Homework 2: MPI concepts

George Bosilca

September 9, 2020
Matrices are usually distributed using regular 2D block/cyclic distributions over a process grid of \( P \times Q \) processes. Use the minimum number of MPI functions to create the row and column communicators (assuming the number of processes per row, \( P \), is provided) as indicated on the picture on the right.

Once you have the correct set of communicators, use the token propagation function from the first homework to move data across each row communicator (red communicators in the picture).

Let’s extend the token movement function, to allow each process to start with a token at the beginning. Upon calling `shift_next` the local token moves to the next rank, and the process receive the token from the previous rank. The API should be

```c
int shift_next(void* sendbuf, void* recvbuf, int count,
               MPI_Datatype type, MPI_Comm comm)
```
Matrix Transpose with MPI

Assume a two dimensional array of types distributed using a block, block pattern across a \( P \times Q \) process grid. Write a function that will transpose the matrix storage, using mostly MPI functions. This means you are not allowed to transpose anything by hand, but instead you should focus on creating description of the memory layout (MPI datatypes) and let the MPI library do the transpose for you.

Also, the transpose should happen in place, but you can use temporary storage.

The expected API of the call is `transpose(void* matrix, int P, int Q, int N, int M, MPI_Comm comm)` where

- `matrix` is the pointer to the local part of the matrix (and should point to a memory area large enough to be able to store all elements the process will need to receive).
- `P` number of processors in the first dimension. the second dimension can then be computed using \( P \) and the size of the communicator.
- `N, M` the size of the first and second dimension of the matrix.
- `comm` the communicator the transpose operation will happen