Class: Final Year (Computer Science and Engineering)

Year: 2022-23 **Semester:** 1

Course: High Performance Computing Lab

Practical No. 7

PRN No : 2019BTECS00110

Name : Pavan Shinde

1. Implement Matrix-Vector Multiplication using MPI. Use a different number of processes and analyze the performance.

Code:

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

// size of matrix
#define N 500

int main(int argc, char *argv[])
{
   int np, rank, numworkers, rows, i, j, k;

   // a*b = c
   double a[N][N], b[N], c[N];
   MPI_Status status;

   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

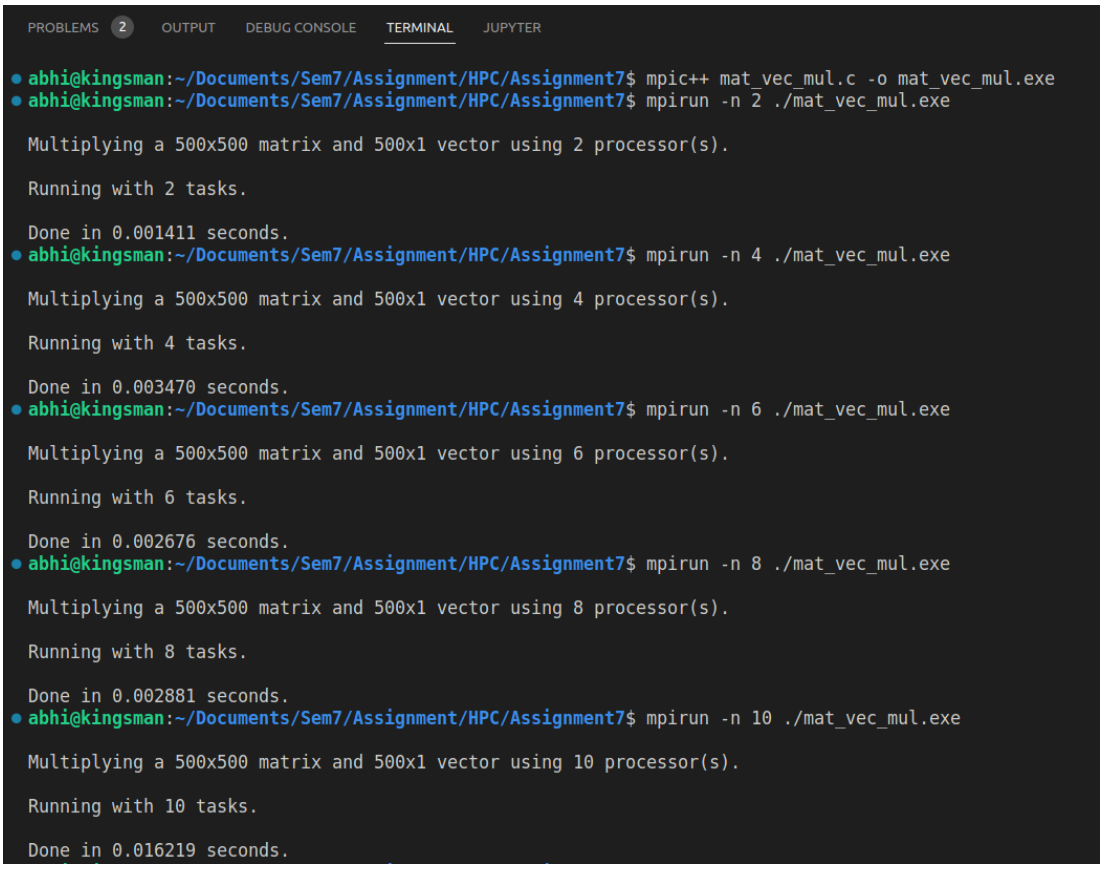
```
   MPI_Comm_size(MPI_COMM_WORLD, &np);


   numworkers = np - 1; // total process - 1 ie process with rank 0


   // rank with 0 is a master process
   int dest, source;
   int tag;
   int rows_per_process, extra, offset;


   // master process, process with rank = 0
   if (rank == 0)
   {
         printf("\nMultiplying a %dx%d matrix and %dx1 vector using %d
processor(s).\n\n", N, N, N, np);


       printf("Running with %d tasks.\n", np);


       // matrix a and b initialization
       for (i = 0; i < N; i++)
           for (j = 0; j < N; j++)
               a[i][j] = 1;


       for (i = 0; i < N; i++)
           b[i] = 1;


       // start time
       double start = MPI_Wtime();


       // Send matrix data to other worker processes
```

