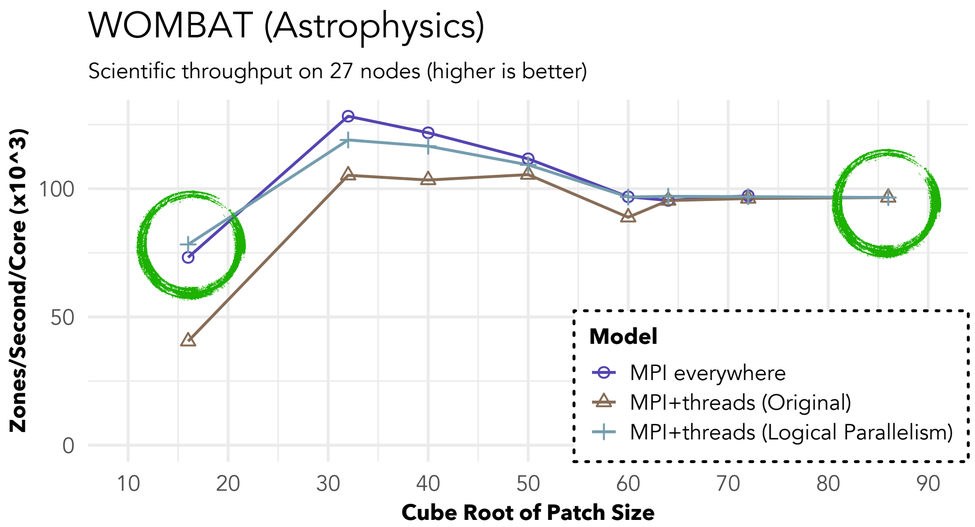


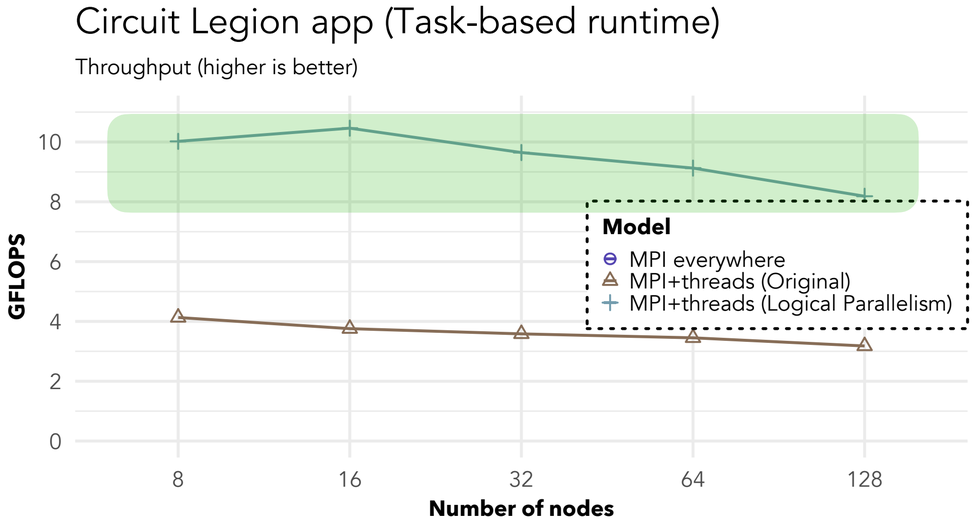
MPI+Threads scales but runs slower



MPI+Threads can achieve best of both worlds







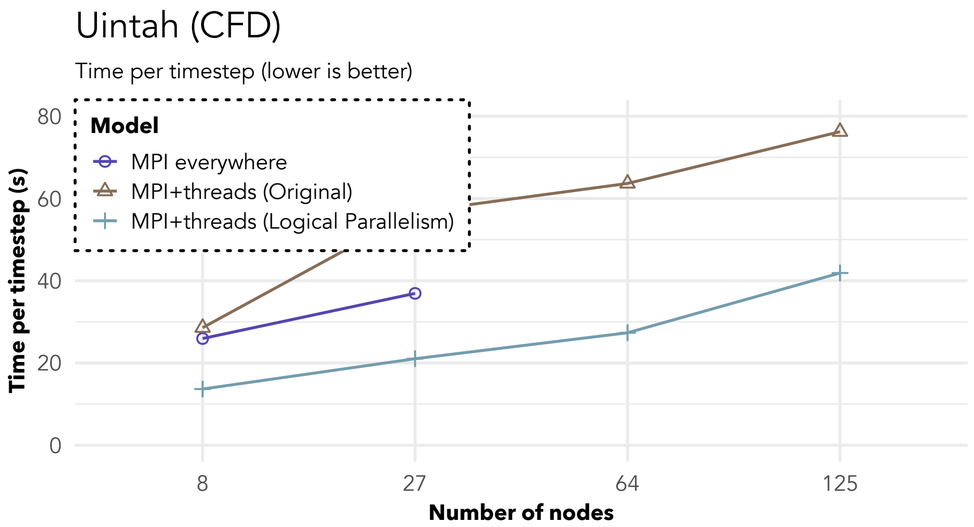
Key: logically parallel communication

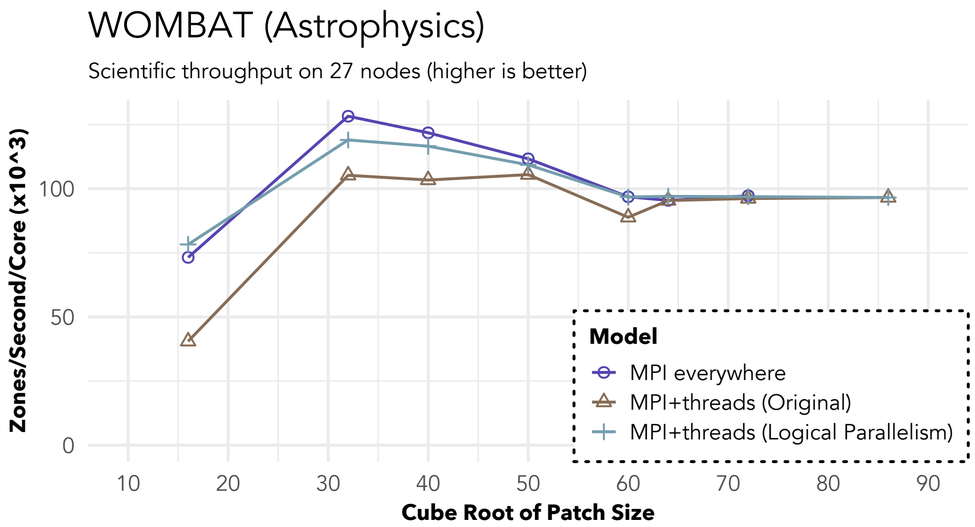
New fast MPI+threads libraries

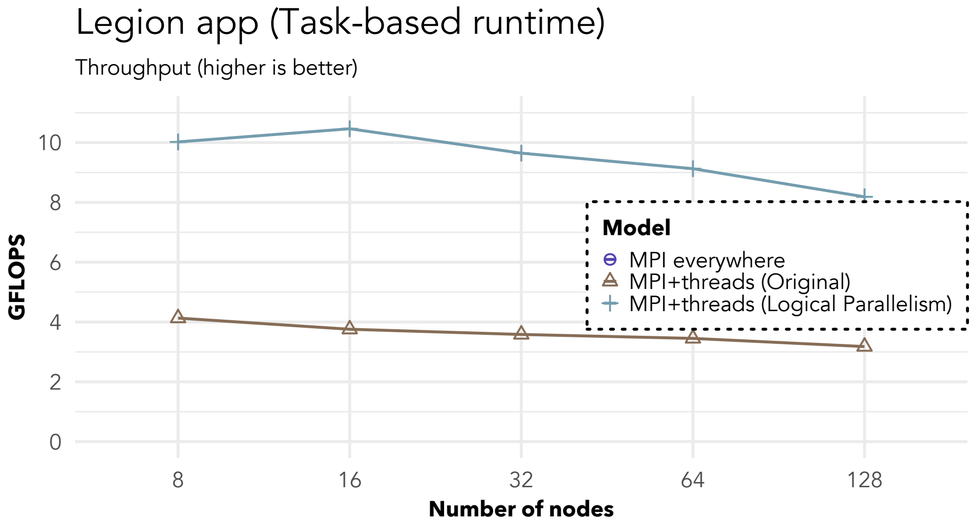
- MPICH VCIs, Open MPI CRIs, Intel MPI

Domain scientists expose communication parallelism to fast MPI+threads libraries

MPI+Threads can achieve best of both worlds







Key: logically parallel communication

New fast MPI+threads libraries

- MPICH VCIs, Open MPI CRIs, Intel MPI

★Domain scientists expose communication parallelism to fast MPI+threads libraries

Which design is best suited for domain scientists, MPI's end users?

Applications considered

Interacted with app developers of each

- Uintah using HYPRE (stencil)
- Smilei (stencil)
- Vite (graph)
- WOMBAT (RMA)
- NWChem (RMA)
