Errata for 'Using MPI'

May 19, 1999

Note: Some of these were corrected in the second printing of *Using MPI*.

various The use of `MPI_Status stat` may cause problems if this is used as a global variable because of the Unix system call `stat`. We recommend replacing `stat` with `status`.

Thanks to wcs@nas.nasa.gov (Bill Saphir).

p 15 in the MPI_Recv call, "dest" should be identified as "source"

Thanks to David Levine <levine@mcs.anl.gov>.

p 23 Variables "sizetype and sumtype" are not used.

p 25 `pi -np 4` should now be `mpirun -np 4 pi`

p 35 The text in the first line of this page, "in the vector b", should be "in the vector c". Similiarly on page 34.

Thanks to Bryan Putnam <bfp@bfp.cc.purdue.edu>.

p 50 There is a missing `;` after `sscanf`.

Thanks to Stacey Smith <smith@mcs.anl.gov>.

p 59 The second sentence reads "programs for the these algorithms" but should read "programs for these algorithms".

Thanks to Bryan Putnam <bfp@bfp.cc.purdue.edu>.

p 66 "perforance" should be "performance".

Thanks to Bryan Putnam <bfp@bfp.cc.purdue.edu>.

p 66 `Mpi_Cart_create` should be `MPI_Cart_create`.

Thanks to Bryan Putnam <bfp@bfp.cc.purdue.edu>.

p 84 thirtytwo needs a space

p 88 The `stridetype` in the example of `MPI_TYPE_VECTOR` is the same item as `stride` in the example on page 90. One of the two should be changed to match.

Thanks to carriero-nicholas@CS.YALE.EDU (Nicholas Carriero).

p 88 The example of `MPI_Type_vector` is formally correct, but the expressions for the count and stride do not match the example in section 4.7. The code should be

```
call MPI_TYPE-VECTOR( ey - sy + 1, 1, ex - sx + 3,
          MPI_DOUBLE_PRECISION, stride, ierr )
call MPI_Type_COMMIT( stride, ierr )
```

The paragraph following this code should have the corresponding values changed.

p 90 Figure 4.20 should be replaced with

```
       subroutine exchng2( a, sx, ex, sy, ey,
     $                     comm2d, stride,
     $                     nbrleft, nbrright, nbrtop, nbrbottom  )
       include "mpif.h"
       integer sx, ex, sy, ey, stride
       double precision a(sx-1:ex+1, sy-1:ey+1)
       integer nbrleft, nbrright, nbrtop, nbrbottom, comm2d
       integer status(MPI_STATUS_SIZE), ierr, nx
c
       nx = ex - sx + 1
c  These are just like the 1-d versions, except for less data
        call MPI_SENDRECV( a(sx,ey),  nx, MPI_DOUBLE_PRECISION,
     $                    nbrtop, 0,
     $                    a(sx,sy-1), nx, MPI_DOUBLE_PRECISION,
     $                    nbrbottom, 0, comm2d, status, ierr )
        call MPI_SENDRECV( a(sx,sy),  nx, MPI_DOUBLE_PRECISION,
     $                    nbrbottom, 1,
     $                    a(sx,ey+1), nx, MPI_DOUBLE_PRECISION,
     $                    nbrtop, 1, comm2d, status, ierr )
c
c This uses the "strided" datatype
        call MPI_SENDRECV( a(ex,sy),  1, stride, nbrright,  0,
     $                     a(sx-1,sy), 1, stride, nbrleft,  0,
     $                     comm2d, status, ierr )
        call MPI_SENDRECV( a(sx,sy),  1, stride, nbrleft,   1,
     $                     a(ex+1,sy), 1, stride, nbrright, 1,
     $                     comm2d, status, ierr )
        return
        end
```

(The change is mostly to use a single array `a` and to make the sense of left and right etc match the (x, y) orientation of the array.

p 99 (This is a clarification, not an errata). Note that table 5.2 does not include the optional data type of MPI_DOUBLE_COMPLEX for Fortran, which is listed in A.2 of the MPI standard but *not* in section 3.2.2 of the MPI standard. Note that this also applies to the list of data types for collective (reduction/scan) operations, in section 4.9.2 of the MPI standard.

p 104-6 The bindings for `MPI_Allgatherv` are incorrect in Tables 5.5 and 5.6. They should be

```
int MPI_Allgatherv(void* sendbuf, int sendcount, MPI_Datatype sendtype,
                   void* recvbuf, int *recvcounts, int *displs,
                   MPI_Datatype recvtype, MPI_Comm comm);
```

and

```
MPI_ALLGATHERV(sendbuf, sendcount, sendtype,
               recvbuf, recvcounts, displs,
               recvtype, comm)
<type> sendbuf(*), recvbuf(*)
integer sendcount, sendtype, recvcount(*), displs(*), recvtype,
        comm
```

Thanks to rvdg@cs.utexas.edu (Robert van de Geijn).

p 139ff In the `Ibcast` examples, the copy and delete callback functions are declared as void but should return an integer. Add the code

```
return(MPI_SUCCESS);
```

to the end of these routines

Thanks to Peter Junglas <Junglas@tu-harburg.d400.de>.

p 140 Need to mention that `MPI_COMM_SPLIT` (and all other communicator creation routines) must be called collectively. (It is possible that they may be implemented in such a way as to avoid the need for any communication, but everyone in the "old" communicator must call these routines).

Thanks to carriero-nicholas@CS.YALE.EDU (Nicholas Carriero).

p 140 In table 6.3 at the bottom of p.140, the type of variable "key" is missing. The type is properly given in Appendix A.1 (p. 257), and the information for the Fortran bindings for this function seems to be correct.