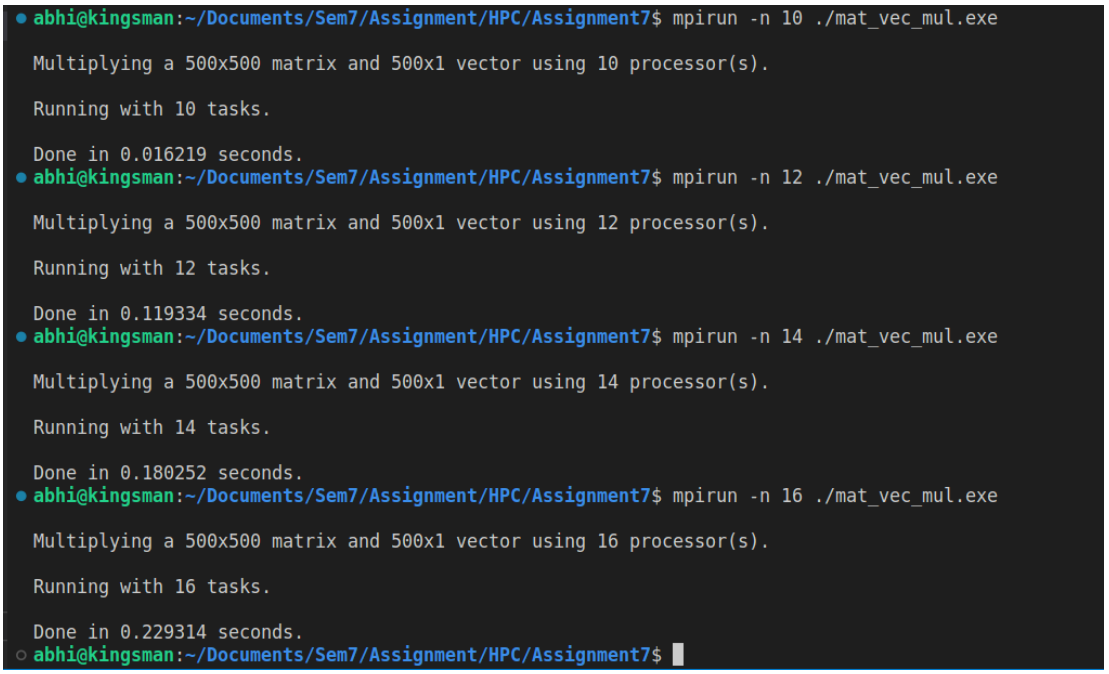
```
        rows_per_process = N / numworkers;
        extra = N % numworkers;

        offset = 0;
        tag = 1;

        // send data to other nodes
        for (dest = 1; dest <= numworkers; dest++)
        {
                    rows = (dest <= extra) ? rows_per_process + 1 :
rows_per_process;

            MPI_Send(&offset, 1, MPI_INT, dest, tag, MPI_COMM_WORLD);
            MPI_Send(&rows, 1, MPI_INT, dest, tag, MPI_COMM_WORLD);

               MPI_Send(&a[offset][0], rows * N, MPI_DOUBLE, dest, tag,
MPI_COMM_WORLD);
            MPI_Send(&b, N, MPI_DOUBLE, dest, tag, MPI_COMM_WORLD);

            offset = offset + rows;
        }

        // receive data from other nodes and add it to the ans matrix c
        tag = 2;
        for (i = 1; i <= numworkers; i++)
        {
            source = i;
              MPI_Recv(&offset, 1, MPI_INT, source, tag, MPI_COMM_WORLD,
&status);
```

```
            MPI_Recv(&rows, 1, MPI_INT, source, tag, MPI_COMM_WORLD,
&status);

                MPI_Recv(&c[offset], N, MPI_DOUBLE, source, tag,
MPI_COMM_WORLD, &status);

      }


      // print multiplication result

      //        printf("Result Matrix:\n");

      //        for (i = 0; i < N; i++)

      //        {

      //            printf("%6.2f   ", c[i]);

      //        }



      printf("\n");


      double finish = MPI_Wtime();

       printf("Done in %f seconds.\n", finish - start); // total time
spent

  }


  // all other process than process with rank = 0

  if (rank > 0)

  {

      tag = 1;

      // receive data from process with rank 0

      MPI_Recv(&offset, 1, MPI_INT, 0, tag, MPI_COMM_WORLD, &status);

      MPI_Recv(&rows, 1, MPI_INT, 0, tag, MPI_COMM_WORLD, &status);

         MPI_Recv(&a, rows * N, MPI_DOUBLE, 0, tag, MPI_COMM_WORLD,
&status);
```

```
        MPI_Recv(&b, N, MPI_DOUBLE, 0, tag, MPI_COMM_WORLD, &status);


        // calculate multiplication of given rows
        for (i = 0; i < rows; i++)
        {
            c[i] = 0.0;
            for (j = 0; j < N; j++)
                c[i] = c[i] + a[i][j] * b[j];
        }
        // send result back to process with rank 0
        tag = 2;
        MPI_Send(&offset, 1, MPI_INT, 0, tag, MPI_COMM_WORLD);
        MPI_Send(&rows, 1, MPI_INT, 0, tag, MPI_COMM_WORLD);
        MPI_Send(&c, N, MPI_DOUBLE, 0, tag, MPI_COMM_WORLD);
    }


    MPI_Finalize();
}
```

