Minutes of MPI meeting
June 5–7
Chicago IL

William Gropp

June 7, 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. Official 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

Rusty Lusk opens meeting with a discussion of the agenda.

Note that the mailing lists have moved to Argonne. The lists are now

mpi-core@mcs.anl.gov MPI Forum

mpi-comm@mcs.anl.gov General MPI

mpi-dynamic@mcs.anl.gov

etc

15 valid voters.

3 Final votes on MPI-1

Distributed a complete revision of MPI-1, with change marks around differences. Note: purely typographic errors will be fixed during final editing by Steve Huss-Lederman, Bill Gropp, and Rusty Lusk.

3.1 MPI Terms and Conventions

Vote on entire chapter: **15-0-0**

3.2 Point to Point

Some discussion about the introduction `MPI_ERR_PENDING` and when a multiple completion routine detects an error in some but not all of the pending communications. General agreement on the draft.

Page 49, lines 7–13, and page 47, lines 35–37 need revision.

Paul Pierce suggests changing "it succeeded" with "no request had an error". Paul also mentions that the wording seems to require certain behavior on failure, which is contrary to the MPI ground rules (no specification of behavior of erroneous programs). Bill Gropp mentions that the environment chapter has a general statement about this. General consensus is to fix this occurrence by using the text on page 47.

Next, consider the two meanings of the extent of a datatype. One is the location of the "highest" and "lowest" bytes, and one, set with the `MPI_UB` and `MPI_LB` elements in `MPI_TYPE_STRUCT`. A straw vote was taken to consider proposals to provide information about both the limits of the data in the datatype, and the mpi extent of the datatype. One proposal from Jim Cownie, adds three routines to get the limits ("span") of the data. Marc Snir's proposal changes the existing specification to make extent the limits of the data, and adding new routines to provide the existing functionality of using the MPI extent to control the stride between uses of the datatype. Lloyd Lewins describes his proposal which also changes existing definitions and introduces new routines to explicitly set the stride between successive instances of a datatype.

Call to vote on do nothing (for MPI-1, leaving open the possibility of additions in MPI2); replaced by several straw votes:

- Keep extent as is *12-6-1*
- Do nothing *6*
- Do something like `MPI_TYPE_SPAN` *enough*

What happens when the datatype has zero size? What does `MPI_TYPE_COUNT` give you? What if you sent 100 items with zero size?

Bill Gropp presents a proposal for 3.12.6 that lets Fortran use only the available bits in an `MPI_ADDRESS`:

Note that in Fortran, Fortran INTEGERs may be too small to contain an address (e.g., 32 bit INTEGERs on a machine with 64bit pointers). Because of this, in Fortran, implementations may restrict the use of absolute addresses to only part of the process memory, and restrict the use of relative displacements to subranges of the process memory.

This text is voted on and passes: **13-0-3**

Steve Lederman-Huss presents an advice to users that reminds them that `sizeof(MPI_DOUBLE)` is not the same as `sizeof(double)`. **6-6-?**. Fails.

Accept the entire chapter (except for proposals on extents/span): **15-0-2**

3.3 Collective Communication

Accept the entire chapter: **17-0-0**

3.4 Groups, Contexts, and Communicators

Accept the entire chapter: **16-0-1**.

3.5 Process Topologies

Accept the entire chapter: **17-0-0**.

3.6 MPI Environmental Management

Accept the entire chapter: **17-0-0**.

3.7 Profiling interface

Accept the entire chapter: **17-0-0**.

End early (!) and make plans for dinner.

4 Tuesday

Rusty begins with the agenda; planning of subcommittee meetings.

16 valid voters

Jim Cownie presents two new routines for MPI1 for getting the locations of the first and last byte represented in a datatype. This gives the lower bound and upper bound ignoring any `MPI_UB` or `MPI_LB`. These can be used to allocate enough space into which an element of the datatype can be received.

Some concern is express about making this a change to MPI1 rather than adding to MPI2. Paul Pierce remarks that the definition of user-defined operations on non-contiguous datatypes may not be what we intended, and fixing this should be left to a more comprehensive proposal in MPI2.

6-7-3

Bill Gropp and Lloyd Lewins suggest that the restriction against negative extents be removed, since an extent is really a stride, and a negative stride is meaningful.

To remove the restriction on negative extents: **11-1-4**

Final vote on the entire document **14-1-1**

5 Dynamic Process Management

We now began the MPI2 meetings, with a subcommittee of the whole on process management.

Rusty Lusk begins with a review of why MPI-1 did not include dynamic process management and why there is now interest in doing so. Reviews goals, which include

- Provide a simple interface
- Provide also a fully functional interface
- Address new requirements
- Postpone "scheduler request language"

Break

Bill Saphir opens discussion about the Gropp/Lusk/Saphir dynamic process proposal.

