Minutes of MPI meeting
March 13–15
Chicago IL

William Gropp

March 23, 1995

1 Introduction

This is an unedited set of minutes taken during this MPI meeting. This contains both a summary of some of the discussions and official, binding votes of the MPI Forum. Offical votes are set in bold type; for example **1-2-3**. Straw votes, which are non-binding and may be voted on by anyone present, are shown as *1-2-3*. Please send any corrections to `gropp@mcs.anl.gov`.

2 Monday

Jack Dongarra opens meeting at 1:30pm and reviews voting rules (present at last two of last three MPI meetings). The last two were Knoxsville and Dallas. Also did introductions and checked voting status.

R Lusk starts actual meeting and reviews agenda. Reviewed process of creating proposals (create, discuss, vote twice; requires at least three meetings, typically four meetings).

T. Skjellum asked if vendors would prefer a commercial standard status for MPI; there was no interest and some concern that doing so would be detrimental to the MPI process.

M. Snir chairs corrections and clarifications. No votes; this is a review session to discuss the issues. Votes Tuesday. Numbers refer to a document distributed by Marc Snir at the meeting.

- (2.1) Once assigned, a request handle iss not modified until set to `MPI_REQUEST_NULL`. Two parts: one is a clarification, another changes the C binding of `MPI_Start`.
- (2.3) Remove padding when computing extent when `MPI_UB`/`MPI_LB` used. (S. Otto points out that referring to default compiler is not what we

actually mean, J. Cownie points out that it really relates to the natural alignment of architecture).

- (2.4) Disallow negative extents.
- (2.5) `MPI_Type_count`. Discussion suggests that it is a left over from an attempt to provide datatype unpacking routines.
- (2.6) Constants in static initializers. Tony S. asks about `MPI_COMM_WORLD`.
- (2.7) Clarification of Bsend and packing units. B. Gropp points out that users need additional information on how big a buffer to provide.
- (2.8) ANSI compatibility of `MPI_Attr_put`
- Internal MPI calls for profiling interface are name shifted. J Cownie points out that this is already discussed in Chapter 8.
- `MPI_IO` is myrank if ok and not `MPI_ANY_PROC` (if some can not do I/O). (Rather than some other process even though you can).
- Asked about aliasing, L Lewins points out that the definition of OUT and INOUT explicitly prohibit aliasing.
- (2.2) Multiple completions containing `MPI_REQUEST_NULL` or inactive persistant requests. Presented a list of changes, containing for example a test of empty list would return flag of true and a null status. Testany would return an index of undefined; testsome and waitsome returns a count of undefined.

Next, Steve Otto reviews errata sheet. (He wants to put everything into the ARM rather than the original specification.) Much discussion on where errata goes (old document, new and unpublished document, or something else). R. Lusk proposes errata to May 5th version; S. Otto states that IJSA version no longer exists in electronic form.

Numbers refer to the October 24th errata sheet.

- 1. Complex for MINLOC, MAXLOC removed
- 2. Status passed as pointer always (even when IN argument)
- 3. `MPI_Aint` used for all storage areas (in case address larger than sizeof(int)).
- 4. typos
- 5. typos
- 6. Unclear wording; no better proposals
- 7. Unclear example; will not work for two processes `MPI_COMM_WORLD`

- 8. typo
- 9. Another unclear example; comes from the time when you could get a communicator that you were not a member of, which is no longer possible. Could fix example by removing a line or two of code.
- 10. Fix to example.
- 11. `MPI_COMM_NULL` is different from (`MPI_Comm`) 0.
- 12. typo
- 13. typo to args
- 14. problems with copy function?
- 15,16 missing from appendix
- 17. typo
- 18. typo
- 19. typo
- 20. typo
- 21. Do null communicators compare as `MPI_IDENT`, `MPI_ERR_COMM`, or unequal
- 22. typos

The following are from the unresolved issues section of the October 24th errata.

