The MPI 3.0 Standard – The Final Stage

Presented by

Rolf Rabenseifner

at the

PRACE Winter School: “Hybrid Programming on Massively Parallel Architectures”

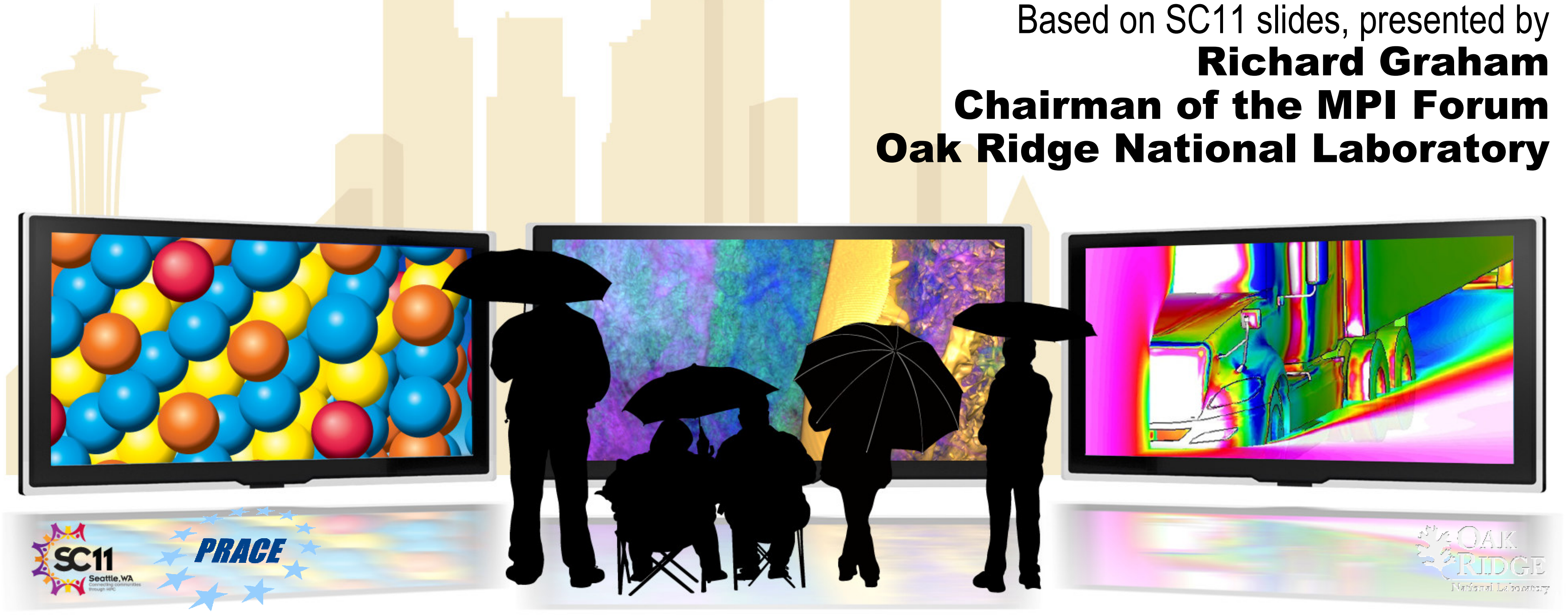
Feb. 6–10, 2011, CINECA, Bologna, Italy

Based on SC11 slides, presented by

Richard Graham

Chairman of the MPI Forum

Oak Ridge National Laboratory



Goal

To produce new versions of the MPI standard that better serves the needs of the parallel computing user community

MPI 3.0 - 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.

Backwards compatibility may be maintained - Routines may be deprecated or deleted

- **Target release date:**
  - **Considering final MPI-3.0 in Sep. 2012**
  - **with incremental draft standard releases (1 to date)**
    - **Posted at http://lists.mpi-forum.org/**
    - **Final version of the standard will be different**

Current Standard: MPI 2.2

2012 Meeting Schedule

Dates	Location	MPI-3.0	MPI-next
Jan 9-11	San Jose, CA		
March 5-7	Chicago, IL	Latest formal reading of tickets, 1st or 2nd vote on tickets that are ready	
May 28-30	Tsukuba, Japan	Latest 1st vote (by institutions) on tickets, Controlling that all finally voted tickets are correctly included into Latex source	Starting with MPI-next
July 16-19	Chicago, IL	Latest 2nd vote (by institutions) on tickets	
Sept 19-21	Vienna, Austria	Chapter votes and finalizing MPI-3.0	
Dec 3-6	Bay Area, CA		

Tracking Forum Activities and Commenting on them

Mailing list: mpi-comments@mpi-forum.org

Subscribe at: http://lists.mpi-forum.org/

One MUST subscribe to the list to post messages to it

All tickets:

https://svn.mpi-forum.org/trac/mpi-forum-web/query

Goal and MPI-3 rule

To produce new versions of the MPI standard that better serves the needs of the parallel computing user community

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.

Backwards compatibility may be maintained - Routines may be deprecated or deleted

Scores: Features

---:	Nonblocking Collectives
14:	RMA
8:	FT stabilization
7:	MPIT
7:	Neighborhood collectives

These features set the clock for MPI-3.0's release.

