CS309 ASSIGNMENT1

Krishanu Saini - (190001029)

September 11, 2021

1 Problem Statement

In this assignment, you will develop MPI program for the following tasks.

1. Write an MPI program with 2/4 processes for multiplying two vectors of size 100k (5 marks).
2. Write an MPI program for multiplying two random matrices of size 5k X 5k (5 marks).

You need to print runtime of your code as well.

2 Multiplying 2 vectors

Suppose we have 2 vectors A_{nX1} and B_{nX1}. We know product of 2 vectors is $A^T B = A_1 \cdot B_1 + A_2 \cdot B_2 + A_3 \cdot B_3 + A_4 \cdot B_4$.
To make it parallel using mpi we will use **data parallelism** steps below.

1. Calculate size of each block = n / number of process.
2. Split A and B into array of size = size_block
3. Accumulate the result in master process using MPI_reduce.

Time Complexity: $O(n/p)$.

3 Code

1. Multiplying 2 vectors : Link

```
#include<stdio.h>
#include<stdlib.h>
#include<mpi.h>

int main(){
  MPI_Init(NULL, NULL);
  int world_rank, world_size;
  MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
  MPI_Comm_size(MPI_COMM_WORLD, &world_size);

  // input 100k size vector
  int n = 100000;
  int *A = (int*)malloc(n*sizeof(int));
  int *B = (int*)malloc(n*sizeof(int));

  // fill data in arr
  for(int i=0;i<n;i++){
    A[i] = (i % 100);
    B[i] = (i % 50);
  }

  int size = n / world_size;

  int *subarray_A = (int*)malloc(size*(sizeof(int)));
  int *subarray_B = (int*)malloc(size*(sizeof(int)));

  MPI_Scatter(A, size, MPI_INT, subarray_A, size, MPI_INT, 0,
      MPI_COMM_WORLD);
  MPI_Scatter(B, size, MPI_INT, subarray_B, size, MPI_INT, 0,
      MPI_COMM_WORLD);

  // dot product here
  int *resultBuf = NULL;
  if(world_rank == 0){
    int *resultBuf = (int*)malloc(world_size*(sizeof(int)));
  }
```

```
  int sub_sum = 0;
  for(int i=0;i<size;i++){
    sub_sum += subarray_A[i]*subarray_B[i];
  }

  printf("Processor %d gives sum %d\n", world_rank, sub_sum);
  MPI_Barrier(MPI_COMM_WORLD);

  int ans = 0;
  MPI_Reduce(&sub_sum, &ans, 1, MPI_INT, MPI_SUM, 0,
      MPI_COMM_WORLD);

  if(world_rank == 0){
    // gather here
    printf("Vector multiplication gives %d\n", ans);
  }

  MPI_Finalize();
  return 0;
}
```

4 OUTPUT

Explanation - For vectors $A = [0, 1, 2, 3]^T$ and $B = [0, 1, 2, 3]^T$
the vector multiplication = $A^T.B$ ie $z = 0{\cdot}0{+}1{\cdot}1{+}2{\cdot}2{+}3{\cdot}3 = 0{+}1{+}4{+}9 = 14$

2 X Processes

```
krishanu2001@LAPTOP-V4CKFTKN:/mnt/c/Users/krishanu/Desktop/sem5/PARALLEL/mpi/lab2$ mpicc vector_mult.c -o vector_mult
krishanu2001@LAPTOP-V4CKFTKN:/mnt/c/Users/krishanu/Desktop/sem5/PARALLEL/mpi/lab2$ mpirun -n 2 ./vector_mult
0 1 2 3
0 1 2 3
Processor 0 gives sum 1
Processor 1 gives sum 13
Vector multiplication gives 14
```

4 X Processes

```
krishanu2001@LAPTOP-V4CKFTKN:/mnt/c/Users/krishanu/Desktop/sem5/PARALLEL/mpi/lab2$ mpirun -n 4 ./vector_mult
0 1 2 3
0 1 2 3
Processor 0 gives sum 0
Processor 2 gives sum 4
Processor 3 gives sum 9
Processor 1 gives sum 1
Vector multiplication gives 14
```

Now for $100K$ sized vector.
This is for input specified in code. A = $[i : i\%100]^T$, B = $[i : i\%50]^T$.

```
krishanu2001@LAPTOP-V4CKFTKN:/mnt/c/Users/krishanu/Desktop/sem5/PARALLEL/mpi/lab2$ mpirun -n 2 ./vector_mult
Processor 0 gives sum 71050000
Processor 1 gives sum 71050000
Vector multiplication gives 142100000
krishanu2001@LAPTOP-V4CKFTKN:/mnt/c/Users/krishanu/Desktop/sem5/PARALLEL/mpi/lab2$ ls
```

5 Multiplying 2 matrices

Suppose we have 2 square matrices A_{nXn} and B_{nXn}. We know product of 2 matrices is $O(n^3)$ Operation using 3 for loops.
To make it parallel using mpi we will use **data parallelism** steps below.

1. Split A_{nXn} into multiple parts having row size r = n / number of processors.
   as $A[1...r][n]$, $A[r+1...2r][n]$ so on.
2. MPI_Send A_{rXn} and B matrix to multiple processes.
3. Calculate matrix $C_{rXn} = A_{rXn} \cdot B_{nXn}$ using matrix multiplication.
4. Master process gathers all C_{rXn}'s in final matrix C.

Time Complexity: $O(n^3/p)$.