- 1. Should Pack count args be int or `MPI_Aint`? (represent args longer than representable in int)
- 2. typo
- 3,4. need `null_delete` and `null_copy`
- 5. Concatenation of packing units, and the receipt with a plain recv after it is packed on the sending end. It appears that the spec is clear.
- 6. `MPI_BOTTOM`
- 7. Request for an additional Attr value for representation of data. J. Cownie suggests an attribute/value for the architecture. R. Lusk suggests generating a proposal.

Break

L. Lewins mentions the problem of errors, say, receives, in `MPI_WAITANY/ALL/SOME` (multiple operations and a single error return). Should errors in this case be in the status?

W. Saphir mentions that MPI should address Fortran 90 binding—Should say supported (as Fortran 77 subset), and provide a Fortran 90 module named MPI (filling the function of mpif.h).

R. Lusk then introduces MPI-2 proposals.

Starts with Gropp/Lusk dynamic process management proposal. Described the proposal with emphasis on job allocator, process management, and MPI's interaction. Uses intercommunicators and provides simple and separate flexible versions. Also has client-server interface.

W. Saphir raises the issue of dying processes, particularly involuntarily.

Marc Snir discusses the following

- Probeany, Probesome
- `Send( , array_of_destinations, )` (multicast)
- Nonblocking collective. Questions include (tags, no tags). Implementations are more awkward (make sure that the blocking versions are compatible with the nonblocking ones)
- Mechanism for user-defined non-blocking operations (using test/wait).
- Receive and call (hrecv) and active messages Big questions in single threaded versus multithreaded semantics
- Put and get; test and set
- Dynamic process scheduling
- C++ and F90 and ADA

Jim Cownie mentions that there could be a different iprobe could return a request that could be waited on (iprobe is unlike the other immediate functions).

S. Fleischman (Convex) introduces distributed combine and collect functions. Much discussion on how these differ from `MPI_Reduce_scatter` and `MPI_Allgatherv`.

Tony Skjellum introduces

- Intercommunicator and topology extensions. Collective operations in intercommunicators.
- Multicasts (receives match broadcasts). Response to Kale's comments.
- Looking at C++ bindings

S. Otto asks how much should be layered on existing code versus adding new functions.

Tony mentions nonblocking collective can be realized with a blocking collective in a separate thread; suggests addressing threads and spawn together.

Oliver McBryan not here for discussion of "regular grid library". Some comment that it can be layered. Another comment that it is an example of what MPI enables. (See Tuesday).

Additional suggestions include

I/O (many requests)

Marc Snir mentions MPI-IO proposal, with working prototype of subset in the next month or so. Only needs the ability to create new request types, could use client-server extensions. Would need access to internals of `MPI_Datatype` to make portable. (note that prototype is using MPICH). Steve Lederman raises the issue of whether MPIF should just give IO to the MPI-IO group. M. Snir responds that it is too early (more research remains to be done). Steve responds that the committee needs a formal position on what the IO strategy is.

L. Lewins mentions that for systems with different kind of memory (for example, cacheable and non-cacheable). Could use `MPI_Malloc` for memory that is better for moving data around. Also have new 'malloc'ed types. Jim Cownie raises the issue that if these are not for portable programs, it need not be in the standard. (Convex remarks that their system has different kinds of memory as well.) T. Skjellum mentions the (different) approach where a receive returns a pointer to memory rather than taking a buffer.

Levins also raises the issue of real time. E.g., message priorities, how to make the system more deterministic from a performance standpoint. J. Cownie asks if these can be done by using attributes, and done without changing the standard (making each implementation use these attributes). Also fully connected channels (two-ended persistant communication).

S. Lederman-Huss raises the issue of threads. Could be either behavior of MPI operations in a multithreaded environment, or even specifying MPI thread operations (e.g., `MPI_Fork`)

Ian Foster will bring proposal tommorrow on generalized communicators (threads with more like processes).

