MPI-3 Coll Workgroup

Status Report and Discussion

MPI Forum

Torsten Hoefler

Indiana University

Sept. 3rd 2008

Dublin, Ireland

Agenda

1) interface progress
2) sparse collective operations
3) non-blocking collectives
4) persistent collectives
5) collective plans
6) dynamic-size collectives

High-level Interface Decisions

Last Forum – Discussion about:

Option 1: ”One call fits all”

vs.

Option 2: ”Calls for everything”

→ we decided to wait, but favored option 2

Can we decide yet?

Last Telecon:

- converged on calls for everything
- With heavy pruning (don't need all functions)
- Didn't decide what to prune yet

Are there any new opinions?

Discussion

Handling Sparsity

Do we really need sparsity for all Collectives?

- only useful if subgroups change fast (otherwise one could create a new communicator)
- optimization potential is limited (need to calculate new communication schedule for each call)
- maybe a subset of operations and some new operations are sufficient?
- do we have an application-driven demand?

Progress at last telecon?

Last Telecon:

- Decided to drop the interface
- Can be implemented by libraries

→ Discussion?

Topological/Sparse Collectives

- Option 1: use information attached to topological communicator
- MPI_Neighbor_xchg(<buffer-args>, topocomm)

- Option 2: use process groups for sparse collectives
- MPI_Exchange(<buffer-args>, sendgroup, recvgroup)

  (each process sends to sendgroup and receives from recvgroup)

Option 1: Topological Collectives

Pro:

- works with arbitrary neighbor relations and has optimization potential (cf. "Sparse Non-Blocking Collectives in Quantum Mechanical Calculations" to appear in EuroPVM/MPI'08)
- enables schedule optimization during comm creation
- encourages process remapping

Con:

- complicated (?) to use (need to create graph communicator)
- dense graphs would be not scalable (are they needed?)

Option 2: Sparse Collectives

Pro:

- simple to use
- groups can be derived from topocomms (via helper functions)

Con:

- need to create/store/evaluate groups for/in every call
- not scalable for dense (and large) communications

New Neighbor Collectives 1/2

MPI_Neighbor_Exchange(v)(<buffer-args>, <comm-or-groups>)

- neighbor exchange
- comm would be (directed) topology communicator
- groups would be sendgroup and recvgroup

MPI_Neighbor_reduce(sbuf, rbuf, count, datatype, op,
<comm-or groups>, comm)

- works with directed graphs (send- and recvgroup)?

New Neighbor Collectives 2/2

MPI_Neighbor_bcast(buf, count, dtype, <comm-or-group>

- broadcasts data to all neighbors

MPI_Neighbor_xxx()

- Do we want more?
- what do users want → add to survey (with explanation)?

Non-blocking Collectives

The best understood MPI-3 issue in the working group!

Vendors and groups start experimental implementations.

We need to begin to make some decisions!

Non-blocking Collectives/Issues

Tags or no Tags?

- we currently want tags for matching clarity and debugging
- is there a performance penalty?
- Implementation in LibNBC would be simple (just use the tag by the user with offset instead of own tags)
- tag-range possibly smaller than p2p (32k) because we need 16+ different ”tag-spaces”

Non-blocking Collectives/Issues

MPI_Requests or no MPI_Requests?

- the group says ”yes”
- we already define two classes of requests (Generalized Requests and P2P Requests)
- are there other opinions?

Non-blocking Collectives/Issues

Multiple outstanding requests or not?

- I heard: ”we don't want multiple outstanding colls because we want to control the network, messages and congestion etc.”
- One does not need to start them:

```
if(!collective_running) {
    start_collective();
    collective_running = 1;
} else {add to list}
```

→ collective_running will be reset if collective completes

Non-blocking Collectives/Issues

Which Prefix?

- forum rejected ”I” (may be confused with p2p)
- group rejected ”A” (it's not necessarily asynchronous)
- new proposal: ”N”, e.g., MPI_Nbcast()
- Better proposals?

Non-blocking Collectives/Issues

Are there any other issues?

Persistent Collectives/Issues

MPI_Startall() ?

- another pro for tags
- in which order do similarly tagged colls match?
- how is it done in the p2p case (not at all)?
- match in ”array-order” or make the operation illegal?

Collective Plans/Schedules

- can we find a better name?
- act as expert interface for advanced users or ...
- ... compilation target
- → Christian (I'll have a different interface)

Collective Plans/Schedules

Oblivious Interface (opaque object):

- MPI_Sched_create(MPI_Sched *sched, MPI_Comm comm)
- MPI_Sched_send(<sendargs>, MPI_Sched *sched, int *id)
- MPI_Sched_recv(<recvargs>, MPI_Sched *sched, int *id)
- MPI_Sched_reduce(<opargs>, MPI_Sched *sched, int *id)
- MPI_Sched_depends(int cause, int action, MPI_Sched *sched)
- MPI_Sched_init(MPI_Sched *sched, MPI_Request *req)

Possible ”helper” functions:

- MPI_Sched_copy(<copyargs>, MPI_Sched *sched, int *id)
- MPI_Sched_pack(<packargs>, MPI_Sched *sched, int *id)
- MPI_Sched_unpack(<unpackargs>, MPI_Sched *sched, int *id)

Collective Plans/Schedules

Advantages:

- seems intuitive to represent a dependency graph
- parameter checking when schedule call is made
- user doesn't need to store items
- less new types (smaller F77/90 interface)

Dynamically-Sized Reductions

- current reductions are fixed-size
- many operations not possible (e.g., compression, string concatenation ...)
- language bindings would benefit from new reductions
- would enable Map/Reduce implementations

→ how do you feel about this?

Dynamically-Sized Collectives

Does anybody speak up for this?

More Comments/Input?

Any items from the floor?

General comments to the WG?

Directional decisions?

How's the MPI-3 process? Should we go off and write formal proposals or wiki pages?