MATRIX MULTIPLICATION

# SOURCE CODE:

#include <stdio.h> #include <stdlib.h> #include <mpi.h> #define N 100 MPI\_Status status;

double a[N][N],b[N][N],c[N][N]; int main(int argc, char \*\*argv)

{

int numtasks,taskid,numworkers,source,dest,rows,offset,i,j,k; float startTime, endTime;

MPI\_Init(&argc, &argv); MPI\_Comm\_rank(MPI\_COMM\_WORLD, &taskid); MPI\_Comm\_size(MPI\_COMM\_WORLD, &numtasks); numworkers = numtasks-1;

if (taskid == 0) {

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

a[i][j]= rand()%N;

b[i][j]= rand()%N;

}

}

startTime = MPI\_Wtime(); rows = N/numworkers; offset = 0;

for (dest=1; dest<=numworkers; dest++)

{

MPI\_Send(&offset, 1, MPI\_INT, dest, 1, MPI\_COMM\_WORLD);

MPI\_Send(&rows, 1, MPI\_INT, dest, 1, MPI\_COMM\_WORLD); MPI\_Send(&a[offset][0], rows\*N, MPI\_DOUBLE,dest,1, MPI\_COMM\_WORLD); MPI\_Send(&b, N\*N, MPI\_DOUBLE, dest, 1, MPI\_COMM\_WORLD);

offset = offset + rows;

}

for (i=1; i<=numworkers; i++)

{

source = i;

MPI\_Recv(&offset, 1, MPI\_INT, source, 2, MPI\_COMM\_WORLD, &status); MPI\_Recv(&rows, 1, MPI\_INT, source, 2, MPI\_COMM\_WORLD, &status); MPI\_Recv(&c[offset][0], rows\*N, MPI\_DOUBLE, source, 2,

MPI\_COMM\_WORLD, &status);

}

endTime = MPI\_Wtime();

printf("Matrix Multiplication Completed. Execution Time is

%f\n",endTime-startTime);

}

if (taskid > 0) { source = 0;

MPI\_Recv(&offset, 1, MPI\_INT, source, 1, MPI\_COMM\_WORLD, &status); MPI\_Recv(&rows, 1, MPI\_INT, source, 1, MPI\_COMM\_WORLD, &status); MPI\_Recv(&a, rows\*N, MPI\_DOUBLE, source, 1, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&b, N\*N, MPI\_DOUBLE, source, 1, MPI\_COMM\_WORLD, &status);

/\* Matrix multiplication \*/ for (k=0; k<N; k++)

for (i=0; i<rows; i++) { c[i][k] = 0.0;

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

c[i][k] = c[i][k] + a[i][j] \* b[j][k];

}

MPI\_Send(&offset, 1, MPI\_INT, 0, 2, MPI\_COMM\_WORLD);

MPI\_Send(&rows, 1, MPI\_INT, 0, 2, MPI\_COMM\_WORLD);

MPI\_Send(&c, rows\*N, MPI\_DOUBLE, 0, 2, MPI\_COMM\_WORLD);

}

MPI\_Finalize(); return 0;

}