MPI-3.0 Fortran Tickets

Rolf Rabenseifner, Jeff Squyres, Craig Rasmussen

Major ideas

- Solving the argument checking problems
  - Allowing also checking of wrong handle types
- Being backward compatible
- Allowing triplet array subscripts a(lb:ub:incr) everywhere, i.e., also in nonblocking routines
  - No need of MPI_Type_create_subarray for buffer descriptions.
  - No local copying
    - Prohibited on compiler level
    - Not required inside the MPI library
    - Although a quick MPI implementation can do such copying, as done currently by the compilers
- Additional features:
  - optional ierror,
  - “status ignore” through overloading
  - INTENT (IN, OUT, INOUT)

Major ideas, continued

- Fixing the Fortran-MPI-incompatibilities
  - At least with advices to users how to use Fortran in combination with MPI
- New **mpi_f08** module with all features
  - Fully Fortran 2008 compatible definition of all MPI routines
- Existing **mpi** module with several enhancements
- Different buffer handling in C-wrappers for
  - Fortran 2008 compilers
  - Older Fortran levels
- The use of mpif.h is strongly discouraged in the future
- In the future, i.e. all Fortran compilers support "assumed-type&rank" and BIND(C) CHARACTER*(*), it is possible to have only one set of C-wrappers for all three: mpi_f08 & mpi module, and mpif.h
- We should install a common source code infrastructure for new mpi_f08

Problems in discussion with Fortran committees:

- The Fortran 2008 method assumed-type & assumed-rank is currently to weak
  - TYPE(*), DIMENSION(..) buf
- Requirement:
  - Existing interface
    - CALL MPI_SEND(any possible actual argument)
    - SUBROUTINE MPI_SEND defined as implicit interface
  - New explicit interface must allow all possible existing calls to MPI_SEND, i.e., all possible actual buffer arguments
- Problems:
  - SUBROUTINE MPI_SEND(…) BIND(C) with TYPE(*). DIMENSION(..) buf does not allow assumed-size arrays as actual arguments
  - BIND(C) does not allow CHARACTER*(*) string dummy arguments
  - → Hardest Problem: MPI_(UN)PACK_EXTERNAL has 2 choice buffers and also a string argument
- Fortran committees WG5 and J3 work on solutions → to be included into TR
  - Expectation: Solution only valid if MPI_.... defined as BIND(C)

#229-A – Overview on all related Fortran Tickets

- Ticket #229-A gives an overview on all related Fortran tickets
- Don’t worry that we have about 25 tickets
- They reflect separate decisions
- Before final voting,
  we can combine them to one single ticket
  (as done with nonblocking-collectives tickets)

- With all Fortran tickets together,
  we can solve all problems reported in MPI-2.2

- https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/229

#230-B - New module "USE mpi_f08"

- New module mpi_f08
  - With all new features:
    - Full compile-time argument checking

Specifically for the MPI Forum, the Fortran 2008 standardization committee developed new syntax TYPE(*), DIMENSION(..) to define choice buffers in a standardized way

- New wording (instead of Basic & Extended Fortran Support)
  - 3 Fortran support methods:
    - include ´mpif.h´
    - use mpi
    - use mpi_f08

Overview on all 3 Methods: **Include file mpif.h**

- Define all named MPI constants & declare MPI functions that return a value.
- For each MPI routine, an implementation can choose to use an implicit or explicit interface.
- The handles are defined as INTEGER
- mpif.h must be valid and equivalent for both fixed- and free- source form.
- **Advice to users: Instead of using mpif.h, the use of the mpi or mpi_f08 module is strongly encouraged. See … Reasons …**

- Almost the same as MPI-2.2
- Only some small additions to MPI-2.2 (based on these tickets)
- Of course, all new stuff from other MPI-3.0 tickets and chapters will be added

Overview on all 3 Methods: **Module mpi**

- Define all named MPI constants & declare MPI functions that return a value.
- **Provide explicit interfaces for all MPI routines → compile-time argument checking.**
- The handles are defined as INTEGER
  - Same values as in mpif.h

An alternative will be in new mpi_f08 module

- An MPI implementation may provide in the mpi module other features that enhance the usability of MPI while maintaining adherence to the standard.
  For example, it may provide argument attributes INTENT(IN,OUT,INOUT) in these interface blocks.