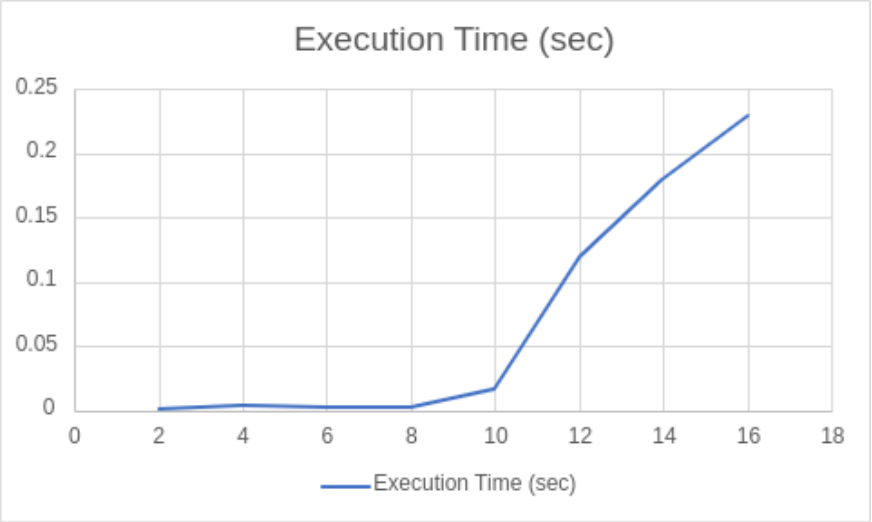
Output:


PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
JUPYTER
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpic++ mat_vec_mul.c -o mat_vec_mul.exe
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -n 2 ./mat_vec_mul.exe
Multiplying a 500x500 matrix and 500x1 vector using 2 processor(s).
Running with 2 tasks.
Done in 0.001411 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -n 4 ./mat_vec_mul.exe
Multiplying a 500x500 matrix and 500x1 vector using 4 processor(s).
Running with 4 tasks.
Done in 0.003470 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -n 6 ./mat_vec_mul.exe
Multiplying a 500x500 matrix and 500x1 vector using 6 processor(s).
Running with 6 tasks.
Done in 0.002676 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -n 8 ./mat_vec_mul.exe
Multiplying a 500x500 matrix and 500x1 vector using 8 processor(s).
Running with 8 tasks.
Done in 0.002881 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -n 10 ./mat_vec_mul.exe
Multiplying a 500x500 matrix and 500x1 vector using 10 processor(s).
Running with 10 tasks.
Done in 0.016219 seconds.


abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -n 10 ./mat_vec_mul.exe
Multiplying a 500x500 matrix and 500x1 vector using 10 processor(s).
Running with 10 tasks.
Done in 0.016219 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -n 12 ./mat_vec_mul.exe
Multiplying a 500x500 matrix and 500x1 vector using 12 processor(s).
Running with 12 tasks.
Done in 0.119334 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -n 14 ./mat_vec_mul.exe
Multiplying a 500x500 matrix and 500x1 vector using 14 processor(s).
Running with 14 tasks.
Done in 0.180252 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -n 16 ./mat_vec_mul.exe
Multiplying a 500x500 matrix and 500x1 vector using 16 processor(s).
Running with 16 tasks.
Done in 0.229314 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$

Analysis:

Processors (Size 500)	Execution Time (sec)
2	0.001411
4	0.00347
6	0.002676
8	0.002881
10	0.016219
12	0.119334
14	0.180252
16	0.229314



Conclusion:

So, as we can see when we increase the number of Processors for execution of the code the execution time also increases. This observation is not because of the computation time but as we increase the number of processes the **communication overhead between the processors increases** and thus there is increase in execution time.

2. Implement Matrix-Matrix Multiplication using MPI. Use a different number of processes and analyze the performance.

Code:

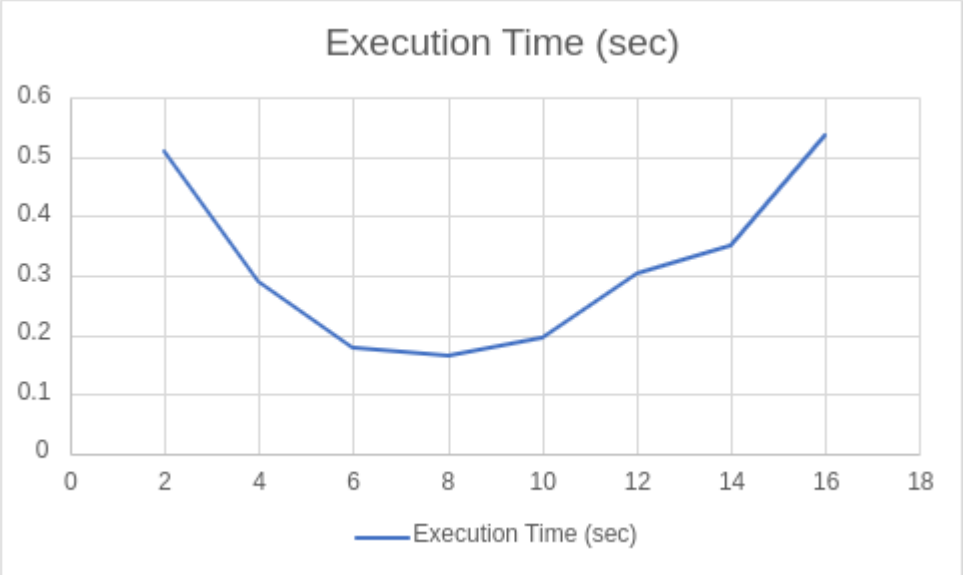
```c
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>

#define MATSIZE 500
#define NRA MATSIZE   /* number of rows in matrix A */
#define NCA MATSIZE   /* number of columns in matrix A */
#define NCB MATSIZE   /* number of columns in matrix B */
#define MASTER 0      /* taskid of first task */
#define FROM_MASTER 1 /* setting a message type */
#define FROM_WORKER 2 /* setting a message type */

int main(int argc, char *argv[])
{
  int numtasks,              /* number of tasks in partition */
      taskid,                /* a task identifier */
      numworkers,            /* number of worker tasks */
      source,                /* task id of message source */
      dest,                  /* task id of message destination */
      mtype,                 /* message type */
      rows,                  /* rows of matrix A sent to each worker */
      averow, extra, offset, /* used to determine rows sent to each worker */
      i, j, k, rc;           /* misc */
  double a[NRA][NCA],        /* matrix A to be multiplied */
      b[NCA][NCB],           /* matrix B to be multiplied */
      c[NRA][NCB];           /* result matrix C */
  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;

  /**************************** master task ************************************/
  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;

    /* Measure start time */
    double start = MPI_Wtime();

    /* Send matrix data to the worker tasks */
    averow = NRA / numworkers;
    extra = NRA % numworkers;
    offset = 0;
    mtype = FROM_MASTER;
    for (dest = 1; dest <= numworkers; dest++)
```

```
    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;
    }

    /* Receive results from worker tasks */
    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);
    }

    /* Print results */
    /*
    printf("******************************************************\n");
    printf("Result Matrix:\n");
    for (i=0; i<NRA; i++)
    {
       printf("\n");
       for (j=0; j<NCB; j++)
          printf("%6.2f   ", c[i][j]);
    }
    printf("\n******************************************************\n");
    */

    /* Measure finish time */
    double finish = MPI_Wtime();
    printf("Done in %f seconds.\n", finish - start);
  }

  /**************************** worker task ************************************/
  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();
}
```

Output:

```
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpicc mat_mat_mul.c -o mat_mat_mul
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -np 4 ./mat_mat_mul
mpi_mm has started with 4 tasks.
Done in 0.200889 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -np 1 ./mat_mat_mul
Need at least two MPI tasks. Quitting...
application called MPI_Abort(MPI_COMM_WORLD, 0) - process 0
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -np 2 ./mat_mat_mul
mpi_mm has started with 2 tasks.
Done in 0.509733 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -np 4 ./mat_mat_mul
mpi_mm has started with 4 tasks.
Done in 0.290073 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -np 6 ./mat_mat_mul
mpi_mm has started with 6 tasks.
Done in 0.176806 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -np 8 ./mat_mat_mul
mpi_mm has started with 8 tasks.
Done in 0.163952 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -np 10 ./mat_mat_mul
mpi_mm has started with 10 tasks.
Done in 0.193650 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -np 12 ./mat_mat_mul
mpi_mm has started with 12 tasks.
Done in 0.303720 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -np 14 ./mat_mat_mul
mpi_mm has started with 14 tasks.
Done in 0.351833 seconds.
abhi@kingsman:~/Documents/Sem7/Assignment/HPC/Assignment7$ mpirun -np 16 ./mat_mat_mul
mpi_mm has started with 16 tasks.
Done in 0.538575 seconds.
```

Processors (Size 512)	Execution Time (sec)
2	0.509733
4	0.290073
6	0.176806
8	0.163952
10	0.19365
12	0.30372
14	0.351833
16	0.538575



Conclusion:

So, as we can see when we increase the number of Processors for execution of the code the execution time decreases and after some time there is increase in the execution time.

This is due to an **increase in the number of processors the execution time decreases till a certain number of processors** after that increase in processors leads to increase in communication time so the execution time increases.