Other scores:

6:	Non-blocking collective IO
4:	Hybrid
4:	Fortran 08 bindings
2:	Persistence
2:	MPI_Count
0:	Const

And of course, all **needed corrections** to **detected** bugs / ambiguities / inconsistencies

Current Forum Activities

Expected state at begin of March 2012 meeting in Chicago
"!" = Some work to be done

Ticket numbers:	
Red:	Unclear
Orange:	Reading is scheduled
Blue:	Reading is passed
Light green:	1st vote passed
Dark Green:	Passed (i.e., 2nd vote)
Grey:	Not part of MPI-3.0

- **Nonblocking collectives**

→Slide 20
  - **Communication, MPI_Ibarrier, MPI_Ibcast, …** htor #109
  - **Correction to #109: Remove C++ interfaces MPI::Ibarrier, …** jsquyres #272
  - **MPI_Icomm_dup** htor #168

→Slides 43-45
  - **I/O** MPI_File_iread/iwrite_all / _at_all / _ordered **chaarawi+koziol** ! #273
    - **And deprecating** MPI_File_iread/write_all_ / _at_all_ / _ordered_**begin/end**

→Slides 46-47
  - MPI_File_iopen / iclose / isync / iset_view/size/info / ipreallocate **chaarawi** #285
    Quincey → Moved to future MPI-next

- **Scalable sparse collectives on process topologies**

→Slide 21
  - **MPI_(I)Neighbor_allgather(v) / alltoall(v/w):** htor #258
  - **MPI_Aint displs in (I)Neighbor_alltoallw (instead of int)** htor #299

- **Scalable irregular collectives:** htor ! #264

→Slide 22
  - MPI_(I)GATHERDV, MPI_(I)SCATTERDV, MPI_(I)ALLGATHERDV, MPI_(I)ALLTOALLDV, MPI_(I)ALLTOALLDW, MPI_(I)REDUCE_SCATTERDV

Current Forum Activities – Cont'd

Ticket numbers:	
Red:	Unclear
Orange:	Reading is scheduled
Blue:	Reading is passed
Light green:	1st vote passed
Dark Green:	Passed (i.e., 2nd vote)

- **Improvements to one-sided communication support**

→Slides 23-28

– **Main ticket –** 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 **gropp** **#270**
  - 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

– **Corrections to Ticket #270 / RMA:**
  - **"Complex" deleted from allowed types for Compare and Swap** **htor** **! #275**
  - **MPI window attribute types** **htor** **! #283**
  - **Adding clarification to MPI_WIN_LOCK_ALL** **htor** **#298**
  - **Fix issues with definition of nonblocking in One Sided Chapter** **htor** **! #300**
  - **Remove sentence on offset in datatypes** **htor** **#308**
  - **Clarify usage of status for request-based RMA operations** **htor** **#309**

→Slides 36+38

– **Shared memory RMA window** **htor** **(see also slide on "hybrid"** →Slides 12+36 **)** **#284**
  - Routine: MPI_Win_allocate_shared, MPI_Win_shared_query
  - Flavor attribute: MPI_WIN_FLAVOR_SHARED

Current Forum Activities – Cont'd

Ticket numbers:	
Red:	Unclear
Orange:	Reading is scheduled
Blue:	Reading is passed
Light green:	1st vote passed
Dark Green:	Passed (i.e., 2nd vote)

- **Support for MPI library failure recovery (FT)**

→Slide 29

Run-Through Stabilization Process Fault Tolerance Proposal jjhursey ! #276

- MPI_Comm/Win/File_group_failed, MPI_Comm_remote_group_failed, MPI_Failhandler_set/get_mode, MPI_Comm/Win/File_set/get_failhandler, MPI_Comm_reenable_any_source, MPI_Comm_any_source_enabled, MPI_(I)Comm_drain/validate/validate_multiple, MPI_(I)Win/File_validate

– **MPI_Errhandler_compare** jjhursey ! #291 → Josh / Chicago

Current Forum Activities – Cont'd

Ticket numbers:	
Red:	Unclear
Orange:	Reading is scheduled
Blue:	Reading is passed
Light green:	1st vote passed
Dark Green:	Passed (i.e., 2nd vote)

- **New Tool Interface**

→Slides 30-35

  - MPI_T_init_thread / finalize / enum_get_info/item, **schulzm** **#266**
    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_Get_library_version **schulzm** **#204**
  - MPI_Comm_dup/set/get_info, MPI_Win_set/get_info **moody** **! #271**
    - Also must duplicate info object in MPI_COMM_DUP.
  - **MPIR (independent document)** **jsquyres+schulzm** **#228**
    - MPIR Specification - An Official Companion Document for the MPI Standard
      "The MPIR Process Acquisition Interface"

Current Forum Activities – Cont'd

Ticket numbers:	
Red:	Unclear
Orange:	Reading is scheduled
Blue:	Reading is passed
Light green:	1st vote passed
Dark Green:	Passed (i.e., 2nd vote)

- Support for clusters of SMP nodes (hybrid support)
  - →Slide 36 – Thread-safe probe → new probe routine and message object
    - MPI_(I)Mprobe, MPI_(I)Mrecv htor #38
    - MPI_Message_f2c / c2f brbarret #274
  - →Slides 36+37 – Topology aware communicator creation
    - MPI_Comm_split_type with split_type=MPI_COMM_TYPE_SHARED balaji #287
  - →Slides 36+38 – Shared memory RMA window htor (see also slide on "one-sided" →Slide 9) #284
    - Routine: MPI_Win_allocate_shared, MPI_Win_shared_query
    - Flavor attribute: MPI_WIN_FLAVOR_SHARED
  - →Slides 36+39 – Support for multiple "MPI processes" within single "operating system process"
    - Clarification on "processes" and new split_type=MPI_COMM_TYPE_ADDRESS_SPACE ! #310
      → New example for MPI+OpenMP → Partial reading of this example snir
    - →Slides 36+40 MPI_Thread_attach to modify/set MPI rank of a thread ! #311
      → New example for MPI+OpenMP → Partial reading of this example snir
  - →Slides 36+41 – Support for helper threads: MPI_Team_create / free / join / leave / sync / break dougmill ! #217