Divide into three steps

Resource Allocation logically external to application potentially complicated specification expensive

Process Management

Establish communication To create MPI jobs

Use `MPI_Resource` instead of `MPI_Node` in the proposal. This is a bag that contains processes.

- May have 0 to N processes (N may be infinite)
- N fixed at allocation time
- No internal structure except N and current number of processes running
- Process slots are reusable

Examples:

1. "My workstation." $N = \infty$ or $N = 1$.
2. "4 processors SMP." $N = \infty$ or $N = 4$.
3. "some (10) paragon nodes." $N = 10$.
4. "specific workstations" $N = 4$ or $N = \infty$.
5. "a promise from the job manager that can start 10 processes" $N = 10$

Note that a resource is a very flexible definition.

Introduced `MPI_Iallocate`. Presented examples of getting specific resources or bags of resources.

Some discussion about the resource descriptions. Tony Skjellum asks whether resources should be like groups; for example, comparisons or formation of sub-resources like sub-groups.

Bill Saphir presents a rationale for these resource objects

- flexible and simple
- separate resources can be used when an application requires them (for example, a specific HiPPI node or file server).
- Portability—allows several different models of computation. For example, a four processor SMP can be used either as four separate `MPI_Resource` or a single `MPI_Resource` that can run four (or more) processes.

Next issue discussed is ownership of processes. Current proposal makes resources owned by a single process. Alternatives include making the ownership collective. Concerns were raised about the possible loss of efficiency, flexibility, and scalability of not having a collective `MPI_Iallocate`.

Carl Kesselman raises the issue of whether the separation of allocation and execution may be difficult to implement (e.g., may require dynamic loading of executable code).

Move on to starting processes with `MPI_Exec`. This takes `MPI_Resource`s and returns `MPI_Process`es.

Note that `MPI_Exec` returns when a process is started; it does not wait until the processes completes (notice that this is unlike a Unix `exec`).

Concern is also expressed about what happens when processes exit. It is noted that some solutions in use have race conditions; in particular, notification by receiving a message can fail if the process exit happens between the test for the "failure" message and the attempt to communicate with the dead process.

Paul Pierce begins a discussion of what should happen when a processes exits unexpectedly. First question is notification. Possibilities are asynchronous notification and the existing MPI error method. First consider the MPI approach, so the error is indicated when an MPI routine uses certain routines that use the communicator (e.g., `MPI_SEND` but not `MPI_COMM_SIZE`).

Lunch, followed by subcommittee meetings.

6 Wednesday

Rusty Lusk opens with agenda for today.

6.1 Future meetings

We begin with a discussion of the next meetings.

Leslie Hart proposes a meeting in Boulder CO. Straw vote to hold MPI Forum meeting the week of October 23–27 passes.

Jim Cownie proposes a meeting in Vienna, Austria at the new computing center there. This would be in January, 1996. Straw vote to hold MPI Forum meeting the end of January passes. Will probably be January 24–26, 1996 (but not confirmed).

Andrew Lumsdaine presents the MPI developers meeting, to be held June 22-23, 1995, and University of Notre Dame.

7 Subcommittee reports

Note that the external resource subcommittee will remain within the dynamic process management subcommittee.

There will also be a separate mailing list, `mpi-leftovers@mcs.anl.gov`, for discussion of MPI-1 clarifications.

7.1 One sided

Pratap Pattnaik presented the current proposals.

Marc Snir's proposal. Three main operations:

- RMC (Remote Memory Copy)
- RMW (Remote Memory Write)
- Commnicate and Call.

Only the first was discussed in the subcommittee meeting.

Presented a non-blocking put operation. Some of the issues include

- What is the coherency requirement?
- How much of the remote memory is exposed to remote access and how is it specified? One suggestion is that there be an `MPI_TARGET` routine to indicate explicitly the memory that can be used as a target of a put operation.

Lloyd Lewins proposal has a `MPI_Shared_create`, which is a collective call. This does not allow arbitrary access to local memory; rather, it provides a sort of shared memory `malloc`.

- When will call fail?
- Asymmetry among processes?

- Should there be shared objects instead?
- Is this an alternative to the Snir proposal or an addition?

7.2 Resource Specification

Rusty Lusk presented the deliberations of the subcommittee on resource specification. This is the subcommittee looking at how resources are specified to the dynamic process management routines. Issues include

- Request for N Whats? Nodes? Virtual nodes? Hosts? CPUs? What is allocated is a license to start processes.
- Scheduler inquiry as well as requests.
- Common cases must be simple.
- MPPs vs. NOWs.

7.3 External interfaces

Steve Huss-Lederman presented the work on generalized requests. Issues include

- Need to protect user functions from simultaneous calls. One solution is to make operations atomic. That is, when MPI is calling a user-specified routine, it will queue any user-requests that could be started, rather than interrupting the user routine to invoke another user routine. Alternatives include using locks. Still under discussion.
- Make sure that the state (system+user) is unique to request/subrequest.
- Need to get count and index straight. Is it global for all calls in request? User may need to know global index to know what finished.
- What happens if the user routine fails?
- Can user cancel request?
- Can this be used to create an `hrecv`? Do we want to relate them?
- wait/test return send/recv type status. What values do we really want and how do we deal with them?

