MPICH: Status and Upcoming Releases
http://www.mpich.org

Ken Raffenetti, **Yanfei Guo**, Hui Zhou, Thomas Gillis and Rajeev Thakur

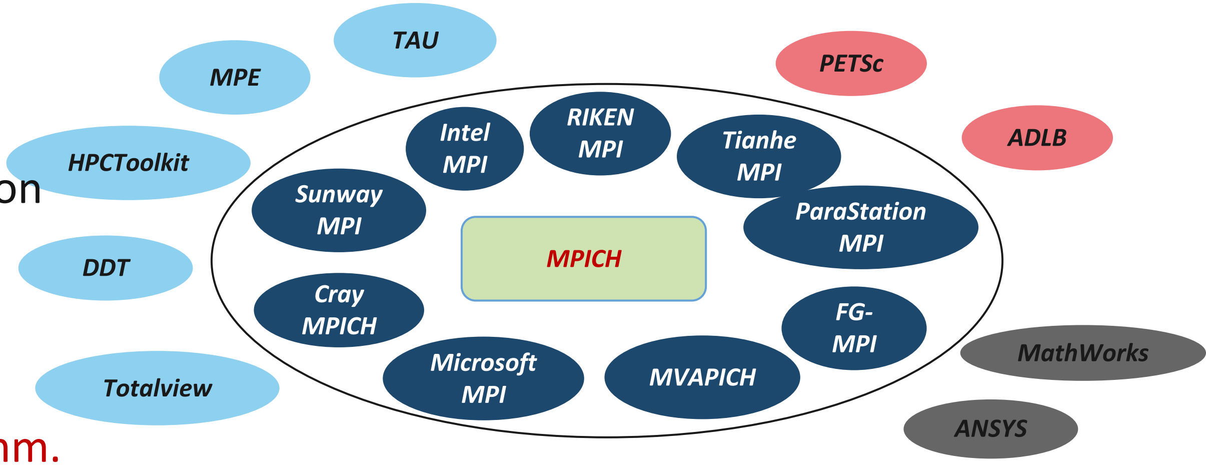
Computer Scientist

Argonne National Laboratory

MPICH turns 31

The MPICH Project

- Funded by DOE for 31 years
- Has been a key influencer in the adoption of MPI
  - First/most comprehensive implementation of every MPI standard
  - Allows supercomputing centers to not compromise on what features they demand from vendors
- DOE R&D100 award in 2005 for MPICH
- DOE R&D100 award in 2019 for UCX (MPICH internal comm. layer)
- MPICH and its derivatives are the world's most widely used MPI implementations



MPICH is not just a software

It's an Ecosystem

MPICH Adoption in Exascale Machines

- Aurora, ANL, USA (Intel MPI for Aurora)
- Frontier, ORNL, USA (Cray MPICH)
- El Capitan, LLNL, USA (Cray MPICH)

MPICH ABI Compatibility Initiative

- Binary compatibility for MPI implementations
  - Started in 2013
  - Explicit goal of maintaining ABI compatibility between multiple MPICH derivatives
  - Collaborators:
    - MPICH (since v3.1, 2013)
    - Intel MPI Library (since v5.0, 2014)
    - Cray MPICH (starting v7.0, 2014)
    - MVAPICH2 (starting v2.0, 2017)
    - Parastation MPI (starting v5.1.7-1, 2017)
- Open initiative: other MPI implementations are welcome to join
- http://www.mpich.org/abi

Argonne NATIONAL LABORATORY

intel

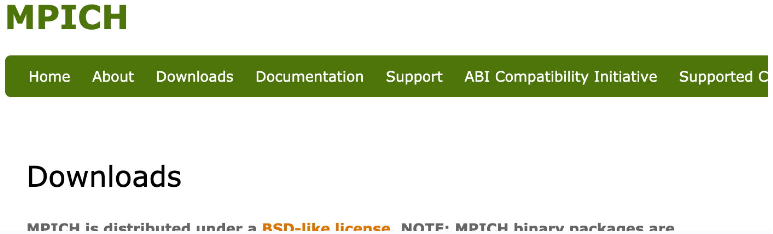
Hewlett Packard Enterprise

MVAPICH

ParaStation MPI

MPICH Distribution Model

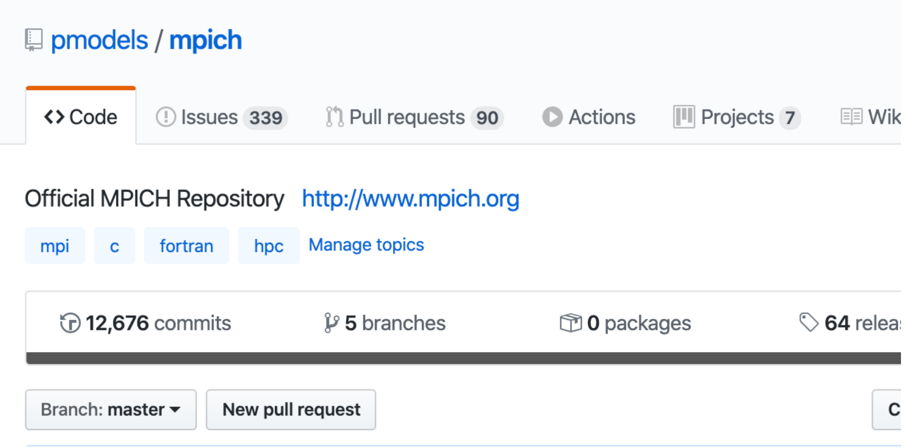
- Source Code Distribution
  - MPICH Website, Github
- Binary Distribution through OS Distros and Package Managers
  - Redhat, CentOS, Debian, Ubuntu, Homebrew (Mac)
- Distribution through HPC Package Managers
  - Spack, OpenHPC, E4S
- Distribution through Vendor Derivatives