Mentioned the email discussion on the progress rule.

Briefly mentioned the desirability of a standardized 'mpirun'. Snir suggests that it be a architecture-specific binding.

Yet more collective operations (`reduce_scatterv`).

Al Geist mentions the name server; consensus is that that is part of the dynamic process management issues.

Dinner

After dinner, start discussion on the errata. Again, no votes, only discussion and review.

Marc Snir leads. Numbers refer to Marc's draft of errata. Page numbers refer to May 5 draft, not IJSA article.

- (2.1) Add to spec "MPI Calls do modify opaque object handles unless explicitly stated otherwise". This is for things like `MPI_Start`. Change includes datatype commit. Consensus is to leave the `MPI_Start` binding unchanged (agreed that it is an error, but too small an error to justify changing the standard). L. Lewins point out the `MPI_Cancel` is another example that takes the address of a request and does not modify it.

- Clarification: passing an address in C does NOT imply OUT or INOUT parameter.

- (2.3) Page 69, line 47; remove the epsilon from the formula, and the accompanying text. P Pierce points out that additional text is probably warrented. L Lewins points out that the datatypes may need to identify unaligned data.

- (2.4) Concensus on no negative extents

- (2.5) Concensus on the elimination of `MPI_Type_count`.

- (2.6) MPI Constants. The problems is static initializers with arrays. Much discussion. Defer until we better understand; Consensus that we make them either all constant or not. (no halfway measures)

- (2.7) Packing units. Much circular discussion about the interrelated issues of packing units and bsend size. P Pierce brings up the need for a Bsend limit, so that users can know how much to allocate for buffers. Much discussion on what a "conservative" estimate is. Everyone was tired. Propose to add a new constant `MPI_BUFFER_OVERHEAD`; a bsend never takes more than the `MPI_Pack_size` size plus this.

  Also, Bsend is allowed to consume system resources (the same as a `MPI_Send` of a zero-length message).

- (2.2) Test and Wait of null/inactive handles. E. Lust expressed concern for changing the behavior of the current routines to be different, without changing the names (thus possibly introducing a change to existing programs). Much debate about what is more natural. P Pierce describes a situation where the current behavior is more natural (a test within a different loop; conclusion of the test is not the termination condition of the loop). L Lewins argues for consistent semantics based on an interpretation of completing a null request. Corresponds to "green" text in Marc's chart.

- Are attributes the same on all processors (`MPI_TAG_UB`, `MPI_Wtick`, error messages, ...). Point out that most of these can not be set by the user.

  Consensus is for `MPI_TAG_UB` the same at all processes in a given communicator.

Discussion of error classes and codes; confusion about portability of an error code/class to a different processor; B Gropp points out that only the names (`MPI_ERR_BUFFER` etc) are defined; portability between processors would require defining the values.

Steve Otto then started (10:00pm) from errata.

- 1. Position and outcount are ints (?) instead of Aint
- 3,4. Names of Null functions. Two names.
- 5. Yet more stuff on packing.
- 6. `MPI_BOTTOM`. Not a standard Fortran constant. Clarification of text.
- 7. MPI Proctype

3 Tuesday

R. Lusk reviews schedule

O McBryan discusses RGL (Regular Grid Library) as a subset of BlockComm and a partial subset of some other packages. Has eight routines; targeted at porting sequential codes to MPI without using any lower-level MPI calls. R Lusk asks if RGL can be layered entirely on MPI; McBryan says yes.

M. Snir begins formal process on clarifications and corrections.

19 valid voters

- MPI Calls do no modify handles to opaque objects except (a) when object is created (b) when object is marked for deletion.

  Pass 19-0-0.

  J. Cownie suggests using the C `const` to clarify the C binding. General suggestion that it should be done and proposed later.