Current Forum Activities – Cont'd

Ticket numbers:	
Red:	Unclear
Orange:	Reading is scheduled
Blue:	Reading is passed
Light green:	1st vote passed
Dark Green:	Passed (i.e., 2nd vote)

→Slides 48-50 **New Fortran bindings** RolfRabenseifner+jsquyres+rasmussen #229

- **Further Scalability features**
  →Slide 51
  – **Non-collective communicator formation:** MPI_Comm_create_group jdinan ! #286
    - **Tags are in a special tag-space.**
    - **This tag-space is also used in existing MPI_INTERCOMM_CREATE.** jdinan ! #305
- **New Datatype Creation Routine:** MPI_Type_create_hindexed_block chaarawi #280
- **Large counts:** type MPI_Count / INTEGER(KIND=MPI_COUNT_KIND), datatype MPI_COUNT ftillier #265
  →Slide 52 MPI_Type_: size_x / get_extent_x / get_true_extent_x, MPI_Get_elements_x, MPI_Status_set_elements_x
- **Reference counted MPI_INIT / INIT_THREAD / FINALIZE** (no new routines) jsquyres ! #302
  → may go to MPI-next
  → Marc Snir makes new ticket for Chicago about MPI_Init ! #313

Current Forum Activities – Cont'd

Ticket numbers:	
Red:	Unclear
Orange:	Reading is scheduled
Blue:	Reading is passed
Light green:	1st vote passed
Dark Green:	Passed (i.e., 2nd vote)

- **Removal of deprecated / optional C++**

Opt. C++

→Slide 53

- **Make C++ bindings optional** ftillier **#279**
  - The C++ language bindings have been deprecated. <u>A compliant MPI implementation providing C++ language bindings must provide the entire set defined in this document.</u>

Text w/o old references

- **Removing deprecated functions from the examples** dries ! #12 (to do in 278)
- **MPI_TYPE_GET_EXTENT is defined via deprecated features** bosilca ! #102 (done in 278)
- **Remove deprecated functions (basically all functions deprecated in 2.0)** ! **#278**
  - This ticket keeps Chapter 15 Deprecated Functions, but removes most links to it ftillier
  - Has to check #12 proposals, update LB/UB, etc.
  - Recommendation, use #278.1 as marker for re-reading
  - Partial Re-reading is needed, especially for pages 97, 112-115, 119-122, 492-494

Real Removal

→Slide 53

- **Remove C++ Bindings (basically moving it to a Removed Chapter)** dougmill ! **#281**
  - All should be removed, except some remarks in the Removed-Chapter → Re-reading
- **Move MPI-1 deprecated functions to new "Removed Interfaces" chapter** ! **#303**
  - Removes: MPI_ADDRESS, MPI_ERRHANDLER_CREATE / GET / SET, ftillier
    MPI_TYPE_EXTENT / HINDEXED / HVECTOR / STRUCT / LB / UB
  - **Fab considers to withdraw this ticket**

Reasons:
- Fortran applications need changes in the data declarations (→ 64bit INTEGER) and mpif.h and application-subroutines may still not use argument checking → high risks
- Fortran rule for "deprecated" means that features should not be used in new applications, but they are still available for existing applications the next ~20 years
  Example: Redundant features in F77, removed in F95: ASSIGN, H, PAUSE, …

Current Forum Activities – Cont'd

Ticket numbers:	
Red:	Unclear
Orange:	Reading is scheduled
Blue:	Reading is passed
Light green:	1st vote passed
Dark Green:	Passed (i.e., 2nd vote)

- **Clarifications (probable #0.5 level?)**
  - **#158 Clarification that MPI_Cart_map and MPI_Graph_map are local calls traff Trivial text changes → Rolf, Chicago (reading based on existing ticket, no pdf)**
  - **#187 For reductions: Multi-language types hubertritzdorf Correction to standard**
    - New wording "Multi-language types" for datatypes MPI_AINT, MPI_OFFSET, MPI_COUNT in reductions
    - <u>All predefined named and unnamed datatypes as listed in Section 5.9.2 on page 164 can be used in all predefined operations independent of the programming language which the MPI routine is called from.</u> → **Rolf, Chicago (reading based on existing ticket, no pdf)**
  - **! #227 + #313 Clarify ambiguous sentence for MPI_FINALIZE jjhursey Text (only) changes**
    - MPI-2.2, Chap. 8.7, MPI_Finalize, page 291, lines 36-37 reads
      - » Each process must call MPI_FINALIZE before **it** exits.

      but should read

      Unclear meaning of "it"

      - » Before each process exits, the process must call MPI_FINALIZE. → **Marc Snir, Chicago**
  - **#256 MPI_PROC_NULL behavior for MPI_PROBE and MPI_IPROBE not defined rlgraham Enhancements to standard** (Only clarification, already indirectly defined via MPI_Recv and MProbe) → **Rolf Rabenseifner + Jeff Squyres, Chicago**
  - **#259 Text updates for MPI_DIST_GRAPH moody20 Text (only) changes**
    - Significantly better re-wording (longer ticket) → **Adam Moody, Chicago**

      (Written or reviewed by Adam Moody, Torsten Hoefler, Rolf Rabenseifner)

Current Forum Activities – Cont'd

Ticket numbers:	
Red:	Unclear
Orange:	Reading is scheduled
Blue:	Reading is passed
Light green:	1st vote passed
Dark Green:	Passed (i.e., 2nd vote)

