Laplace - MPI

Parallel Programming

2D Laplace Equation

```
#include ...
#define  ...
int main(int argc, char *argv[])
{
...
   laplace_matrix_creation (A, Anew, n, m)
   laplace_init (A, n, m);

  while ( error > tol && iter < iter_max )
  {
    laplace_step (A, Anew, n, m);

    error = laplace_error (A, Anew, n, m);

    laplace_copy (Anew, A, n, m);
   }

   print_results();
   free (A, Anew);
}
```

2D Laplace Equation

```
MPI_Init()
Comm_Rank(&me)
Comm_Size(&nprocs)
Determine initial (first_row) and final (final_row) rows for each process
Preparation of matrix A and Anew.
laplace_init (A, n, m)
while error > tol && iter < iter_max  do
    if me > 0 then
        send( A[first_row], me - 1)
        recv( A[first_row-1], me - 1 )
    if me < nprocs-1 then
        send( A[final_row], me + 1)
        recv( A[final_row+1], me + 1)
    laplace_step( A, Anew, first_row, final_row );
    my_error = laplace_error( A, Anew, first_row, final_row );
    Interchange and determine global error
    laplace_copy( A, Anew, first_row, final_row )
MPI_Finalize();
```

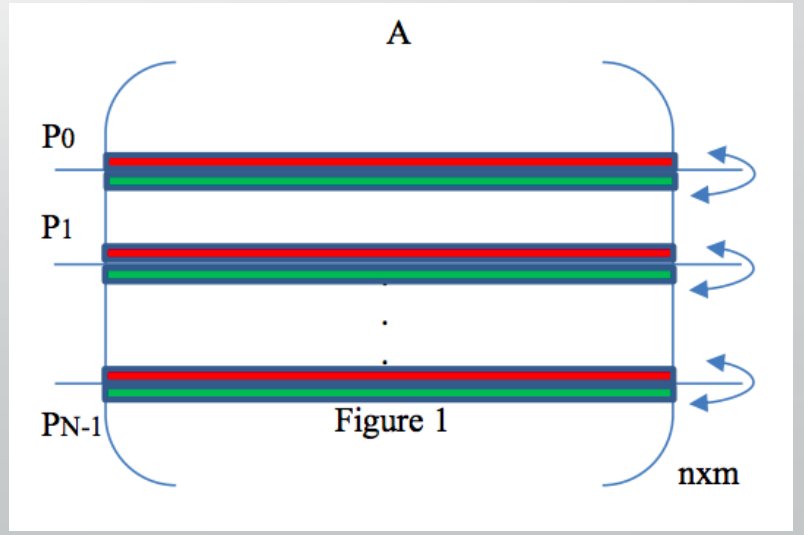

Figure 1

2D Laplace Equation

Assumption: Number of rows is divisible by number of processes

Each process Pi takes care of the computation of n/N rows of the matrix

Each process needs the last row from the previous process and the first row of the next process

Note:
- Take care about the process 0 (zero) and the last process (nprocs– 1)

- Take care about possible deadlocks while using blocking communication

- Take care about the size of matrix and where the initial and final rows start

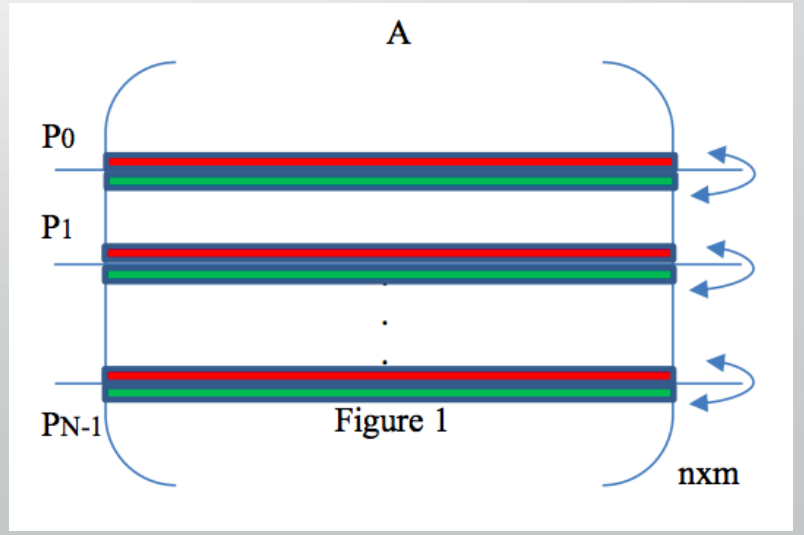

Figure 1