- (rounding of datatype extents for alignments) Proposed

$$lb(typemap) = \begin{cases} \min disp_j if no MPI\_LB \\ \min(disp_j or type_j if MPI\_LB) \end{cases} \tag{1}$$

$$ub(typemap) = \begin{cases} \max disp_j + sizeof(type_j) + \epsilon(no MPI\_UB) \\ \max(disp_j : type_j if MPI\_UB) \end{cases} \tag{2}$$

$$extent = ub - lb \tag{3}$$

  A type specification with negative extent is erroneous.

  Approved 19-0-0.

- Deleteing `MPI_Type_count`

  Approved vote 16-1-0.

- Buffer send space. Proposed `MPI_BSEND_OVERHEAD` attribute; a value such that the amount of space used by Bsend is less than or equal to `MPI_BSEND_OVERHEAD` + `MPI_PACK_SIZE`(count,datatype,comm).

  Question raised about making `MPI_BSEND_OVERHEAD` a constant or an attribute.

  Revised proposal is `MPI_BSEND_OVERHEAD` is a constant with same definition.

  Approved 17-0-1.

- `void **` versus `void *` in `MPI_Attr_get`. B Gropp will draw up a proposal (or collection of proposals) for this and the interpretation of the output value.

- `MPI_TAG_UB` provides thesame value on all members of the same communicator

  Approved 18-0-1.

- `MPI_Error_string` applies to both error classes and codes. (Remark that different codes with the same class may have different text.)

  Approved 19-0-0.

- MPI defined constants for error classes are the same on all processes.

  MPI defined strings for error classes same on all processes.

  Much discussion about how adopting this may impeed the development of interoperable MPI implementations, by requiring a standardized list of error classes.

  Moved to the to-do list for proposals.

- Redefining the behavior of `MPI_Test` etc for empty lists. Proposal: `MPI_Test` and `MPI_Testany` returns `flag=true` on empty list.

  `MPI_Testsome(...., flag)` will return `flag = true` on empty list

  `MPI_Test` returns `flag = false` whenever corresponding wait returns

  Empty list signalled using

null status	TEST/WAIT
outcount = 0	SOME
index=`MPI_UNDEFINED`	ANY
null statuses	ALL

L Lewins mentions that a consistent view of blocking requests is that the never complete (rather than being a special case).

Another proposal is that WAIT on an empty list is erroneous (and gets an MPI error class).

Another position is that both TEST and WAIT on an empty list is erroneous. The argument is that it is probably a user error, and does not provide any particular service.

Could change TESTSOME to not add a flag argument, but allow outcount to be `MPI_UNDEFINED`.

Straw votes: Make test and wait consistent *16-8-5*

Empty list is ok *16-5*

Change in behavior of test/wait some.

3-way straw vote: Testsome returns success via flag = true *6 for* Testsome and waitsome return outcount = `MPI_UNDEFINED`. *1 for* Both return outcount = 0. *9 for*

M Snir will bring a more complete proposal (see Wednesday).

P Pierce will write about MPI defined constants being compile-time constants

Rolf Hempel will add `MPI_NODE_TYPE` attribute and how the values are chosen.

`MPI_GET_ERROR`(status) returns error code, if nonfatal error occured (for multiple wait and test routines). L Lewins will do.

Change C bindings to use const (J Cownie, who proposes to do it for C++ only). Recommendation to implementers to use const, but not a requirement.

The progress rule. G Burns indicated interest, but left open

B Gropp to look at all `**` in C binding

Error classes - implementors will try to standardize. B Gropp to coordinate. Will handle cases of extensibility of error classes.

Fortran 90 binding - W. Saphir will write.

Mentioned that all will be added to (mpi-core@cs.utk.edu) (mpi-comm@cs.utk.edu is the general list; also comp.parallel.mpi)

Break
S. Otto leads formal votes on unresolved part of errata

- (1) **18-1-0** for Position and outcount to be int, not Aint
- (3) Separate null functions approved **16-0-2**

- (5) Pointed out that since `MPI_PACKED` is a datatype, recursive creation of packed data.
- (6) `MPI_BOTTOM` needs clarification for Fortran; Bill Gropp will provide text.

