# Parallel and Distributed COMPUTING PROJECT

**Yung-Chi Liu,** A20364639

**ABSTRACT**

implementing two-dimensional convolution on parallel computers using different parallelization techniques and models. The project is to design and implement parallel programs using different models.

(a) Implement the 2D convolution using SPMD model and use MPI send and receive operations.

(b) Implement 2D convolution using SPMD model but use MPI collective communication functions wherever possible.

(c) Implement 2D convolution model using SPMD model using hybrid programming with MPIopenMP.

(d) Implement 2D convolution model using a Task and Data Parallel Model.

And last, compare each executed time with different number of processer (1, 2, 4 and 8). Then, compare all models executed time with 8 processer

**Design and Implementation**

1. **ArrayInverter(A**)

Invert array A, change A[i][j] with A[j][i].

This function’s goal is let c\_fft1d( ) work by column.

1. **Data Structure**

A = (complex\*)malloc(NSIZE\*NSIZE\*sizeof(complex));

Store 2D array into 1D array with N\*N size. In this case A[x,y] is equal to A[ x\*N+y]. And when use send and receive operation, the size could be modifying as 2\*N while means size of 2 row of data.

1. **Sequential model**

===========================================

Do c\_fft1d (A,-1)

ArrayInverter(A) // do column FFT

Do c\_fft1d (A,-1) // by invert X-axis and Y-axis

ArrayInverter(A) // (task1)

Do c\_fft1d (B)

ArrayInverter(B,-1)

Do c\_fft1d (B)

ArrayInverter(B,-1) // (task2)

Do multiple A and B, then put to C

// (task3)

Do c\_fft1d (C)

ArrayInverter(C,+1)

Do c\_fft1d (C)

ArrayInverter(C,+1) // (task4)

===========================================

1. **MPI send and receive operations**

===========================================

Send from processor 0 to other. Other processor receives from processor 0.

Do c\_fft1d (A,-1) // Do with N/P row of data

All processor send to processor 0. Processor 0 receive.

ArrayInverter(A) // do column FFT

Send from processor 0 to other. Other processor receives from processor 0.

Do c\_fft1d (A,-1) // by invert X-axis and Y-axis

All processor send to processor 0. Processor 0 receive.

ArrayInverter(A) // (task1)

Task 2 use same method as Task 1.

Send A and B array from processor 0 to other processors with size N/P

Do multiple A and B, then put to C // Do N/P size

All processors send data to P0. // (task3)

Task 4 use same method as Task 1.

===========================================

1. **MPI collective communication functions**

Replace MPI\_Send and MPI\_Recv with MPI\_Scatter and MPI\_Gather.

1. **hybrid programming with MPIopenMP**

USE openMP to parallel the data in FFT. In this case, each processor work with N size in FFT. However, data divide into N/thread in each processor with openMP.

1. **Task and Data Parallel**

P0, P1 handle data in array A. then send data to P4, P5

P2, P3 handle data in array B. Than send data to P4, P5

P4 multiple A and B with half data. P5 multiple other part.

And combine all data in array C to P4. Then send to P6, P7

P6, P7 handle data in array C. And collection the final result at P6.

# Results and Analysis

1.The project is tested by my laptop with Microsoft Visual Studio 2015.

2.For test, the data were default to 1.0 in array A and B. And collective time of

computation plus communication.

3. All the test were average of 5 times test.

A. Send and Receive



As my result, the Speed Up is best by 2processors than decrease. One of reasons may be I don’t code communication part well, so take too much time by passing the data.

**B. MPI collective communication functions**



As the result Gather and Scatter method is more efficient than simple use send and receive. In my algorithm, it takes too much on send and receive. In FFT array A 2 times, I had use 6 send and receive.

C. hybrid programming



The Speed up not had a lot of change. However, speed up is lower when use 8 processors. In my opinion, it takes some time when processor create threads. With 8 processors it take more resource in system and take more time.

D. **Task and Data Parallel**

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Procs | sequntial\_time(sec) | Parallel\_time(sec) | Speed Up |
| 8 | 0.1238 | 0.0882 | 1.403628 |

1. Compare all model with 8 processors



# Improvement

1. In this project, the diversification of speed up is not obviously. I’ll try other strategy and test the result.
2. By my implementation, the function InvertArray() is not paralleled. If parallel the function could get more speed up.
3. In Task and Data parallel, I use Send and Receive to block the task. It could be change to other way, like use MPI Barrier to block tasks.