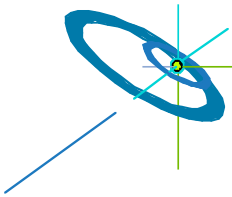
MPI 2.1
at
MPI Forum
Chicago, March 10-12, 2008
Ballot 3

Rolf Rabenseifner
rabenseifner@hlrs.de
(Chairman of MPI 2.1 Task)

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



1st official vote:

Do you accept the MPI-2.1 Ballot 3 Items 1-4, 7-14 ?

Number of insti-tutions:

24

Yes:

23

No:

0

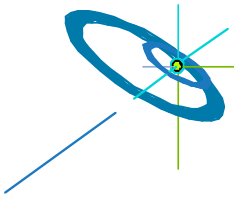
Abstain:

1

First official vote: MPI-21. Ballot 3, Items 1-4 and 7-14

- 1. MPI_COMM_PARENT instead of MPI_COMM_GET_PARENT
- 2. MPI_UNPACK_EXTERNAL
- 3. Additional C++ binding errors
- 4. MPI_REQUEST_CANCEL used where MPI_CANCEL intended
- (5. Intercommunicator collective and datatypes
  → moved to Ballot 4 → no further proposal → moved to MPI 2.2)
- (6. const in C++ specification of predefined MPI objects
  → moved to Ballot 4)
- 7. Error in MPI_Scan Example
- 8. Missing newline in Fortran binding
- 9. Misspelled argument in Fortran binding
- 10. Error in MPI-1, Example 3.12
- 11. Error in MPI-1, Example 3.34
- 12. Change MPI-2, page 343, lines 22-23
- 13. MPI 1.1, strlen in first pt-to-pt example
- 14. Formatting error on MPI 1.1, page 58

Following slides:
Results and votes from
January 2008
Meeting



Ballot 3 –
1. MPI_COMM_PARENT instead of MPI_COMM_GET_PARENT

Question:

Do you accept this entry?

Yes:

All-2

No:

0

Abstain:

2

Mail discussion, proposed by Bill Gropp and Rusty Lusk, Mar 18, 2004

http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/commparent/

MPI-2, page 179, lines 4-5 change

> **Thus, the names of MPI_COMM_WORLD, MPI_COMM_SELF, and MPI_COMM_PARENT will have the default of MPI_COMM_WORLD, MPI_COMM_SELF, and MPI_COMM_PARENT.**

to

> **Thus, the names of MPI_COMM_WORLD, MPI_COMM_SELF, and the communicator returned by MPI_COMM_GET_PARENT (if not MPI_COMM_NULL) will have the default of MPI_COMM_WORLD, MPI_COMM_SELF, and MPI_COMM_PARENT.**

MPI-2, page 94, line 3-5, change

> *** The manager is represented as the process with rank 0 in (the remote**
> *** group of) MPI_COMM_PARENT. If the workers need to communicate among**
> *** themselves, they can use MPI_COMM_WORLD.**

to

> *** The manager is represented as the process with rank 0 in (the remote**
> *** group of) the parent communicator. If the workers need to communicate**
> *** among themselves, they can use MPI_COMM_WORLD.**

Reason: MPI_COMM_PARENT is used where the communicator returned by MPI_COMM_GET_PARENT is meant. This reflects, I believe, an earlier version of the parent where we had a MPI_COMM_PARENT similar to MPI_COMM_WORLD.

Ballot 3 –
2. MPI_UNPACK_EXTERNAL

Question:

Do you accept this entry?

Yes:

all

No:

0

Abstain:

0

Mail discussion, proposed by Hubert Ritzdorf, May 09, 2001

http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/unpackext/

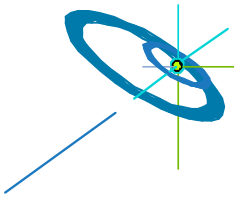
MPI-2, page 79, line 11 is

MPI_UNPACK_EXTERNAL (datarep, inbuf, incount, datatype, outbuf, outsize, position)

but should be

MPI_UNPACK_EXTERNAL (datarep, inbuf, insize, position, outbuf, outcount, datatype)

Reason: Wrong and inconsistent with rest of the definition of MPI_UNPACK_EXTERNAL.