MPICH

Downloads

MPICH is distributed under a BSD-like license. NOTE: MPICH binary packages are



pmodels / mpich

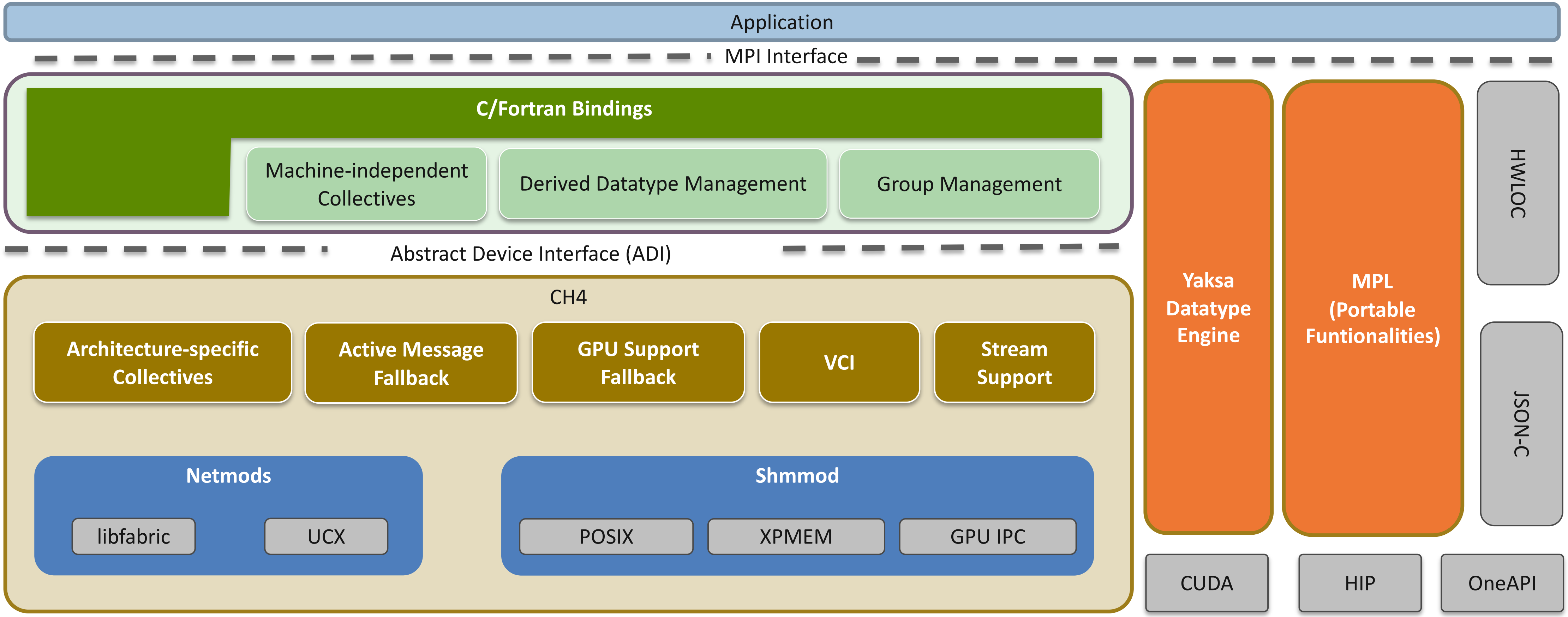
Official MPICH Repository http://www.mpich.org

12,676 commits 5 branches 0 packages 64 relea

MPICH Releases

- MPICH now aims to follow a 12-month cycle for major releases (4.x)
  - Minor bug fix releases for the current stable release happen every few months
  - Preview releases for the next major release happen every few months
  - Branching off when beta is released (feature freezed)
- Current stable release is in the 4.1.x series
  - mpich-4.1.2 was in Jun 2023
- Upcoming major release is in the 4.2.x series
  - mpich-4.2.0b1 released last week
  - rc1 and GA release coming soon

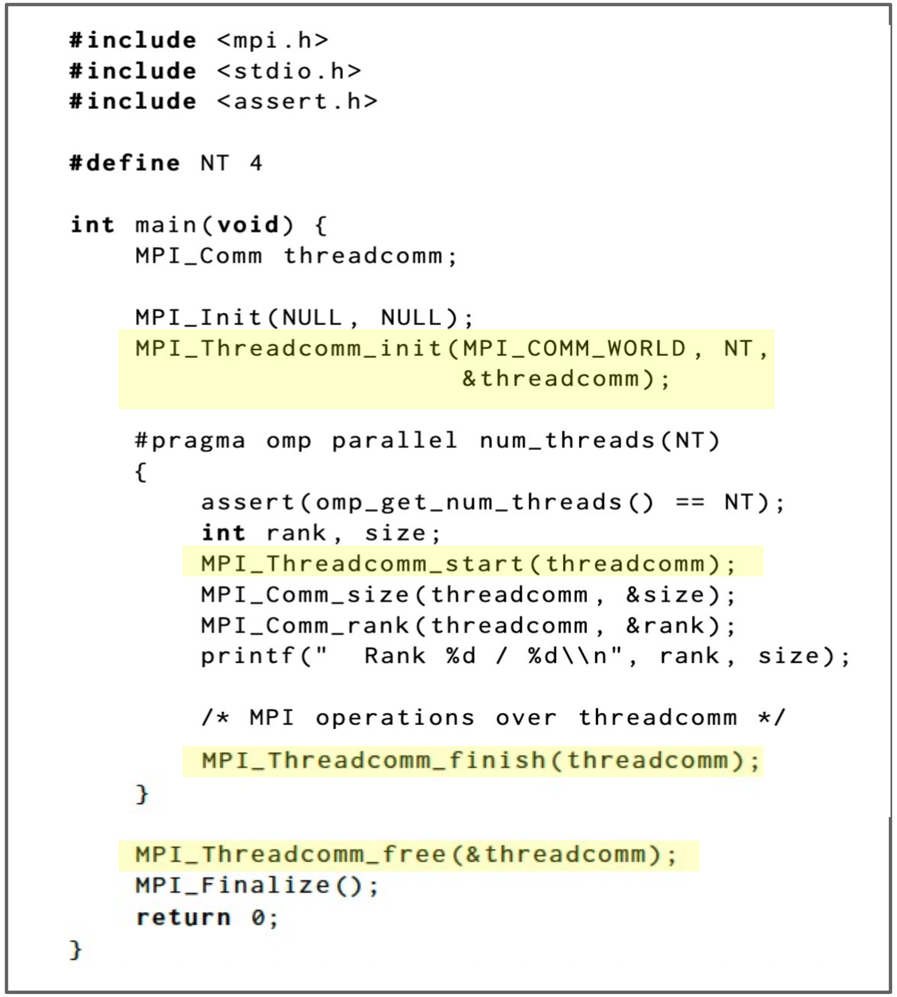
MPICH Layered Structure



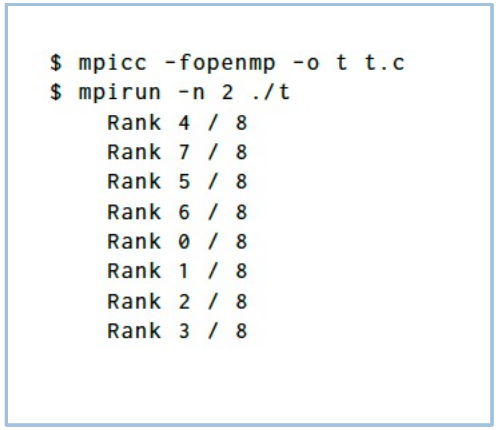
MPICH 4.2

- Full support for MPI 4.1 specification
  - `mpi_memory_alloc_kinds` info hint
  - `MPI_Request_get_status_{all,any,some}`
  - `MPI_Remove_error_{class,code,string}`
  - `MPI_{Comm,Session}_{attach,detach}_buffer`
  - `MPI_BUFFER_AUTOMATIC`
  - Split type `MPI_COMM_TYPE_RESOURCE_GUIDED`
- New experimental features
  - MPI Thread communicator
  - MPI datatype iov query
  - Reduction operator `MPIX_EQUAL`
- Enhanced GPU (esp. ZE) support
- Unified PMI-{1,2,x} support

