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

Ken Raffenetti, Yanfei Guo, Hui Zhou, Mike Wilkins and Rajeev Thakur

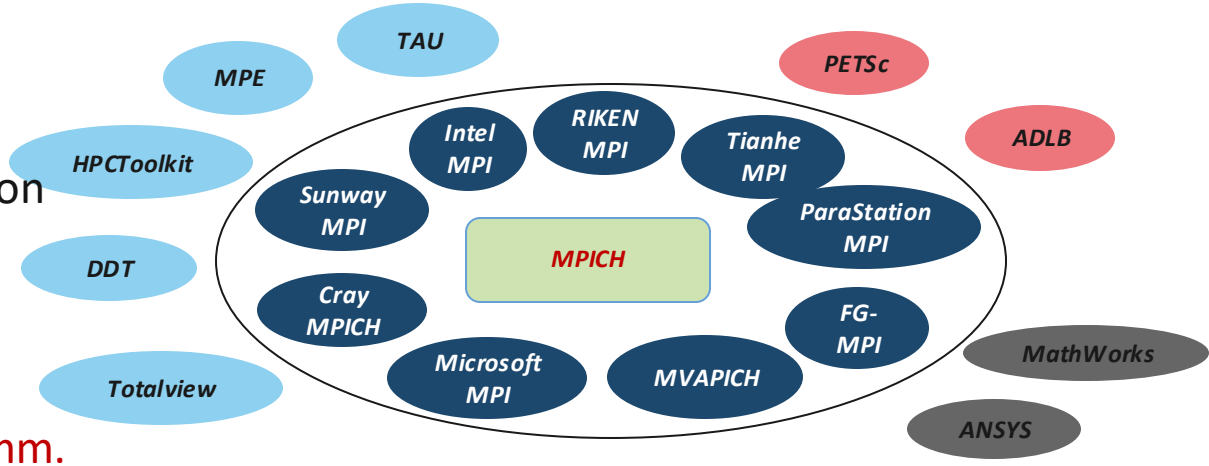
Computer Scientist

Argonne National Laboratory

MPICH turns 32

The MPICH Project

- Funded by DOE for 32 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 in 7 of Top 10 Supercomputers

1. **El Capitan, LLNL, USA (Cray MPI)**
2. **Frontier, ORNL, USA (Cray MPI)**
3. **Aurora, ANL, USA (Intel MPI for Aurora)**
4. Eagle, Microsoft, USA
5. **HPC 6, Eni S.p.A, Italy (Cray MPI)**
6. Fugaku, RIKEN, Japan
7. **Alps, CSCS, Switzerland (Cray MPI)**
8. **LUMI, EuroHPC/CSC, Finlnad (Cray MPI)**
9. Leonardo, EuroHPC/CINECA, Italy
10. **Tuolumne, LLNL, USA (Cray MPI)**

