

# MPI-3.0 Overview

Rolf Rabenseifner



## Acknowledgements

- Some detailed slides are provided by the
  - Ticket authors,
  - Chapter authors, or
  - Chapter working groups.
- Richard Graham, chair of MPI-3.0.

## Goal & Scope

- Goal:
  - To produce new versions of the MPI standard that better serves the needs of the parallel computing user community
- Scope:
  - Additions to the standard that are needed for better **platform** and **application** support.
  - These are to be consistent with MPI being a library providing process group management and data exchange. This includes, but is not limited to, issues associated with scalability (performance and robustness), multi-core support, cluster support, and application support.
  - And of course,  
all needed corrections to detected bugs / ambiguities / inconsistencies
  - Backwards compatibility may be maintained —  
Routines may be deprecated or deleted.

## Planned Schedule to MPI-3.0

- July 16-19, 2012, Chicago: Final votes for draft 2
  - Some additional days for finishing the draft document
- Aug. 2, 2012: mpi3.0\_draft\_2.pdf is released
  - See [http://meetings.mpi-forum.org/MPI\\_3.0\\_main\\_page.php](http://meetings.mpi-forum.org/MPI_3.0_main_page.php)
  - **Request for comments** sent out to the public
    - Subscribe to the mpi-comments mailing list
    - Send mail to [mpi-comments@lists.mpi-forum.org](mailto:mpi-comments@lists.mpi-forum.org)
  - Public comment period ends on Sep. 6, 2012
- Sep. 7 – 12, 2012: Late changes based on public comments
- Sep. 12, 2012: mpi3.0\_draft\_3.pdf
- MPI-3 Forum meeting, Sep. 20-21, 2012 in Vienna:
  - Final chapter votes on Thursday, Sep. 20
  - Final standard vote on Friday, Sep. 21 (state of this schedule: Aug. 18, 2012)

# Improved Collective Communication

- Scalable sparse collectives on process topologies
  - `MPI_(l)Neighbor_allgather(v) / alltoall(v/w):`
  - `MPI_Aint` displs in `(l)Neighbor_alltoallw` (instead of int) →Slide  
14 #258<sub>21</sub>  
#299<sub>21</sub>
  
- Nonblocking collectives
  - Communication: `MPI_Ibarrier`, `MPI_Ibcast`, ... →Slide  
15 #109<sub>13</sub>  
#168<sub>15</sub>
  - `MPI_Icomm_dup`

Background information, see:

- <https://svn.mpi-forum.org/trac/mpi-forum-web/query>
- #Item in MPI-3.0, Change-Log, Annex B.1.1 (E1-E6) and B.1.2 (1-42)  
 The item numbers in this document are based on MPI-3.0 Draft 2, Aug. 2, 2012:  
[http://meetings.mpi-forum.org/draft\\_standard/mpi3.0\\_draft\\_2.pdf](http://meetings.mpi-forum.org/draft_standard/mpi3.0_draft_2.pdf)  
 Final change-log item numbers may be different.

## Improvements to one-sided communication support

Main ticket

→Slides  
16-21 #270<sub>24</sub>

- **Routines:** MPI\_Rget / Rput / Raccumulate / Get\_accumulate / Rget\_accumulate / Fetch\_and\_op / Compare\_and\_swap  
MPI\_Win\_allocate / create\_dynamic / attach / detach / lock\_all / unlock\_all / flush / flush\_all / flush\_local / flush\_local\_all / sync
  - **New operation** MPI\_NO\_OP
  - **New attribute** MPI\_WIN\_CREATE\_FLAVOR = MPI\_WIN\_FLAVOR\_CREATE / ALLOCATE / DYNAMIC
  - **New attribute** MPI\_WIN\_MODEL = MPI\_WIN\_SEPARATE / UNIFIED
- 
- Shared memory RMA window (see also slide on “hybrid” →Slide  
7 ) #284<sub>24</sub>
    - **Routine:** MPI\_Win\_allocate\_shared, MPI\_Win\_shared\_query →Slides  
22-23
    - **Flavor attribute:** MPI\_WIN\_FLAVOR\_SHARED

## Support for clusters of SMP nodes (hybrid support)

- Topology aware communicator creation →Slide  
21 #287<sub>17</sub>
  - **MPI\_Comm\_split\_type with split\_type=MPI\_COMM\_TYPE\_SHARED**
- Shared memory RMA window (see also slide on “one-sided” →Slide  
6) #284<sub>24</sub>
  - **Routine: MPI\_Win\_allocate\_shared, MPI\_Win\_shared\_query** →Slides  
22+23
  - **Flavor attribute: MPI\_WIN\_FLAVOR\_SHARED**
- Thread-safe probe → new probe routine and message object #38<sub>11</sub>
  - **MPI\_(I)Mprobe, MPI\_(I)Mrecv** →Slide  
24
  - **MPI\_Message\_f2c / c2f** #274<sub>11</sub>