New Extension – MPIX Thread Communicator



```c
#include <mpi.h>
#include <stdio.h>
#include <assert.h>

#define NT 4

int main(void) {
    MPI_Comm threadcomm;

    MPI_Init(NULL, NULL);
    MPI_Threadcomm_init(MPI_COMM_WORLD, NT,
                        &threadcomm);

    #pragma omp parallel num_threads(NT)
    {
        assert(omp_get_num_threads() == NT);
        int rank, size;
        MPI_Threadcomm_start(threadcomm);
        MPI_Comm_size(threadcomm, &size);
        MPI_Comm_rank(threadcomm, &rank);
        printf("  Rank %d / %d\\n", rank, size);

        /* MPI operations over threadcomm */

        MPI_Threadcomm_finish(threadcomm);
    }

    MPI_Threadcomm_free(&threadcomm);
    MPI_Finalize();
    return 0;
}
```

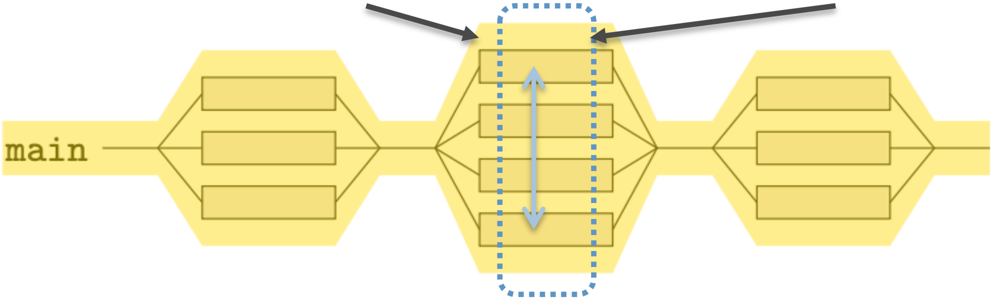


```
$ mpicc -fopenmp -o t t.c
$ mpirun -n 2 ./t
  Rank 4 / 8
  Rank 7 / 8
  Rank 5 / 8
  Rank 6 / 8
  Rank 0 / 8
  Rank 1 / 8
  Rank 2 / 8
  Rank 3 / 8
```

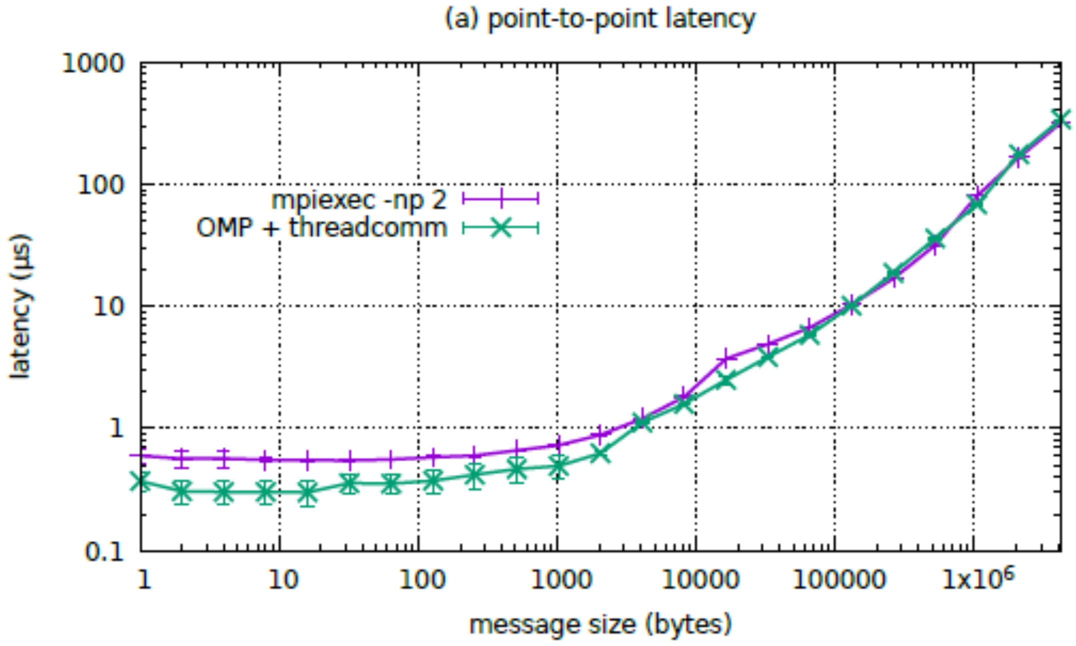
- MPI × Threads paradigm
- Blocking pt2pt ✓
- Nonblocking pt2pt ✓
- Blocking collectives ✓
- In progress –
  - Nonblocking collectives
  - RMA

