

Errata for ‘Using MPI, 2nd Edition’

September 19, 2003

p xvii The third line from the bottom should be “MPICH. Appendix C”.

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

p 6 The 5th line from the bottom has an extra “the” in it; the end of that line should be “not the same”.

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

p 12 The last sentence should end

in a more tutorial approach in [66].

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

p 42 The 5th line from the top reads

```
if (rank .gt. rows)
```

but should read

```
if (myid .gt. rows)
```

Thanks to Weiqun Zhang <zhang@ucolick.org>.

p 48 The declaration of `ans` should use `MAX_BCOLS`:

```
double precision buffer(MAX_ACOLS), ans(MAX_ACOLS)
```

should be

```
double precision buffer(MAX_ACOLS), ans(MAX_BCOLS)
```

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

p 48–51 The example code here uses user-defined state numbers in the MPE logging calls (e.g., `MPE_DESCRIBE_STATE`). While this works with stand-alone logging, it is not compatible with the automatic logging that is available with MPICH. Applications should consider using the routine `MPE_Log_get_event_number` to get state numbers rather than defining them.

Thanks to Anthony Chan <chan@mcs.anl.gov>.

p 48–51 There are a few errors in the examples in Figures 3.10, 3.11, and 3.12.

In Figure 3.10, the variable `master` should be initialized to zero.

If Figure 3.12, all of the calls to `MPI_LOG_EVENT` need an additional argument `IERR` at the end of the argument list; i.e.,

```
call MPE_LOG_EVENT(2, 0, "bend", ierr)
```

Note that later versions of the Fortran MPE routines return the error code as the value of the function, and hence would need

```
ierr = MPE_LOG_EVENT( 2, 0, "bend" )
```

In Figure 3.11 on page 50, the call to `MPI_BCAST` is not shown in the code. The call is referred to at the line

```
.... initialization of a and b, broadcast of b
```

but the code itself is not shown. It is the same as in Figure 3.12:

```
do 85 i = 1, bcols
call MPI_BCAST(b(1,i), brows, MPI_DOUBLE_PRECISION, master, &
               MPI_COMM_WORLD, ierr)
85   continue
```

Thanks to Hartmut Kapitza <hartmut.kapitza@gkss.de>.

p 57 The value of `INT_MAX` must correspond to the range used by the random routine. Linux users can use `RAND_MAX`; for greatest portability, values returned from `random` should be discarded if they are too large. The following code replaces the code in the master for filling the `rands` array:

```
if (request) {
  for (i = 0; i < CHUNKSIZE; ) {
    rands[i] = random();
    if (rands[i] <= INT_MAX) i++;
  }
}
```

Thanks to Kevin T. Pedretti <pedretti@eng.uiowa.edu>.

p 58 The call to `MPI_Comm_free` should be moved into the worker process branch of the `if`.

Thanks to Kevin T. Pedretti <pedretti@eng.uiowa.edu>.

p 64 In the second paragraph from the bottom, “some compilers” should be “Some compilers”.

p 65 In the second item (Expecting `argc` etc.), “this *allows*” should be “This *allows*”.

p 72 The code in Figure 4.2 has a misplaced parenthesis and should be

```
integer i, j, n
double precision u(0:n+1,0:n+1), unew(0:n+1,0:n+1)
do 10 j=1, n
  do 10 i=1, n
    unew(i,j) = &
      0.25*(u(i-1,j)+u(i,j+1)+u(i,j-1)+u(i+1,j)) - &
      h * h * f(i,j)
  10 continue
```

Thanks to Steven Knudsen <sknudsen@softwareresearch.org>.

p 97 The legend for Figure 4.17 should say

Predicted time for a 1-D (solid) and 2-D (dashed) decomposition of the 2-D Poisson problem

Thanks to Paul Hovland <hovland@mcs.anl.gov>.

p 99 The near the bottom of the page has a misplaced parenthesis and should be

```
integer i, j, n
double precision u(sx-1:ex+1,sy-1:ey+1), &
  unew(sx-1:ex+1,sy-1:ey+1)
do 10 j=sy+1, ey-1
  do 10 i=sx+1, ex-1
    unew(i,j) = &
      0.25*(u(i-1,j)+u(i,j+1)+u(i,j-1)+u(i+1,j)) - &
      h * h * f(i,j)
  10 continue
```

p 100 The first argument to the `MPI_TYPE_VECTOR` call in the middle of the page should be `ey-sy+1` since only the interior points of the mesh are communicated. The text in the last full paragraph on the page should read

...In this example, there is one double-precision item per block; the double-precision values are $ex + 1 - (sx - 1) + 1 = ex - sx + 3$ apart, and there are $ey - (sy) + 1 = ey - sy + 1$ of them, since only the interior elements are needed.
...