- Legion (task-based runtime)

“Its not about the cores, its not about the network, its about the people.”

Dorian Arnold
Reinventing High-Performance Computing, SC’22 panel

Lessons Learned on MPI+Threads Communication

Outline

High performance and high scalability with MPI+threads

Designs

Lessons learned

- Parallelism vs. MPI semantics
- Intuition
- Heterogeneous computing

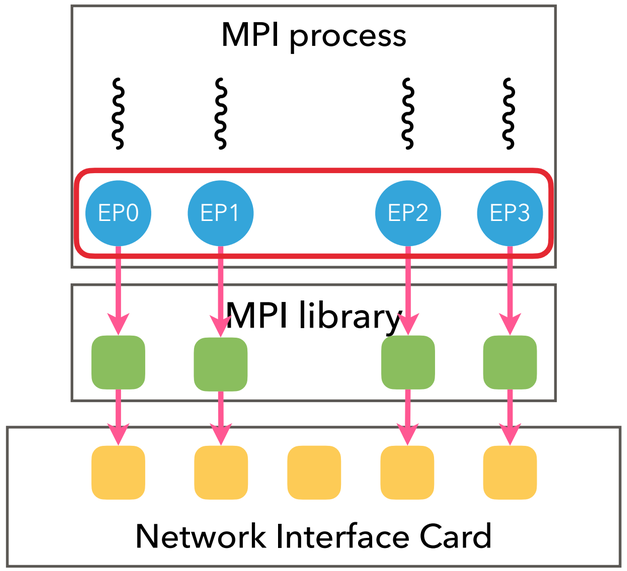
Takeaways

Ways to expose logically parallel communication

- User-visible solutions: Ports (Foster et al., 1996), or MPI Endpoints (Dinan et al., 2013)

Control-path operations (new API):

```
MPI_Comm_create_endpoints
(parent_comm, num_ep, info,
new_comm_handles)
```



Data path operations (no new APIs):

```
MPI_Isend/Irecv
(…,dest_ep, tag, comm[ep])
```

- Oracle for performance

MPI Communicator MPI Endpoint MPI library communication context Network hardware context

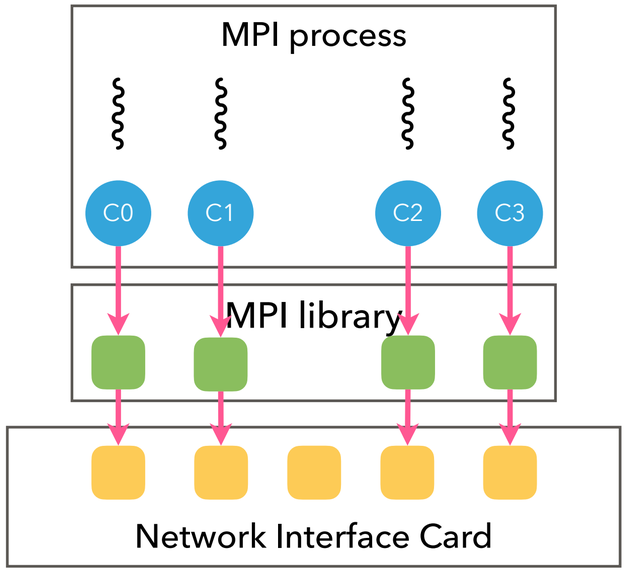
Ways to expose logically parallel communication

