# MPI-2 Overview

Rolf Rabenseifner
**rabenseifner@hlrs.de**

University of Stuttgart
High-Performance Computing-Center Stuttgart (HLRS)
www.hlrs.de



## MPI-2 Outlook

- MPI-2, standard since July 18, 1997
- Chapters:
  - Version 1.2 of MPI (Version number, Clarifications)
  - Miscellany (Info Object, Language Interoperability, New Datatype Constructors, Canonical Pack & Unpack, C macros)
  - Process Creation and Management (MPI_Spawn, ...)
  - One-Sided Communications
  - Extended Collective Operations
  - External interfaces (..., MPI and Threads, ...)
  - I/O
  - Language Binding (C++, Fortran 90)
- All documents from http://www.mpi-forum.org/
  ( or from www.hlrs.de/mpi/ )

## Dynamic Process Management

- Goals
  - starting new MPI processes
  - connecting independently started MPI processes
- Issues
  - maintaining simplicity, flexibility, and correctness
  - interaction with operating systems, resource manager, and process manager
- Spawn interfaces:
  - at initiators (parents):
    - **Spawning new processes is *collective*, returning an intercommunicator.**
    - **Local group is group of spawning processes.**
    - **Remote group is group of spawned processes.**
  - at spawned processes (children):
    - **New processes have own MPI_COMM_WORLD**
    - **MPI_Comm_get_parent() returns intercommunicator to parent processes**



## One-Sided Operations

- Goals
  - PUT and GET data to/from memory of other processes
- Issues
  - Synchronization is separate from data movement
  - Automatically dealing with subtle memory behavior: cache coherence, sequential consistency
  - balancing efficiency and portability across a wide class of architectures
    - **shared-memory multiprocessor (SMP)**
    - **clusters of SMP nodes**
    - **NUMA architecture**
    - **distributed-memory MPP's**
    - **workstation networks**
- Interface
  - PUTs and GETs are surrounded by special synchronization calls

## Extended Collective Operations

- In MPI-1, collective operations are restricted to ordinary (intra) communicators.
- In MPI-2, most collective operations are extended by an additional functionality for intercommunicators
  - e.g., Bcast on a parents-children intercommunicator: sends data from one parent process to all children.
- Provision to specify "*in place*" buffers for collective operations on intracommunicators.
- Two new collective routines:
  - generalized all-to-all
  - exclusive scan



## MPI - I/O

- Goals:
  - reading and writing files in parallel
- Rich set of features:
  - Basic operations: open, close, read, write, seek
  - noncontiguous access in both memory and file
  - logical view via *filetype* and *element-type*
  - physical view addressed by hints, e.g. "striping_unit"
  - explicit offsets / individual file pointers / shared file pointer
  - collective / non-collective
  - blocking / non-blocking or split collective
  - non-atomic / atomic / explicit sync
  - "native" / "internal" / "external32" data representation

## Other MPI-2 Features (1)

- Standardized Process Startup: **mpiexec**
- C / C++ / Fortran language interoperability
- Datatypes:
  - New constructors:
    - **MPI_Type_create_darray / ..._subarray / ..._indexed_block**
  - new routines due to incorrect Fortran binding in MPI-1:
    - **INTEGER (KIND=MPI_ADDRESS_KIND) ... in MPI-2**
  - new predefined datatypes:
    - **MPI_WCHAR, MPI_SIGNED_CHAR, MPI_UNSIGNED_LONG_LONG**
- Null values:
  - MPI_Init(NULL,NULL)
  - MPI_STATUS(ES)_IGNORE instead of (&)status



## Other MPI-2 Features (2)

- C/C++/Fortran Language interoperability support
  - between languages in same processes
  - messages transferred from one language to another
- (P)MPI_Wtime and ..._Wtick may be implemented as macros in C

## Process Startup

- Current MPI implementations provide the “mpirun” as a startup command which is not standard and not portable
- MPI-2 specifies an “mpiexec” startup command (recommended)

```
mpiexec  {-n <maxprocs>  -soft <      >
          -host <        >  -arch <     >
          -wdir <        >  -path <     >
          -file <        >  …….  Command
         } : { …… } : { …… }
```



