Treating threads as MPI processes thru registration/deregistration

Alexander Supalov

Intel Corporation

Contents

- Rationale
- Proposal
- MPI_COMM_THEAD_REGISTER bindings
- Example: OpenMP parallel section

Rationale

- Increasing prevalence of multi- and manycore processors calls for extended MPI facilities for dealing with threads as first class MPI entities.
- So far the MPI standard treats threads as second-class citizens that cannot be directly addressed and whose presence is merely tolerated.
- This leads to issues like the Probe/Recv consistency issue described in https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/38 and partial solution proposed therein,
- This proposal seeks to introduce a powerful and convenient way of direct addressing of the threads as MPI processes.

Proposal

- A new collective routine **MPI_Comm_thread_register()** is introduced to create a communicator in which existing threads become MPI processes with unique ranks.
  - The way in which the existing threads were created or are going to be terminated is out of the scope of this proposal. See Posix threads, OpenMP, etc.
- The existing routine **MPI_Comm_free()** is extended to operate on the resulting communicators.
- All power of MPI is retained. For example:
  - All communicator and group manipulation routines can work on the resulting communicators recursively, combining and rearranging processes and threads as needed to the user.
  - All communication routines (pt2pt, collectives, 1-sided, file I/O) can work on the new communicators, and may optionally take advantage of the fact that the data should not cross the process boundary.
- Prerequisite: MPI_THREAD_MULTIPLE thread support level.

MPI_Comm_thread_register (language invariant binding)

MPI_Comm_thread_register(comm,
  local_thread_index, local_num_threads, newcomm)

IN comm	original communicator
IN local_thread_index	index of the calling thread (0 to local_num_threads – 1) on the current MPI process in comm
IN local_num_threads	total number of threads issuing this call on the current MPI process in comm
OUT newcomm	new communicator based on threads

MPI_Comm_thread_register (basic language bindings)

C:

```
int MPI_Comm_thread_register(MPI_Comm comm,
  int local_thread_index, int local_num_threads,
  MPI_Comm *newcomm)
```

Fortran:

```
MPI_COMM_THREAD_REGISTER(INTEGER COMM,
  INTEGER LOCAL_THREAD_INDEX, INTEGER
  LOCAL_NUM_THREADS, INTEGER NEWCOMM,
  INTEGER IERROR)
```

Example: OpenMP parallel section

```
!$OMP parallel num_threads(4)
call MPI_COMM_THREAD_REGISTER(       &
MPI_COMM_WORLD,OMP_GET_THREAD_NUM(),
        & OMP_GET_NUM_THREADS(),NEWCOMM)
!
!       Whatever MPI operations on and in NEWCOMM
!
        call MPI_COMM_FREE(NEWCOMM)
!$OMP parallel end
```