- Backward compatible to MPI-2.2 for applications with bug-free syntax
- Details later

Overview on all 3 Methods: **Module mpi_f08**

- Define all named MPI constants & declare MPI functions that return a value.
- **Provide explicit interfaces for all MPI routines → compile-time argument checking.**
- **All handles are defined with named types.**
- **If the Fortran compiler provides *assumed type* and *assumed rank:***
  - **All choice buffers are declared with TYPE(*), DIMENSION(..)**
  - **MPI_SUBARRAYS = MPI_SUBARRAYS_SUPPORTED**

  → non-contiguous sub-arrays are also <u>valid</u> in nonblocking routines.

  Only if the target compiler does **not** support <u>*assumed type* and *assumed rank*</u> or <u>an equivalent non-standard alternative</u>:
  - Same requirements as for mpi module
  - MPI_SUBARRAYS = MPI_SUBARRAYS_NOT_SUPPORTED

  → non-contiguous sub-arrays are **not** valid also in nonblocking routines

```
Advice to users.
IF (MPI_SUBARRAYS ==
    MPI_SUBARRAYS_SUPPORTED)
    …
ELSE IF (MPI_SUBARRAYS ==
    MPI_SUBARRAYS_NOT_SUPPORTED)
    …
ELSE
   PRINT *, 'value not yet standardized'
ENDIF
```

- With this module, **new Fortran 2008 definitions are added for each MPI routine**, except for routines that are deprecated in MPI-2.2.
- **Each argument has an INTENT=IN, OUT, or INOUT attribute** if appropriate.
- **All ierror output arguments are declared as optional**, except for user-defined callback functions (e.g., comm_copy_attr_fn) and their predefined callbacks (e.g., MPI_NULL_COPY_FN).

Available with all 3 methods

- Each application routine can freely choose one of the 3 methods.
- **Compile-time constant MPI_F08_SUBARRAYS is .TRUE.**
  if all choice buffer arguments are implemented with TYPE(*), DIMENSION(..)
  - Can be set different within all 3 methods
- MPI_SIZEOF, MPI_TYPE_MATCH_SIZE, MPI_TYPE_CREATE_F90_INTEGER, MPI_TYPE_CREATE_F90_REAL and MPI_TYPE_CREATE_F90_COMPLEX.
- **New MPI_SYNC_REG as one of several methods to solve register optimization problems.**
- Handle values are equivalent in *mpif.h* & *USE mpi*
- **Handles easily convertible to/from *USE mpi_f08***
- All new MPI-3.0 features are implemented in all 3 Fortran support methods

#231-C - Fortran argument checking with individual handles

- **This is the first major topic of new mpi_f08 module.**
- In mpi_f08, all handles use named types:

```
TYPE MPI_Comm
   SEQUENCE
   INTEGER   :: MPI_VAL
END TYPE MPI_Comm
```

- Trivial conversion between old-style (module mpi) and new handles:

```
comm_new%MPI_VAL = comm_old
```

- Same C-binding: one Fortran integer
  - The new handles do not require any changes for the wrappers written in C

Straw	Yes	No	Abstain
Votes:	11	-	5

#232-D: Existing module "USE mpi" with argument checking

- **Compile-time argument checking will now be mandatory for mpi module**
  - Can be done because most relevant Fortran compilers know directives like

