#include <mpi.h>

#include <iostream>

#include <cstdlib>

#define N 1024

using namespace std;

void matrix\_mult(double\* A\_local, double\* B, double\* C\_local, int rows\_per\_process) {

for (int i = 0; i < rows\_per\_process; i++) {

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

C\_local[i \* N + j] = 0;

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

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

}

}

}

}

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

MPI\_Init(&argc, &argv);

int rank, size;

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

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

int rows\_per\_process = N / size;

double\* A\_local = new double[rows\_per\_process \* N];

double\* B = new double[N \* N];

double\* C\_local = new double[rows\_per\_process \* N];

if (rank == 0) {

double\* A = new double[N \* N];

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

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

A[i \* N + j] = rand() % 100;

B[i \* N + j] = rand() % 100;

}

}

MPI\_Scatter(A, rows\_per\_process \* N, MPI\_DOUBLE, A\_local, rows\_per\_process \* N, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);

delete[] A;

} else {

MPI\_Scatter(nullptr, rows\_per\_process \* N, MPI\_DOUBLE, A\_local, rows\_per\_process \* N, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);

}

MPI\_Bcast(B, N \* N, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);

double start\_time = MPI\_Wtime();

matrix\_mult(A\_local, B, C\_local, rows\_per\_process);

double end\_time = MPI\_Wtime();

double elapsed\_time = end\_time - start\_time;

if (rank == 0) {

double\* C = new double[N \* N];

MPI\_Gather(C\_local, rows\_per\_process \* N, MPI\_DOUBLE, C, rows\_per\_process \* N, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);

delete[] C;

} else {

MPI\_Gather(C\_local, rows\_per\_process \* N, MPI\_DOUBLE, nullptr, rows\_per\_process \* N, MPI\_DOUBLE, 0, MPI\_COMM\_WORLD);

}

if (rank == 0) {

cout << "Time taken for matrix multiplication: " << elapsed\_time << " seconds." << endl;

}

delete[] A\_local;

delete[] B;

delete[] C\_local;

MPI\_Finalize();

return 0;

}