## Other Forum Activities

- New Fortran bindings: mpi\_f08 module →Slides 25-37 #229<sub>26-42</sub>
- Non-collective communicator formation: MPI\_Comm\_create\_group #286<sub>16</sub>
  - Tags are in a special tag-space. →Slide 38
  - MPI\_INTERCOMM\_CREATE has its own tag-space #305<sub>18</sub>
- New Datatype Creation Routine: MPI\_Type\_create\_hindexed\_block →Slide 39 #280<sub>12</sub>
- Large counts: #265<sub>6,8</sub>
  - type MPI\_Count / INTEGER(KIND=MPI\_COUNT\_KIND) →Slide 40
  - datatype MPI\_COUNT
  - New routines: MPI\_Type::size\_x / get\_extent\_x / get\_true\_extent\_x, MPI\_Get\_elements\_x, MPI\_Status\_set\_elements\_x
- MPI\_Init, MPI\_Init\_thread, and MPI\_Finalize were clarified. #313<sub>22</sub>
  - New predefined info object **MPI\_INFO\_ENV** holds arguments from mpiexec or MPI\_COMM\_SPAWN



## New Tool Interface

- Main ticket #266<sub>25</sub>
  - Slides 41-46
    - MPI\_T\_init\_thread / finalize / enum\_get\_info/item, MPI\_T\_cvar\_: get\_num/info / handle\_alloc/free / read / write, MPI\_T\_pvar\_: get\_num/info / session\_create/free / handle\_alloc/free / start / stop / read / write / reset / readreset, MPI\_T\_category\_: get\_num/info/cvars/pvars/categories / changed
    - MPI\_T routines are callable before MPI\_Init & after MPI\_Finalize #266<sub>23</sub>
    - MPI\_Get\_library\_version #204<sub>5</sub>
    - MPI\_Comm\_dup\_with\_info / set\_info / get\_info, MPI\_Win\_set/get\_info
      - MPI\_COMM\_DUP must also duplicate info hints. #271<sub>14</sub>
  - MPIR (independent document, not part of the MPI standard) #228
    - “The MPIR Process Acquisition Interface”
    - a commonly implemented interface primarily used by debuggers to interface to MPI parallel programs

## Deprecated functionality *moved* to Chapter 16. Removed Interfaces

- Removed MPI-1.1 functionality (deprecated since MPI-2.0): #303<sub>1</sub>
  - Routines: MPI\_ADDRESS, MPI\_ERRHANDLER\_CREATE / GET / SET, MPI\_TYPE\_EXTENT / HINDEXED / HVECTOR / STRUCT / LB / UB
  - Datatypes: MPI\_LB / UB
  - Constants MPI\_COMBINER\_HINDEXED/HVECTOR/STRUCT\_INTEGER
  - Removing deprecated functions from the examples and definition of MPI\_TYPE\_GET\_EXTENT #278<sub>1</sub>
- C++ Bindings are removed (deprecated since MPI-2.2) #281<sub>2</sub>

→Slide  
47

  - C++ applications are supported through the MPI C language binding
  - Special C++ types are supported through additional MPI predefined datatypes #340<sub>E1</sub>
    - MPI\_CXX\_BOOL bool
    - MPI\_CXX\_FLOAT\_COMPLEX std::complex<float>
    - MPI\_CXX\_DOUBLE\_COMPLEX std::complex<double>
    - MPI\_CXX\_LONG\_DOUBLE\_COMPLEX std::complex<long double>

## Minor Corrections and Clarifications

- Consistent use of [] for input and output arrays #125, #126<sub>7</sub>
  - Exception: MPI\_INIT and MPI\_INIT\_THREAD: char \*\*\*argv
- Add const keyword to the C bindings. “IN” was clarified. #140<sub>3</sub>
- MPI\_STATUSES\_IGNORE can be used in MPI\_(I)(M)PROBE #229.2<sub>9</sub>
- MPI\_PROC\_NULL behavior for MPI\_PROBE and MPI\_IPROBE #256<sub>10</sub>
- MPI\_UNWEIGHTED should not be NULL #294<sub>4</sub>
- MPI\_Cart\_map with num\_dims=0 #162<sub>20</sub>
- MPI\_MAX\_OBJECT\_NAME used in MPI\_Type/win\_get\_name #219<sub>19</sub>
- New wording in reductions:  
Multi-language types MPI\_AINT, MPI\_OFFSET, MPI\_COUNT #187
- MPI\_TYPE\_CREATE\_RESIZED should be used for “arrays of struct” #229.2<sub>32</sub>
  - The MPI alignment rule cannot guarantee to calculate the same alignments as the compiler

## Six MPI-2.2 errata items, see Change-Log Annex B.1.1

- C++ corrections were added as MPI-2.2 errata, because MPI-2.2 is the last version that includes the MPI C++ language binding.
  - Item 1: **MPI\_CXX\_BOOL** and **MPI\_CXX\_FLOAT/DOUBLE/LONG\_DOUBLE\_COMPLEX** are added to the list of predefined datatype handles (in C and Fortran) #340<sub>E1</sub>
  - The nonstandard C++ types `Complex<...>` were substituted by the standard types `std::complex<...>`. #340<sub>E1</sub>
  - Items 3, 5, 6: minor corrections #192<sub>E3</sub> #166<sub>E5</sub> #202<sub>E6</sub>
- `MPI_C_COMPLEX` was added to the *Complex* reduction group (Item 2) #340<sub>E2</sub>
- The `MPI_C_BOOL` "external32" representation is 1-byte (Item 4) #171<sub>E4</sub>