OpenMP parallel regions

MPI communicator

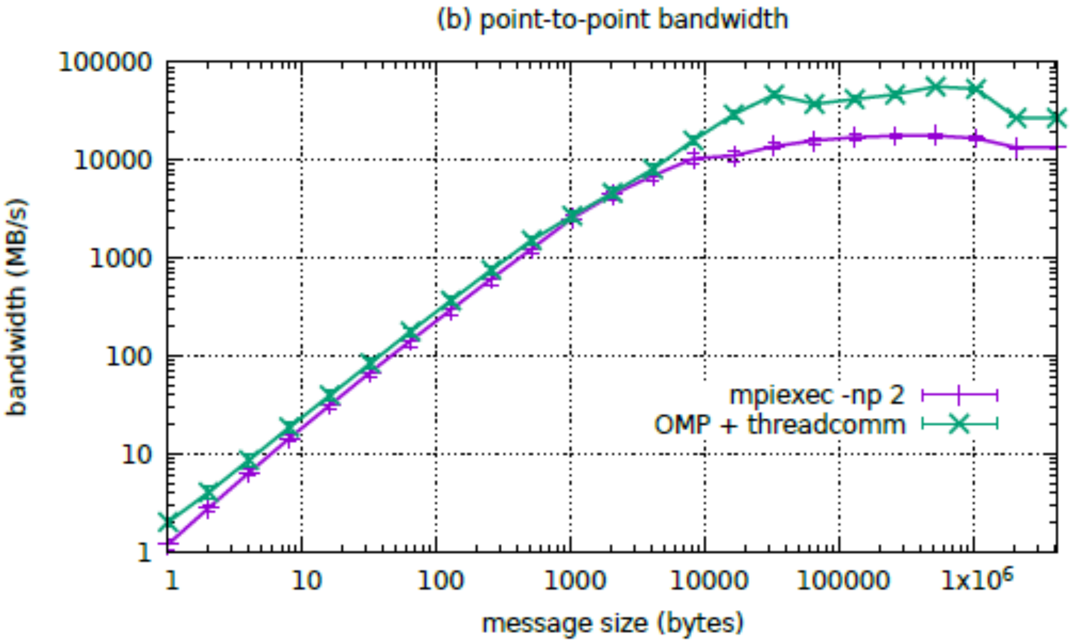


Latency and bandwidth



MPI on threads ***VS***

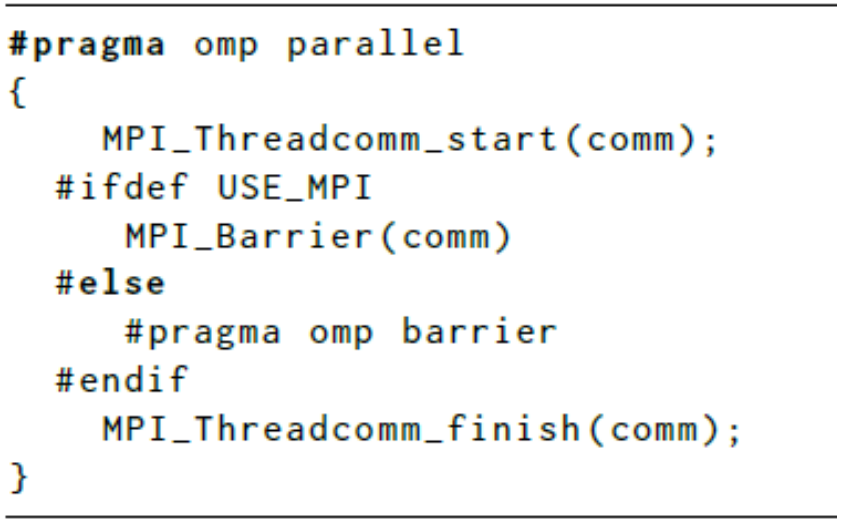
MPI on processes



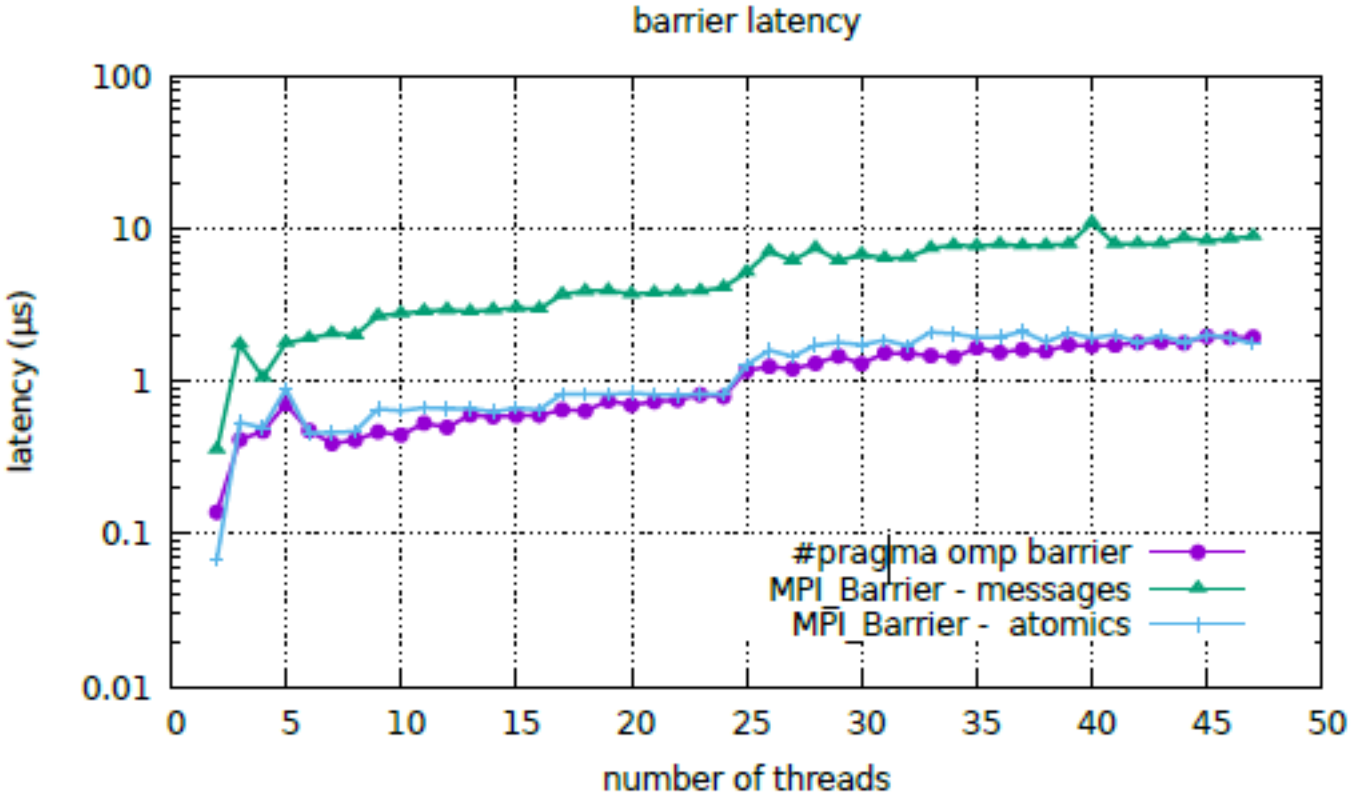
- Only practical difference
- No fundamental difference
- See paper for detailed discussions

Hui Zhou, Ken Raffenetti, Junchao Zhang, Yanfei Guo, Rajeev Thakur**. Frustrated With MPI+Threads? Try MPIxThreads!** . EuroMPI '23: Proceedings of the 30th European MPI Users' Group Meeting, https://doi.org/10.1145/3615318.3615320