- User-visible solutions (Ports, or MPI Endpoints)
- Existing MPI mechanisms (MPI 4.0)
  - Communicators/Tags for point-to-point and collectives, Windows for RMA

Control-path operations (no new APIs):

```
mip_assert_no_any_source
mpi_assert_no_any_tag
mpi_accumulate_ordering_none

<mpi_library_specific_hints>
```



Data path operations (no new APIs):

```
MPI_Isend/Irecv
(…,dest, tag, comm[tid])
```

- Equivalent performance to user-visible endpoints (Zambre et al., 2021)

MPI Communicator

MPI library communication context

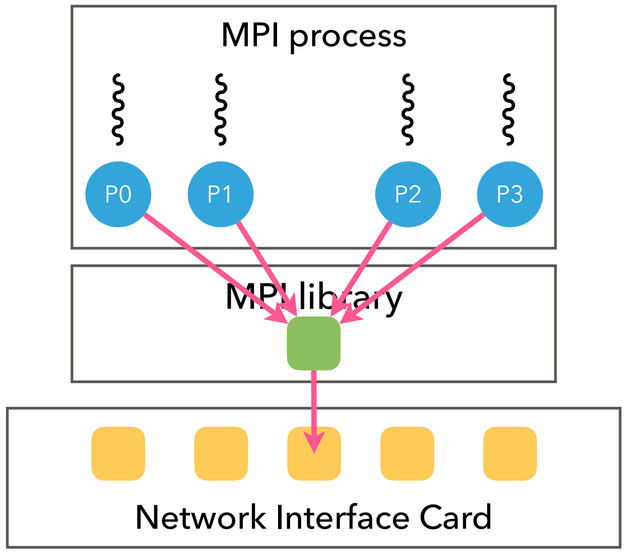
Network hardware context

Ways to expose logically parallel communication

- User-visible solutions (Ports, or MPI Endpoints)
- Existing MPI mechanisms (MPI 4.0)
- Partitioned operations (MPI 4.0)

Control-path operations (new APIs):

```
MPI_Psend_init(<partition_info>,
               <dest_info>)
MPI_Precv_init(<partition_info>,
               <src_info>))
```



Data path operations (new APIs):

```
MPI_Pready  (partition,request)
MPI_Parrived(partition,request,flag)
```

- Outperforms “MPI+Threads (Original)” (Grant et al., 2019)

MPI Partition

MPI library communication context

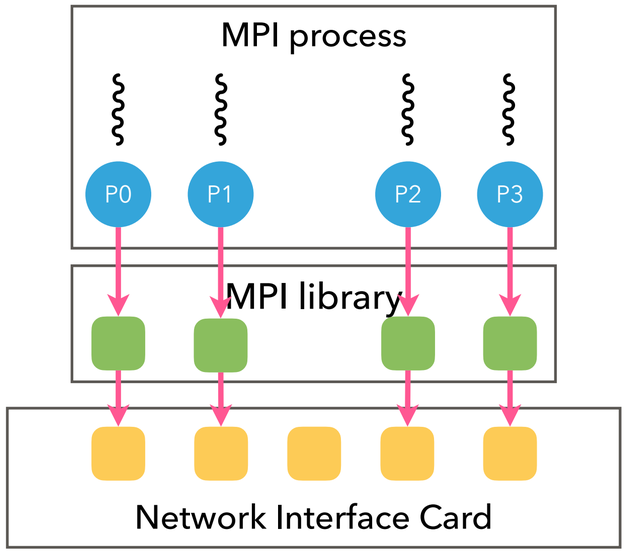
Network hardware context

Ways to expose logically parallel communication

- User-visible solutions (Ports, or MPI Endpoints)
- Existing MPI mechanisms (MPI 4.0)
- Partitioned operations (MPI 4.0)

Control-path operations (new APIs):

```
MPI_Psend_init(<partition_info>,
               <dest_info>)
MPI_Precv_init(<partition_info>,
               <src_info>))
```



Data path operations (new APIs):

```
MPI_Pready  (partition,request)
MPI_Parrived(partition,request,flag)
```

- Outperforms “MPI+Threads (Original)” (Grant et al., 2019)
- Performance studies utilizing network parallelism not yet available

MPI Partition

MPI library communication context

Network hardware context

Lessons Learned on MPI+Threads Communication

Outline

High performance and high scalability with MPI+threads

Designs

Lessons learned

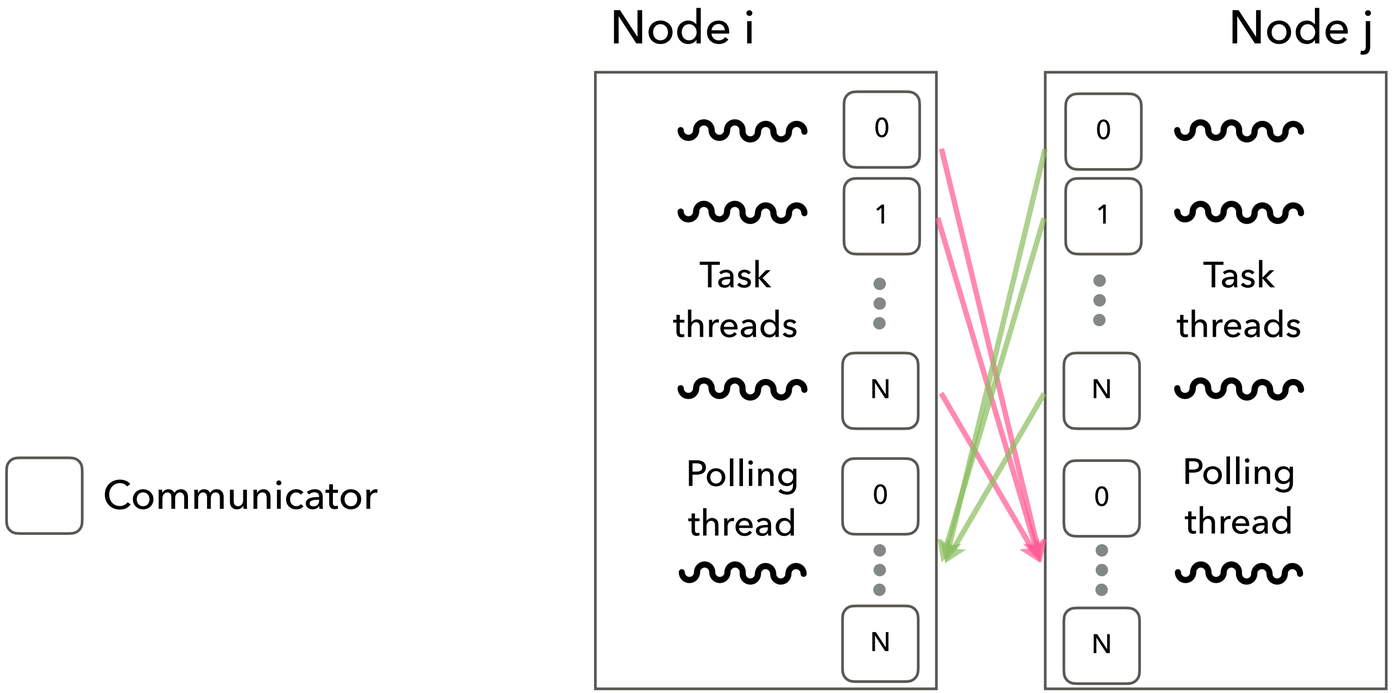
- Parallelism vs. MPI semantics
- Intuition
- Heterogeneous computing

