MPI3: Migrating Threads Across MPI Processes

Hybrid Programming Working Group

December 13, 2011

0.0.1 Rationale

MPI implicitly assumes that a thread and all its children are in a fixed association with one MPI processes. TThis assumption is natural when an MPI process corresponds to one address space, but is less natural when multiple endpoints reside whithin the same address space. Consider, for example, an OpenMP program that uses multiple MPI endpoints: It is natural for the program to start and end with only one thread running; in order to use multiple endpoints, the other threads should be associated dynamically to those. This section adds an addtiioon function to achieves this purpose. It also slgithly increase the flexibility of MPI intialization and completion, to handle a more dynamic environment.

0.0.2 Thread Migration

The function MPI_THREAD_ATTACH can bus used to attach a thread to an endpoint.

MPI_THREAD_ATTACH(rank, comm)

IN	rank	rank of endpoint (integer)
IN	comm	communicator containing endpoint (handle)

```
int MPI_Thread_attach(int rank, MPI_Comm comm)
```

```
MPI_THREAD_ATTACH (RANK, COMM, IERROR)
    INTEGER RANK, COMM, IERROR
```

The thread performing this call detaches from the current endpoint it is attached to and attaches to the endpoint identified by the arguments comm and rank. The call is erroneous if the new endpoint is not in the same address space as the old endpoint. It is also erroneous if the thread support level is MPI_THREAD_SINGLE and there already is a thread attached to the target endpoint.

> *Advice to implementors.* An implementation that supports multiple endpoints within the same address space must maintain the association of threads to endpoints. For example, it can store in a thread-private location a pointer to the data structure associated with this endpoint. This information is inherited fro parent to child. In addition, it can be modified by a call to MPI_THREAD_ATTACH. Noite that this introduces one addtional memory access for each MPI call. (*End of advice to implementors.*)

Bibliography