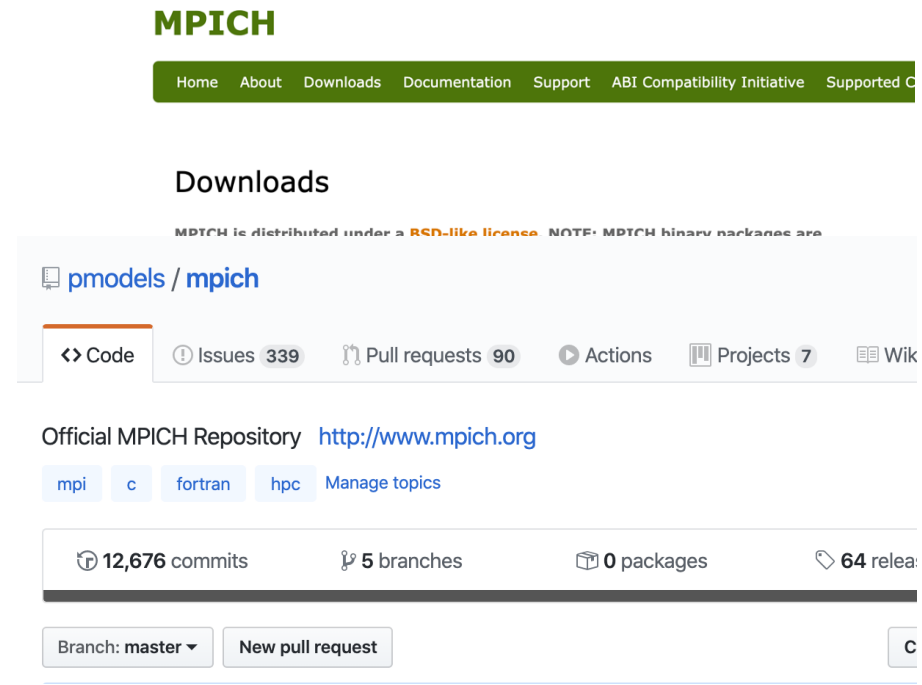
MPICH ABI Compatibility Initiative

- 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

MPICH Distribution Model

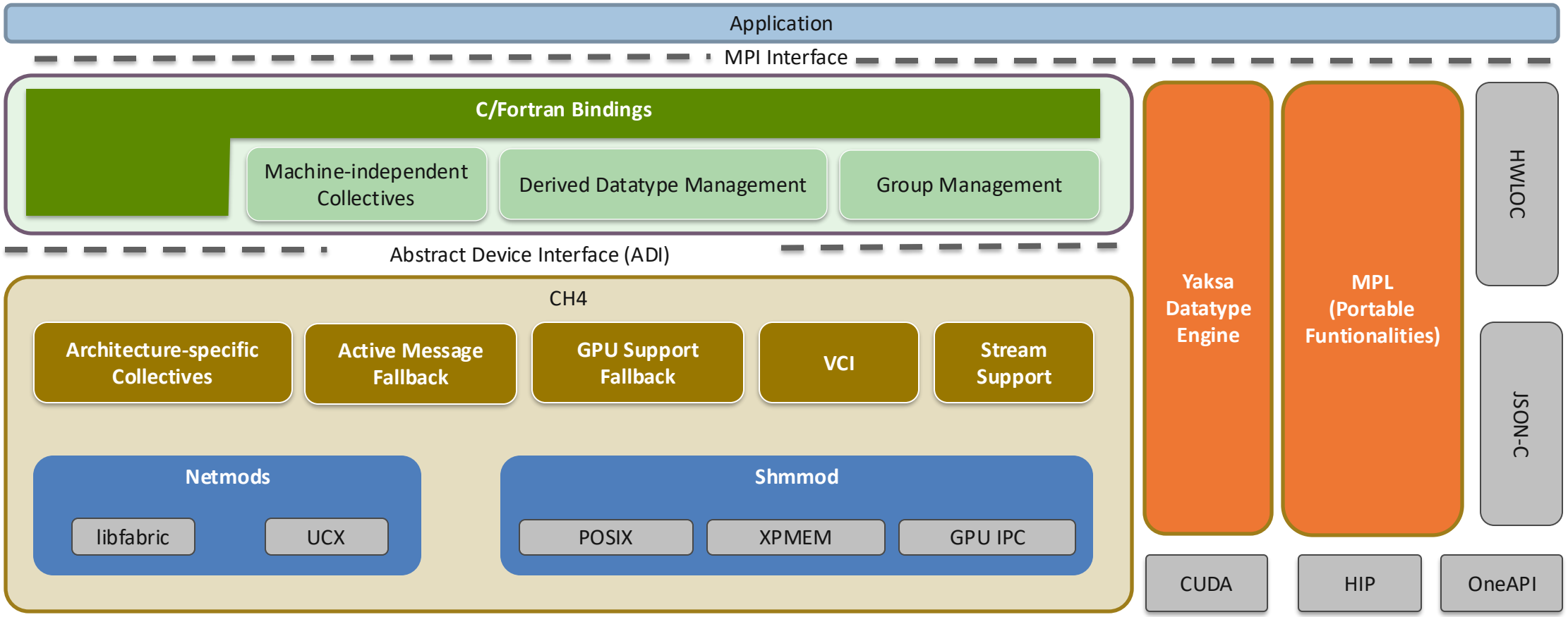
- Source Code Distribution
  - MPICH Website, Github
- Binary Distribution through OS Distros and Package Managers
  - Redhat, CentOS, Debian, Ubuntu, Homebrew (Mac)
- Distribution through HPC Package Managers
  - Spack, OpenHPC, E4S
- Distribution through Vendor Derivatives