Ballot 3 –
3. Additional C++ binding errors

Question:

Do you accept this entry?

Yes:

all

No:

0

Abstain:

0

Mail discussion proposed by Rolf Rabenseifner, Jul 17, 2001

http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/CxxBindings/

MPI-2, page 337, line 31-32 reads

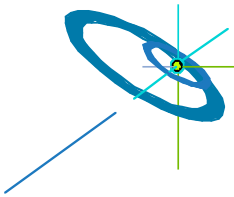
bool MPI::Win::Get_attr(const MPI::Win&win, int win_keyval, void* attribute_val) const

but should read

bool MPI::Win::Get_attr(int win_keyval, void* attribute_val) const

---

Reason: same as adopted correction in Ballot 1&2 → MPI 2.0 page 204, line 30



Ballot 3 –
4. MPI_REQUEST_CANCEL used where MPI_CANCEL intended

Question:

Do you accept this entry?

Yes:

all

No:

0

Abstain:

0

Mail discussion, proposed by Jeff Squyres and Rajeev Thakur, Oct. 31, 2006

http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/req-cancel/

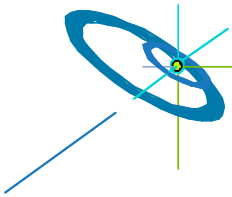
On page 172, line 37 in section 8.2, change

MPI_REQUEST_CANCEL

To

MPI_CANCEL

Reason: Typo



Ballot 3 – 5. Intercommunicator collective and datatypes

Moved into Ballot 4

Still TODO !

Question:

Do you accept this entry?

Yes:

No:

Abstain:

Mail discussion, proposed by Bill Gropp, Feb 25, 2000, **modified Jan 14, 2008**

http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/iccoll/

MPI-2, page 162, line 47-48 reads (in MPI_ALLREDUCE)

Both groups should provide the same count value.

but should read

....

We may counter-check with following MPI-1.2 text whether the proposed new text is okay.

Still TODO !

Ballot 3 – 5. Intercommunicator collective and datatypes (continued)

→ **Moved into Ballot 4**

Still TODO !

Question:

Is the merging decision for MPI-2 Sect.3.2.7 okay?

Yes:

No:

Abstain:

- **Background is the MPI-1.2 text on MPI_Reduce datatype/count usage**

Blue: MPI 1.1, page 114, lines 1,28-30 Purple: MPI-2.0, page 26, lines 22-34

The routine is called by all group members using the **same** arguments for **count**, **datatype**, op, root and comm.

Bold font highlighting from me

…

The datatype argument of MPI_REDUCE must be compatible with op. Predefined operators work only with the MPI types listed in Section 4.9.2 and Section 4.9.3. **Furthermore, the datatype and op given for predefined operators must be the same on all processes**..

Note that it is possible for users to supply different user-defined operations to MPI_REDUCE in each process. MPI does not define which operations are used on which operands in this case. User-defined operators may operate on general, derived datatypes. In this case, each argument that the reduce operation is applied to is one element described by such a datatype, which may contain several basic values. **This is further explained in Section 4.9.4**

Advice to users. Users should make no assumptions about how MPI_REDUCE is implemented. Safest is to ensure that the same function is passed to MPI_REDUCE by each process. *(Advice to users.)*

Overlapping datatypes are permitted in ``send'' buffers. Overlapping datatypes in ``receive'' buffers are erroneous and may give unpredictable results.

Ballot 3 – 5. Intercommunicator collective and datatypes (continued)

Moved into Ballot 4

Still TODO !

Question:

Do you accept this entry?

Yes:

No:

Abstain:

Mail discussion, proposed by Bill Gropp, Feb 25, 2000, **modified Jan 14, 2008**

http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/iccoll/

MPI-2, page 163, line 22-24 reads (in MPI_REDUCE_SCATTER)

> **Within each group, all processes provide the same recvcounts argument, and the sum of the recvcounts entries should be the same for the two groups.**

but should read

> **Within each group, all processes provide the same type signature as defined by the recvcounts and datatype arguments, and the recvcounts entries and datatype should specify the same type signature for the two groups.**

Reason: Several of the intercommunicator collective operations contain statements along the lines of "Both groups should provide the same count value". However, what is really required is that the (count,datatype) tuples describe the same type signature. See MPI_Allreduce and MPI_Reduce_scatter. I propose a clarification that replaces the text that refers only to count to "Both groups should provide count and datatype arguments that specify the same type signature."