## The MPI-3.0 Change-Log, Annex B.1.2 has two parts:

- Items 1-25 (1-26 in final 3.0): General changes
- Times 26-33 (27-34 in final 3.0): Changes related to Fortran

## Details about most & important topics

- Slide 14: Sparse and scalable irregular collectives
- Slide 15: Nonblocking collectives
- Slides 17-21: One-sided communication – enhancements
- Slides 22-23: Shared memory extensions (on clusters of SMP nodes)
- Slide 24: Mprobe for hybrid programming on clusters of SMP nodes
- Slides 25-37: Fortran interface
- Slide 38: Group-Collective Communicator Creation
- Slide 39: `MPI_TYPE_CREATE_HINDEXED_BLOCK`
- Slide 40: Large Counts
- Slides 41-46: New tools interface
- Slide 47: Removing C++ bindings from the Standard

Background information, see: <sup>nn</sup>  
#Item in MPI-3.0, Change-Log, B.1.2 <sup>(1-42)</sup>  
The item numbers in this document are based on MPI-3.0 Draft 2, Aug. 2, 2012.  
[http://meetings.mpi-forum.org/draft\\_standard/mpi3.0\\_draft\\_2.pdf](http://meetings.mpi-forum.org/draft_standard/mpi3.0_draft_2.pdf)  
Final change-log item numbers may be different.

## Sparse Collective Operations on Process Topologies

21

- MPI process topologies (Cartesian and (distributed) graph) usable for communication
  - `MPI_NEIGHBOR_ALLGATHER(V)`
  - `MPI_NEIGHBOR_ALLTOALL(V,W)`
  - Also nonblocking variants
  - Accepted for MPI-3 in Sept. 2011
  - Reference implementation exists
- If the topology is the full graph, then neighbor routine is identical to full collective communication routine
  - Exception: `s/rdispls` in `MPI_NEIGHBOR_ALLTOALLW` are `MPI_Aint`
- Allow for optimized communication scheduling and scalable resource binding

Courtesy of Torsten Hoefler and Richard Graham

## Nonblocking Collective Communication and MPI\_ICOMM\_DUP

13+15

- Idea
  - Collective initiation and completion separated
  - Offers opportunity to overlap computation and communication
  - Each blocking collective operation has a corresponding nonblocking operation
  - May have multiple outstanding collective communications on the same communicator
  - Ordered initialization
- Voted into the MPI-3.0 draft standard in June 2009,
  - Reference Implementation (LibNBC) stable
  - Several implementations pending
- Parallel MPI I/O: The split collective interface may be substituted in the next version of MPI

Courtesy of Torsten Hoefler and Richard Graham

# MPI One-Sided Communication Interface

24

Courtesy of the MPI-3 One-sided working group



## Background of MPI-2 One-Sided Communication

- MPI-2's one-sided communication provides a programming model for put/get/update programming that can be implemented on a wide variety of systems
- The “public/private” memory model is suitable for systems without local memory coherence (e.g., special memory in the network; separate, non-coherent caches between actors working together to implement MPI One-Sided)
- The MPI-2 interface, however, does not support some other common one-sided programming models well, which needs to be fixed
- Good features of the MPI-2 one-sided interface should be preserved, such as
  - Nonblocking RMA operations to allow for overlap of communication with other operations
  - Support for non-cache-coherent and heterogeneous environments
  - Transfers of noncontiguous data, including strided (vector) and scatter/gather
  - Scalable completion (a single call for a group of processes)

## Goals for the MPI-3 One-Sided Interface

- Address the limitations of MPI-2 RMA by supporting the following features:
  - In order to support RMA to arbitrary locations, no constraints on memory, such as symmetric allocation or collective window creation, should be required
  - RMA operations that are imprecise (such as access to overlapping storage) must be permitted, even if the behavior is undefined
  - The required level of consistency, atomicity, and completeness should be flexible
  - Read-modify-write and compare-and-swap operations are needed for efficient algorithms

## Major New Features in the MPI-3 One-sided Interface

- New types of windows
  - `MPI_Win_allocate` – returns memory allocated by MPI; permits symmetric allocation
  - `MPI_Win_create_dynamic` – allows any memory to be attached to the window dynamically as needed
  - `MPI_Win_allocate_shared` – creates a window of shared memory that enables direct load/store accesses with RMA semantics to other processes in the same shared memory domain (e.g., the same node)
- New atomic read-modify-write operations
  - `MPI_Get_accumulate`, `MPI_Fetch_and_op`, `MPI_Compare_and_swap`
- New synchronization and completion calls, including:
  - Wait and test on request-based one-sided operations
  - Completion of pending RMA operations within passive target access epochs (`MPI_Win_flush` and variants)

## Major New Features – cont'd

- Query for new attribute to allow applications to tune for cache-coherent architectures
  - Attribute `MPI_WIN_MODEL` with values
    - `MPI_WIN_UNIFIED` on cache-coherent systems
    - `MPI_WIN_SEPARATE` otherwise
- Relaxed rules for certain access patterns
  - Results undefined rather than erroneous; matches other shared-memory and RDMA approaches
- Ordering of Accumulate operations
  - Change: ordering provided by default
  - Can be turned off for performance, using a new info key

## Status of the MPI-3 One-sided Interface

- Passed final 2<sup>nd</sup> (final) vote at the January 2012 meeting
- Example implementation on top of Portals-4 available
  - thanks to Brian Barrett of Sandia National Labs
- Other implementations in progress

## Shared Memory Extensions to MPI

17

- Step 1: Topology-aware communicator creation
  - Allows you to create a communicator whose processes can create a shared memory region
  - `MPI_Comm_split_type( comm, comm_type, key, info, split_comm)`
  - More generally: it splits a communicator into subcommunicators `split_comm` of a certain type:
    - `MPI_COMM_TYPE_SHARED`: shared memory capability
    - Other implementation specific types are possible: rack, switch, etc.
      - But not yet standardized.