MPICH 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 frozen)
- Current stable release is in the 4.2.x series
  - mpich-4.2.3 was in Oct 2024
- Upcoming major release is in the 4.3.x series
  - mpich-4.3.0b1 released 11/15/2024
  - rc1 and GA release coming soon

MPICH Layered Structure



MPICH 4.3 Release

MPICH 4.3 Release

- 4.3.0b1 released
- 4.3.0 RC and GA releases soon
- Full support for MPI-4.1
- Support for MPI-4.2 ABI Proposal
- Experimental New Extensions
- Enhancing CH4 Netmods with AM
- Benchmark Tests in MPICH test suite

- Topology-aware shmem communication
- Improved dynamic process support
  - Added CH4 UCX and Shared Memory
- Improved separation between libmpi.so and libmpifort.so
- Lazy GPU (yaksa) initialization
- Improved progress debugging

Full Support for MPI-4.1

MPI 4.1

Feature	MPICH
(P)MPI_WTI{ME,CK} and handle conversion macros are symbols	✅
`MPI_REQUEST_GET_STATUS` makes progress	✅
Status field get/set functions	✅
Send with automatic buffering features	✅
Getting multiple statuses from an array of requests	✅
`MPI_TYPE_GET_VALUE_INDEX`	✅
`MPI_COMM_TYPE_RESOURCE_GUIDED` and `mpi_pset_name`	✅
`mpi_assert_strict_persistent_collective_ordering`	✅
Memory alloc kinds	✅
Getting names for null handles	✅
`MPI_GET_HW_RESOURCE_INFO`	✅
`MPI_ERR_ERRHANDLER`	✅
Removing error class, code and string	✅
`MPI_WIN_SHARED_QUERY` supports more window types	✅
`mpi_accumulate_granularity`	✅

Support for Memory Allocation Kinds Side Document

- https://github.com/mpi-forum/mem-alloc
  - Defines CUDA, ROCm, and Level Zero memory kinds
  - Extends mpi and system kinds defined in MPI-4.1

- MPICH supports all three kinds (with or without restrictors) defined in the side document.
  - GPU support remains controlled by the MPIR_CVAR_ENABLE_GPU environment variable. Requesting GPU support is not required (for now).
  - Queries return the kinds supported, e.g. mpi,system,cuda. MPICH does not differentiate any restrictors at this time, but will return them if requested.

```
/* test if MPI_COMM_WORLD gets the right value */
MPI_Info info;
MPI_Comm_get_info(MPI_COMM_WORLD, &info);
MPI_Info_get(info, "mpi_memory_alloc_kinds", MPI_MAX_INFO_VAL,
             value, &flag);
MPI_Info_free(&cinfo);
if (flag) {
    printf("mpi_memory_alloc_kinds = \"%s\"\n", value);
}
```

```
pmrs-gpu-240-01% mpiexec -n 1 ./memory_alloc_kinds
mpi_memory_alloc_kinds value is "mpi,system,cuda"
pmrs-gpu-240-01% mpiexec -n 1 -memory-alloc-kinds cuda:device ./memory_alloc_kinds
mpi_memory_alloc_kinds value is "mpi,system,cuda,cuda:device"
pmrs-gpu-240-01%
```

