MPI 3.0 Fortran Bindings:

a high-level summary for non-Fortran programmers

Craig Rasmussen (Los Alamos)
Rolf Rabenseifner (HLRS)
Jeff Squyres (Cisco Systems)

A very brief overview of the requirements for new MPI 3.0 Fortran bindings

- Requirements
  - comply with Fortran standard (for the first time)
  - enhance type safety
  - suppress argument checking for choice buffers
  - guarantee of correct asynchronous operations
  - for user convenience
    - provide users with convenient migration path
    - allow some optional arguments (e.g., ierror)
    - support sub-arrays
  - for vendor convenience
    - allow vendors to take advantage of the C interoperability standard

Requirement 1: comply with fortran standard

- Requirement
  - comply with Fortran standard (for the first time)
- Example of non-compliance
  - call MPI_Send(real_buffer_argument, ...)
  - call MPI_Send(integer_buffer_argument, ...)
- Do you agree with this requirement

☐ yes

☐ no (why not)

Requirement 2: enhance type safety

- Requirement
  - enhance type safety (for user)
- Example
  - call MPI_Send(buf,count,datatype,dest,comm,tag,ierror)
  - call MPI_Send(buf,count,datatype,dest,tag,comm,ierror)
  - call MPI_Send(buf,count,datatype,dest,comm,ierror)
- Do you agree with this requirement

☐ yes

☐ no (why not)

Requirement 3: solve mapping to void* buffers

- Requirement
  - have standardized possibility to suppress argument checking for choice buffers (for implementors)
- Example
  - currently no good standards-based solution with explicit interfaces
    - Fortran interface is equivalent to a C prototype
    - must define interface for all combinations of (type X,kind X dim)
      (if the compiler does not support a non-standardized IGNORE option)
- Do you agree with this requirement

☐ yes

☐ no (why not)

Requirement 4: provide ability to write correct programs using asynchronous operations

- Requirement
  - guarantee of correct asynchronous operations
- Example of incorrect behavior
  - call MPI_Irecv(buf, ...)
  - compiler may be required to provide temporary copy of buf
    - problem: temporary copy deallocated on return from MPI_Irecv
- Do you agree with this requirement

☐ yes

☐ no (why not)

Requirement 4: provide ability to write correct programs using asynchronous operations (2)

- Requirement
  - guarantee of correct asynchronous operations
- Example of incorrect behavior
  - call MPI_Irecv(buf, ..., rq ,...)
  - ...
  - call MPI_Wait(rq)
  - a = buf(1)
  - Problem: code movement of reference to buf moved above MPI_Wait()

Requirement 5: provide a convenient migration path for users

- Requirement
  - provide users with convenient migration path
- Examples
  - allow “include mpif.h” and “use mpi_f08” in the same program
  - allow easy conversion between new and old Fortran handles
  - make usage as similar as possible
    - subroutines not functions
- Do you agree with this requirement

☐ yes

☐ no (why not)

Requirement 6: take advantage of some features in Fortran as a convenience to users

- Requirement
  - allow some optional arguments (e.g., ierror)
- Example of common error pattern
  - call MPI_Send(buf,count,datatype,dest,tag,comm)
  - call MPI_Send(buf,count,datatype,dest,tag,comm,ierror)
  - forgetting the ierror return argument will no longer be an error
- Do you agree with this requirement

☐ yes

☐ no (why not)

Requirement 7: take advantage of sub-arrays for convenience and performance

- Requirement
  - support sub-arrays
- Example
  - call MPI_Irecv(Array(1,:), ...)
  - instead:

    set up all the arguments for creating subarray dt
    call MPI_Type_create_subarray(...., dt, ierror)
    call MPI_Irecv(Array(:,:),1,dt,....)
- Do you agree with this requirement

☐ yes

☐ no (why not)

Requirement 8: allow vendors access to existing C implementations of Fortran wrappers

- Requirement
  - allow vendors access to existing C implementations of Fortran wrappers
- Example
  - Existing mpif.h and mpi module routine is, e.g., `mpi_send__`
  - For the new mpi_f08 module with existing Fortran 2003 compilers one should only do
    - substitute `*ierror` = ... by `if (ierror != NULL) *ierror` = ...
    - add an alias `mpi_send_f08__` to `mpi_send__`
- Do you agree with this requirement

☐ yes

☐ no (why not)

A very brief overview of design tradeoffs

- Now we take a look at some of the design tradeoffs made in implementing the requirements

Requirement: comply with fortran standard

- Requirement
  - comply with Fortran standard (for the first time)
- Example of non-compliance
  - call MPI_Send(real_buffer_argument, ...)
  - call MPI_Send(integer_buffer_argument, ...)
- Design decision
  - take advantage of new features in Fortran
    - TYPE(*), DIMENSION(..)
- Tradeoffs
  - wait for compiler support for full implementation of nonblocking routines

Requirement: enhance type safety

- Requirement
  - enhance type safety
- Example
  - call MPI_Send(buf,count,datatype,dest,comm,tag,ierror)
  - call MPI_Send(buf,count,datatype,dest,tag,comm,ierror)
- Design decision
  - MPI handles are user-defined types
    - TYPE(MPI_Comm)
- Tradeoffs
  - users must convert code to use new handle types

Requirement: provide ability to write correct programs using asynchronous operations

- Requirement
  - guarantee of correct asynchronous operations
- Example of incorrect behavior
  - call MPI_Irecv(buf, ...)
  - compiler may be required to provide temporary copy of buf
    - problem: temporary copy deallocated on return from MPI_Irecv
- Design decision
  - use DIMENSION(..) and ASYNCHRONOUS attribute for choice buffers
- Tradeoffs
  - vendors must convert implementation to access array descriptor

Requirement: provide a convenient migration path for users

- Requirement
  - provide users with convenient migration path
- Examples
  - allow easy conversion between new and old Fortran handles
  - make usage as similar as possible
- Design decision
  - handles are user defined types (think C structs) with an integer component
  - subroutines not functions
- Tradeoffs
  - wrapper functions still needed because of choice to use subroutines

Requirement: take advantage of some features in Fortran as a convenience to users

- Requirement
  - allow some optional arguments (e.g., ierror)
- Example of common error pattern
  - call MPI_Send(buf,count,datatype,dest,tag,comm)
  - call MPI_Send(buf,count,datatype,dest,tag,comm,ierror)
  - forgetting the ierror return argument will no longer be an error
- Design decision
  - ierror argument made optional
- Tradeoffs
  - None

Requirement: take advantage of sub-arrays for convenience and performance

- Requirement
  - support sub-arrays
- Example
  - call MPI_Irecv(Array(1,:), ...)
  - receive into top row of array
- Design decision
  - choice buffers declared with DIMENSION(..)
- Tradeoffs
  - vendors must convert implementation to access array descriptor

Decision for C programmers: to support or not to support MPI 3.0 Fortran bindings

- Do you agree with all or most of the requirements?
- Do the technical tradeoffs seem reasonable?
- Since you may not understand the technical details, do you trust the Fortran experts to make good decisions regarding the low-level technical details?
- Fortran Experts
  - WG5 Fortran committee agrees with general changes but don't follow details
  - Nick, Steve Lionel
  - Rolf (and contacts...)
  - Craig and Jeff