Q4. WAP for matrix mutiplication with MPI

#include"mpi.h"

#include<stdio.h>

#include<stdlib.h>

#define N 4

int A[N][N], B[N][N], C[N][N], D[N];

void fill\_matrix(int m[N][N])

{

static int n=0;

int i, j;

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

for (j=0; j<N; j++)

m[i][j] = n++;

}

void print\_matrix(int m[N][N])

{

int i, j = 0;

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

printf("\n\t| ");

for (j=0; j<N; j++)

printf("%2d ", m[i][j]);

printf("|");

}

}

int main(int argc, char \*argv[])

{

int myrank, P, from, to, i, j, k;

MPI\_Status status;

MPI\_Init (&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &myrank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &P);

if (N%P!=0) {

if (myrank==0) printf("Matrix size not divisible by number of processors\n");

MPI\_Finalize();

exit(-1);

}

from = myrank \* N/P;

to = (myrank+1) \* N/P;

if (myrank==0) {

fill\_matrix(A);

fill\_matrix(B);

}

for (i=from; i<4;i++)

{

for (j=0; j<N; j++) {

C[i][j]=0;

for (k=0; k<N; k++)

C[i][j] += A[i][k]\*B[k][j];

}

from=myrank \* N/P;

to=(myrank+1)\*N/P;

}

if (myrank==0) {

printf("\n\n");

print\_matrix(A);

printf("\n\n\t \* \n");

print\_matrix(B);

printf("\n\n\t = \n");

print\_matrix(C);

printf("\n\n");

}

MPI\_Finalize();

return 0;

}

**OutPut**

| 0 1 2 3 |

| 4 5 6 7 |

| 8 9 10 11 |

| 12 13 14 15 |

\*

| 16 17 18 19 |

| 20 21 22 23 |

| 24 25 26 27 |

| 28 29 30 31 |

=

| 152 158 164 170 |

| 504 526 548 570 |

| 856 894 932 970 |

| 1208 1262 1316 1370 |