Support for MPI-4.2 ABI Proposal

- MPI ABI Working Group
  (https://github.com/mpiwg-abi)

- Test drive MPI ABI
  with MPICH today!

```
$ ./autogen.sh
$ ./configure --prefix=[path] --enable-mpi-abi
$ make install

$ mpicc_abi -o cpi examples/cpi.c
$ mpirun -n 2 ./cpi
$ ldd ./cpi |grep libmpi

libmpi_abi.so.0 => /opt/MPI/lib/libmpi_abi.so.0
(0x00007ff1883e9000)
```

Experimental New Extensions

- MPIX_Op_create_x
- MPIX_Comm_create_errhandler_x
- MPIX_File_create_errhandler_x
- MPIX_Session_create_errhandler_x
- MPIX_Win_create_errhandler_x
- MPIX_Request_is_complete
- MPIX_Async_start
- MPIX_Async_get_state
- MPIX_Async_spawn

```
int MPIX_Op_create_x(MPIX_User_function_x *user_fn_x,
                     MPIX_Destructor_function *destructor_fn,
                     int commute, void *extra_state, MPI_Op *op);

typedef void (MPIX_User_function_x) (void *invec, void *inoutvec,
                                     MPI_Count count,
                                     MPI_Datatype datatype,
                                     void *extra_state);
```

- Addition of user context (extra_state)
- C interface instead of FORTRAN-style reference
- Targeting language bindings (Python, Rust, Julia)
- Fortran w. INTEGER-8 default now works

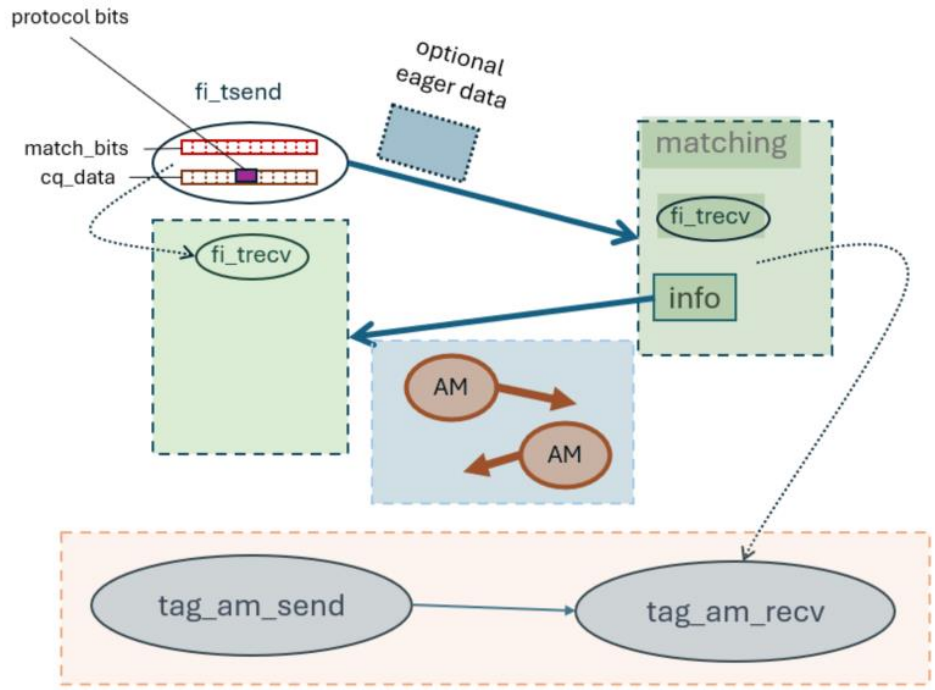
```
int MPIX_Async_start(MPIX_Async_poll_function poll_fn,
                     void *extra_state, MPIX_Stream stream);

enum {
    MPIX_ASYNC_PENDING = 0,
    MPIX_ASYNC_DONE = 1,
};
typedef int (MPIX_Async_poll_function)(MPIX_Async_thing thing);
```