## Shared Memory Extensions to MPI – Cont'd


24

- Step 2: Creation and manipulation of shared memory (based on MPI RMA semantics)
- `MPI_WIN_ALLOCATE_SHARED( size, info, split_comm, baseptr, win )`
  - Collective call that allocates memory of least size bytes that is shared among all processes in comm
  - Returns locally allocated region pointed to by baseptr that can be used for load/store access on the calling process
  - Consistent view of shared memory can be created in the RMA unified memory model by using window synchronization functions or by calling `MPI_WIN_FLUSH()`

## Thread-safe probe: MPI\_(I)MPROBE & MPI\_(I)MRECV

11

- MPI\_PROBE & MPI\_RECV together are not thread-safe:
  - Within one MPI process, thread A may call MPI\_PROBE
  - Another thread B may steal the probed message
  - Thread A calls MPI\_RECV, but may not receive the probed message
- New thread-safe interface:
  - MPI\_IMPROBE(source, tag, comm, flag, message, status) or
  - MPI\_MPROBE(source, tag, comm, message, status)together with
  - MPI\_MRECV(buf, count, datatype, message, status) or
  - MPI\_IMRECV(buf, count, datatype, message, request)



Message handle,  
e.g., stored in a thread-  
local variable









## MPI 3.0 Fortran Bindings

26-42



- A high-level summary for non-Fortran programmers
- Details for Fortran programmers

## 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

Slide: Courtesy of Jeff Squyres and Craig Rasmussen

## Three methods of Fortran support

- USE mpi\_f08  26
  - This is the only Fortran support method that is consistent with the Fortran standard (Fortran 2008 + TR 29113 and later).
  - This method is highly recommended for all MPI applications.
  - Mandatory compile-time argument checking & unique MPI handle types.
  - Convenient migration path.
- USE mpi
  - This Fortran support method is **inconsistent** with the Fortran standard, and its use is therefore **not recommended**.
  - It exists only for backwards compatibility.
-  INCLUDE 'mpif.h' 39
  - The use of the include file mpif.h is **strongly discouraged** starting with MPI-3.0. 40
  - Does not guarantee compile-time argument checking.
  - Does not solve the optimization problems with nonblocking calls,
  - and is therefore inconsistent with the Fortran standard.
  - It exists only for backwards compatibility with legacy MPI applications.

# The mpi\_f08 Module



Mainly for implementer's reasons.  
Not relevant for users.

- Example:

MPI\_Irecv(buf, count, datatype, source, tag, comm, request, ierror) BIND(C)

```

TYPE(*), DIMENSION(..), ASYNCHRONOUS :: buf
INTEGER, INTENT(IN) :: count, source, tag
TYPE(MPI_Datatype), INTENT(IN) :: datatype
TYPE(MPI_Comm), INTENT(IN) :: comm
TYPE(MPI_Request), INTENT(OUT) :: request
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
    
```

Fortran compatible buffer declaration allows correct compiler optimizations

Unique handle types allow best compile-time argument checking

INTENT → Compiler-based optimizations & checking

MPI\_Wait(request, status, ierror) BIND(C)

```

TYPE(MPI_Request), INTENT(INOUT) :: request
TYPE(MPI_Status) :: status
INTEGER, OPTIONAL, INTENT(OUT) :: ierror
    
```

Status is now a Fortran structure, i.e., a Fortran derived type

OPTIONAL ierror: MPI routine can be called without ierror argument

## Major changes

- Support method:

USE mpi or INCLUDE 'mpif.h'



→ USE mpi\_f08

26

- Status

INTEGER, DIMENSION(MPI\_STATUS\_SIZE) :: status

→ TYPE(MPI\_Status) :: status

30

status(MPI\_SOURCE)

→ status%MPI\_SOURCE

status(MPI\_TAG)

→ status%MPI\_TAG

status(MPI\_ERROR)

→ status%MPI\_ERROR

**Additional routines and declarations are provided for the language interoperability of the status information between**

- C,
- Fortran mpi\_f08, and
- Fortran mpi & mpif.h

## Major changes, continued

- Unique handle types, e.g.,
  - INTEGER new\_comm
- Handle comparisons, e.g.,
  - req.EQ. MPI\_REQUEST\_NULL

**new**

Same names as in C

TYPE, BIND(C) :: MPI\_Comm  
INTEGER :: MPI\_VAL  
END TYPE MPI\_Comm

27

→ **TYPE(MPI\_Comm) :: new\_comm**

No change through overloaded operator

→ req.EQ. MPI\_REQUEST\_NULL

- Conversion in mixed applications:

– Both modules (mpi & mpi\_f08) contain the declarations for all handles.

```

SUBROUTINE a
USE mpi
INTEGER :: splitcomm
CALL MPI_COMM_SPLIT(..., splitcomm)
CALL b(splitcomm)
END

SUBROUTINE b(splitcomm)
USE mpi_f08
INTEGER :: splitcomm
TYPE(MPI_Comm) :: splitcomm_f08
CALL MPI_Send(..., MPI_Comm(splitcomm))
! or
splitcomm_f08%MPI_VAL = splitcomm
CALL MPI_Send(..., splitcomm_f08)
END
    
```

```

SUBROUTINE a
USE mpi_f08
TYPE(MPI_Comm) :: splitcomm
CALL MPI_Comm_split(..., splitcomm)
CALL b(splitcomm)
END


SUBROUTINE b(splitcomm)
USE mpi
TYPE(MPI_Comm) :: splitcomm
INTEGER :: splitcomm_old
CALL MPI_SEND(..., splitcomm%MPI_VAL)
! or
splitcomm_old = splitcomm%MPI_VAL
CALL MPI_SEND(..., splitcomm_old)
END
    
```

## Major changes, continued



- SEQUENCE **and BIND(C)** derived application types can be used as buffers in MPI operations.
- Alignment calculation of basic datatypes:
  - In MPI-2.2, it was undefined in which environment the alignments are taken.
  - There is no sentence in the standard.
  - **It may depend on compilation options!**
  - In MPI-3.0, still undefined, but recommended to use a BIND(C) environment.
  - Implication **(for C and Fortran!)**:
    - If an array of structures (in C/C++) or derived types (in Fortran) should be communicated, it is recommended that
    - the user creates a portable datatype handle and
    - applies additionally MPI\_TYPE\_CREATE\_RESIZED to this datatype handle.

## Other enhancements

- Unused ierror  
INCLUDE 'mpif.h'  
! wrong call:  
CALL MPI\_SEND(...., MPI\_COMM\_WORLD)  
! → terrible implications because ierror=0 is written somewhere to the memory
- With the new module  
USE mpi\_f08  
! Correct call, because ierror is **optional**:   
CALL MPI\_SEND(...., MPI\_COMM\_WORLD)

29



## Other enhancements, continued

- With the `mpi` & `mpi_f08` module:



- Positional and **keyword-based** argument lists

33

- CALL `MPI_SEND(sndbuf, 5, MPI_REAL, right, 33, MPI_COMM_WORLD)`
- CALL `MPI_SEND(buf=sndbuf, count=5, datatype=MPI_REAL, dest=right, tag=33, comm=MPI_COMM_WORLD)`



The keywords are defined in the language bindings.  
Same keywords for both modules.

- Remark: Some keywords are changed since MPI-2.2

33

- For consistency reasons, or
- To prohibit conflicts with Fortran keywords, e.g., `type`, `function`.

## Major enhancement with a full MPI-3.0 implementation


- The following features require Fortran 2003 + TR 29113
  - Subarrays may be passed to nonblocking routines  28
    - This feature is available if the LOGICAL compile-time constant `MPI_SUBARRAYS_SUPPORTED == .TRUE.`
  - Correct handling of buffers passed to nonblocking routines,  37
    - if the application has declared the buffer as `ASYNCHRONOUS` within the scope from which the nonblocking MPI routine and its `MPI_Wait/Test` is called,
    - and the LOGICAL compile-time constant `MPI_ASYNC_PROTECTS_NONBLOCKING == .TRUE.`
  - These features **must** be available in MPI-3.0 if the target compiler is Fortran 2003+TR 29113 compliant.
    - For the `mpi` module and `mpif.h`, it is a question of the quality of the MPI library.

## Minor changes

- MPI\_ALLOC\_MEM, MPI\_WIN\_ALLOCATE, MPI\_WIN\_ALLOCATE\_SHARED and MPI\_WIN\_SHARED\_QUERY return a base\_addr. 35
  - In MPI-2.2, it is declared as INTEGER(KIND=MPI\_ADDRESS\_KIND) and may be usable for non-standard Cray-pointer, see Example 8.2 of the use of MPI\_ALLOC\_MEM
  - In MPI-3.0 in the mpi\_f08 & mpi module, these routines are overloaded with a routine that returns a TYPE(C\_PTR) pointer, see Example 8.1
- The buffer\_addr argumet in MPI\_BUFFER\_DETACH is incorrectly defined and therefore unused. 31
- Callbacks are defined with explicit interfaces PROCEDURE(MPI\_...) BIND(C) 41+42
- A clarification about comm\_copy\_attr\_fn callback, 34  
see MPI\_COMM\_CREATE\_KEYVAL:
  - Returned flag in Fortran must be LOGICAL, i.e., .TRUE. or .FLASE.

