Chapter 8 MPI topic: One-sided communication

8.1

Window: An area of user-space memory which is declared that is accessible to other processors.

Other processes can only ‘get’ data from a window or ‘put’ it into a window; all the other memory is not reachable from other processes.

Window’s characters:

a) the window is defined on a communicator, so the create call is collective;

b) the window size can be set individually on each process. A zero size is allowed, but since window creation is collective, it is not possible to skip the create call;

c) the datatype can also be set individually on each process. This makes it possible to use a derived type on one process, for instance for copying strided data inyo a contiguous buffer;

d) the window is the target of data in a put operation, or the source of data in a get operation;

e) there can be memory associated with a window, so it needs to be free explicitly.

8.2

Epochs

The process of letting the target know the state of affairs is called ‘synchronization’.

MPI_Win_fence, the interval between two fences is known as an epoch. There can be only one remote process that does a PUT; multiple ACCUMULATE accesses are allowed. As a further restriction, you cannot mix GET with PUT or ACCUMULATE calls in a single epoch.

8.3

PUT, GET, ACCUMULATE

MPI_Put routine is used to put data in the window of a target process.

MPI_Get routine is used to get data in the window of a target process.

MPI_Accumulate

8.4

MPI_Win_post

MPI_Win_wait

MPI_Win_start

MPI_Win_complete

Questions

a) how can the recipient know the data I want to put to the window has already been put?

b) can I call an MPI_Get and an MPI_Put within an epoch?

Questions

1. Window: An area of user-space memory which is declared that is accessible to other processors.

2. Only one.

Exercises

8.2 Please see the attached code.