Collectives Working Group
– April’08 Report and Discussions –

Torsten Hoefler and Andrew Lumsdaine

Open Systems Lab
Indiana University
Bloomington, USA

3rd MPI Forum Meeting April

Chicago, IL, USA

April, 28-30th 2008

Topics

- Topological/Sparse Collectives (Jesper, Torsten)
- Non-Blocking Collectives (Torsten)
- Persistent Collectives (Jesper, Torsten, Christian)
- Dynamic-sized (Vector) Collectives (Hans-Joachim, Alexander)

All-in-one sentence

We will propose a new interface that is able to handle topological/sparse, non-blocking and persistent collective operations and only adds one new interface function per collective.

Topics

- Topological/Sparse Collectives (Jesper, Torsten)
- Non-Blocking Collectives (Torsten)
- Persistent Collectives (Jesper, Torsten, Christian)
- Dynamic-sized (Vector) Collectives (Hans-Joachim, Alexander)

All-in-one sentence

We will propose a new interface that is able to handle topological/sparse, non-blocking and persistent collective operations and only adds one new interface function per collective.

All-in-one Interface

- MPI_Bcast_init(..., group, info, request)
- supports:
  - non-blocking
  - sparse/topological
  - persistent
  - multiple optimization possibilities
- several open issues, for example:
  - tags?
  - ordering in startall?
  - re-using MPI_Requests?
  - ... some more
- $\Rightarrow$ join our discussions on Wed. 9:30am

All-in-one Interface

... jointly developed and proposed by Jesper Larsson Traeff (NEC) and Torsten Hoefler (Indiana University)!

- MPI_Bcast_init(..., group, info, request)
- general:
  - _init calls are collective (also if rank is not in group)
  - _init calls can involve communication or not
- the MPI_Group argument:
  - assembled process-local
  - (in/out) data-ordering is determined by order in group
  - must be identical on all ranks
- the MPI_Info argument:
  - allows hints to the implementation
  - e.g., can the _init call be collective?

Info Hints

- **coll_init**: _init call can be collective (enables collective schedule optimization)
- **no_coll_init**: force _init call to be local (reduces number of synchronization points)
- **non-blocking**: optimize for non-blocking usage (overlap computation)
- **blocking**: optimize for lowest latency in the blocking case (no overlap needed)
- **reuse**: similar arguments will be reused (e.g., group and sizes stay identical, only addresses are changed)
- **previous**: look for a similar operation in cache

Topologies

- enable optimized process mapping
- changes to enhance scalability:
  - MPI_Graph_create will only accept a neighbor list
  - represent more general directed graph (change in MPI-2.1?)
  - query functions will not have rank argument
- group query functions:
  - get a neighbor group from a communicator (for sparse collectives)
  - convenience function to encourage graph/cart usage
  - MPI_Cart_neighbor_group(selected_dims, distance, comm, group)
  - MPI_Graph_neighbor_group(comm, group)

Open Questions

- tags?
- using MPI_Requests?

Dynamic-sized Collectives

Dynamic-sized Collectives

Hans-Joachim, Alexander?