Additional items (these were changed in IJSA version). (S Lederman raised)

- The operations `MPI_MIN`, max, minloc, maxloc on `MPI_Complex` are not allowed. **Approved 19-0-0**.
- Accepted by acclamation
- Use of Aint for addresses and differences of addresses. Lots of disscussion about use of int; several people pointed out that even on systems with 64 bit addresses, ints may be 32 bits and longs 64.

  Seems like consensus for Aint for extent, displacement (byte) and int for size, displacement (element count version).

  (remove rationale on page 70 (it is bogus)) This changes `MPI_Type_extent`, `MPI_Type_size`, `MPI_Type_lb`, `MPI_Type_ub`.

  Approved 17-0-2.
- S Lederman points out that we voted that examples are *not* part of the standard. In other words, if an example and the text disagree, the text is definitive.
- (9) Value of `MPI_COMM_RANK` when given `MPI_COMM_NULL`

  Suggest `MPI_UNDEFINED`. **Approved 17-0-2**. More discussion and revoted that behavior is erroneous. B. Gropp pointed out that a discussion about `MPI_COMM_COMPARE` caused an implementation to *change* to view an input of `MPI_COMM_NULL` as erroneous (even though he prefered the original spec).

  Approved 13-0-3.
- 14. Question is where error code should be returned (value or parameter) in Fortran. Deferred to B Gropp as part of fixing attribute routine bindings.

Lunch

R Lusk opens discussion about continuing the MPI process.

Possible dates May 1-3, June 12-14, July 24-26, Sep 4-6(Labor day), Oct 16-18 (every six weeks). One suggestion is T-Th to miss M/F busy travel days; for over weekend travel, M-W or W-F is preferable.

P. Pierce discussed the view of MPI-2 as having separate, orthogonal modules, with them released for test and comments before announcing MPI-2.

Much concern about time frame and timing of MPI 2.

One mechanism for deciding what to work on is to identify things that can not be done by layering on the existing MPI and system features (for example, could argue that parallel I/O can be layered on MPI and the basic file system).

Straw vote on scope of MPI-2 effort

- Dynamic Process Management, client server. Vote *34 for*
- Collective Extensions (nonblocking, combine). Vote *30 for*
- One-sided point-to-point, threads, channels. Vote *33 for*
- C++ and Fortran 90 and Ada Bindings. Vote *30 for*
- Topology and Intercommunicator extensions. Placed under collective extensions
- Realtime. Vote *17 for*
- I/O. Vote *4 for.*
- Graphics. Vote *0 for.*
- Debugging facilities and externalization of opaque structures. *Vote 23* for external interfaces.

W. Sapir suggests MPI-2 draft for supercomputing 96; dynamic process management draft (for comment) by SC'95.

Move dates to W-F.

Vote on Finish by SC 96, meet every 6 weeks, work on these topics. *34-0-1.* 31 said they will come to the meetings.

Break

Form working groups.

- Dynamic Process Management - B Gropp, R. Lusk, J Cownie
- Collective Extensions - Al Geist, Tony Skjellum, Paul Pierce
- One-sided operations, thread, and channels - Marc Snir
- Bindings for other languages - Nathan Doss for C++, Bill Saphir
- External interfaces - Steve H-L.
- Docmument integrator - Steve Huss-Lederman
- Mailing lists - David Walker
- Minutes - Bill Gropp, Bill Saphir

Beginning of discussion of Dynamic Process Management proposal. R. Lusk leads, using proposal of Gropp and Lusk.

Start with generalized request (in the appendix of the proposal).

Cownie points out that the out type in the MPI_NEW_REQUEST_TYPE is provided by the implementation, just like a keyval in attributes.