    ```
    !DEC$ ATTRIBUTES NO_ARG_CHECK :: BUF
    !$PRAGMA IGNORE_TKR BUF
    REAL, DIMENSION(*) :: BUF
    ```

    - To be checked: Those directives imply same argument handling as with implicit interfaces in mpif.h for choice buffers
      → May need a Fortran complier-specific flag!
    - (such additional compiler flags are okay for compiling the module, but not a good idea for compiling the user application)
  - Only if the compiler has no such method, then implicit interface is still okay
  - No checking of MPI handle types (all are INTEGER)

Straw	Yes	No	Abstain
Votes:	7	-	11

#233-E: **The use of 'mpif.h' is strongly discouraged**

- **Advice to users: Instead of using mpif.h, the use of the mpi or mpi_f08 module is strongly encouraged. See … Reasons …**
  - Most have no compile-time argument checking
  - Too many bugs are in MPI applications, e.g., due to
    - Missing IERROR as last additional argument in Fortran
    - Status only as INTEGER,
      instead of INTEGER, DIMENSION(MPI_STATUS_SIZE)
    - Passing the wrong MPI handle types
    - …
  - Easy migration to USE mpi
    - For syntax-correct programs
- We do not deprecate mpif.h because
  - All new features should be still added to mpif.h
  - Because there is only **one** interface per routine for mpif.h and USE mpi

Straw Votes:	Yes	No	Abstain
	9	-	10

#234-F: Choice buffers through "TYPE(*), DIMENSION(..)"

- **This is the second major topic of new mpi_f08 module.**
- **All choice buffers are declared with new Fortran 2008**
  - **TYPE(*), DIMENSION(..) = assumed type & assumed rank**
- Internally, a dope-vector (i.e., a descriptor) is passed to the Fortran wrapper (The dope-vector is generated by the complier and describes the buffer.)
- Based on this dope-vector, a virtual or real copying from non-contiguous to contiguous scratch arrays can be done.
- This contiguous scratch buffer is under control of MPI
- With non-blocking routines, it must not be released before MPI_Wait
- Implication:
  - All Fortran sub-arrays can be used also in non-blocking MPI routines
  - Handling of sub-arrays can be done through Fortran syntax instead of MPI derived datatypes
- Exception: If compiler without TYPE(*), DIMENSION(..) → MPI_F08_SUBARRAYS=.FALSE.

Many thanks to the Fortran 2008 standardization committee

How stable is a TR?

Is it official?

Straw Votes:	Yes	~~No~~	Abstain
	13	-	2

Provided that TYPE(*), DIMENSION(..) will have Fortran Standard Quality

#235-G: Corrections to "Problems with Fortran Bindings" (p. 481) and "Problems Due to Strong Typing" (p.482)

- No decisions, only new wording that is more correct.
- Corrections specific to the 3 methods (mpi_f08, mpi, mpif.h)

#236-H: Corrections to "Problems Due to Data Copying and Sequence Association" (MPI-2.2 page 482)

- No decision, mainly new wording to reflect the new methods and the constant MPI_F08_SUBARRAYS

```
USE mpi_f08
REAL :: buf(100)
CALL MPI_Irecv(buf(1:100:7), 15, MPI_REAL, …, req)
…
CALL MPI_Wait(req, …)
```

buf(1), buf(8), buf(15), buf(22), … buf(99)

Triplet-subscripts can now be used in non-blocking routines if compile-time constant MPI_F08_SUBARRAYS is .TRUE.

- New unsolved minor problem:
  Vector-subscripts still not usable in nonblocking calls, e.g. buf([1,2,24,25,33,34])
  - It is not planned to solve this problem, i.e, it is okay that only triplets are solved

#237-I: Corrections to "Problems Due to Fortran 90 Derived Types" (MPI-2.2 page 484)

- No decision, only new wording that correct this wrong section.
- Currently, MPI-2.2 says
  - “MPI does not explicitly support passing Fortran 90 derived types to dummy arguments. …
    Use of the SEQUENCE attribute may help here, somewhat.
- Reality is that MPI datatypes
  - work correctly with Fortran sequence derived types, and
  - are not guaranteed to work for Fortran non-sequence derived types.
- The section must be therefore corrected.

