MPI parallelization of convolution computation

IN4200 - Home exam 2, Spring 2021

Information on the files, and how to compile them

Here is a brief overview of the files

- `MPI_double_layer_convolution.c` Performs the double layer convolution on the input matrix, contains the function `MPI_single_layer_convolution` , this function is called inside the `MPI_double_layer_convolution` function
- `MPI_main.c` Main file, shows how the `MPI_double_layer_convolution` function is used
- `functions.h` Header file, contains all the functions. There are also some printing functions that I used while I was developing the code, however these functions are not called when testing with larger values of M and N. This file also contains the serial function `single_layer_convolution` , which is used when checking if the parallel and serial execution give the same output matrix in MPI_main.c
- `makefile` Makefile to make it easier to run the codes

Code compilation

To compile and run the codes, you can use the makefile. Use the command

```
make main_mpi
```

The command line arguments are of the form `mpirun -np <tot_proc> <M_rows> <N_col> <K1> <K2>` If you wish to change the command line arguments, you have to go into the makefile and type them in, they are given as shown below

```
/usr/lib64/openmpi/bin/mpirun -np <tot_processes> ./a.out <M_rows> <N_columns> <K1_size> <K2_size>
```

Implementation details

In this assignment, we have to implement a parallelized implementation of convolution, using MPI.

To implement the parallelization in the `MPI_single_layer_convolution` function, I have used the `MPI_Scatterv` function. Here I have scattered the input matrix with overlapp, this way each process will have all the necessary information to compute its part of the output matrix. In the convolution computation, the center is chosen to be the top leftmost element of the kernel(i.e. element [0,0] of the kernel). This way I can find the overlapping rows to be K-1 (where K is the size of the kernel). When I scatter the matrix I add K-1 rows to each process to add the additional rows needed for the computation. Once the convolution computation is done, I use `MPI_Gatherv` to gather the output from each process onto the root process. To check if my parallel implementation is correct, I have also implemeted a serial version. After the parallel computation on the input matrix, I do a serial computation and check if the two output matrices match.

Test output

Here are some examples of my code, I have shown the output I get for some different values of M,N,K1 and K2, as well as the number of processes. If my implementation is correct, then the error will be 0. All the output shown here was produced by running my code on ifi's linux servers.

- **Example 1**

```
[terminal]$ make main_mpi
/usr/lib64/openmpi/bin/mpicc MPI_main.c MPI_double_layer_convolution.c
/usr/lib64/openmpi/bin/mpirun -np 15 ./a.out 8 8 3 3

The input values for the parameters are
---------------------------------------------
M(rows)=8, N(columns)=8, K1=3 and K2=3 on rank=0

Checking the computations on rank 0
----------------------------------------
The error in the output matrix, from the
serial and parallel execution is
Error  =  0.000000
----------------------------------------
```

- **Example 2**

```
[terminal]$ make main_mpi
/usr/lib64/openmpi/bin/mpicc MPI_main.c MPI_double_layer_convolution.c
/usr/lib64/openmpi/bin/mpirun -np 15 ./a.out 1000 1000 4 13
```

```
The input values for the parameters are
---------------------------------------------
M(rows)=1000, N(columns)=1000, K1=4 and K2=13 on rank=0

Checking the computations on rank 0
----------------------------------------
The error in the output matrix, from the
serial and parallel execution is
Error  =  0.000000
----------------------------------------
```

- Example 3

```
[terminal]$ make main_mpi
/usr/lib64/openmpi/bin/mpicc MPI_main.c MPI_double_layer_convolution.c
/usr/lib64/openmpi/bin/mpirun -np 6 ./a.out 1000 200 7 11

The input values for the parameters are
---------------------------------------------
M(rows)=1000, N(columns)=200, K1=7 and K2=11 on rank=0

Checking the computations on rank 0
----------------------------------------
The error in the output matrix, from the
serial and parallel execution is
Error  =  0.000000
----------------------------------------
```

- Example 4

```
[terminal]$ make main_mpi
/usr/lib64/openmpi/bin/mpicc MPI_main.c MPI_double_layer_convolution.c
/usr/lib64/openmpi/bin/mpirun -np 10 ./a.out 1000 1000 7 11

The input values for the parameters are
---------------------------------------------
M(rows)=1000, N(columns)=1000, K1=7 and K2=11 on rank=0

Checking the computations on rank 0
----------------------------------------
The error in the output matrix, from the
serial and parallel execution is
Error  =  0.000000
----------------------------------------
```

Evaluation of my implementation

My implementation is working fine when I vary the number of processors. It is also working fine when I vary the parameter M, K1 and K2. However, when I try to run it with a value of N>1000, I get a very large error. There seems to be something wrong in my implementation when N>1000, or maybe there is something wrong with the way I calculate the error between the two output matrices. Besides this, it works fine and I do get an error of 0.00 with the other values for M,N,K1 and K2.