W. Saphir suggested that it be passed to the external interface committee. M. Snir then introduced his version of the user-defined requests, which has a different specification about when completion functions are called. R. Lusk asks if there is any guarentee about when these functions will be called; M. Snir answers no.

R. Lusk then goes to the beginning of the spawn proposal, starting with the simple MPI_COMM_MODIFY, then MPI_SPAWN. Intercommunicators are used to identify the spawners and spawnees. Introduces MPI_COMM_PARENT, the intercommunicator that contains the parent group (MPI_COMM_NULL if there is no parent).

W. Saphir asks that we step back and identify the issues that we need to address. These include the race conditions in creation of communicators. Another issue is the resource allocation and process creation (handled in the proposal). Another issue is what happens when a process dies (particularly unexpectedly).

M. Snir wants saftey (communication state of a process does not change without that process calling a routine). Also concerned about the interface between MPI and the resource manager. Is it left unspecified, do we specify, or something in between? Can you communicate with the resource manager with MPI?

W. Saphir asks about the programs that we wish to support. Asks if it is reasonable to try to list those applications. One specific application, not covered in the current proposal, is the case where the resource allocator asynchronously notifies the user that new resources are available.

M. Snir asked how much if any of the interface to the allocator is specified. B. Gropp answers that none is, but a separate spec for a default interface could be added.

R. Lusk continues with the client-server discussion.

M Snir points out that handling the exceptions generated by failure (e.g., a process dies) needs handling out-of-band notification. Need to describe the properties that the out-of-band notifcation have.

A Elster points out that work with database servers may have relevent ideas.

B Gropp concludes with reference to the example programs in the proposal, and requests that alternatives also present sample programs.

4 Wednesday

R. Lusk begins day.

M. Snir chairs corrections and clarifications.

15 voters.

Behavior of test/wait. Two page description of four proposals distributed, with the fifth proposal to leave the standard unchanged. In general, these address the behavior on empty lists. Additional discussion about the proposals.

Quick summary

1. Original proposal of Snir. Changes syntax of `MPI_TESTSOME`. **6-8-3**
2. Proposal of Pierce. Does not change syntax of `MPI_TESTSOME`. **5**
3. Variant of 2, uses `MPI_UNDEFINED` for empty list. **7** Proposals 2 and 3 were voted together, with 6 abstentions.
4. Wait blocks on empty list. Does not require an additional test after testsome. **1-8-8**.
5. No change. **4-11-0**.

Proposal 3 passed.

Proposal 3 is:

Operation	Success	Failure	Empty list
WAIT	good status	blocks	null status
TEST	flag = T, good status	flag = F	flag = T; null status
WAITANY	good index	blocks	index = `MPI_UNDEFINED`
TESTANY	flag = T; good index	flag = F	index = `MPI_UNDEFINED`
WAITSOME	outcount > 0	blocks	outcount = `MPI_UNDEFINED`
TESTSOME	outcount > 0	outcount = 0	outcount = `MPI_UNDEFINED`
WAITALL	good statuses	blocks	null statuses
TESTALL	flag = T; good statuses	flag = F	flag = T; null statuses

R. Lusk introduces disccusions of three working groups, starting with one-sided ops, lead by M. Snir.

Introduces a put that includes MPI datatypes (for both source and destination), and a flag to indicate that transfer is completed (on the receiver).

```
put( source_buf, src_count, src_datatype, dest, comm, dest_buf,
dest_count, dest_datatype, dest_flag )
```

J Cownie raises issue of what the `dest_flag` is (boolean or request). Note that this is another address on the destination.

get is symmetric. There are non-blocking versions of these, as well as a read-modify-write operation. (full proposal distributed at meeting.)

T. Skjellum raises the issue of whether the communicator should contain information on what parts of the destination memory are accessible with put or get; he has a proposal presented at SC'94.

Some discussion about how the destintation of a put knows that the operation has completed; approaches mentioned include the `dest_flag`, use of barriers, or counters of operations.