## Detailed description of problems, mainly with the old support methods, or if the compiler does not support TR 29113:

37

- 17.1.8 Additional Support for Fortran Register-Memory-Synchronization
- 17.1.10 Problems With Fortran Bindings for MPI
- 17.1.11 Problems Due to Strong Typing
- 17.1.12 Problems Due to Data Copying and Sequence Association with Subscript Triplets
- 17.1.13 Problems Due to Data Copying and Sequence Association with Vector Subscripts
- 17.1.14 Special Constants
- 17.1.15 Fortran Derived Types
- 17.1.16 Optimization Problems, an Overview
- 17.1.17 Problems with Code Movement and Register Optimization
  - Nonblocking Operations
  - One-sided Communication
  - MPI\_BOTTOM and Combining Independent Variables in Datatypes
  - Solutions
  - The Fortran ASYNCHRONOUS Attribute
  - Calling MPI\_F\_SYNC\_REG (new routine, defined in Section 17.1.7) 
  - A User Defined Routine Instead of MPI\_F\_SYNC\_REG
  - Module Variables and COMMON Blocks
  - The (Poorly Performing) Fortran VOLATILE Attribute
  - The Fortran TARGET Attribute
- 17.1.18 Temporary Data Movement and Temporary Memory Modication
- 17.1.19 Permanent Data Movement
- 17.1.20 Comparison with C

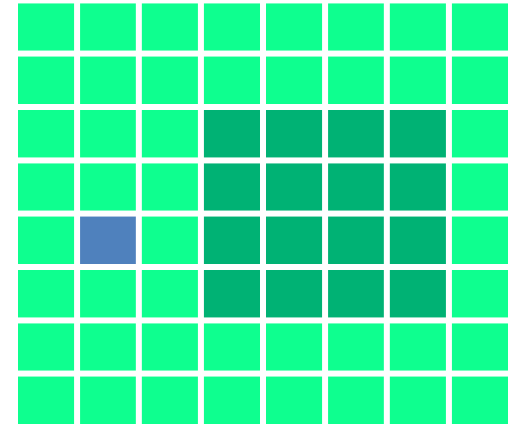
## Status

- Passed final 2<sup>nd</sup> (final) vote at the March 2012 meeting
- Status of reference implementation
  - an initial implementation of the MPI 3.0 Fortran bindings are available in Open MPI
  - a full implementation will not be available until compilers implement new Fortran syntax added specifically to support MPI
    - need ASYNCHRONOUS attribute for nonblocking routines
    - need TYPE(\*), DIMENSION(..) syntax to support subarrays
      - e.g. MPI\_Irecv( Array(3:13:2), ...)

## Group-Collective Communicator Creation

16

- MPI-2: Comm. creation is collective
- MPI-3: New group-collective creation
  - Collective only on members of new comm.
- Avoid unnecessary synchronization
  - Enable asynchronous multi-level parallelism
- Reduce overhead
  - Lower overhead when creating small communicators
- Recover from failures
  - Failed processes in parent communicator can't participate
- Enable compatibility with Global Arrays
  - In the past: GA collectives implemented on top of MPI Send/Recv



Courtesy of Jim Dinan and Richard Graham

## MPI\_TYPE\_CREATE\_HINDEXED\_BLOCK

12

- MPI\_TYPE\_CREATE\_HINDEXED\_BLOCK is identical to MPI\_TYPE\_CREATE\_INDEXED\_BLOCK, except that block displacements in array\_of\_displacements are specified in bytes, rather than in multiples of the oldtype extent:

MPI\_TYPE\_CREATE\_HINDEXED\_BLOCK(count, blocklength, array\_of\_displacements, oldtype, newtype)

IN	count	length of array of displacements (non-negative integer)
IN	blocklength	size of block (non-negative integer)
IN	array_of_displacements	<b>byte</b> displacement of each block (array of integer)
IN	oldtype	old datatype (handle)
OUT	newtype	new datatype (handle)

## Large Counts

- MPI-2.2
  - All counts are int / INTEGER
  - Producing longer messages through derived datatypes may cause problems
- MPI-3.0
  - New type to store long counts:
    - MPI\_Count / INTEGER(KIND=MPI\_COUNT\_KIND)
  - Additional routines to handle “long” derived datatypes:
    - MPI\_Type\_size\_x, MPI\_Type\_get\_extent\_x, MPI\_Type\_get\_true\_extent\_x
  - “long” count information within a status:
    - MPI\_Get\_elements\_x, MPI\_Status\_set\_elements\_x
  - Communication routines are not changed !!!
  - Well-defined overflow-behavior in existing MPI-2.2 query routines:
    - count in MPI\_GET\_COUNT, MPI\_GET\_ELEMENTS, and size in MPI\_PACK\_SIZE and MPI\_TYPE\_SIZE is set to MPI\_UNDEFINED when that argument would overflow.

6

8



## Tool Interfaces for MPI-3

25

- ▶ Goals of the tools working group
  - ▶ Extend tool support in MPI-3 beyond the PMPI interface
  - ▶ Document state of the art for de-facto standard APIs

Courtesy of the MPI-3 Tools working group

## The MPI Performance Interface (MPI\_T)

- Goal: provide tools with access to MPI internal information
  - Access to configuration/control and performance variables
  - MPI implementation agnostic: tools query available information
- Information provided as a set of variables
  - Performance variables (design similar to PAPI counters)  
Query internal state of the MPI library at runtime
  - Configuration/control variables  
List, query, and (if available) set configuration settings