Much concern about the difficulty in implementing this. Also mentioned is that a solution (and the MPI-1 position) is that at least the `hrecv` is just an `MPI_Recv` in a separate thread, and should be left to the thread library.

Another issue is access to handles, particularly in interlanguage environments.

- Add Fortran versions?
- How to deal with user values/routines associated with handles (example, attributes and callback functions). Warn user? Remove them? Signal error?
- Other languages (C++, Fortran 90, etc.)

Also mentioned is getting access, in a single language environment, to internal MPI data. For example, a context id for use by a visualization system. Another is user access to derived datatypes. A proposal produces an ASCII string representing the derived datatype.

- Like the idea. Check what stored in current implementations to make sure that the set is complete.
- Compare to how LAM does this. Others?
- Restrict to once datatype is committed.
- Provide function to take ASCII representation and get MPI datatype (inverse).
- Provide function to get datatype id without decode. Allows quick check if seen before.

Lloyd Lewins made some other proposals which were not discussed. These were "get global rank" and "provide context id". Noted that the existing translate ranks routine, using `MPI_COMM_WORLD` provides 'get global rank". The problem with "provide context id" is that an implementation may not have a context id and would suffer a performance hit if one was mandated.

7.4 Real time

Tony Skjellum presented the realtime MPI subcommittee report.

Motivations

- COTS initiative
- Potential of MPI to MPI-lite (standard concept, elements of standard) to impose practice in segments of RT community.
- Some RT applications are moving to MPI

Things to note

- Common practice not strong basis for standardization
- Big theory-practice gap

- Big opportunity to make improvements

What is realtime MPI-2?

- MPI-2 with additional promises (semantic restrictions, quality of service, features to support application development and porting)
- A completely new message passing system, developed under MPI-F auspices, designed specifically for realtime applications
- Compromise between the above

Includes

- Efficient
- Quality of service (timing, scheduling, performance, verifiability, access to realtime controlling mechanisms (time driven, event driven, priority based))
- Huge range of operating environments

Possibilities

- Soft realtime/Large memory. Want send/recv with timeout, RTP-like semantics, priority messages.
- Soft realtime/Limited memory. Want subset of above, memory conscious.
- Hard realtime, want strong guarantees to schedule everything

Two starting points

1. Soft realtime, Medium grain. MPI-2 + some features and priorities.
2. Fine Grain, memory limited. Less than MPI-1; some of MPI-2.

Some additional discussion about what "soft realtime" and "hard realtime" mean. Distinction between best effort, data that has no value after a deadline, and whether meeting deadlines are critical or not.

Subcommittee agreed to prepare draft chapter for the next meeting, and to try and get more representatives from the realtime community (such as tool builders and vendors).

Mailing list will be `mpi-realtime@mcs.anl.gov`.

8 MPI -2

Steve Huss-Lederman closes with the traditional `MPI --`.

9 Attendees

Partial list:

Name	Institution	email
Joel Clark	Intel	joel@ssd.intel.com
Lyndon Clarke	EPCC	lyndon@epcc.ed.ac.uk
Jim Cownie	Meiko	jim@meiko.co.uk
Raja Daoud	OSC	raja@tbag.osc.edu
Nathan Doss	MSU	doss@erc.msstate.edu
Ian Foster	ANL	foster@mcs.anl.gov
Robert George	Army Research Lab	george@cs.nps.navy.mil
Bill Gropp	ANL	gropp@mcs.anl.gov
Leslie Hart	NOAA/FSL	hart@fsl.noaa.gov
Steve Huss-Lederman	CCS	lederman@super.org
Arkady Kanevsky	MITRE	arkady@mitre.org
Karl Kesselman	Caltech	carl@compbio.caltech.edu
Susan Kraus	NEC	kraus@hstc.necsyl.com
Lloyd Lewins	Hughes	llewins@msmail4.hac.com
Andrew Lumsdaine	Notre Dame	lumsdaine@nd.edu
Rusty Lusk	ANL	lusk@mcs.anl.gov
Peter Madams	nCUBE	peterm@ncube.com
Jarek Nieploche	PNL	j_nicploche@pnl.gov
Pratap Pattnaik	IBM	pratap@watson.ibm.com
Paul Pierce	Intel	prp@ssd.intel.com
Peter Rigsbee	Cray Research	par@cray.com
Eric Salo	SGI	salo@sgi.com
Bill Saphir	NAS/NASA Ames	wcs@nas.nasa.gov
Tony Skjellum	MSU	tony@cs.msstate.edu
Jeff Squyres	Notre Dame	squyres@nd.edu
MaryDell Tholburn	LANL	marydell@lanl.gov