Minutes of MPI meeting
September 6–8
Chicago IL

William Gropp

September 15, 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 Wednesday

Rusty Lusk opens meeting with a discussion of the agenda.

2.1 Dynamic Processes

Al Geist begins discussion of dynamic process management with a review of resource management in MPI. Jim Cownie expresses concern about the resource ownership; in particular, the resources may be owned only by rank 0 in `MPI_COMM_WORLD`. Much discussion about the preallocated resources returned by resource discovery. Some additional concern about the scope of the resource management; Rusty Lusk reminds us that the original proposals were more general and the committee had backed away from that generality.

Moved on to discussion of process manager. The principle issue is whether the started process is an MPI process or not, and whether an MPI process is connected to the process that started it. A recent addition is the `MPI_NOTIFY` which specifies a way for a process to be notified when a child dies or some resource fails.

Then moved on to the section on "attaching independant processes;" this used to be the client-server section. Two issues came up. The first is the absence of the nonblocking versions present in the earlier proposals. The second is the issue of fault tolerance in terms whether these routines can be used to implement general client-server systems or just applications written by a single user and that have more limited reliability requirements.

Jim Cownie asks that Al Geist comment on how the PVM experience affected the MPI proposals. Al comments that one issue is that in PVM, all groups are dynamic and can change at any time, making it difficult (and expensive) to identify a process by `(group,rank)`. This led to a discussion of whether there should be `MPI_Groups`. Marc Snir asserts that any MPI-1 program can be re-written to not use `MPI_Group` functions, relying on `MPI_Comm_split` etc.

Closing the circle, we return to the two proposals for resource management. Proposal 1 has a resource description; proposal 2 leaves the description opaque to MPI.

3 Thursday

Subcommittee meetings today with parallel sessions: one-sided and real-time in the morning, collective and binding in the early afternoon, and external and dynamic processes in the late afternoon.

In the one-sided subcommittee meeting, led by Marc Snir, there was much discussion on the need and implications of an `MPI_HRECV`. The alternatives are either threads or generalized requests. After some discussion, straw votes were made to test the interest in the more direct forms. The results were: For an `MPI_HRECV`: *9-5-10*, and for active messages in MPI: *5-6-13*.

After this, discussion turned to the put and get operations. An initialization function to control the part of memory (the window) that is accessable to remote processes has been proposed. A major issue is whether the window is specified as a contiguous array of memory or by an arbitrary MPI datatype (allowing the specification on windows with holes). This led to much discussion on the use of datatypes in a heterogeneous environments, and how data would be converted in one-sided operations. In particular, this seems to mandate either a sender (on put) or receiver (on get) conversion of data, or the transmission of a type signature as part of the operation, where MPI message-passing deliberately allows the implementation to decide when the

Are windows always untyped (no datatype on windows) *0*? Are windows always typed *4* or is there an escape that also allows untyped windows *13*?

Break.

Marc Snir reopens the one-sided discussion by raising the issues of how general the description of the source and destination buffers, particularly when combined with general MPI datatypes for scatter/gather operations. The problem for a remote buffer is that the MPI datatype is most naturally defined at the local process. Alternately, it might be defined at the remote process, possibly requiring additional communication. Paul Pierce notes that the specification of the buffer on a heterogeneous system has related difficulties.

4 Friday morning

(Notes taken by Rusty Lusk and edited by Bill Gropp.)

The Next meeting in Denver, Oct 23-25, at the Executive tower Inn; You need to reserve 2 weeks before as MPI Forum.

The Vienna meeting is Jan 24-26, meeting all day Thurs-Fri.

4.1 Subcommittee reports

4.2 Report from One-sided

Report presented by Marc Snir.

Discussion of typed windows, will probably require typed windows plus escape remote datatypes.

remote address buffer address data type

reminder: `MPI_Aint` not available in Fortran

Discussion of hrecv -heavyweight (handler can call MPI routines, etc.).

Discussion of amrecv - lightweight, performance-oriented but with limited functionality.

4.3 Report from Dynamic

Al Geist presented.

Still have resources, process, connections

Major changes from existing draft on resources.

- No resource ownership
- pre-allocated visible to all.
- anyone can spawn
- single resource pool.
- new allocations go into pool

Resource(object, description, license, thing) needed?

Needed–a way to find out about resources ability to construct description to pass to spawn.