One suggestion is for a persistent put/get (to eliminate argument checking and perhaps any underlying implementation issues).

W. Saphir asks whether adding a new model is a good idea (how will programmers know what to use) or whether this should be part of MPI. P Pierce mentions that the remote memory reference model fits some applications, even on message-passing system. B Gropp mentions that some applications may use message-passing to estabilish the addresses, then may use one-sided operations for the data exchange.

L Lewins mentions the desirability of distinguishing between sharable and non-sharable memory (such as marking it as non-cacheable).

Next, discussed hrecv and active messages. hrecv is a regular receive with a handler to be called, handler chosen by the receive. The active message in M. Snir proposal has a number of differences with "classical" active messages (pointed out by P Pierce and J Cownie). In particular, routine to call still specified by receiver instead of sender. The similarity is that the message is passed directly as arguments to the handler.

Other issues include what thread the handler runs in, can a polling implementation be used, what if there is a communication processor with a different instruction set, are there limits on what can be used in the handler, is there a limit to how much time/resources the handler uses?

Much discussion of the relative merits of active messages; one comment is that the real issue is what kind of "interrupt-driven receive" might be provided by MPI.

R. Lusk discusses the mechanism for discussion between meetings; suggestions that all discussions take place on `mpi-core@cs.utk.edu`.

Discuss scheduling. The hotel is available April 24–26, June 5–7, and July 26–28.

Next, Steve Lederman-Huss discusses the external interfaces working group.

Want to support layered libraries and tools

- Access to MPI opaque objects (derived datatypes, communicators, etc.)
- MPI internal system info (queues, etc). Very system specific; may be difficult to do. This is particularly for debugger support.
- Support of external interfaces, such as I/O, debuggers, external topologies, graphics, profiling and tracing tools.
- Also need to consider scope and solicit input from the community.
- Nonblocking operations and user-defined requests

Suggest that this group be named extensible interface issues, not external. L Lewins suggests that support for special tools (debuggers, libraries, etc) be segregated so that ordinary users are not burdened with a large number of routines intended for specialists.

Break

A Geist presents the collective extensions working group. Desire to limit the number of new routines. Currently under discussion

- Multisend or multicast; blocking and nonblocking versions; use list of destinations or a group.
- Nonblocking versions of the fourteen existing collective functions
- Collective wait and test on new request types
- New collective operations (combine and collect). Revisiting whether IN and OUT arguments are separated.
- Intercommunicator extensions for collective operations and topologies. An example is a broadcast from a root in the local group to the entire remote group. Proposal introduces few new functions; mostly generalizes the existing collective operations.

S. Lederman-Huss closes with the traditional MPI minus one comments, which met with much applause.

R. Lusk concludes by announcing that meetings confirmed for April 24–26, June 5–7, and July 26–28.

5 Attendees

alexho@almaden.ibm.com
babb@cs.du.edu
dongarra@cs.utk.edu
elster@slb.com
fineberg@nas.nasa.gov
foster@mcs.anl.gov
gdburns@tbag.osc.edu
golan@convex.com
gropp@mcs.anl.gov
gst@ornl.gov
hart@fsl.noaa.gov
hch@pallas.de
hempel@gmd.de
hender@fsl.noaa.gov
hlg@cray.com
ho@almaden.ibm.com
j-nieplocha@pnl.gov
jim@meiko.com
lederman@super.org

llewins@msmail4.hac.com
lusk@mcs.anl.gov
mcbryan@cs.colorado.edu
otto@cse.ogi.edu
par@cray.com
peter@usfca.edu
peterm@ncube.com
prp@ssd.intel.com
raja@tbag.osc.edu
rbarrett@lanl.gov
salo@sgi.com
sfleisch@convex.com
shuler@cs.sandia.gov
smith@epcc.ed.ac.uk
snir@watson.ibm.com
tony@cs.msstate.edu
walker@msr.epm.ornl.gov
wcs@nas.nasa.gov