COMP 6xx (Fall 2021): Compensation Project
Name: Hossein Alishah
Student ID: 202048619
Section: Wed (7:00 pm – 9:45 pm)

Calculate the product of two 8-by-8 matrices using MPI

Answer:

I have implemented parallel programming using MPI for product of square matrix (8×8):

The parallel program that I implemented, do the calculation of each row of the result is one induvial process, so we need 8 free slots available to run this program.

For an example, I put the following matrixes A and B:

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

And the product of 8×8 matrix is:

$$C = A \times B = \begin{bmatrix} 36 & 72 & 108 & 144 & 180 & 216 & 252 & 288 \\ 36 & 72 & 108 & 144 & 180 & 216 & 252 & 288 \\ 36 & 72 & 108 & 144 & 180 & 216 & 252 & 288 \\ 36 & 72 & 108 & 144 & 180 & 216 & 252 & 288 \\ 36 & 72 & 108 & 144 & 180 & 216 & 252 & 288 \\ 36 & 72 & 108 & 144 & 180 & 216 & 252 & 288 \\ 36 & 72 & 108 & 144 & 180 & 216 & 252 & 288 \\ 36 & 72 & 108 & 144 & 180 & 216 & 252 & 288 \end{bmatrix}$$

So, I am going to implement this product by a parallel MPI algorithm:

```c
//Hossein alishah
// Comp 620- Fall 2021
// to run this multiplication we need 8 free slots on processor,
//so, each processor calculate on row of the final matrix
// to compile on mac use the following command line:
//mpicc matrix.c
// to run on mac use the following command line:
//mpirun -np 8 ./a.out
//mpirun --oversubscribe -np 8 ./a.out


#define N 8
#include <stdio.h>
#include <math.h>
#include <sys/time.h>
#include <stdlib.h>
#include <stddef.h>
#include "mpi.h"


void print_matrix(char *prompt, int arr[N][N], int r, int s);

int main(int argc, char *argv[])
{
    int i, j, k, rank, size, tag = 99, blksz, sum = 0;
    // 8 by 8 matrix
    int a[N][N]=
        {{1,2,3,4,5,6,7,8},{1,2,3,4,5,6,7,8},{1,2,3,4,5,6,7,8},{1,2,3,4,5,6,7,8},
        {1,2,3,4,5,6,7,8}, {1,2,3,4,5,6,7,8}, {1,2,3,4,5,6,7,8}, {1,2,3,4,5,6,7,8}};
    // 8 by 8 matrix
    int b[N][N]=
        {{1,2,3,4,5,6,7,8},{1,2,3,4,5,6,7,8},{1,2,3,4,5,6,7,8},{1,2,3,4,5,6,7,8},
        {1,2,3,4,5,6,7,8}, {1,2,3,4,5,6,7,8}, {1,2,3,4,5,6,7,8}, {1,2,3,4,5,6,7,8}};
    // the matrix that will have the results of the multiplication
    int c[N][N];
    // scatter send a copy of matrrix a to aa
    int aa[N],cc[N];

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    //scatter rows of first matrix to different processes
    MPI_Scatter(a, N*N/size, MPI_INT, aa, N*N/size, MPI_INT,0,MPI_COMM_WORLD);

    //broadcast second matrix to all processes
    MPI_Bcast(b, N*N, MPI_INT, 0, MPI_COMM_WORLD);

    MPI_Barrier(MPI_COMM_WORLD);
```

```
46
47      //perform vector multiplication by all processes
48      for (i = 0; i < N; i++)
49          {
50                  for (j = 0; j < N; j++)
51                  {
52                          sum = sum + aa[j] * b[j][i];  //MISTAKE_WAS_HERE
53                  }
54                  cc[i] = sum;
55                  sum = 0;
56              }
57
58      //we are gather all process to the process with the rank=0, so to access the
59      //result of multiplication, we need to just get the c[][] from ran=0
60      MPI_Gather(cc, N*N/size, MPI_INT, c, N*N/size, MPI_INT, 0, MPI_COMM_WORLD);
61
62
63      MPI_Barrier(MPI_COMM_WORLD);
64      MPI_Finalize();
65
66      //because of gattering all calculation on C on processor rank=0,
67      //we print only C from processor rank=0
68      if (rank == 0)
69          print_matrix("C = ", c, rank, size);
70
71  }
72
73  // this function print all elements of a N by N matrix
74  void print_matrix(char *prompt, int arr[N][N], int r, int s)
75  {
76      int i, j;
77
78      printf ("\n\n%s\n", prompt);
79      for (i = 0; i < N; i++) {
80              for (j = 0; j < N; j++) {
81                      printf(" %d", arr[i][j]);
82              }
83              printf ("\n");
84      }
85      printf ("\n\n");
86      printf("processor rank is  %d\n", r);
87      printf("processor size is  %d\n", s);
88      printf ("\n\n");
89  }
90
91
```

Since this code is gathering all the results at the process having rank 0, there is a statement *if (rank == 0)* before calling the *print_results(...)* function to print the matric C or the product A × B.

I am going to run this program and show the result here:

```
hosseinalishah@Hosseins-MacBook-Pro ~ % mpicc matrix.c
hosseinalishah@Hosseins-MacBook-Pro ~ % mpirun --oversubscribe -np 8 ./a.out


C =
 36 72 108 144 180 216 252 288
 36 72 108 144 180 216 252 288
 36 72 108 144 180 216 252 288
 36 72 108 144 180 216 252 288
 36 72 108 144 180 216 252 288
 36 72 108 144 180 216 252 288
 36 72 108 144 180 216 252 288
 36 72 108 144 180 216 252 288

processor rank is  0
processor size is  8
```