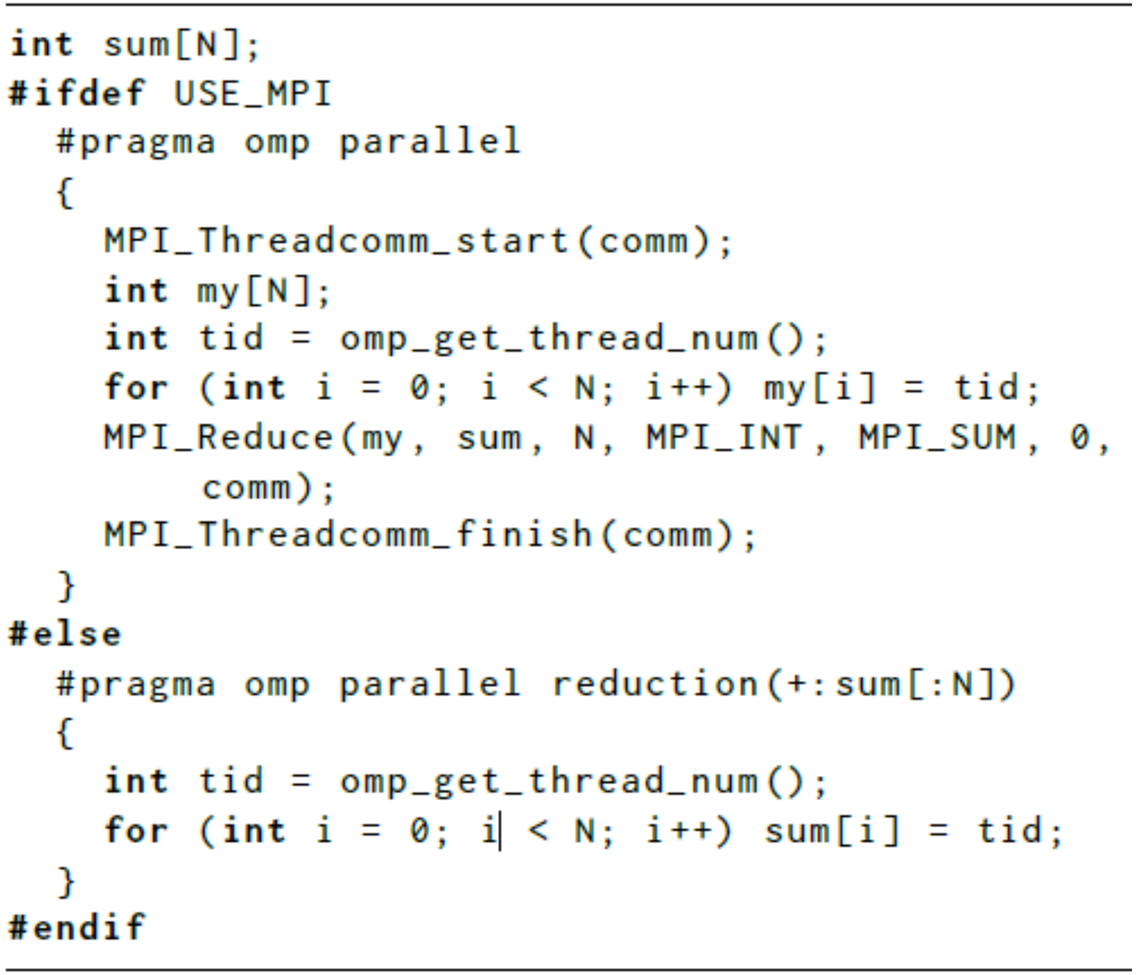
Barrier



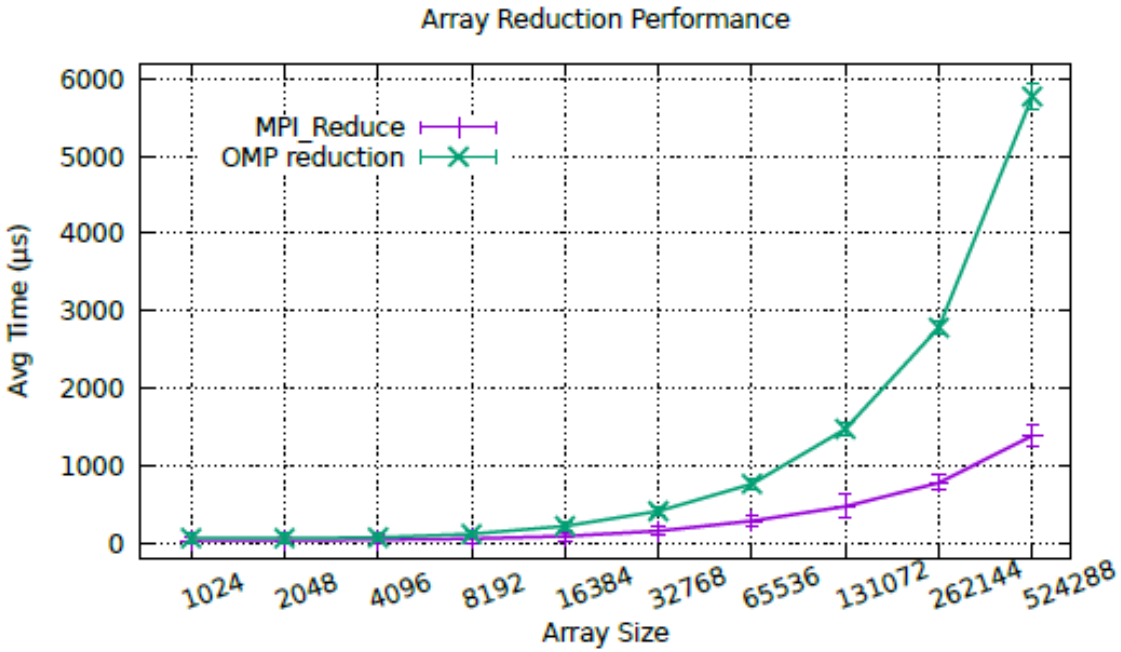
```
#pragma omp parallel
{
    MPI_Threadcomm_start(comm);
  #ifdef USE_MPI
     MPI_Barrier(comm)
  #else
     #pragma omp barrier
  #endif
    MPI_Threadcomm_finish(comm);
}
```



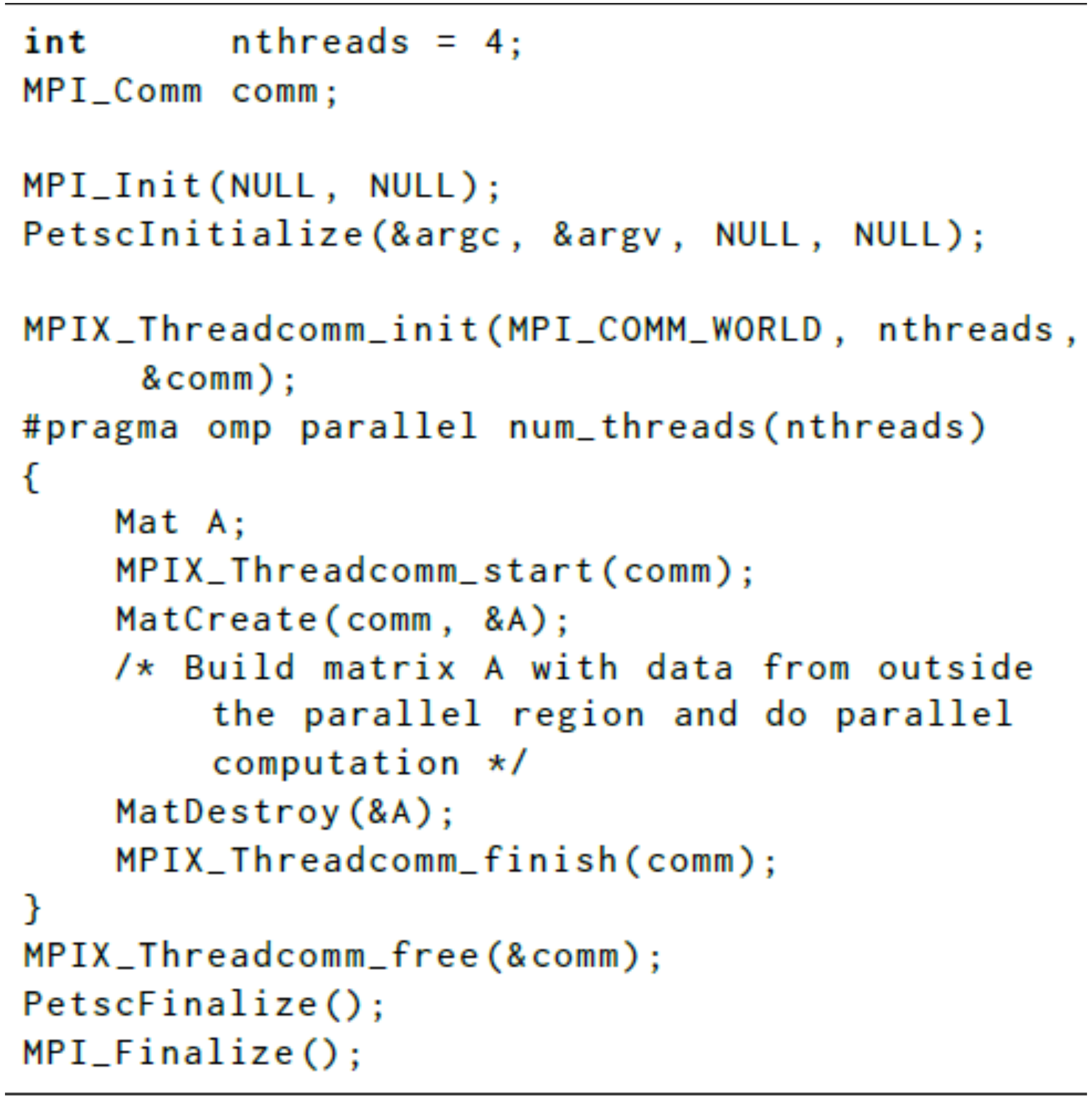
REDUCTION



```
int sum[N];
#ifdef USE_MPI
  #pragma omp parallel
  {
    MPI_Threadcomm_start(comm);
    int my[N];
    int tid = omp_get_thread_num();
    for (int i = 0; i < N; i++) my[i] = tid;
    MPI_Reduce(my, sum, N, MPI_INT, MPI_SUM, 0,
        comm);
    MPI_Threadcomm_finish(comm);
  }
#else
  #pragma omp parallel reduction(+:sum[:N])
  {
    int tid = omp_get_thread_num();
    for (int i = 0; i < N; i++) sum[i] = tid;
  }
#endif
```