Thanks to puri@cs.msstate.edu (Puri Bangalore).

p 143 The lines preceding `MPI_Cart_sub()` are transposed. That is, the remaining dimensions are eliminated in the opposite order than they should be. Here is how the correct fragment looks:

```
/* Use cartesian sub-topology mechanism to get row/col comms: */
remain_dims[0] = FALSE; remain_dims[1] = TRUE;
MPI_Cart_sub(comm_2d, remain_dims, &row);
remain_dims[0] = TRUE; remain_dims[1] = FALSE;
MPI_Cart_sub(comm_2d, remain_dims, &col);
```

The examples have been updated to reflect this correction as well.

Thanks to schneid@tc.cornell.edu (David J. Schneider).

p 150 Figure 6.16: The declaration of `ddot_stride` doesn't give a type for `*y`, so in ANSI C it will be `int`, not `LA_Dvector`.

p 159 The routines `MPE_Counter_nxtval` contains a superfluous calls to `MPI_Comm_rank` and declaration of `myid`.

Thanks to carriero-nicholas@CS.YALE.EDU (Nicholas Carriero).

p 159 This routine has a test for `smaller_comm` *after* using `*smaller_comm` in an `MPI_Barrier`. The `MPI_Barrier` call should be within the test (actually the test on the comparison against `MPI_UNEQUAL` as well).

Thanks to carriero-nicholas@CS.YALE.EDU (Nicholas Carriero).

p 163 There is a missing `)` in the second paragraph of section 7.2. It should read

> The ability to compute FCI wave functions thus confers the ability to adjudicate among all approximate methods (e.g., SCF, many-body methods, and truncated CI*) and, by comparison with experiment, permits assessment of deficiencies in the one-particle basis set and the Hamiltonian approximations.

Thanks to carriero-nicholas@CS.YALE.EDU (Nicholas Carriero).

p 166 Second line from bottom; there is an extraneous comma after `MPI_Op_create`.

p 166 Second box, the calling sequence of User_function is not int ... but should be void ... Also see box on next page for Fortran again a suspicious int.

Thanks to Barry Smith.

p 167 Table 7.6 has typesetting error.

p 167 Line 2: "intercommunicators" and "intracommunicators" should be italicized.

p 167 In the second paragraph in "Other Collective Operations", in the discussion of `MPI_Reduce_scatter`, the word "combines" should be "combine".

p 168 Line 18: operation(s).

p 168 Line 14 "When receiving..." is wrong or, at best, misleading.

p 176 Figure 7.13, the `MPI_Pcontrol` declaration is missing `const` and has a semi-colon at the end.

p 177 Table 7.10: `MPI_PCONTROL` is missing ierr return value.

p 180 At bottom of page `MPI_Get_errhandler` should be `MPI_Errhandler_get` and `MPI_Set_errhandler` should be `MPI_Errhandler_set`.

Thanks to Barry Smith <bsmith@mcs.anl.gov>.

Chapter 9 Chapter 9 often recommends using `MPI_Send` as a replacement for other system's send routines. However, because `MPI_Send` may block when sending messages when no matching receive is posted (see Chapter 4 for a full discussion), some applications in some systems may need to use `MPI_Bsend` instead. This is particularly true for P4, PVM and Intel NX. The necessary `MPI_Buffer_attach` call can be added to the initialization (i.e., immediately after the `MPI_Init`).

Thanks to carriero-nicholas@CS.YALE.EDU (Nicholas Carriero).

p 246 Section 10.5 should be amended to reflect the MPI2 process.

p 258 The prototype for MPI_Get_processor_name is inconsistent with the MPI standard (p. 217). The C and FORTRAN bindings in the standard appear consistent, but the FORTRAN binding is inconsistent with the C binding in *Using MPI*. The binding should be

```
int MPI_Get_processor_name( char *name, int *resultlen )
```

Thanks to schneid@tc.cornell.edu (David J. Schneider).

p 263 The binding for `MPI_Topo_test` should be

```
int MPI_Topo_test( MPI_Comm comm, int *top_type )
```

p 268 "nt" in front of "MPI_Comm_split" is extraneous.

Thanks to schneid@tc.cornell.edu (David J. Schneider).

p 275 The binding for `MPI_Topo_test` should be

```
MPI_Topo_test( comm, top_type, ierr )
integer comm, top_type, ierr
```

p 280-282 The MPICH model implementation now recommends using the `mpirun` command instead of the system-specific information given in section B.2.

p 290 The MPI FAQ is now at `http://www.erc.msstate.edu/mpi/mpi-faq.html`

p 292 E.1.3, in the sentence "If we want to send or...a(sx:ex,sy:ez,k)", ez should be ey.

Thanks to David Levine <levine@mcs.anl.gov>.

This isn't really a correction, but if the example of sending parts of `a(nx,ny,nz)` sends the entire side (for example, `sx=sy=sz=1` and `ex=nx`, `ey=ny`, and `ez=nz`), then the three forms can be simplified to

```
a(1:nx,1:ny,k):
MPI_Type_contiguous(nx*ny,MPI_DOUBLE_PRECISION,newz,ierr)
a(i,1:ny,1:nz):
MPI_Type_vector(ny*nz,1,nx,MPI_DOUBLE_PRECISION,newx,ierr)
a(1:nx,j,1:nz):
MPI_Type_vector(nz,nx,nx*ny,MPI_DOUBLE_PRECISION,newy,ierr)
```

Note in particular the form for `newx`; this exploits the fact that all of the elements of this plane are separated by `nx` elements. This is not true for the more general case, which leads to the use of `hvector` in the example.

A.2 The FORTRAN binding for MPI_TYPE_COMMIT has been omitted from Appendix A.2.

Thanks to schneid@tc.cornell.edu (David J. Schneider).