#238-J: Corrections to "Registers and Compiler Optimizations" (p. 371) and "A Problem with Register Optimization" (page 485)

- Additional advice about already known "Problems with MPI and Fortran optimization“ (or any future method)
  - New advice with the Fortran **TARGET** attribute
    - Solves problems with: nonblocking calls, MPI_BOTTOM, and 1sided
  - New advice with the Fortran **ASYNCHRONOUS** attribute
    - Solves problems with: nonblocking calls
  - Additional helper routine **MPI_F_SYNC_REG** to substitute the user-written DD (buf)
  - Module data and common blocks also work for all three problems
- New problem with "Temporary Memory Modifications"
  - Solved with **ASYNCHRONOUS**, but not with TARGET attribute, DD, MPI_SYNC_REG, module data, or common blocks
- **This is the third major topic of new mpi_f08 module: It is about correctness of MPI nonblocking, 1-sided, and MPI_BOTTOM based calls in Fortran.**

Additional Problem:

buf1 + buf2 in one datatype

MPI_Send(buf1,1,datatype,…)

Straw Votes:	Yes	No	Abstain
	6	-	9

#239-K: IERROR optional

- In the current MPI Fortran interface, the IERROR dummy argument is mandatory.
- In the MPI C interface, the MPI routines can be called as
  - a function (i.e., the ierror value is returned), or
  - as a procedure (i.e., ignoring the ierror value),

  and therefore the ierror is optional.
- **With this ticket, the Fortran IERROR dummy argument is declared as optional** in all MPI routines that provide an IERROR.
  - Exception: For user-defined callback functions (e.g., comm_copy_attr_fn) and their predefined callbacks (e.g., MPI_NULL_COPY_FN), ierror should not be optional
- An MPI implementation can also choose to use function overloading instead of implementing IERROR as optional
  - Advantage: Compile-time decision
  - Drawback: Doubling number of wrappers

- Implementation: We have to check that wrappers in C can test for this **optional** Fortran IERROR argument !!!

Straw Votes:	Yes	No	Abstain
	14	-	1

#240-L: New syntax used in the description of Fortran general interfaces

- New Fortran 95 style, e.g., INTEGER :: MPI_VERSION
- New wordings

Straw Votes:	Yes	No	Abstain
	10	-	-

#241-M: Not including old deprecated routines

- Not to include deprecated routines into the new Fortran 2008 bindings.

Straw	Yes	No	Abstain
Votes:	by acclamation		

#242-N: Arguments with INTENT=IN, OUT, INOUT

- **Use of INTENT=IN, OUT, INOUT attributes in all new Fortran 2008 bindings.**
- The Fortran attribute INTENT(IN) is used for all arguments that are IN arguments in the language-independent notation.
- For OUT or INOUT arguments in the language-independent notation, the Fortran attributes INTENT(OUT) or INTENT(INOUT) are used, with following exceptions:
  - If there exists a constant that can be provided as actual argument, then an INTENT attribute is not specified. Examples:
    - MPI_BOTTOM and MPI_IN_PLACE for buffer arguments;
    - MPI_UNWEIGHTED in sourceweights and destweights in MPI_Dist_graph_neighbors.
  - If the argument is a handle type argument and is implemented in C with call-by-value, then INTENT(IN) is specified. Examples:
    - All file-handles in MPI_Write routines;
    - the request in MPI_Grequest_complete.
    - Exception: MPI_Cancel with INTENT(IN) request
  - Buffers without INTENT
- An MPI implementation is allowed to use more restrictive INTENT
  - E.g., INTENT(IN) for send buffers

Straw	Yes	No	Abstain
Votes:	11	-	3

#243-O: MPI_Status as a Fortran derived type

- **TYPE(MPI_Status) status**

Reasons:

- The existing status(MPI_STATUS_SIZE) array is awkward
  - status%MPI_SOURCE versus status(MPI_SOURCE)
- Wrong accesses through integer indexes cannot be detected at compile-time
  - status(1) instead of status(MPI_SOURCE)
- Wrong arrays size is also not detected at compile-time
  - INTEGER status(1) instead of INTEGER status(MPI_STATUS_SIZE)

Advice to implementors.
With the SEQUENCE attribute, one can implement this Fortran derived type with the same memory layout as the Fortran status (MPI_STATUS_SIZE) array and the C structure MPI_Status.

Straw	Yes	No	Abstain
Votes:	9	-	5

#244-P: MPI_STATUS(ES)_IGNORE with function overloading

With USE mpi_f08, the user can freely choose

- CALL MPI_Recv(buf,cnt,datatype,src,tag,comm,status,ierror)
- CALL MPI_Recv(buf,cnt,datatype,src,tag,comm, ierror)
- CALL MPI_Recv(buf,cnt,datatype,src,tag,comm,status)
- CALL MPI_Recv(buf,cnt,datatype,src,tag,comm)

- Some routines are often in the critical path:
  - Function overloading is at compile-time
  - → no conditional branch at run-time
  - → Function overloading is more efficient
- Only 36 routines with status output argument
- Same API cannot be done with OPTIONAL status argument, i.e., with OPTIONAL status, users must write
  - CALL MPI_File_write(fh,buf,count,datatype, IERROR=ierror)

  instead of
  - CALL MPI_File_write(fh,buf,count,datatype, ierror)

Note that here, ierror may be needed, because in all I/O routines, ERORS_RETURN is the default!

Straw Votes:	Yes	No	Abstain
	5	**2**	5

#245-Q: Upper and lower case letters in new Fortran bindings

- The new interfaces in mpi_f08 look like:

```
SUBROUTINE MPI_Recv(buf, count, datatype, source, tag, comm, status, ierror)
        TYPE(*), DIMENSION(..) :: buf
        TYPE(MPI_Datatype), INTENT(IN)  :: datatype
        TYPE(MPI_Comm), INTENT(IN)  :: comm
        INTEGER, INTENT(IN)  :: count, source, tag
        TYPE(MPI_Status), INTENT(OUT) :: status
        INTEGER, OPTIONAL, INTENT(OUT) :: ierror
END
```

- Bill and Adam “hate" this → We will discuss the reasons

Straw Votes:	Yes	No	Abstain
	--- deferred ---		

#246-R: MPI_ALLOC_MEM and Fortran

- How to use MPI_ALLOC_MEM together with C-Pointers in Fortran. (instead of non-standard Cray-Pointers)

```
SUBROUTINE MPI_Alloc_mem(size, info, baseptr, ierror)
USE, INTRINSIC :: ISO_C_BINDING
INTEGER(KIND=MPI_ADDRESS_KIND), INTENT(IN) :: size
TYPE(MPI_Info), INTENT(IN) :: info
TYPE(C_PTR), INTENT(OUT) :: baseptr
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
END
```

```
SUBROUTINE MPI_Free_mem(base, ierror)
TYPE(C_PTR), INTENT(IN) :: base
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
END
```

- New interface that can be used together with ALLOCATABLE arrays

Example: To be done

#247-S: All new Fortran 2008 bindings – Part 1

- This ticket shows the principles and some special details

#248-T: All new Fortran 2008 bindings – Part 2

Half-automatically generated from the existing Fortran bindings!