Ballot 3 – → Moved into Ballot 4

Still TODO !

6. const in C++ specification of predefined MPI objects

Question:

Do you accept this entry?

Yes:

All-5

No:

0

Abstain:

5

Mail discussion, by Richard Treumann and Rolf Rabenseifner, Jun 13 – Jul 26, 2001

http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/cxxconstdtype/

- MPI-2, page 345, line 37: **Remove** the const from **const MPI::Op**.
- MPI-2, page 346, line 20: **Remove** the const from **const MPI::Group**.
- MPI-2, page 346, add after line 34:
  Advice to implementors. If an implementation does not change the value of predefined handles while execution of MPI_Init, the implementation is free to define the predefined operation handles as const MPI::Op and the predefined group handle MPI::GROUP_EMPTY as const MPI::Group. Other predefined handles must not be "const" because they are allowed as INOUT argument in the MPI_COMM_SET_NAME/ATTR and MPI_TYPE_SET_NAME/ATTR routines. (End of advice to implementors.)

- Reason: MPI_Init may change the predefined handles, because MPI 1.1, page 10, lines 9-10 says: "Opaque objects accessed by constant handles are defined and do not change value between MPI initialization (MPI_INIT() call) and MPI completion (MPI_FINALIZE() call)." Therefore they must not be defined as const in the MPI standard.
  I would allow one exception: The predefined ...._NULL handles, because as fare as I know, all implementations handle ..._NULL as (zero) constant of arbitrary datatype. See MPI-2, page 346, lines 4, 10, 12, 14, 16 (const in Ballot 1&2).

Ballot 3 –
7. Error in MPI_Scan Example

Question:

Do you accept this entry?

Yes:

all

No:

0

Abstain:

0

Mail discussion, proposed by A. Ayhan Kanmaz, May 14, 2003

http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/scanexample/

MPI 1.1, page 128, line 11, in MPI-1.1 has an extraneous root argument.
That line should be

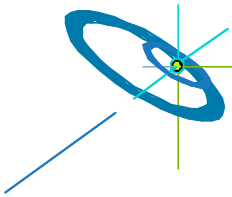
MPI_Scan(a, answer, 1, sspair, myOp, comm);

(instead of

MPI_Scan(a, answer, 1, sspair, myOp, root, comm);

)

Reason: MPI_Scan hasn‘t a root argument.



Ballot 3 –
8. Missing newline in Fortran binding

Question:

Do you accept this entry?

Yes:

all

No:

0

Abstain:

0

MPI-2, page 223, line 19. Change

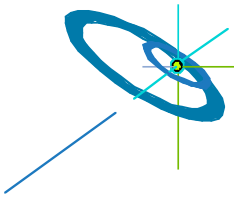
MPI_FILE_GET_VIEW(FH, DISP, ETYPE, FILETYPE, DATAREP, IERROR)
INTEGER FH, ETYPE, FILETYPE, IERROR
CHARACTER*(*) DATAREP, INTEGER(KIND=MPI_OFFSET_KIND) DISP

to

MPI_FILE_GET_VIEW(FH, DISP, ETYPE, FILETYPE, DATAREP, IERROR)
INTEGER FH, ETYPE, FILETYPE, IERROR
CHARACTER*(*) DATAREP
INTEGER(KIND=MPI_OFFSET_KIND) DISP

in io-2.tex. (Replace the comma after the declaration of datarep)

Reason: Formatting error



Ballot 3 –
9. Misspelled argument in Fortran binding

Question:

Do you accept this entry?

Yes:

all

No:

0

Abstain:

0

MPI-2, page 66, line 26, change

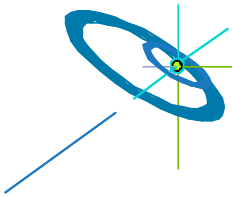
MPI_TYPE_CREATE_HVECTOR(COUNT, BLOCKLENGTH, STIDE, OLDTYPE, NEWTYPE, IERROR)
INTEGER COUNT, BLOCKLENGTH, OLDTYPE, NEWTYPE, IERROR
INTEGER(KIND=MPI_ADDRESS_KIND) STRIDE

to

MPI_TYPE_CREATE_HVECTOR(COUNT, BLOCKLENGTH, STRIDE, OLDTYPE, NEWTYPE, IERROR)
INTEGER COUNT, BLOCKLENGTH, OLDTYPE, NEWTYPE, IERROR
INTEGER(KIND=MPI_ADDRESS_KIND) STRIDE

in misc-2.tex (Replace STIDE with STRIDE).