- **Further tickets → All for Chicago**
  - ! **#159 Fortran extra_state parameter in attribute callback functions** Correction → **Jeff Sqyres**
  - ! **#163 MPI_IN_PLACE in Gather** Enhancements to standard → **Torsten Hoefler**
  - **#194 Ensure MPI_Dims_create is really suitable for MPI_Cart_create** **gropp** → **I ask Bill**
  - ! **#195 Topology awareness in MPI_Dims_create** **balaji** New routines → **Pavan Balaji**
  - ! **#222 Requiring same level of thread support on all MPI processes** **gropp** Correct. → **Pavan B.**
  - ! **#293 K&R style on 6.7.6** **jhammond** Text (only) changes → **Fab Tillier**
  - ! **#304 Fix the attribute disaster** **ftillier** Text (only) changes → **Fab Tillier**
  - **#221 Undeprecate the C++ Bindings in MPI 3.x** **tony** → **Withdrawn**
- **Add MPI_Timer requests** **ftillier** New routines → **future MPI-next** **! #277**
  - MPI_Timer_create, MPI_Timer_reset (convenience fuction for cancel+create)
- **Glossary** **Terry Jones (trj)** Text (only) changes → **future MPI-next** **! #78**
  - Text is only for illustration purpose, i.e., does not change / add to the standard
  - Same rules as for examples, helpful for new readers of the MPI standard
  - Text should be taken mainly from the standard → wording can/should not be wrong
  - Therefore only level 0.5?

Current Forum Activities – Cont'd

Ticket numbers:	
Red:	Unclear
Orange:	Reading is scheduled
Blue:	Reading is passed
Light green:	1st vote passed
Dark Green:	Passed (i.e., 2nd vote)

- **Other Corrections (also #0.5 level?)**
  - **#140 Add const Keyword to the C bindings ftillier Enhancements to standard**
  - **! #294 MPI_UNWEIGHTED should not be NULL goodell Correction to standard** → was read Oct. 2011 → 1st vote in Chicago → text changed → Reading in Chicago
  - **#162 MPI 2.1 Clarification - MPI_Cart_map with num_dims=0 smithbr Text (only) changes**
    - Was overseen in MPI-2.2 when correcting all zero-dimensional Cartesian cases → Rolf
  - **#182 s/legal/something_better/g gropp Text (only):** All prepared by Bill Gropp (→**Bronis**)
  - **#219 MPI_MAX_OBJECT_NAME and MPI_Type/win_get_name dries Text (only) changes**
    - Currently undefined, i.e., needs to be defined! → **Rolf Rabenseifner**
  - **! #125 Use of [] in output arrays gropp Trivial text changes:** Needs work by some chapter authors → **Fab Tillier**
  - **! #126 Consistent use of [] for input arrays gropp Enhance.:** Needs work by some chapter authors → **Fab Tillier**
  - **! #218 "same arguments" to MPI_Reduce goodell Text (only) changes → Dave Goodell**
    - Correction to an overseen inconsistency when MPI_IN_PLACE was added to MPI_Reduce

Current Forum Activities – Cont'd

Ticket numbers:	
Red:	Unclear
Orange:	Reading is scheduled
Blue:	Reading is passed
Light green:	1st vote passed
Dark Green:	Passed (i.e., 2nd vote)

- Small corrections which should/must not be overseen
  - Trivial (level #0 changes)
    - (#0) Trivial changes read+voted in one step (mainly proposed by chapter authors)
    - #185 Mention C++ typedef deprecations RolfRabenseifner Text (only) changes
    - #207 Minor Text Correction / MPI_GET_PROCESSOR_NAME schulzm Trivial text changes
    - #215 MPI-2.2 typo - missing parenthesis on page 272, line 25 bosilca Trivial text changes
    - #254 Typo in the description of MPI_GROUP_INCL rlgraham Trivial text changes
    - #255 Missing headline for MPI_GET_PROCESSOR_NAME bosilca Trivial text changes
    - #262 Minor typos in pt2pt chapter rlgraham Trivial text changes
    - #263 wrong formula for the upper bound bosilca Text (only) changes
    - #267 Typo in One-Sided chapter 11.4.4 jjhursey Trivial text changes
    - #268 Formatting issue in One-Sided chapter 11.4.4 jjhursey Trivial text changes
    - #312 Typo-Correction in ALLTOALLV RolfRabenseifner Trivial text changes

Details about most & important topics

- Slides 20-23: Nonblocking, sparse and scalable irregular collectives
- Slides 23-28: One-sided communication – enhancements
- Slide 29: Fault-tolerance
- Slides 30-35: New tools interface
- Slides 36-42: Hybrid Programming
- Slides 43-47: Nonblocking collective I/O
- Slides 48-50: Fortran interface
- Slide 51: Group-Collective Communicator Creation
- Slide 52: Large Counts
- Slide 53: Removing C++ bindings from the Standard

Slides – courtesy of the ticket owners / committees

Nonblocking Collective Operations

- 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 draft standard
  - Reference Implementation (LibNBC) stable
  - Several implementations pending

Sparse Collective Operations on Process Topologies

- MPI process topologies (Cartesian and (distributed) graph) usable for communication
  - MPI_Sparse_gather(v)
  - MPI_Sparse_alltoall(v,w)
  - Also nonblocking variants
- Allow for optimized communication scheduling and scalable resource binding
- New feature to enhance scalability and performance of MPI
  - Accepted for MPI-3 in Sept. 2011
  - Reference implementation exists

Scalable Irregular Collectives

- Distribute argument lists of vector collectives
  - Simple interface extension
  - Lower overhead at scale
  - Reduce memory overhead from O(P) to O(1)

- Proposal forthcoming
  - "distributed vector collectives"
    - Scatterdv(), Gatherdv() … including Alltoalldv()
  - Reference implementation on the way

MPI One-Sided Communication Interface

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

- **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_SEPARATE and MPI_WIN_UNIFIED**

- **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

- Passed first vote at the July meeting
- Example implementation on top of Portals-4 available
  - thanks to Brian Barrett of Sandia National Labs
- Other implementations in progress
- Second (and final) vote planned for the next Forum meeting in January

Fault Tolerance Working Group

Define a set of semantics and interfaces to enable fault tolerant applications and libraries to be portably constructed on top of MPI.