A.4 Fortran 2008 Bindings with module mpi_f08

A.4.1 Point-to-Point Communication Fortran Bindings

```
SUBROUTINE MPI_Bsend(buf, count, datatype, dest, tag, comm, ierror)  BIND(C)
TYPE(*), DIMENSION(..) :: buf
INTEGER, INTENT(IN) :: count, dest, tag
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Comm), INTENT(IN) :: comm
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
END

SUBROUTINE MPI_Bsend_init(buf, count, datatype, dest, tag, comm, request, ierror)  BIND(C)
TYPE(*), DIMENSION(..) :: buf
INTEGER, INTENT(IN) :: count, dest, tag
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
END
...
```

#250-V: Minor Corrections in Fortran Interfaces

- Typo in the existing Fortran Interface of MPI_INTERCOMM_MERGE:
  - INTRACOMM → **NEW**INTRACOMM
- Remove double definition of request in the Fortran binding type declaration part of MPI_SEND_INIT and MPI_BSEND_INIT

- Regular errata – in all three Fortran support methods

Straw Votes:	Yes	No	Abstain
	by acclamation		

#252-W: Substituting dummy argument name "type" by "datatype" or "oldtype"

- To mimimize conflicts with language keywords (TYPE in Fortran), the dummy argument name "type" is substituted by "datatype" or "oldtype".

- Note, with explicit interfaces, the user can freely choose between
  - Positional argument lists
    - CALL MPI_Send(buf,cnt,datatype,src,tag,comm,ierr)
  - Keyword-based argument lists and mixed lists
    - CALL MPI_Send(buf,cnt,datatype,source=src, &
      &   tag=13,comm=MPI_COMM_WORLD,ierror=ierr)

→ Dummy argument names *should be done correctly* & *should make sense*

Straw Votes:	Yes	No	Abstain
	by acclamation		

Conclusion

- Although separated into tickets, it is one big packet.
- We needed 3 years to detect most of the MPI-Fortran-problems (1994-1997)
- We needed additional 13 year to hopefully solve them all (1997-2010) !
- And we still detect new ones ☺ ☹ ☺ ☹ ☺

- Thanks to the Fortran Standardization Committees WG5 and J3
  for their working together to solve the MPI-Fortran incompatibility problems.

Appendix

- List of all Fortran tickets

Tickets related to new MPI-3.0 Fortran Interface

- #229-A - Overview over all related tickets
- #230-B - New module "USE mpi_f08"
- #231-C - Fortran argument checking with individual handles
- #232-D - Existing module "USE mpi" with argument checking
- #233-E - The use of ‘mpif.h’ is strongly discouraged
- #234-F - Choice buffers through TYPE(*) DIMENSION(..) declarations
- #235-G - Corrections to "Problems with Fortran Bindings" (p.481) & "Strong Typing" (482)
- #236-H - Corrections to "Problems Due to Data Copying and Sequence Association" (482)
- #237-I - Corrections to problems due to "Fortran 90 Derived Types" (MPI-2.2 page 484)
- #238-J - Corrections to "A Problem with Register Optimization" (pages 371 and 485)
- #239-K - IERROR optional
- #240-L - New syntax used in all three (mpif.h, mpi, mpi_f08)

https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/229 .. 253

Tickets related to new MPI-3.0 Fortran Interface

- #241-M - Not including old deprecated routines from MPI-2.0 - MPI-2.2
- #242-N - Arguments with INTENT=IN, OUT, INOUT
- #243-O - MPI_Status as a Fortran derived type
- #244-P - MPI_STATUS(ES)_IGNORE with function overloading
- #245-Q - Upper and lower case letters in new Fortran bindings
- #246-R - MPI_ALLOC_MEM and Fortran
- #247-S - All new Fortran 2008 bindings - Part 1
- #248-T - All new Fortran 2008 bindings - Part 2
- #249-U - Alternative formulation for Section 16.2 Fortran Support
- #250-V - Minor Corrections in Fortran Interfaces
- #252-W - Substituting dummy argument name "type" by "datatype" or "oldtype"
- #253-X - mpi_f08 Interfaces for new MPI-3.0 routines (not yet done)

- #251 is currently a helper ticket: Printable version of all tickets together