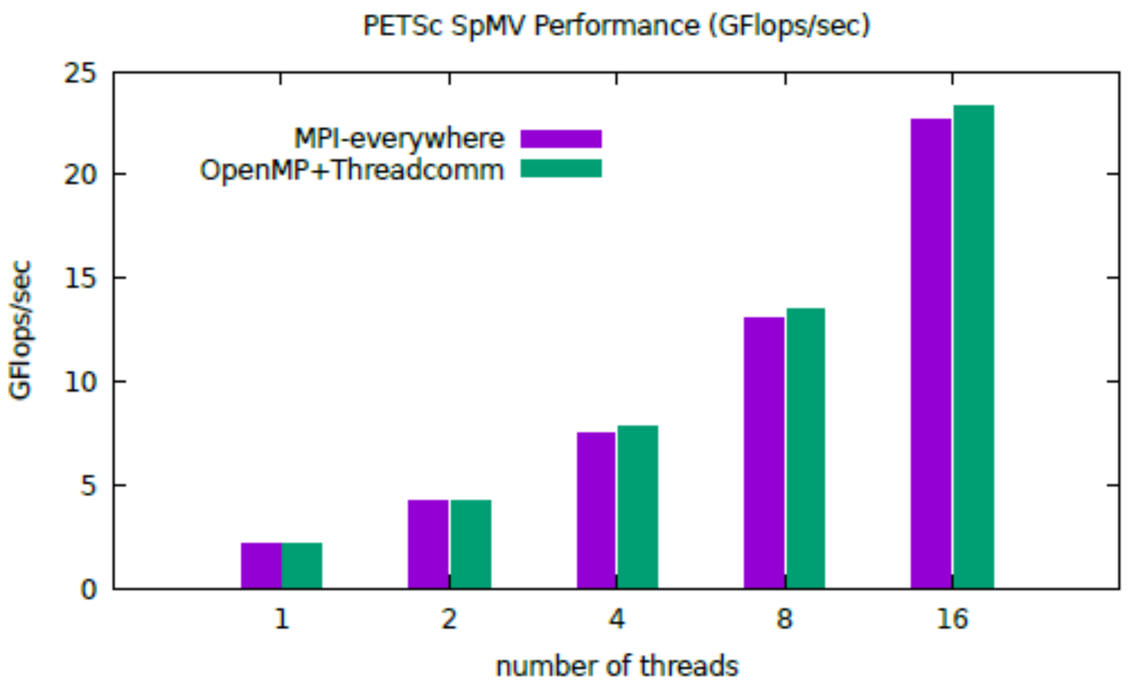
Using PETSc with threadcomm

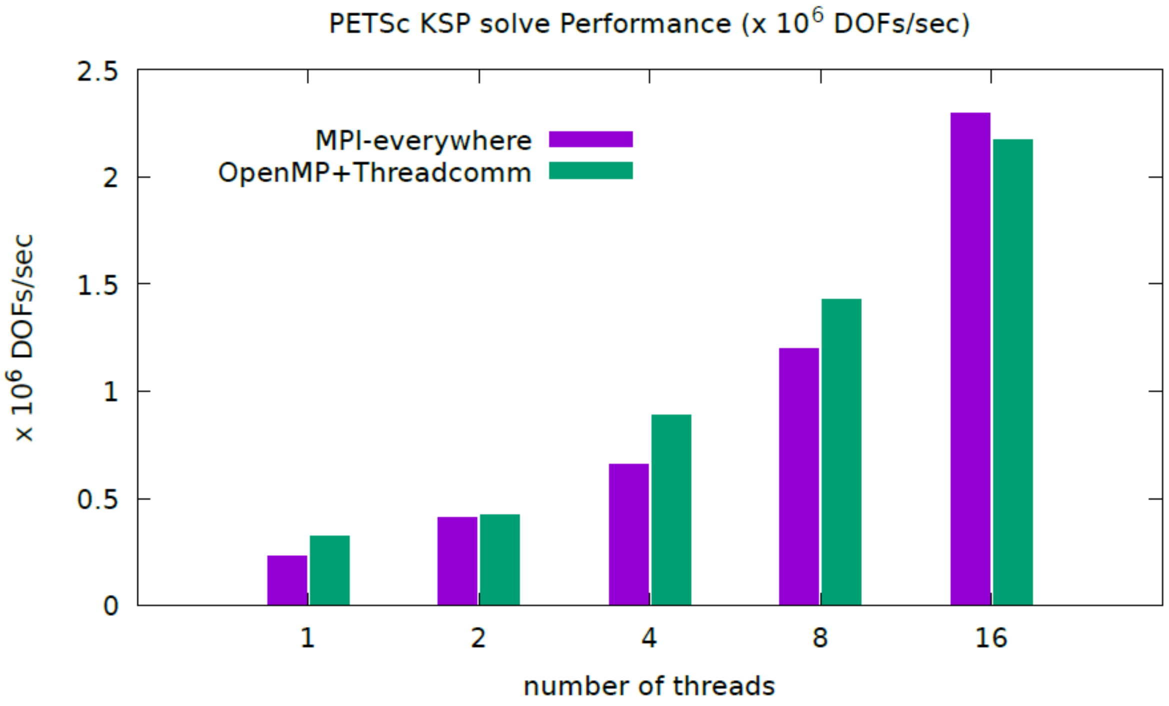


```
int       nthreads = 4;
MPI_Comm  comm;

MPI_Init(NULL, NULL);
PetscInitialize(&argc, &argv, NULL, NULL);

MPIX_Threadcomm_init(MPI_COMM_WORLD, nthreads,
     &comm);
#pragma omp parallel num_threads(nthreads)
{
    Mat A;
    MPIX_Threadcomm_start(comm);
    MatCreate(comm, &A);
    /* Build matrix A with data from outside
        the parallel region and do parallel
        computation */
    MatDestroy(&A);
    MPIX_Threadcomm_finish(comm);
}
MPIX_Threadcomm_free(&comm);
PetscFinalize();
MPI_Finalize();
```