- **Application involved fault tolerance** (not transparent FT)
  - Natural & Algorithm Based Fault Tolerance (ABFT)
- **Fail-stop process failure:**
  - MPI process permanently stops communicating with other processes.
- **Two Complementary Proposals:**
  - **Run-Through Stabilization:** (*Target: MPI-3.0*)
    - Continue running and using MPI even if one or more MPI processes fail
  - **Process Recovery:** (*Target: MPI-3.1*)
    - Replace MPI processes in existing communicators, windows, file handles
- **Prototype in Open MPI available to interested applications (MPICH2 in progress)**

MPI Forum Fault Tolerance Working Group:
https://svn.mpi-forum.org/trac/mpi-forum-web/wiki/FaultToleranceWikiPage

Tool Interfaces for MPI-3

- 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

The MPI Performance Interface (MPI_T)

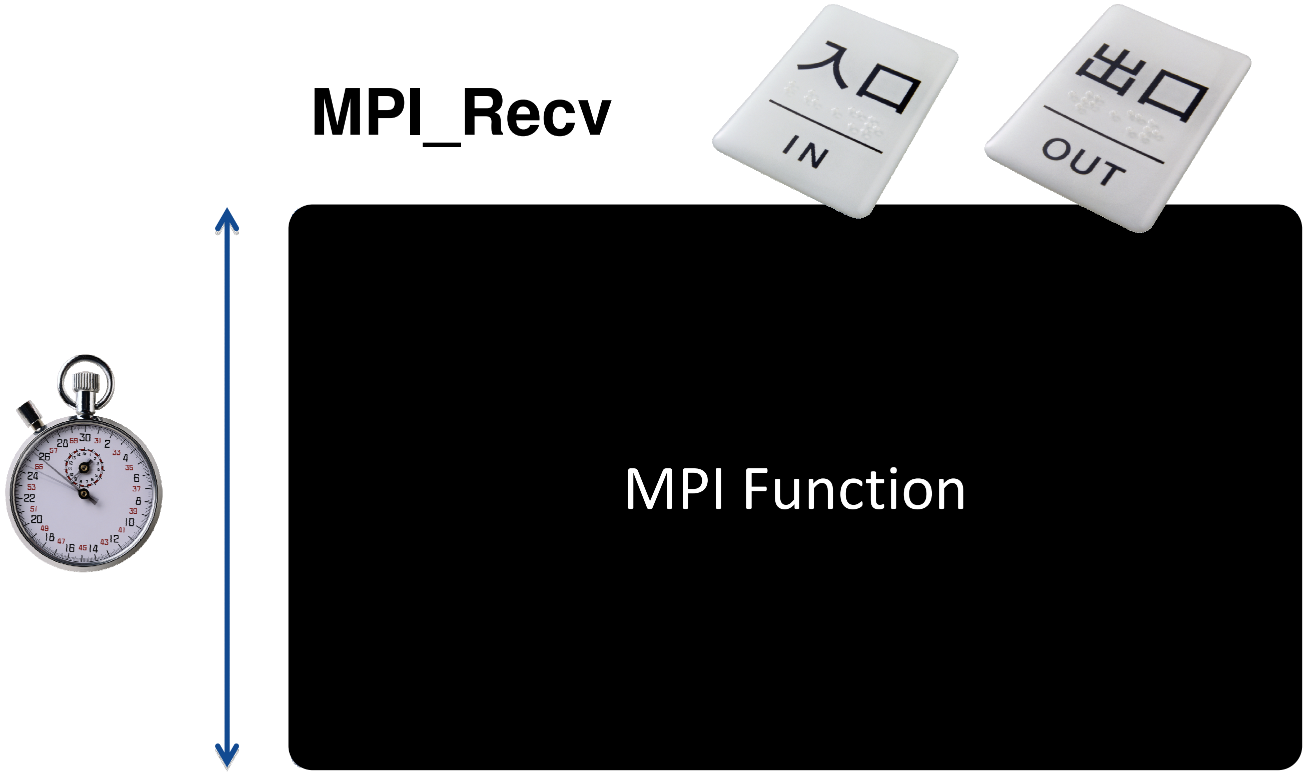
- **Goal: provide tools with access to MPI internal information**
  - **Access to configuration/control and performance variables**
  - **MPI implementation agnostic: tools query available information**
- **Information provided as a set of variables**
  - **Performance variables (design similar to PAPI 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**