Takeaways

Parallelism vs. MPI semantics

Point-to-point: Existing MPI mechanisms

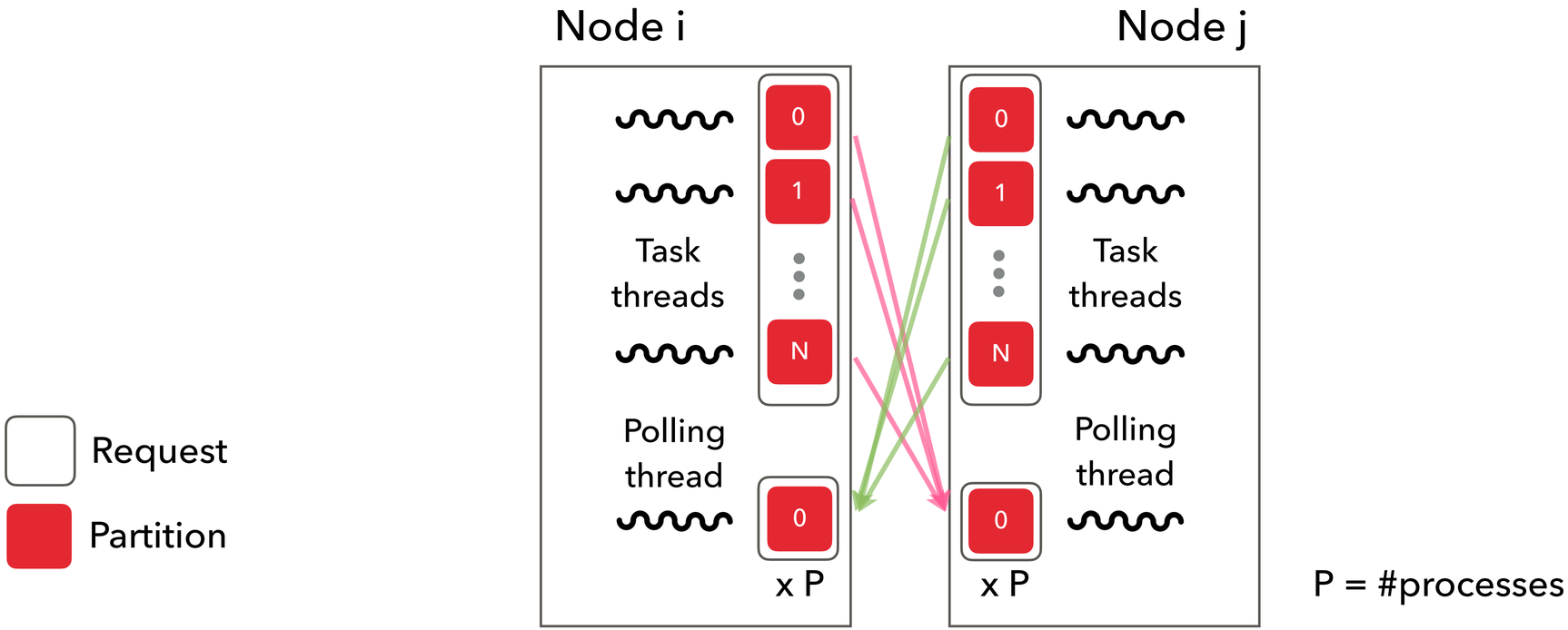
- Communicators constrain parallelism information with matching information