- PETSc is not thread-safe
  - Use thread-local storage
  - Global init, then read-only
  - Logging and debugging
    - Need mutexes
    - Need threadcomm-aware
- The lessons apply to all MPI-only applications
- The changes required by adaptation are minimal

PETSC+Threadcomm performance



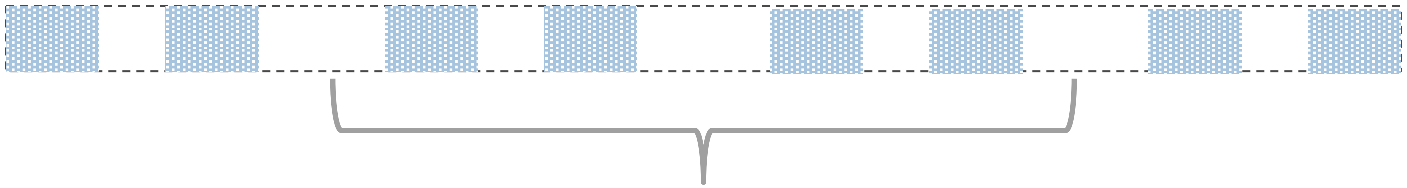


New Extension - Datatype IOV

- MPI datatypes provide efficient abstractions for data layout.
- However, it is opaque to users and only can be used in MPI communications.
- An IOV extension enables users to benefit from MPI datatypes beyond MPI usages.

```
int MPIX_Type_iov_len (MPI_Datatype datatype, MPI_Count max_iov_bytes,
                       MPI_Count *iov_len, MPI_Count *actual_iov_bytes);

int MPIX_Type_iov (MPI_Datatype datatype, MPI_Count iov_offset,
                   MPIX_Iov iov[], MPI_Count max_iov_len,
                   MPI_Count *actual_iov_len);
```

```
Typedef struct MPIX_Iov {
    void *iov_base;
    MPI_Aint iov_len;
} MPIX_Iov;
```



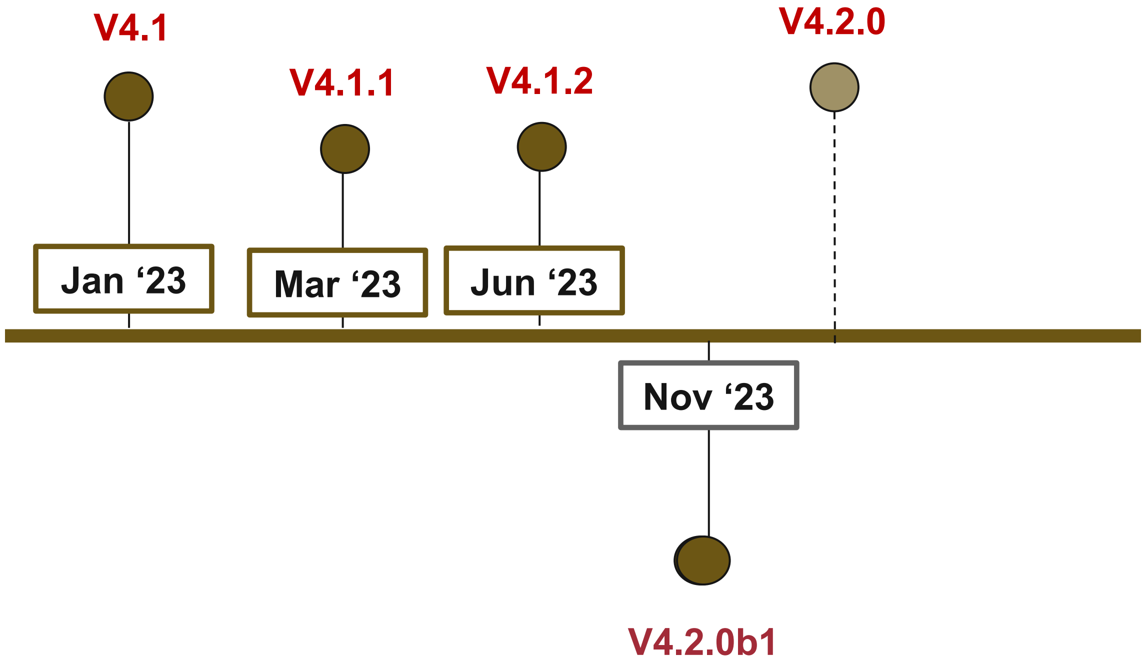
New Extension - MPIX_EQUAL

```
/* Usage example */

int data[NUM];
int is_equal;

MPI_Allreduce(data, &is_equal, NUM, MPI_INT, MPIX_EQUAL, comm);

MPI_Reduce(data, &is_equal, NUM, MPI_INT, MPIX_EQUAL, root, comm);
```

- The missing but most basic logical operator
- The output buffer always points to an int (boolean)
- Compare to the alternative:
  - Allgather -> local comparison via loops
  - Allreduce(MPI_MAX) -> local comparison -> Allreduce(MPI_LAND)
  - User-defined operator that return special values representing equal (or unequal)

MPICH 4.2.0 Roadmap

- MPICH-4.2.0b1 released last week
  - 4.2.x branch is created
- GA release in late 2023/early 2024
- Critical bug fixes are backported to 4.1.x



MPICH 4.3 Series Plans (RFC)

- Support dynamic VCIs
- Optimize partitioned communication
- Enhance support for MPI sessions
- Better code design for hierarchical collective algorithms
- Support runtime loading of selected dependency libraries (e.g. CUDA, libfabric, UCX)
- Support `mpi_memory_alloc_kinds` side document specification
- Continue prototyping standard MPI ABI
- Performance improvements

Ongoing research: MPI-RMA refactoring

RMA should be used for GPU-to-GPU communication

Exploration of the best target completion protocol:

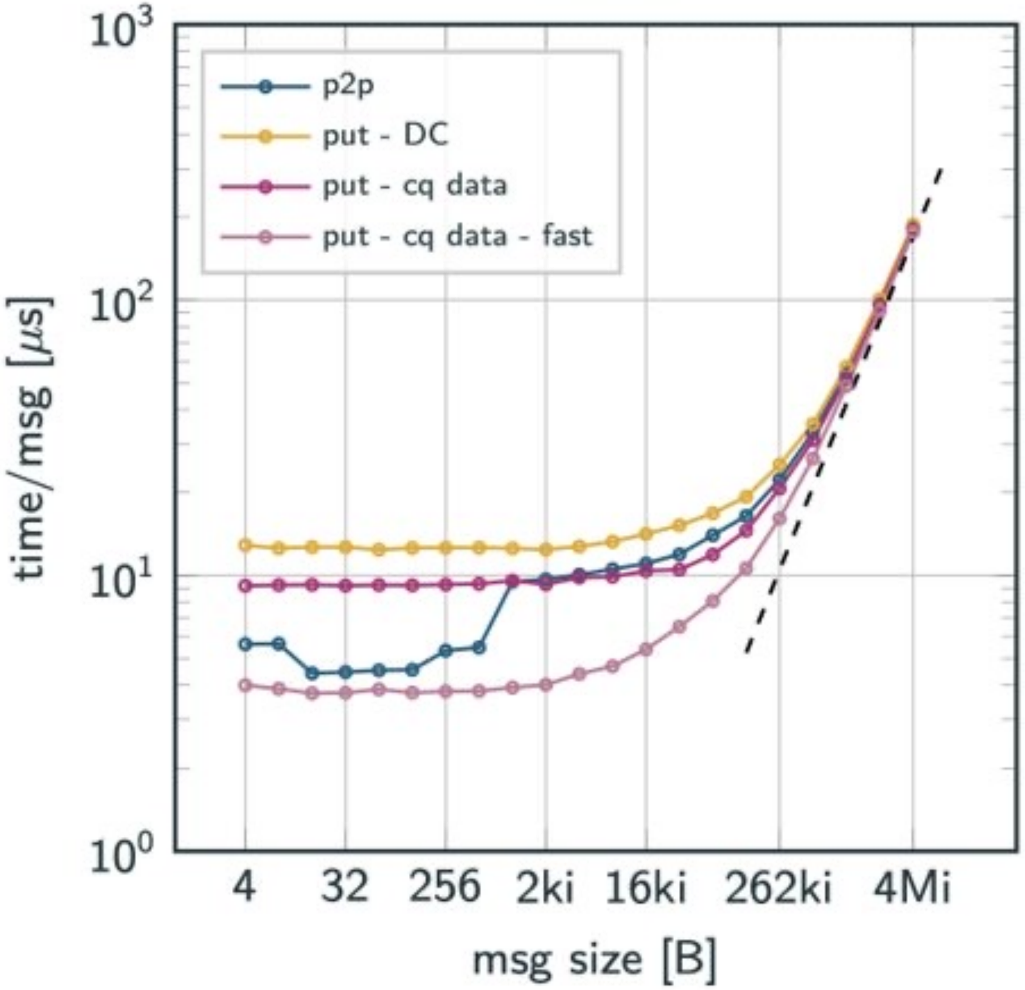
completion ack required: (origin-based completion)

- **delivery complete** (DC) always available, MPICH default
- **fence** similar to DC, no-op if ordered, only UCX and CXI

No-completion ack: (target-based completion)

- **immediate data** (cq_data) - not supported with UCX and CXI
- **remote counter** only CXI

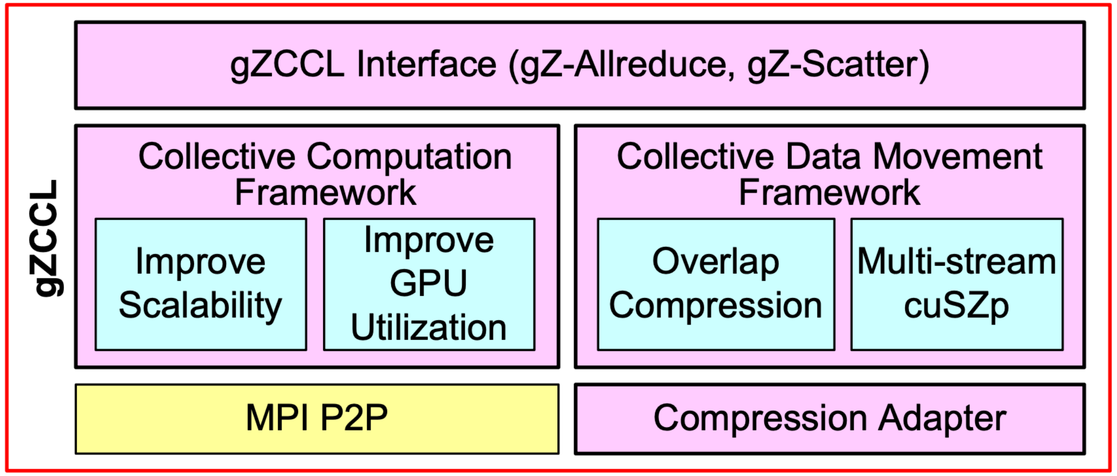
Ongoing work with the MPI Forum RMA working group (led by Joseph Schuchart, UTK)

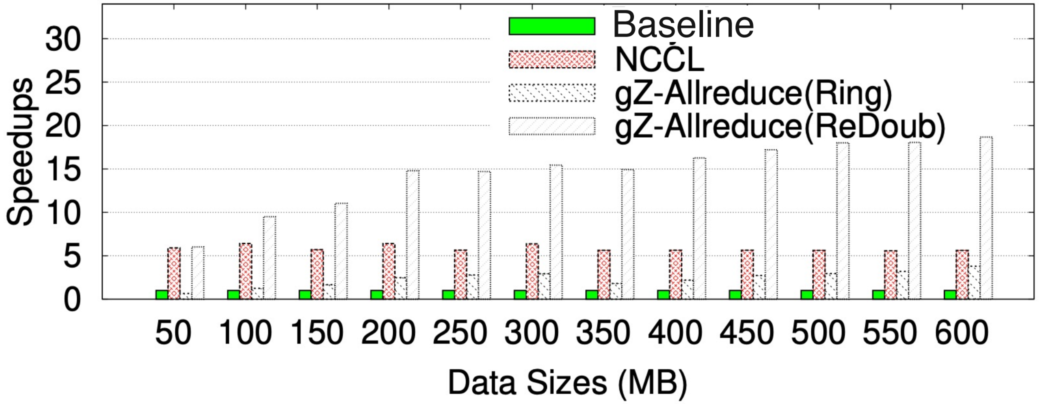


Latency for 1 msg (GPU-to-GPU)
MeluXina – 200 Gb/s

Ongoing research: MPI Collective with Lossy Compression

- Integrating Lossy Compression with MPI Collective for Large Message Transfer
- Efficient Scheduling of Compression and Communication
- Relying on Regular MPI P2P





Jiajun Huang, et el. **gZCCL: Compression-Accelerated Collective Communication Framework for GPU Clusters,** https://arxiv.org/abs/2308.05199

Thank you!

- https://www.mpich.org
- Mailing list: discuss@mpich.org
- Issues and Pull requests: https://github.com/pmodels/mpich
- Weekly development call every Thursday at 9am (central): https://bit.ly/mpich-dev-call