p 100 The sentence after the call `MPI_TYPE_COMMIT` should read

The arguments to `MPI_Type_vector` describe a *block*, which consists of a number of (contiguous) copies of the input datatype given by the third argument. The first argument is the number of blocks; the second is the number

Thanks to Hartmut Kapitza <hartmut.kapitza@gkss.de>.

p 104 The code in Figure 4.22 has a misplaced parenthesis and should be

```
integer i, j, n
double precision u(sx-1:ex+1,sy-1:ey+1), &
                unew(sx-1:ex+1,sy-1:ey+1)
do 10 j=sy+1, ey-1
  do 10 i=sx+1, ex-1
    unew(i,j) = &
      0.25*(u(i-1,j)+u(i,j+1)+u(i,j-1)+u(i+1,j) - &
        h * h * f(i,j))
  10 continue
```

p 105 The computed goto in the example should refer to `status(MPI_TAG)`, not `status(MPI_TAG,idx)`, since for `MPI_Waitany`, only a single status value is returned. However, because 4 of the requests are sends instead of receives, it would be better to use the `idx` value returned by `MPI_Waitany`; this indicates which request has completed. A cleaner solution moves the wait on the sends to a separate step, using `MPI_Waitall` for them (this uses the MPI-2 `MPI_STATUSES_IGNORE` to simplify the `Waitall` call):

```
do 100 k=1, 4
  call MPI_WAITANY(4,requests,idx,status,ierr)
  goto (1,2,3,4), idx
  ...
enddo
call MPI_WAITALL(4, requests(4), MPI_STATUSES_IGNORE, ierr)
```

Thanks to jose@frenadol.ti.uam.es.

p 114 "descuss" should be "discuss" in the first line of the page

Thanks to Rusty Lusk <lusk@mcs.anl.gov>.

p 121 In the second example using `MPI_Allgather`, `counts` should be `totalcount`, where `totalcount` is the sum of the values in the array `counts`:

```
MPI_Allgather( myparticles, count, particletype,
              allparticles, totalcount, particletype, MPI_COMM_WORLD );
```

Thanks to Dinesh Kaushik <kaushik@mcs.anl.gov>.

p 123 Not actually an errata, but a clarification. After the example at the end of section 5.2.1, add a paragraph that explains why there are MPI datatypes for both the sender and receiver, particularly for `MPI_Gather` and `MPI_Allgather`. In case you are wondering, the reason is that MPI requires only that the type signatures match; for example, by using a

different MPI datatype for the send and receive types, the data can be transposed “on the fly”. For example, the send type could specify a row of a matrix (stored in column-major, with n columns) with a sendcount of 1 and the receive type could specify contiguous doubles with a receive count of n and receive type of MPI_DOUBLE.

p 124 Line 10 reads

```
MPI_Start( request );
```

but should read

```
MPI_Start( &request );
```

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

p 155 The second line from the bottom is missing a close parenthesis

Thanks to Rusty Lusk <lusk@mcs.anl.gov>.

p 158 In the 2nd paragraph from the bottom, “particular master/worker” should be “particularly master/worker”

p 167 The bindings for the `MPI_Copy_function` and the `MPI_Delete_function` should not use pointers to the communicators. This was a change from MPI 1.0 to MPI 1.1. The correct code is

```
typedef int MPI_Copy_function(MPI_Comm oldcomm, int *keyval,
void *extra_state, void *attribute_value_in, void *attribute_value_out,
int *flag)
```

```
typedef int MPI_Delete_function(MPI_Comm comm, int *keyval,
void *attribute_value, void *extra_state)
```

Thanks to Emil Ong <ong@mcs.anl.gov>.

p 233–268 The following typographic errors were found by Jeffrey Oldham <oldham@codesourcery.com>:

page	paragraph	line	incorrect	correct
233	1	1	code or class)	code or class
233	3	3	Fotran	Fortran
234	1	4	declearations	declarations
237	4	4	unusal	unusual
241	4	6	data	date
242	2	7	in	if
245	1	2	1.0	2.0
245	2	1	introudce	introduce
250	0	5	rougly	roughly
255	3	5	send	sent
266	1	5	long as	as long as
268	2	1	the a	the

p 237 "stop code)" should be "stop(code))".

Thanks to Rusty Lusk <lusk@mcs.anl.gov>.

p 240 "gloabl" should be "global" in the last line.

p 268 In Section 9.2.2, "rather is combines" should be "rather it combines"

p 270 At the top of the page, add a comma after the section number "in Section 5.2.5,"