Process management: same basic functions. Three start cases: need 3 functions or a flag to one function

1. non-MPI

2. MPI no talk to parent

3. MPI talk to parent

Much objection to this, except distinguish between talk and no talk (can do with two functions in existing draft)

Signals - Posix with comment that at least KILL be supported if system isn't Posix compliant

Process Object s (group,rank) to identify tasks

Discussion of `MPI_Process` vs. (Group, rank)

(group,rank)	`MPI_Process`
Pro:	
already exists	natural abstraction
nice syntax	no aliasing
process group	cache attributes
Con:	
groups unnecessary	new object
groups with holes	array + count inconvenient
aliasing	
process groups don't overlap	
would need new function to compare processes	
	would need translation functions

Straw vote *14-6* in favor of group-rank representation.

Connecting/communicating: no changes from draft

4.4 Report from Collective

Report from Steve Huss-Lederman.

Cancel - should collective operations have cancel like pt2pt? Choices:

- global operation: clean up on all processes - hard
- not allowed: no fault tolerance, symmetric with pt2pt
- local operation: free all resources on process logically equivalent to cancel of each pt2pt, logically equivalent to cancel of each pt2pt can easily hang code if not called by all processes

Persistent collective operations considered good.

mcast: current proposal seems to have lost ability to do a list of processes. whole committee thinks this is OK.

Use different names for inter-collective since different semantics.

No tags in non-blocking, otherwise can't easily mix blocking/non-blocking

4.5 Report from misc/external

Report from Steve Huss-Lederman.

Interlanguage: want better C-Fortran interaction in MPI-1

min: can translate handle to other languages max: init, attributes,.. will need to merge 3 proposals for next time. Generalized requests: do with hrecv, hsend, hwait

Datatype compacting: do users need? layer?

Handle caching: add handle type to call for Fortran

Callback: can do without if have handle caching

Portable MPI debuggers: want access to msg state, datatypes, comms, options: 1. no additions to implementation: hard/impossible 2. keep "shadow" info for debugger with special library, adds expense 3. add decode functions (what if break at awkward point?) Plan: pursue 2 and 3

Communicator id's wanted but no good way yet

4.6 Report from binding

Presented by Nathan Doss

Discussed current draft. C++ class libraries part of standard? Will propose one for next meeting.

4.7 Report from Real-time

Presented by Lloyd Lewins

The goal is to provide predictability, guaranteed timing performance, and portability.

The approach is to provide hooks so that application (or middleware) can schedule communication over the network resource.

Three commonly used paradigm:

- time driven
- event driven
- priority based

Subcommittee discussed time-driven in most detail. Assume scheduler is separate from MPI, e.g. Maruti OS. (similar to approach to threads in MPI-1)

Some ideas of changes:

1. Globally synchronous clock. Need to know maximum skew (attribute)
2. Earliest start time for operation: `MPI_send(..., start_time,...)`
3. timeouts for operations: `MPI_Wait-delay(..,timeout,...)` (combines wait and test) (`waitall`, `wait_some`, `wait_any`, probe)

5 Attendees

Name	Institution	email
Robert Babb		
Ron Brightwell	MSU	
Dennis Cottel		
Joel Clark	Intel	joel@ssd.intel.com
Jim Cownie	BBN	
Laurie Costello	Cray	
Raja Daoud		
Nathan Doss	MSU	doss@erc.msstate.edu
Stephen Fleischman	Convex	sfleish@convex.com
Al Geist	ORNL	gst@ornl.gov
David Greenberg	Sandia	dsgreen@cs.sandia.gov
Bill Gropp	ANL	gropp@mcs.anl.gov
Leslie Hart		
Hans-Christian Hoppe	PALLAS	hch@pallas.de
Steve Huss-Lederman	CCS	lederman@super.org
Suresh Damodovan Kamal		
Arkady Kanevsky	MITRE	arkady@mitre.org
Susan Kraus	NEC	kraus@nstc.necsyl.com
Lloyd Lewins	Hughes Aircraft	llewins@msmail4.hac.com
Andrew Lumsdaine	Notre Dame	lumsdaine.1@nd.edu
Rusty Lusk	ANL	lusk@mcs.anl.gov
Nicholas Nevin	OSC	
Peter Pacheco		
Pratap Pattnaik	IBM	pratap@watson.ibm.com
Paul Pierce	Intel	prp@ssd.intel.com
James Pruyve		
William Saphir	NAS (NASA Ames)	wcs@nas.nasa.gov
Marc Snir	IBM	
Jeffrey Squires	Notre Dame	
David Taylor	TMC	taylor@think.com