MPI
Application Binary Interface (ABI) Standardization

Jeff Hammond
Principal Software Architect, NVIDIA
Co-chair, MPI Forum ABI WG
jeff.science@gmail.com

What problem are we solving?

Break the dependency between how you build your MPI libraries and applications and how you run them.

If you build with Open MPI 3.x, you need to run with Open MPI 3.x.

If you build with MVAPICH, you need to run with MVAPICH…

…or another MPICH-based implementation. Why does this work?

It’s not just you who is building MPI software: package managers, Spack and ISVs ship binaries.

API versus ABI

API

int MPI_Bcast(void * buffer, int count, MPI_Datatype d, int root, MPI_Comm c);

MPI_Datatype and MPI_Comm are unspecified types

ABI

typedef **struct ompi_datatype_t *** MPI_Datatype; // Open MPI family

typedef **int** MPI_Datatype; // MPICH family

Lots of other stuff like SO names, SO versioning, calling convention, etc.

MPI ABI Status Quo

MPI is an **API** standard, which defines the source code behavior in C (C++) and Fortran. The **compiled** representation of MPI features is implementation-defined.

If you **compile** with one of the following MPI families, you MUST **run** with the same.

1. MPICH / Intel MPI / MVAPICH / Cray MPI
2. Open MPI / NVIDIA HPC-X / Amazon MPI / IBM Spectrum MPI

Family 1 exists because there was a demand for interoperability with Intel MPI due to the prevalence of usage in ISV codes.

Family 2 is not guaranteed to be consistent, especially across major versions.

1 = https://www.mpich.org/abi/

Why?

Modern software use cases:

- Third-party **language** support, e.g. Python, Julia, Rust, etc.
- **Package** distribution, e.g. Spack, Apt, etc.
- **Tools** become implementation-agnostic
- **Containers**
- More efficient **testing** (build only once)

We can:

- Architectural reasons not to are gone
- Two platform ABIs cover >90% of HPC platforms

MPI Application Binary Interface Standardization

Jeff R. Hammond
NVIDIA Helsinki Oy
Helsinki, Finland
NVHPC SDK, Fortran

Lisandro Dalcin
Extreme Computing Research Center
KAUST
Thuwal, Saudi Arabia
dal Python l.com

Erik Schnetter
Perimeter Institute for Theoretical Physics
Waterloo, Ontario, Canada
esc Julia, MPItrampoline e.ca

Marc Pérache
CEA, DAM, DIF
Arpajon, France
wi4mpi, containers, MPC

Jean-Baptiste Besnard
ParaTools
Bruyères-le-Châtel, France
jbbes TAU, E4S s.fr

Jed Brown
University of Colorado Boulder
Boulder, Colorado, USA
j PETSc, Rust

Gonzalo Brito Gadeschi
NVIDIA GmbH
Munich, Germany
Rust, containers

Joseph Schuchart
University of Tennessee, Knoxville
Knoxville, Tennessee, USA
sc Open MPI du

Simon Byrne
California Institute of Technology
Pasadena, California, USA
simonby Julia tech.edu

Hui Zhou
Argonne National Laboratory
Lemont, Illinois, USA
zh MPICH ov

A preprint will be on arXiv tomorrow.

This link will work in the near future:
https://doi.org/10.1145/3615318.3615319

Disclaimer

- Many of the design decisions have been debated extensively and reflect very strong consensus among MPI Forum members.
- Some of the design decisions are still being debated extensively and consensus among MPI Forum members has not been achieved.
- The ABI proposal will be read for the first time September and may change significantly before it is ratified.

This is not a tutorial. You can’t rely on any of this until MPI 4.2 is published.

MPI 4.2 ABI Design

- MPI integer types
  - Only standardize 32/64-bit platforms for now
  - MPI_Aint is intptr_t because that satisfies all of the requirements
  - MPI_Offset is int64_t because that will be sufficient for ~30 years
  - MPI_Count is int64_t
- Handles are defined like Open MPI (type-safety) but with compile-time constant handles like MPICH (OS portability)
- Constants use a Huffman code with information encoded in values like MPICH
  - 0 is never a legal handle (detect uninitialized)
  - Fixed-size types have size encoded in constant value
- Integer constants are globally unique to allow nice error messages