### Examples of Performance Vars.

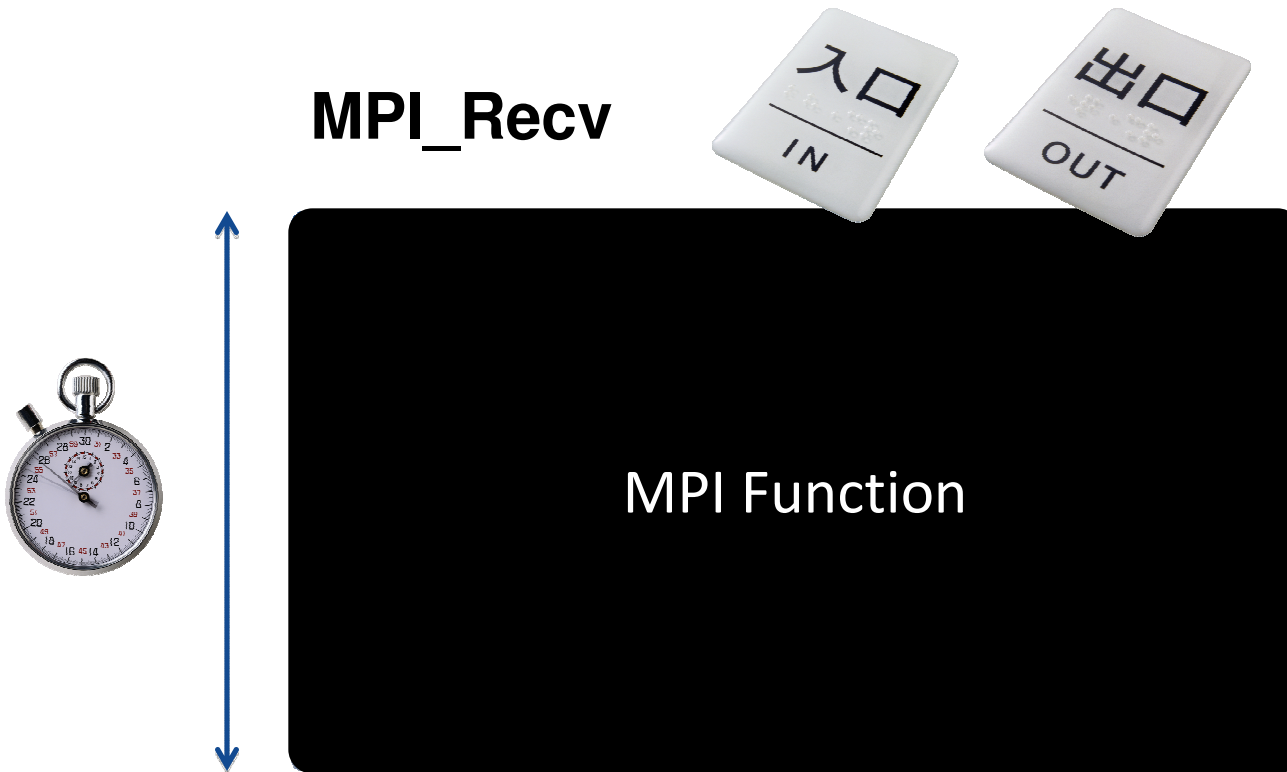
- ▶ Number of packets sent
- ▶ Time spent blocking
- ▶ Memory allocated

### Examples for Control Vars.

- ▶ Parameters like Eager Limit
- ▶ Startup control
- ▶ Buffer sizes and management

- Complimentary to the existing PMPI Interface

## Granularity of PMPI Information

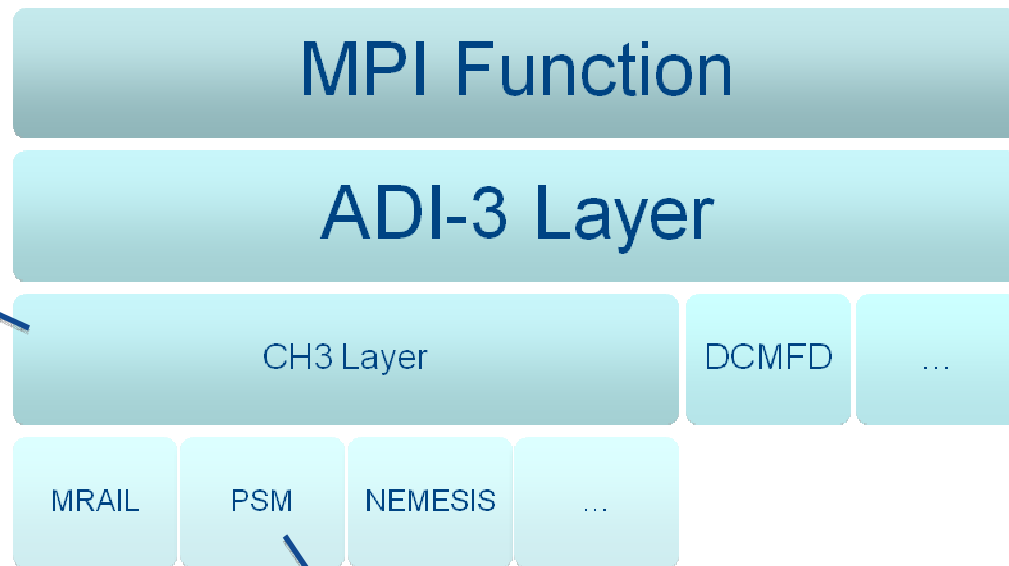


- + Information is the same for all MPI implementations
- MPI implementation is a black box

# Granularity of MPI\_T Information

Example: MVAPICH2

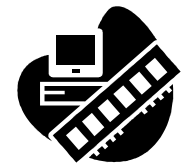
## MPI\_Recv



Polling Counter,  
Queue Length &  
Time, ...