## C++ Language Bindings

- C++ bindings match the new C bindings
- MPI objects are C++ objects
- MPI functions are methods of C++ classes
- User must use MPI create and free functions instead of default constructors and destructors
- Uses shallow copy semantics
  (except MPI::Status objects)
- C++ exceptions used instead of returning error code
- declared within an MPI namespace (**`MPI::...`**)
- C++/C mixed-language interoperability

## Fortran 90 Support, I.

- MPI-2 Fortran bindings are Fortran 90 bindings that are “Fortran 77 friendly” (most cases)
- Fortran 90 and MPI have several incompatibilities
  - strong typing vs. choice arguments
  - data copying vs. sequence association
  - special values vs. special constants
  - …..
- MPI-2 standard documents the “do’s” and “don’ts” while using Fortran 90 / 77 features
  - *Chap. 10.2.2* “Problems with Fortran Bindings for MPI”



## Problems Due to Data Copying and Sequence Association

- Example 1:

```
real a(100)
call MPI_Irecv( a(1:100:2), MPI_REAL, 50, ...)
```

  - First dummy argument of MPI_Irecv is an assumed-size array ( `<type> buf(*)` )
  - a(1:100:2) is copied into a scratch array, which is freed after the end of MPI_Irecv
  - Afterwards, until MPI_Wait, there is no chance to store the results into a(1:100:2)

## Problems Due to Data Copying and Sequence Association

- Example 2:

```
real a
call user1(a,rq)
call MPI_Wait(rq,status,ierr)
write (*,*) a
subroutine user1(buf,request)
call MPI_Irecv(buf, ..., request, ...)
end
```

  - the compiler has to guarantee, that it makes no copy of the scalar a
    - **neither in the calling procedure**
    - **nor in the called procedure**
  - check for compiler flags!
  - guarantee is necessary for
    - **MPI_Get_address**
    - **all non-blocking MPI routines**



## Fortran Problems with Register Optimization and 1-sided

| Source of Process 1 | Source of Process 2 | Executed in Process 2 |
|---|---|---|
| bbbb = 777 | buff = 999 | register_A := 999 |
| call MPI_WIN_FENCE | call MPI_WIN_FENCE | |
| call MPI_PUT(bbbb into buff of process 2) | | stop application thread<br>buff := 777 in PUT handler<br>continue application thread |
| call MPI_WIN_FENCE | call MPI_WIN_FENCE | |
| | ccc = buff | ccc := register_A |

- Fortran register optimization
- Result ccc=999, but expected ccc=777
- How to avoid: (see MPI-2, Chap. 6.7.3)
  - window memory declared in COMMON blocks
    i.e. MPI_ALLOC_MEM cannot be used
  - declare window memory as VOLATILE
    (non-standard, disables compiler optimization)
  - Calling MPI_Address(buff, idummy_addr, ierror) after 2nd FENCE in process 2

## Fortran 90 Support, II.

- Different support levels:
  - Basic Fortran support
    - **mpif.h is valid with free-form and fixed-form**
  - Extended Fortran support
    - **an `mpi` module: `USE mpi` instead of `include mpif.h`**
    - **datatype generation routines for KIND-parameterized Fortran types:**
      - `MPI_TYPE_CREATE_F90_INTEGER`
      - `MPI_TYPE_CREATE_F90_REAL`
      - `MPI_TYPE_CREATE_F90_COMPLEX`
    - **alternative concept [*not appropriate for heterogeneous platforms*]:**
      - `MPI_SIZEOF`, `MPI_TYPE_MATCH_SIZE`



## Summary of MPI-2

- MPI-2 standard document available since July 1997
- Provides extensions to MPI-1, does not replace MPI-1
- Provides a wide variety of functionality, some still untested
- Implementation of parts of the MPI-2 standard are available
- Nearly full implementations were available by mid-2000 on
  - Fujitsu
  - NEC
  - Hitachi
- Subset examples:
  - Cray: mpt.1.4.0.1 (MPI-I/O [romio], one-sided)
  - MPICH
  - LAM: with dynamic process start
- List of MPI-2 implementations and subsets: www.mpi.nd.edu/MPI2/

## Other MPI Activities

- Metacomputing
  - MPI-PACX, MetaMPI, HARNESS, Globus, StaMPI, ...
  - IMPI (Interoperable Massage Passing Interface)
- Real-time MPI
- Additional MPI Language Bindings
  - Java
  - Ada-95
  - COM
- The DARPA Data Reorganization Standard
- MPI on PC clusters

(see www.hlrs.de/mpi/ )