Reason: Typo



Ballot 3 –
10. Error in MPI-1, Example 3.12

Question:

Do you accept this entry?

Yes:

all

No:

0

Abstain:

0

Proposed by NN, and Bill Gropp, Jan. 3, 2008

Mail discussion: Examples in Chapter 3 of MPI 1.1 require several fixes.

MPI 1.1, Example 3.12, page 43, line 47 and page 44, lines 1, 5, 8, 10, and 13, the communicator argument **comm** must be added before the req argument.

Mail discussion: The **ierr** argument must be added at the end of the argument list in the calls to MPI_COMM_RANK and MPI_WAIT in MPI 1.1, page 43, line 43, and page 44, lines 6 and 14.

Mail discussion: The **ierr** argument must be added at the end of the argument list in the calls to MPI_WAIT in MPI 1.1, page 44, lines 35 and 36.

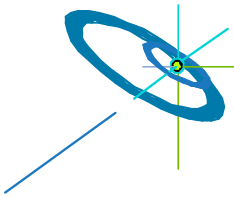
Mail discussion: The lines in MPI 1.1, page 52, line 45, and page 53, line 17

IF (status(MPI_SOURCE) = 0) THEN

should be

IF (status(MPI_SOURCE) .EQ. 0) THEN

Reasons: Obvious / Syntax error



Ballot 3 –
11. Error in MPI-1, Example 3.34

Question:

Do you accept this entry?

Yes:

all

No:

0

Abstain:

0

Mail discussion, proposed by Bettina Krammer

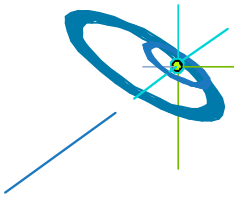
http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/ex334/

MPI 1.1, page 80, line 2,

The variable base should be declared as **MPI_Aint**, not **int**, in Example 3.34.

Reason:

The variable base (declared on this line) is used to store the address output from MPI_Address. On systems with addresses longer than 32 bit, a truncation will cause wrong execution of the program.



Ballot 3 –
12. Change MPI-2, page 343, lines 22-23

Mail discussion, proposed by Jeff Squyres, Nov. 27, 2007

http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/constbottom/

Prelimi-nary question:

Do you accept Proposal 2 ?

1st vote on 1/14

Yes:

(All-6)

No:

(1)

Abstain:

(5)

Change MPI-2, page 343, lines 22-23

// Type: const void * MPI::BOTTOM

to (Proposal 1)

// Type: void * MPI::BOTTOM

to (Proposal 2)

// Type: void * const MPI::BOTTOM

Questions: (2nd votes on 1/16/2008)

Do we remove the const before void?

Yes: all-1 No: 0 Abstain: 1

Do we add the const before MPI::BOTTOM?

Yes: all-14 No: 1 Abstain: 13

Reason:

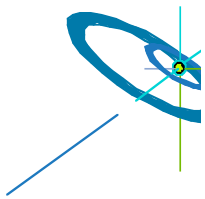
See mail discussion on next slides

Jeff Squyres + Alexander Supalov + Erez Haba are reviewing the topic:

e.g., const allows optimized allocation of the value in read-only pages

This declaration must reflect the rule defined in MPI 1.1, page 10, lines 7-11:

All named constants, with the exception of MPI_BOTTOM in Fortran, can be used in initialization expressions or assignments. These constants do not change values during execution. Opaque objects accessed by constant handles are defined and do not change value between MPI initialization (MPI_INIT() call) and MPI completion (MPI_FINALIZE() call).