Parallelism vs. MPI semantics

Point-to-point: Partitioned operations

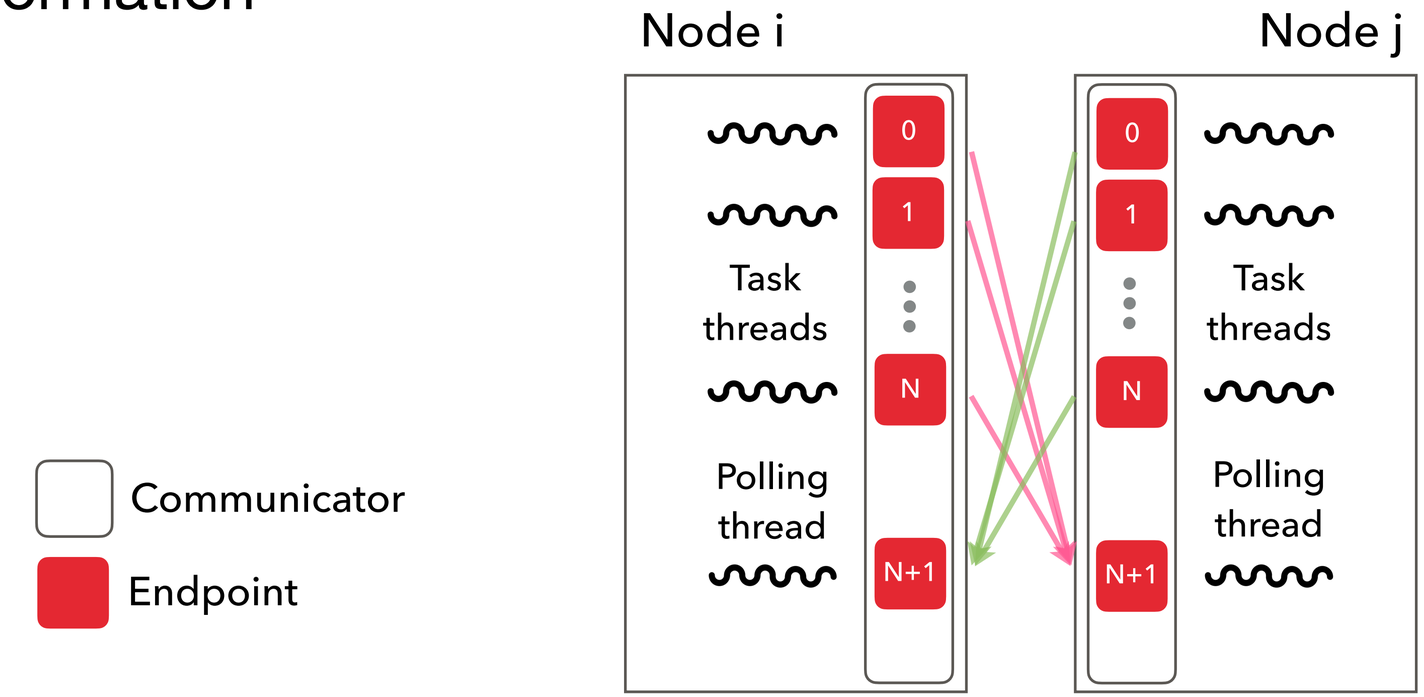
- Partitioned operations constrain parallelism information with their semantics: lack of wildcards, and shared request.



Parallelism vs. MPI semantics

Point-to-point: User-visible endpoints

- Endpoints distinguish between parallelism information and matching information

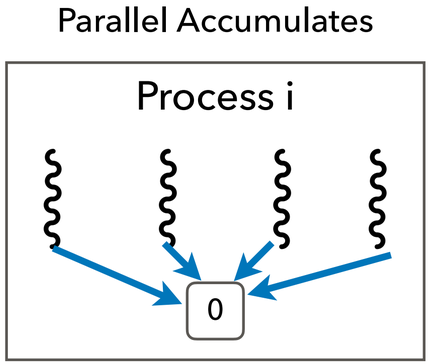


Parallelism vs. MPI semantics

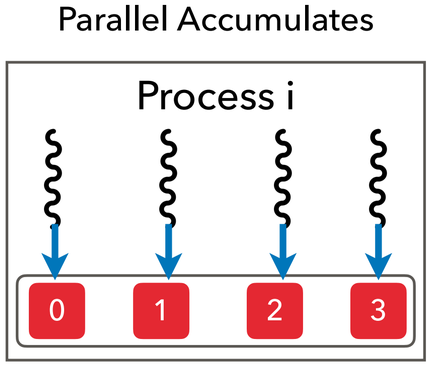
RMA: Parallel accumulates

- Windows: forced to use a single window for parallel accumulates
- Endpoints: enables parallelism *while* maintaining atomicity

Existing MPI mechanisms



User-visible endpoints

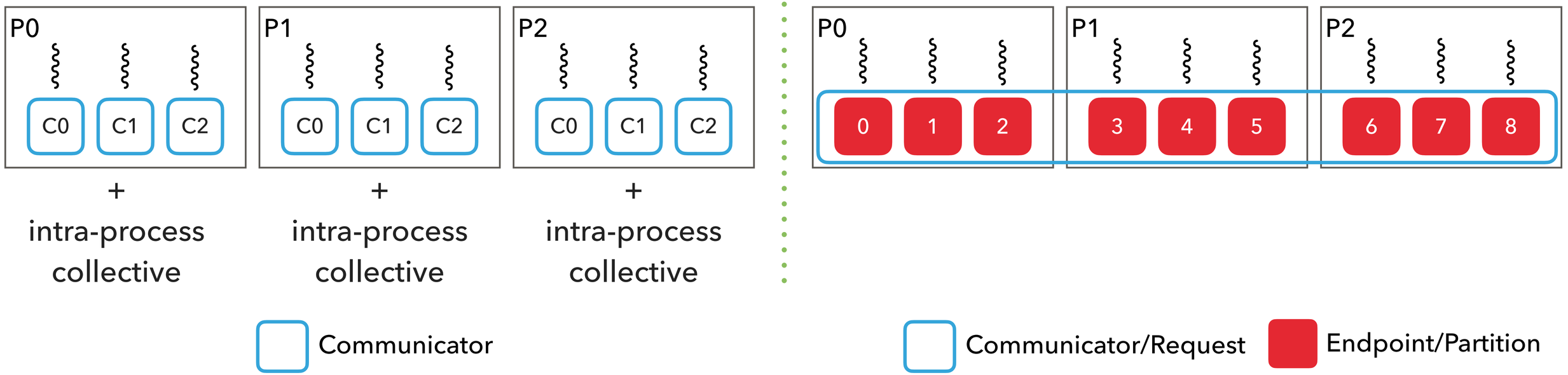


Window Endpoint

Parallelism vs. MPI semantics

Collectives

- Communicators: users forced to perform intra-process collective
- Partitions/Endpoints: offload intra-process collective to MPI library