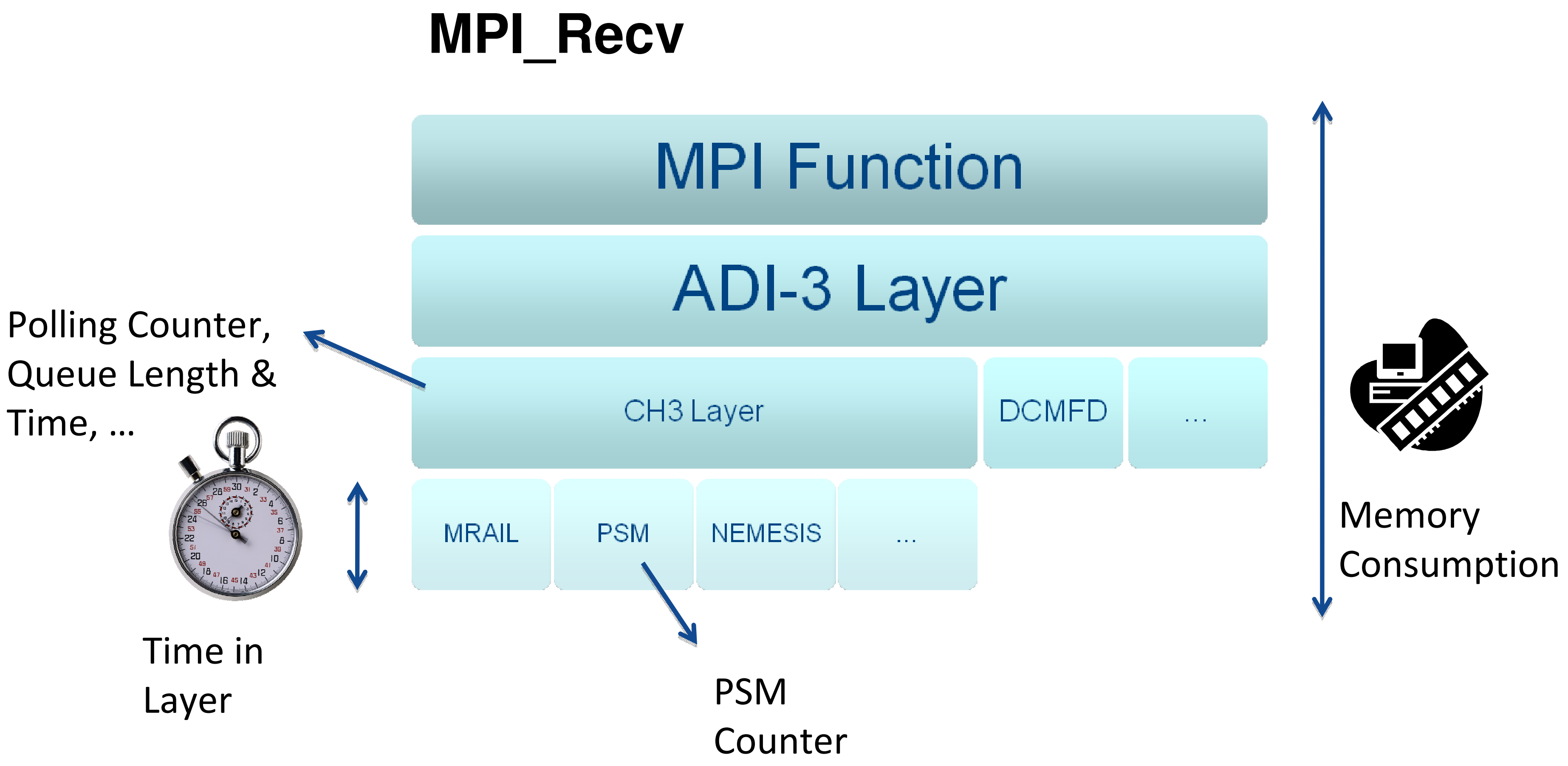
Granularity of PMPI Information



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

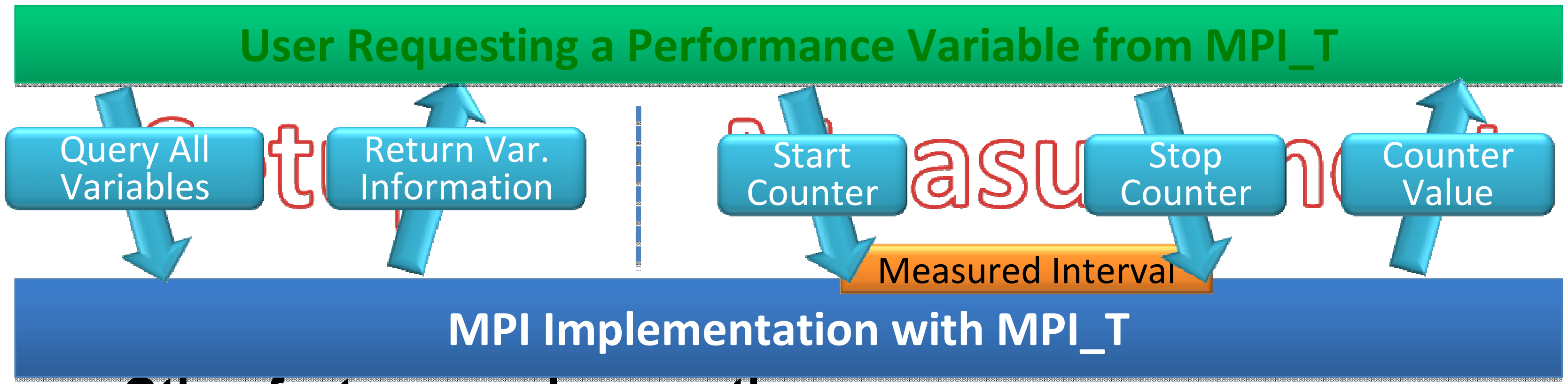
Granularity of MPI_T Information

Example: MVAPICH2



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 first vote in October 2011
  - 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*