MPI ABI Packaging

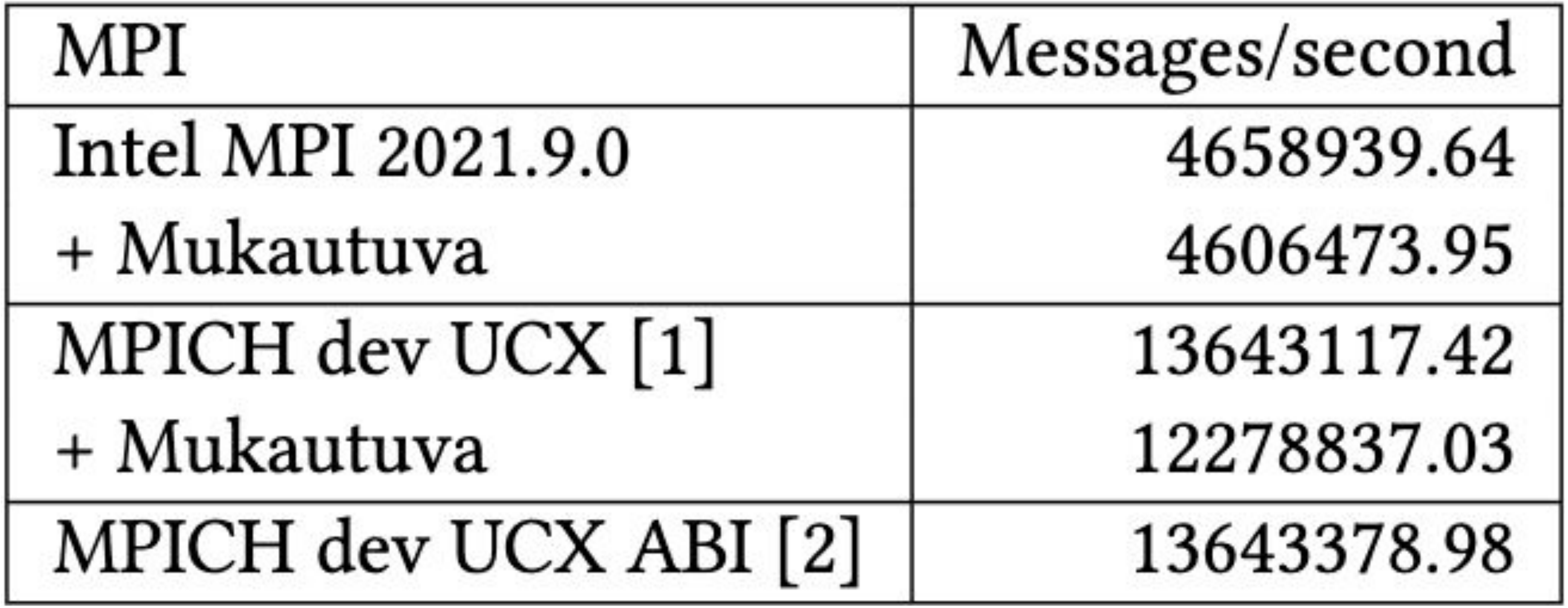
- The header is abi/mpi.h
  - #include <mpi.h> still works - no code changes required to adopt ABI
  - #include <abi/mpi.h> allows users to force the use the standard ABI
  - The Forum may distribute a standard header for convenience
- The library is libmpi_abi.so
  - Implementations are instructed to use platform-specific SO versioning conventions
  - The Forum may distribute a standard SO for convenience
- The ABI is versioned independently from the API
  - ABI starts with 1.0
  - Backwards-compatible changes (e.g. new handle type) increment the minor version
  - Backwards-incompatible changes increment the major version
  - Adding a new function to the API does not change the ABI

MPI 4.2 ABI Design - Fortran

- Fortran isn’t tied to platform ABI like C
- Integer constants are required to match C
- Trivial conversions for *predefined* handles, like MPICH
- Simple lookup overhead for other handles, like Open MPI
- Sentinels aren’t part of the ABI
- MPI_<Handle>_{f2c,c2f} and MPI_Status_{f2c,c2f} depend on MPI_Fint, which will be defined to be C int in order to have a fixed ABI; if INTEGER doesn’t match C int because of compiler options, users have to deal with that.

Implementing the standard ABI

1. **Standalone:** dlopen MPI, dlsym everything, translate everything at runtime.
    - wi4mpi (CEA)
    - MPItrampoline (Erik Schnetter)
    - Mukautuva (me)
2. **Integrated:** the MPI library implements the ABI in a separate header+library and does all the conversions to the existing ABI internally.
    - MPICH has done this already
3. **Native:** the MPI library implements the ABI throughput.



MPI	Messages/second
Intel MPI 2021.9.0	4658939.64
+ Mukautuva	4606473.95
MPICH dev UCX [1]	13643117.42
+ Mukautuva	12278837.03
MPICH dev UCX ABI [2]	13643378.98

1. --enable-error-checking=no --enable-fast=Os --enable-g=none --with-device=ch4:ucx
2. Same as 1 plus --enable-mpi-abi

https://github.com/jeffhammond/mukautuva

When?

- Targeting MPI 4.2 as a single-feature ABI-only release (early 2024?).
- Mukautuva, wi4mpi, and MPItrampoline can support this immediately.
- MPICH has a prototype already.
- Open MPI has not implemented this but they say it’s easy.

Diffusion: upstream -> release -> packaging, etc.

FAQ

- Launchers are not part of the ABI. There are at least two options:
  - Slurm and PBS launchers are supported by all the major MPIs already.
  - mpirun can set the shared library to use, in which case the launcher and library will match.
- Wrapper scripts (e.g. mpicc) are not standard but the ecosystem will probably have “mpicc_abi” or “mpicc -abi”.
- MPICH and Open MPI will continue to support their existing ABIs.

The End