6 Code

2. Multiplying 2 matrices : Link

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>

#define MATSIZE 4
#define NRA MATSIZE
#define NCA MATSIZE
#define NCB MATSIZE
#define MASTER 0
#define FROM_MASTER 1
#define FROM_WORKER 2

int main(int argc, char *argv[])
{
  int numtasks,
      taskid,
      numworkers,
```

```
    source,
    dest,
    mtype,
    rows,
    averow, extra, offset,
    i, j, k, rc;
double a[NRA][NCA],
    b[NCA][NCB],
    c[NRA][NCB];
MPI_Status status;

MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &taskid);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
if (numtasks < 2)
{
  printf("Need at least two MPI tasks. Quitting...\n");
  MPI_Abort(MPI_COMM_WORLD, rc);
  exit(1);
}
numworkers = numtasks - 1;
if (taskid == MASTER)
{
  printf("mpi_mm has started with %d tasks.\n", numtasks);
  printf("Initializing arrays...\n");
  for (i = 0; i < NRA; i++)
    for (j = 0; j < NCA; j++)
      a[i][j] = i + j;
  for (i = 0; i < NCA; i++)
    for (j = 0; j < NCB; j++)
      b[i][j] = i * j;
  double start = MPI_Wtime();
  averow = NRA / numworkers;
  extra = NRA % numworkers;
  offset = 0;
  mtype = FROM_MASTER;
  for (dest = 1; dest <= numworkers; dest++)
  {
    rows = (dest <= extra) ? averow + 1 : averow;
    printf("Sending %d rows to task %d offset=%d\n", rows, dest,
```

```
      offset);
    MPI_Send(&offset, 1, MPI_INT, dest, mtype, MPI_COMM_WORLD);
    MPI_Send(&rows, 1, MPI_INT, dest, mtype, MPI_COMM_WORLD);
    MPI_Send(&a[offset][0], rows * NCA, MPI_DOUBLE, dest, mtype,
             MPI_COMM_WORLD);
    MPI_Send(&b, NCA * NCB, MPI_DOUBLE, dest, mtype,
      MPI_COMM_WORLD);
    offset = offset + rows;
  }
  mtype = FROM_WORKER;
  for (i = 1; i <= numworkers; i++)
  {
    source = i;
    MPI_Recv(&offset, 1, MPI_INT, source, mtype, MPI_COMM_WORLD,
      &status);
    MPI_Recv(&rows, 1, MPI_INT, source, mtype, MPI_COMM_WORLD,
      &status);
    MPI_Recv(&c[offset][0], rows * NCB, MPI_DOUBLE, source, mtype,
             MPI_COMM_WORLD, &status);
    printf("Received results from task %d\n", source);
  }
  printf("Result Matrix:\n");
  for (i = 0; i < NRA; i++)
  {
    printf("\n");
    for (j = 0; j < NCB; j++)
      printf("%6.2f  ", c[i][j]);
  }
  double finish = MPI_Wtime();
  printf("Done in %f seconds.\n", finish - start);
}
if (taskid > MASTER)
{
  mtype = FROM_MASTER;
  MPI_Recv(&offset, 1, MPI_INT, MASTER, mtype, MPI_COMM_WORLD,
    &status);
  MPI_Recv(&rows, 1, MPI_INT, MASTER, mtype, MPI_COMM_WORLD,
    &status);
  MPI_Recv(&a, rows * NCA, MPI_DOUBLE, MASTER, mtype,
    MPI_COMM_WORLD, &status);
```

```
    MPI_Recv(&b, NCA * NCB, MPI_DOUBLE, MASTER, mtype,
       MPI_COMM_WORLD, &status);

    for (k = 0; k < NCB; k++)
      for (i = 0; i < rows; i++)
      {
        c[i][k] = 0.0;
        for (j = 0; j < NCA; j++)
          c[i][k] = c[i][k] + a[i][j] * b[j][k];
      }
    mtype = FROM_WORKER;
    MPI_Send(&offset, 1, MPI_INT, MASTER, mtype, MPI_COMM_WORLD);
    MPI_Send(&rows, 1, MPI_INT, MASTER, mtype, MPI_COMM_WORLD);
    MPI_Send(&c, rows * NCB, MPI_DOUBLE, MASTER, mtype,
       MPI_COMM_WORLD);
  }
  MPI_Finalize();
}
```

7 OUTPUT

$$\text{Matrix A} = \begin{matrix} 0 & 1 & 2 & 3 \\ 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \end{matrix} \qquad \text{Matrix B} = \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 \\ 0 & 2 & 4 & 6 \\ 0 & 3 & 6 & 9 \end{matrix}$$

```
PROBLEMS 6   OUTPUT   DEBUG CONSOLE   TERMINAL
Initializing arrays...
Sending 2 rows to task 1 offset=0
Sending 1 rows to task 2 offset=2
Sending 1 rows to task 3 offset=3
Received results from task 1
Received results from task 2
Received results from task 3
Result Matrix:

  0.00    14.00    28.00    42.00
  0.00    20.00    40.00    60.00
  0.00    26.00    52.00    78.00
  0.00    32.00    64.00    96.00   Done in 0.000106 seconds.
krishanu2001@LAPTOP-V4CKFTKN:/mnt/c/Users/krishanu/Desktop/sem5/PARALLEL/mpi/lab2$
```

$$\text{Matrix A} = \begin{matrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{matrix} \qquad \text{Matrix B} = \begin{matrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{matrix}$$

```
mpi_mm has started with 4 tasks.
Initializing arrays...
Sending 1 rows to task 1 offset=0
Sending 1 rows to task 2 offset=1
Sending 1 rows to task 3 offset=2
Received results from task 1
Received results from task 2
Received results from task 3
Result Matrix:

  5.00     8.00    11.00
  8.00    14.00    20.00
 11.00    20.00    29.00   Done in 0.000059 seconds.
krishanu2001@LAPTOP-V4CKFTKN:/mnt/c/Users/krishanu/Desktop/sem5/PARALLEL/mpi/lab2$
```