Time in  
Layer

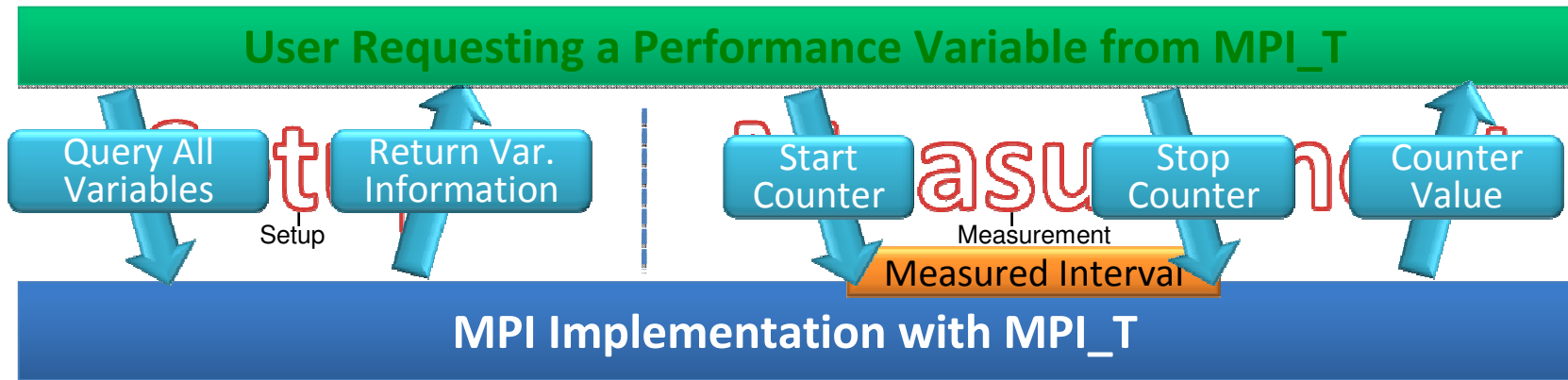


Memory  
Consumption

PSM  
Counter

## Some of MPI\_T's Concepts

- Query API for all MPI\_T variables / 2 phase approach
  - Setup: Query all variables and select from them
  - Measurement: allocate handles and read variables



- Other features and properties
  - Ability to access variables before MPI\_Init and after MPI\_Finalize
  - Optional scoping of variables to individual MPI objects, e.g., communicator
  - Optional categorization of variables

## Status and Next Steps for the Tools WG

- Status of the MPI Tool Information Interface
  - Complete proposal published (MPI ticket #266)
  - Passed final 2<sup>nd</sup> (final) vote at the January 2012 meeting
  - Two prototypes (MVAPICH2 and MPICH2)
  - Next step: integration into tools (in progress)
- Possible next topics for the MPI-3 Tools WG
  - Low-level tracing options in MPI\_T
  - Extended version of MPI\_Pcontrol
  - Piggybacking (in collaboration with the FT group)
  - Companion document to describe the message queue interface
  - Standardization of a more scalable process acquisition API
- Other suggestions/contributions welcome!
  - Documents, Minutes, Discussion on WG Wiki:  
<http://svn.mpi-forum.org/> → MPI 3.0, Tools Workgroup

## Removing C++ bindings from the Standard

2

- Technical aspects of deprecated in MPI 2.2:
  - Supports MPI namespace
  - Support for exception handling
  - Not what most C++ programmers expect
- Decision in MPI-3.0:
  - Remove the C++ support from the standard
  - Use the C bindings – what most C++ developers do today
  - Preserve/add additional MPI predefined datatype handles in C and Fortran to support C++ types that are not provided by C
  - Preserve the MPI:: namespace and names with the meaning as defined in MPI-2.2 + MPI-2.2 errata, see MPI-3.0 Annex B.1.1
  - Perhaps provide the current bindings as a standalone library sitting on top of MPI,
  - or as part of MPI-3.0 libraries.

## Further information

- [www.mpi-forum.org](http://www.mpi-forum.org)
- <https://svn.mpi-forum.org/>
  - View tickets (see headline boxes) → Custom query (right below headline boxes)
    - <https://svn.mpi-forum.org/trac/mpi-forum-web/query>  
→ Filter → Version = MPI-3.0 **or** MPI-2.2-errata
- <http://meetings.mpi-forum.org/>
  - At a glance → All meeting information
    - [http://meetings.mpi-forum.org/Meeting\\_details.php](http://meetings.mpi-forum.org/Meeting_details.php)
  - MPI-3.0 Wiki
    - [http://meetings.mpi-forum.org/MPI\\_3.0\\_main\\_page.php](http://meetings.mpi-forum.org/MPI_3.0_main_page.php)
    - Chapter Working groups:  
[http://meetings.mpi-forum.org/mpi3.0\\_chapter\\_wgs.php](http://meetings.mpi-forum.org/mpi3.0_chapter_wgs.php)

Thank you for your interest