MPI-3 Hybrid Programming: In a Nutshell

- Three broad proposals in the works:
  - MPI Shared memory interoperability
    - Functions to create and manipulate shared memory between a few MPI processes (e.g., those that reside on the same node or same NUMA domain)
  - MPI Endpoints
    - Ability to have many "MPI processes" (implemented as threads) within one "OS process"
    - Some of these MPI processes form a "clique" and can communicate with each other using thread communication semantics or using MPI
  - MPI Helper threads
    - Allow applications to lend threads to MPI for parallelizing internal MPI processing
- Thread-safe probe: MPI_Mprobe / MPI_Improbe & MPI_Mrecv / MPI_Imrecv

Shared Memory Extensions to MPI

- Step 1: Topology-aware communicator creation
  - Allows you to create a communicator whose processes can create a shared memory region
  - More generally: it splits a communicator into subcommunicators of a certain *type*
    - "shared memory capability" is one type
    - Other implementation specific types are possible: rack, switch, etc.
  - MPI_Comm_split_type(comm, comm_type, key, info, new_comm)

Shared Memory Extensions to MPI – Cont'd

- Step 2: Creation and manipulation of shared memory (based on MPI RMA semantics)
- MPI_WIN_ALLOCATE_SHARED(size, info, 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()

MPI Endpoints: Assumptions

- Idea is to have multiple "MPI processes" within one "OS process"
  - Number of threads per node will increase dramatically
  - Nodes will have multiple NICs
  - Intranode communication will use shared memory (MPI+X)
    - May not be coherent and will be very NUMA
- Multiple MPI processes within one address space gives more flexibility
  - Improve message rate (avoid long queues and conflicts) and deal with NUMA or non-coherent memory
  - Facilitate resource sharing (TLBs, cores, NICs)
  - Provide proper MPI interface to OpenMP, TBB, PGAS, etc.
- Note: MPI never required that MPI processes run in distinct address spaces
  - MPI1 explicitly mentions possibility of multiple MPI processes within one address space – we are just reviving the same concept

MPI Endpoints: Proposal

- Make sure that MPI is properly defined when MPI processes can share memory
  - No MPI issue found, so far
- Applications which were assuming MPI process == OS process might need some corrections if they want to use threads
  - It is unlikely MPI implementations will drop support for the “older MPI process == OS process model” anytime soon
- Allow for “migration” of a thread from one MPI process to another, whenever possible

MPI_THREAD_ATTACH(rank,comm)

  - Thread detaches from its current MPI process and attaches to the MPI process identified by <rank,comm>
- This design enables MPI to work with OpenMP, TBB or PGAS

MPI_Team – a.k.a. Helper Threads

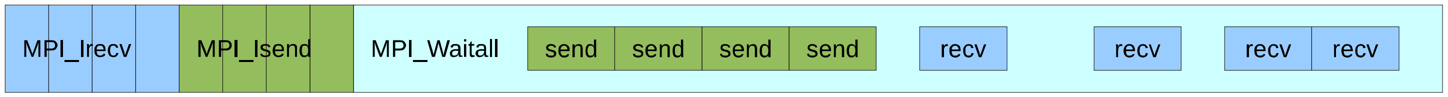
- Multi-threading has become a necessity in order to fully utilize hardware.
- Over-subscribing hardware threads negatively impacts performance in HPC.
- Therefore, there is a need to increase cooperation and coordination between user threads and an MPI implementation which may need threads.

The Proposal:

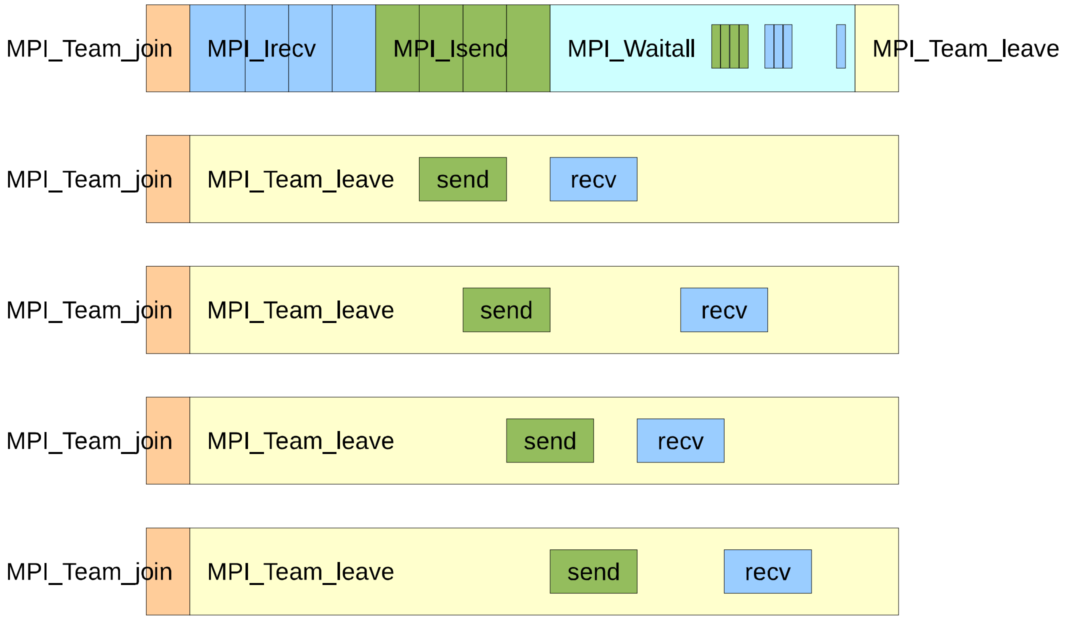
- Allows a user thread to lend itself to MPI when it is otherwise idle.
- Similar to how OpenMP work sharing constructs allow assigning tasks to threads.
- MPI_Team_join begins an MPI parallel region.
- Analogous to "#pragma omp xxx {"
- MPI_Team_leave ends an MPI parallel region and synchronizes threads.
- Analogous to matching "}" of pragma.
- MPI calls between join and leave may divide work between team threads.