Lessons Learned on MPI+Threads Communication

Outline

High performance and high scalability with MPI+threads

Designs

Lessons learned

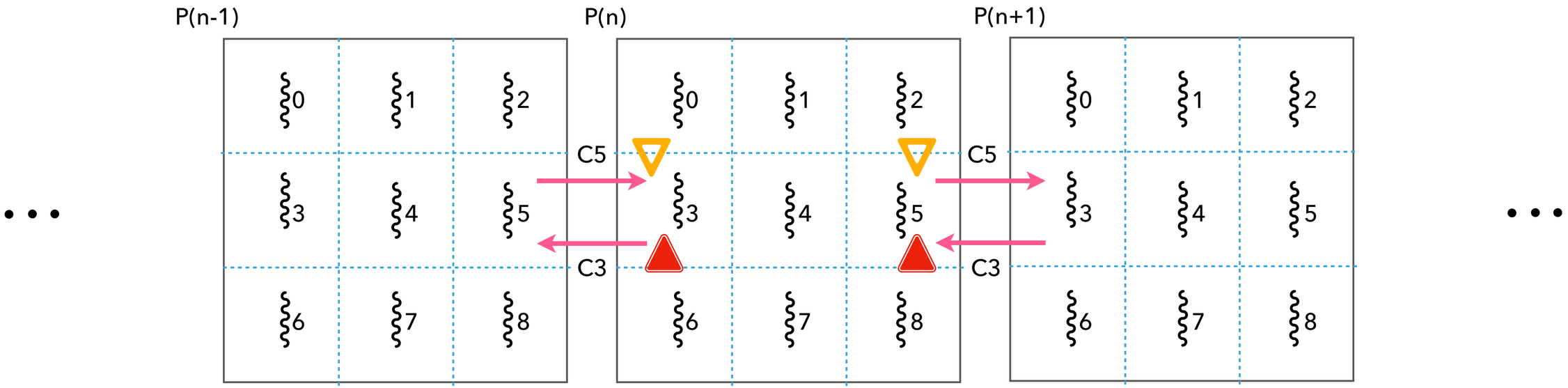
- Parallelism vs. MPI semantics
- Intuition
- Heterogeneous computing

Takeaways

Intuition

Point-to-point: Communicators

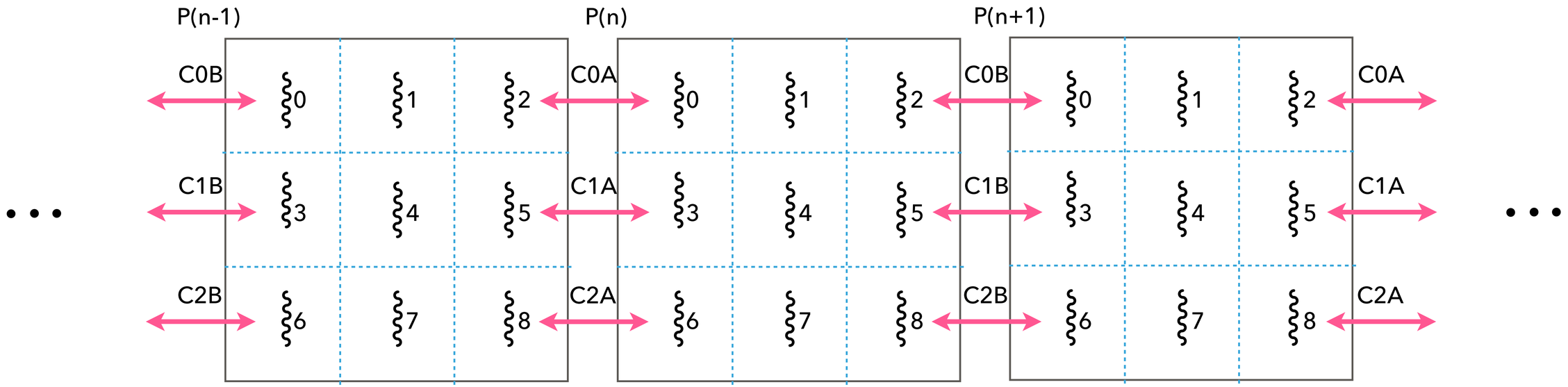
- Communicators are not intuitive to use.
  - App developers first exposed only half of the available parallelism



Intuition

Point-to-point: Communicators

- Communicators are complex to use.



Intuition

Point-to-point: Tag

- Tags are intuitive to use.
  - MPI_THREAD_MULTIPLE apps already encode thread ID information
- Optimal use of tags is tedious and non-portable across MPI libraries

```
1 /* Existing THREAD_MULTIPLE stencil apps */
2 #pragma omp parallel num_threads(N_THREADS)
3 {
4   app_tag = src_tid << (NUM_TID_BITS + NUM_APP_BITS)
5           | dst_tid <<  NUM_APP_BITS
6           | app_tag;
7   MPI_Send(dest_proc, tag, app_comm);
8 }
```

```
1 /* Leveraging parallelism info in tags of existing apps */
2 //(1) Relax unneeded MPI semantics with MPI 4.0 hints
3 MPI_Info_set(info,"mpi_assert_no_any_tag",    "true");
4 MPI_Info_set(info,"mpi_assert_no_any_source","true");
5 //(2) Achieve optimal mapping with MPI library hints
6 MPI_Info_set(info,"mpich_num_vcis",              N_THREADS);
7 MPI_Info_set(info,"mpich_num_tag_bits_vci", NUM_TID_BITS);
8 MPI_Info_set(info,"mpich_place_tag_bits_local_vci","MSB");
9 MPI_Info_set(info,"mpich_tag_vci_hash_type","one-to-one");
10 MPI_Comm_dup_with_info(app_comm, info, &tag_par_app_comm);
11 #pragma omp parallel num_threads(N_THREADS)
12 {
13   ... // (Tag encoding same as above)
14   MPI_Send(dest_proc, app_tag, tag_par_app_comm);
15 }
```

Which bits?

How to map?

Intuition

Point-to-point: Endpoints

- Endpoints are intuitive to use.
  - Domain scientists innately familiar with the concept of MPI ranks.
- Portable across MPI implementations (also with partitioned operations)

Lessons Learned on MPI+Threads Communication

Outline

High performance and high scalability with MPI+threads

Designs

Lessons learned

- Parallelism vs. MPI semantics
- Intuition
- Heterogeneous computing

Takeaways

Heterogeneous computing

All three designs

- Accelerator-initiated communication: need lightweight interfaces
- Pready/Parrived good candidates for point-to-point
  - How will they work with persistent kernels?
- Windows/endpoints good candidates for RMA (following NVSHMEM/ROC_SHMEM)

- Lessons still being learned

Lessons Learned on MPI+Threads Communication

Outline

High performance and high scalability with MPI+threads

Designs

Lessons learned

- Parallelism vs. MPI semantics
- Intuition
- Heterogeneous computing

Takeaways

Characteristics	Existing MPI mechanisms			Partitioned operations	User-visible endpoints
	Communicators	Tags	Windows		
Semantics don't hurt parallelism	One-to-one communication	Tag overflows	Accumulates	Shared request	✓
Intuitive	Complex	✓	✓	?	✓
Portable	✓	MPI library specific	✓	✓	✓
Dynamic communication	One-to-one communication	✓	✓	Persistence	✓
Resource efficient	One-to-one communication	✓	✓	No wildcards	✓
No intranode collective	User-defined collective	-	-	✓	✓
No collective buffer duplication	✓	-	-	✓	Compute collectives
Heterogenous computing	?	?	✓ / ?	✓ / ?	✓ / ?