Ballot 3 –
12. Change MPI-2, page 343, lines 22-23 (discussion)

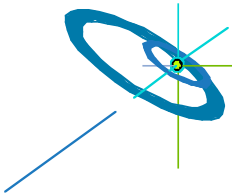
Jeff Squyres, Nov. 27, 2007

A user recently raised an issue that I just looked into and discovered a problem with the C++ binding for MPI::BOTTOM. In the spec, MPI::BOTTOM is defined to be of type (const void*). However, all receive buffers are defined to be of type (void*) -- such as for the various flavors of point-to-point receive, the receive buffer for collectives, etc. This means that you'll get a compiler error when trying to use MPI::BOTTOM as a receive buffer:

bottom.cc:81: error: invalid conversion from const void*' to void*'

bottom.cc:81: error: initializing argument 1 of virtual void MPI::Comm::Bcast(void*, int, const MPI::Datatype&, int) const'

A user can cast away the const-ness of MPI::BOTTOM, but that seems inelegant/wrong. I don't yet have a solution to this problem; I raise it here so that it gets added to the list of issues to be addressed in MPI-2.1.



Ballot 3 –
12. Change MPI-2, page 343, lines 22-23 (discussion)

Dave Goodell, Nov. 27, 2007

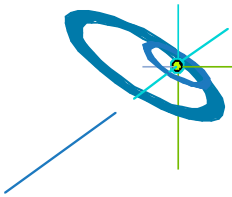
Looks like the const is on the wrong side of the declaration. That is, unless my C++ is too rusty it should instead be something like:

```
namespace MPI {
    ...
    extern void * const BOTTOM;
    ...
}
```

"const TYPE * FOO" indicates that the data pointed to by FOO is read-only. So "*FOO = BAR;" would be an illegal statement.

"TYPE * const FOO" indicates that the memory holding the value of FOO is read-only. So "FOO = &BAR;" would be an illegal statement.

The latter seems to be what is desired for MPI::BOTTOM: an address that cannot be changed but a the data that it references can.

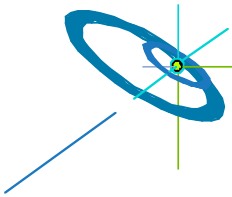


Ballot 3 –
12. Change MPI-2, page 343, lines 22-23 (discuss., cont'd)

Jeff Squyres, Nov. 28, 2007

> Good point. I think you're right -- I ran a few tests to convince myself that changing the type of MPI::BOTTOM to (void * const) won't break anything in terms of the other existing bindings.
>
> However, in terms of what MPI::BOTTOM *should* be, shouldn't it be *both* consts? We don't want the value to change, nor do we want the pointed-to-contents where it points to change:
>
> extern const void * const BOTTOM;
>
> Technically, though, with your suggestion, you couldn't change the pointed-to contents without casting anyway (because you can't assign to *(void*)). So this might be a good enough solution.

My opinion: Dave Goodell is right. → Therefore back to the proposel-slide.



Ballot 3 –
13. MPI 1.1, strlen in first pt-to-pt example

Question:

Do you accept this entry?

Yes:

all

No:

0

Abstain:

0

Mail discussion, proposed by Bill Gropp, Jan 2, 2008

http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/strlen/

In MPI 1.1, page 16, line 23, use

strlen(message) + 1

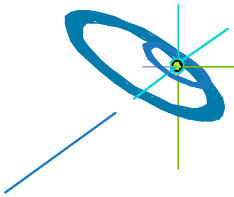
instead of

strlen(message)

in the MPI_Send call.

Reason:

In the MPI-1 document, on page 16 (first page of chapter 3), the example uses strlen(message) for the number of characters in the string message to send, and then uses printf to print that message when received. This fails to send the trailing null, so in the MPI_Send call, the length should be strlen(message) + 1 on line 33.



Ballot 3 –
14. Formatting error on MPI 1.1, page 58

Question:

Do you accept this entry?

Yes:

all

No:

0

Abstain:

0

Mail discussion, proposed by Bill Gropp, Jan 3, 2008

http://www.cs.uiuc.edu/homes/wgropp/projects/parallel/MPI/mpi-errata/discuss/persistypo/

A LaTeX line break is needed in MPI 1.1, page 58, line 44, in Section 3.9.

The text should read

be invoked in a sequence of the form,

Create (Start Complete)* Free

where * indicates zero or more repetitions. If the same communication ...

Reason: Formatting error