- Custom async operations integrated with MPI progress
- Extend MPI without compromising performance
- Ref: “MPI Progress for All!”, https://arxiv.org/abs/2405.13807

Enhancing Native OFI with Active Messages

- Default path is the native libfabric/ucx path
  `fi_tsend`/`fi_trecv`
  `ucp_tag_send_nbx`/`ucp_tag_recv_nbx`
- Optionally switching to Active Message by sending a meta-flag
- Native matching
- Potential: software RNDV, “huge” messages, “pipelining”, RDMA write



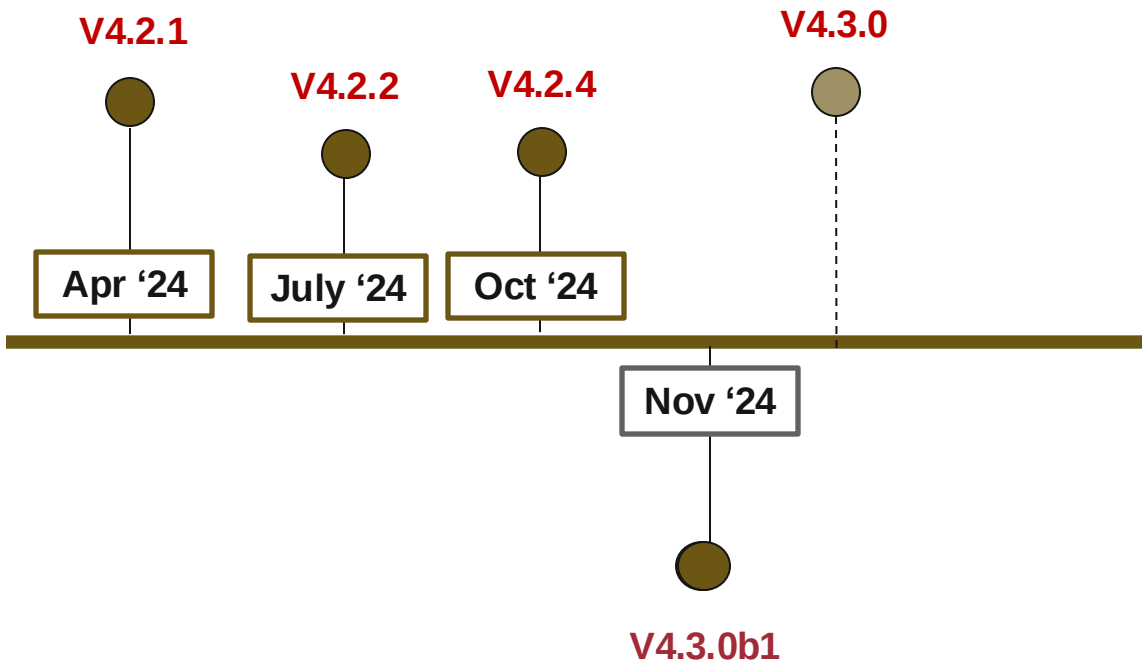
Benchmark Test in MPICH Test Suite

- Track performance in CI testing
- Based on OSU Microbenchmarks
- Integrated GPU testing – CUDA/HIP/ZE
- Simple interfaces
  - Single source file
  - Built-in warm-up
  - Always report uncertainties

```
mpicc -o ./p2p_latency ./p2p_latency.c -lm && mpirun -n 2 ./p2p_latency
TEST p2p_latency:
     msgsize    latency(us)  sigma(us)    bandwidth(MB/s)
           0          0.465      0.002              0.000
           1          0.475      0.001              2.105
           2          0.476      0.002              4.205
           4          0.478      0.019              8.360
           8          0.476      0.002             16.800
          16          0.475      0.002             33.694
          32          0.474      0.002             67.451
          64          0.476      0.001            134.424
         128          0.505      0.003            253.422
         256          0.523      0.004            489.442
         512          0.608      0.005            842.436
        1024          0.776      0.003           1318.895