Which design is best suited for domain scientists, MPI's end users?

Characteristics	Existing MPI mechanisms			Partitioned operations	User-visible endpoints
	Communicators	Tags	Windows		
Semantics don't hurt parallelism	One-to-one communication	Tag overflows	Accumulates	Shared request	✓
Intuitive	Complex	✓	✓	?	✓
Portable	✓	MPI library specific	✓	✓	✓
Dynamic communication	One-to-one communication	✓	✓	Persistence	✓
Resource efficient	One-to-one communication	✓	✓	No wildcards	✓
No intranode collective	User-defined collective	-	-	✓	✓
No collective buffer duplication	✓	-	-	✓	Compute collectives
Heterogenous computing	?	?	✓ / ?	✓ / ?	✓ / ?

User-visible solutions warrant reconsideration

- Goal: Make the adoption of MPI+threads seamless for domain scientists
- Potential solutions moving forward:
  - MPI Rankpoints
    - Combats preconceived notions of endpoints being direct handles to network resources
  - Promising new proposals in MPI's HACC working group
  - Addressable partitions?

Characteristics	Existing MPI mechanisms			Partitioned operations	User-visible endpoints
	Communicators	Tags	Windows		
Semantics don't hurt parallelism	One-to-one communication	Tag overflows	Accumulates	Shared request	✓
Intuitive	Complex	✓	✓	?	✓
Portable	✓	MPI library specific	✓	✓	✓
Dynamic communication	One-to-one communication	✓	✓	Persistence	✓
Resource efficient	One-to-one communication	✓	✓	No wildcards	✓
No intranode collective	User-defined collective	-	-	✓	✓
No collective buffer duplication	✓	-	-	✓	Compute collectives
Heterogenous computing	?	?	✓ / ?	✓ / ?	✓ / ?