MPI_Team Example

Typical single-threaded (non-blocking) send/recv



Multi-threaded send/recv using MPI_Team



MPI-I/O

Nonblocking Collective I/O Routines

- Current routines are in the form of split collectives:
  - MPI_FILE_XXX_ALL_BEGIN
  - MPI_FILE_XXX_ALL_END
- Main restriction:
  - Each file handle may have at most one active split collective operation at any time.

Nonblocking Collective I/O Routines

- In MPI-3.0 we propose to:
  - Add immediate versions of the split collective operations
  - Deprecate the split collective routines
- The following routines will be added:
  - MPI_File_iread_all (MPI_File file, void *buf, int count, MPI_Datatype type, MPI_Request *req);
  - MPI_File_iwrite_all (MPI_File file, void *buf, int count, MPI_Datatype type, MPI_Request *req);
  - MPI_File_iread_at_all (MPI_File file, MPI_Offset offset, void *buf, int count, MPI_Datatype type, MPI_Request *req);
  - MPI_File_iwrite_at_all (MPI_File file, MPI_Offset offset, void *buf, int count, MPI_Datatype type, MPI_Request *req);
  - MPI_File_iread_ordered (MPI_File file, void *buf, int count, MPI_Datatype type, MPI_Request *req);
  - MPI_File_iwrite_ordered (MPI_File file, void *buf, int count, MPI_Datatype type, MPI_Request *req);

Asynchronous File Manipulation Routines

- In MPI-3.x we propose to add asynchronous file manipulation routines to the standard.
- All file manipulation routines would have an asynchronous version.
- The intent is to hide the I/O step and to achieve maximum overlap of computation/communication and I/O.
- The semantics of those operations are still being worked out.
- This is a proposal in the works, and the MPI Forum has not approved it yet.

Asynchronous File Manipulation Routines

- A preliminary set of routines are listed:
  - MPI_FILE_IOPEN
  - MPI_FILE_ISET_VIEW
  - MPI_File_ISYNC
  - MPI_FILE_ISET_SIZE
  - MPI_FILE_IPREALLOCATE
  - MPI_FILE_ISET_INFO
  - MPI_FILE_ICLOSE
- The current consistency semantics and nonblocking behavior in MPI do not apply to those routines.
  - the name will probably change
  - the request handle might not be applicable here since MPI would be recording dependencies between the function calls internally (i.e. queuing).

MPI 3.0 Fortran Bindings:

a high-level summary for non-Fortran programmers

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

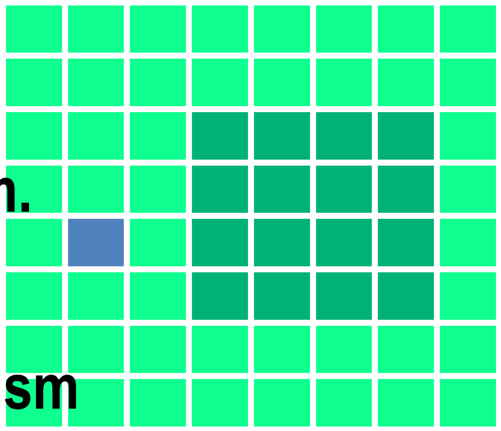
- Requirements
  - comply with Fortran standard (for the first time)
  - enhance type safety
  - suppress argument checking for choice buffers
  - guarantee of correct asynchronous operations
  - for user convenience
    - provide users with convenient migration path
    - allow some optional arguments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

Status: MPI 3.0 Fortran binding specification is complete and a reference implementation exists

- Status of specification
  - been through several "final" revisions and is ready for a first reading
  - document available for outside review
- 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

- MPI-2: Comm. creation is collective
- MPI-3: New group-collective creation
  - Collective only on members of new comm.



1. Avoid unnecessary synchronization
   - Enable asynchronous multi-level parallelism
2. Reduce overhead
   - Lower overhead when creating small communicators
3. Recover from failures
   - Failed processes in parent communicator can't participate
4. Enable compatibility with Global Arrays
   - In the past: GA collectives implemented on top of MPI Send/Recv

Large Counts

- **MPI-2.2**
  - **All counts are int / INTEGER**
  - **Producing longer messages through derived datatypes may cause problems**
- **MPI-3.0**
  - **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
  - **MPI_Count / INTEGER (KIND=MPICOUNT_KIND)**
    - **New type to store long counts**
  - **Communication routines are not changed !!!**

Removing C++ bindings from the Standard

- Current state
  - Deprecated in MPI 2.2
  - Technical aspects
    - Supports MPI namespace
    - Support for exception handling
    - Not what most C++ programmers expect
- Proposal
  - Remove the C++ support from the standard
  - Use the C bindings – what most C++ developers do today
  - Perhaps provide the current bindings as a standalone library sitting on top of MPI

Further information

- 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
- http://meetings.mpi-forum.org/
  - At a glance → All meeting information
    - http://meetings.mpi-forum.org/Meeting_details.php
  - MPI-3.0 Wiki
    - http://meetings.mpi-forum.org/MPI_3.0_main_page.php
    - Chapter Working groups: http://meetings.mpi-forum.org/mpi3.0_chapter_wgs.php

Thank you for your interest