        2048          0.932      0.003           2196.811
        4096          1.273      0.004           3216.882
        8192          1.949      0.006           4202.567
       16384          4.143      0.017           3954.333
       32768          5.674      0.016           5775.437
       65536          8.632      0.018           7592.587
      131072         14.502      0.050           9038.108
      262144         26.104      0.052          10042.183
      524288         50.209      0.104          10442.031
     1048576        106.563      0.234           9839.961
     2097152        219.804      0.385           9541.020
     4194304        476.780      1.431           8797.154
```

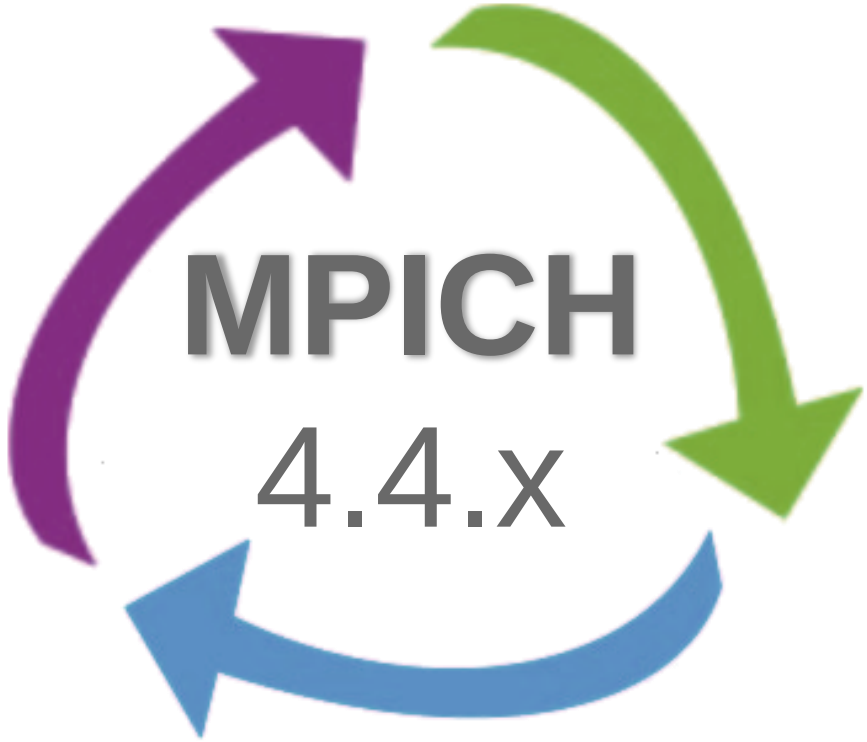
MPICH 4.3.0 Roadmap

- MPICH-4.3.0b1 released last week
  - 4.3.x branch is created
  - Main branch start 4.4 development
- GA release in late 2024/early 2025
- Critical fixes will be backported to 4.3.x



MPICH 4.4 Series Plans (RFC)

- Continuous improving
- Unified pipelining for all communications (CPU, GPU)
  - GPUs are designed to use pipelining, MPI was not
  - Based on generalized async progress engine
- Hierarchical Collectives
  - Unifying MPIR/MPID layers
  - Extend beyond node level
  - Allow more flexible and dynamic algorithms
- Partial datatype support
  - (`buffer, count, datatype, offset, length`)
- MPI for AI
  - MPI is well designed and HPC native
  - HPC now includes AI
  - We need make AI stack on MPI *performant* and *competitive*



Argonne NATIONAL LABORATORY

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

U.S. DEPARTMENT OF ENERGY