Extra

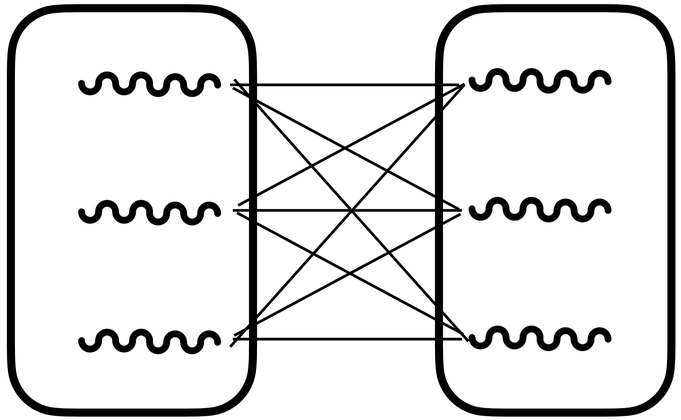
Point-to-point lessons

Communicators

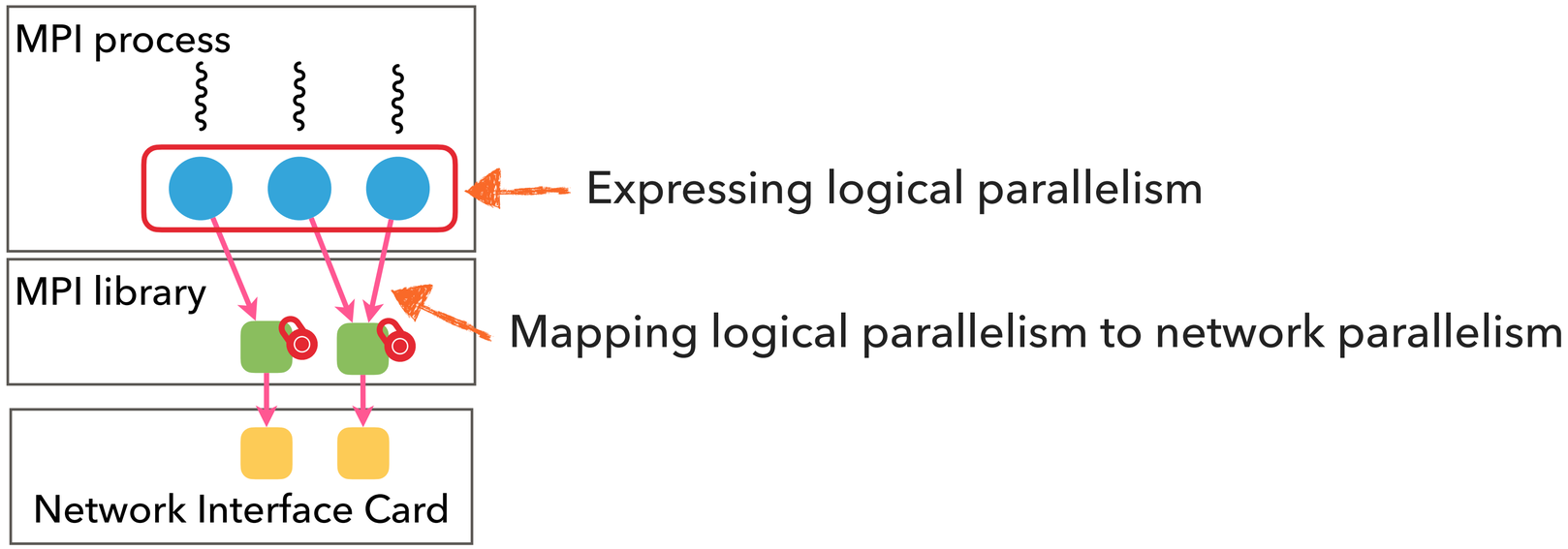
- High network resource requirements

  For all-to-all communication between N threads per process: N^2

  Example: For 96 threads per process, we need 9216 communicators

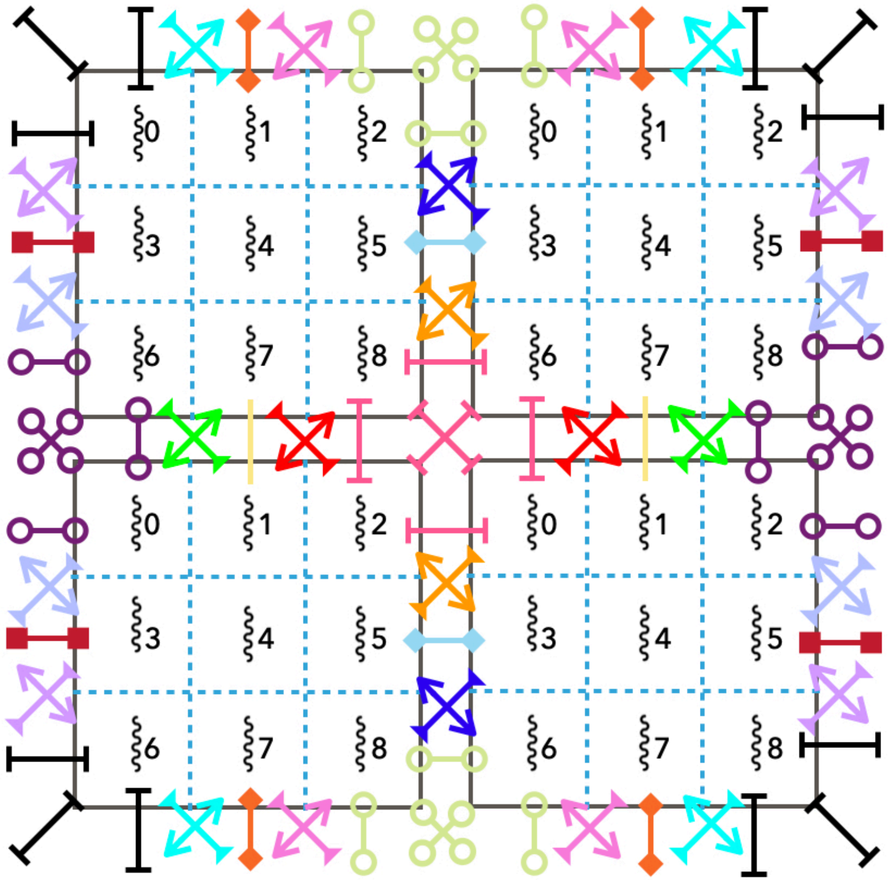


MPI Rankpoints to combat domain scientists' concerns



MPI Communicator MPI Rankpoint Virtual Communication Interface Network hardware context

Category 2: hypre for Uintah



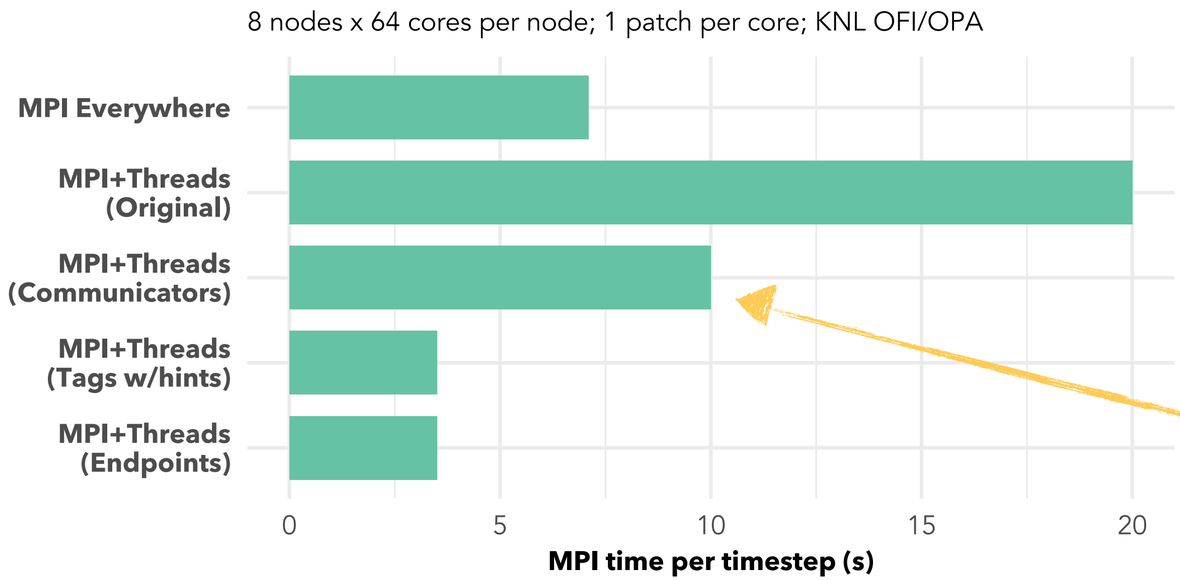
Communicators

- #bi-directions x #communicating threads x 2 (odd-even)
- If [X,Y,Z] represents cubic arrangement of threads

#communicators
= 2 * (XY + YZ + XZ ← Faces
+ 4(XY+YZ+XZ - 1) ← Corner diagonals
+ 2(XZ + YZ - Z) + 2(XY+YZ-Y) + 2(XY+XZ-Z))

Edge diagonals

Category 2: hypre for Uintah



Intel KNL node: 64 cores ([4,4,4] threads)

Intel Omni-Path network: 160 contexts

Ideal number of VCIs required

- Endpoints: 64
- Communicators: 808
- Tags with hints: 64

Point-to-point lessons

- Stencil: very complex to use communicators, high resource usage
  - Partitioned operations
  - Endpoints separate out matching information from parallelism information
- Legion: very complex to use communicators, or partitioned operations
  -

RMA lessons

- Accumulate

Collective lessons

- User needs to perform an extra step with existing mechanisms
- Endpoints help but can lead to duplication
- Partitioned operations can achieve best of both
  - However duplication of collective data has not yet caused apps to run out of memory

Heterogeneous computing

- Accelerator-initiated communication
- Needs lightweight communication.
- Partitioned operations allow for that.
  - The Wait can be enqueued
  - But how this would work with persistent kernels.

“Rule of thumb for UX: More options, more problems”

Scott Belsky