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

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

int N = 16; // Total number of elements int array[N];

int local\_sum = 0, total\_sum = 0;

MPI\_Init(Cargc, Cargv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, Crank); MPI\_Comm\_size(MPI\_COMM\_WORLD, Csize);

int elements\_per\_proc = N / size; int remaining\_elements = N % size;

if (rank == 0) {

for (int i = 0; i < N; i++) { array[i] = i + 1;

}

printf("Original array: "); for (int i = 0; i < N; i++) { printf("%d ", array[i]);

}

printf("\n");

}

int local\_size = elements\_per\_proc + (rank < remaining\_elements ? 1 : 0);

int\* local\_array = (int\*)malloc(local\_size \* sizeof(int));

int displs[size], send\_counts[size]; if (rank == 0) {

int offset = 0;

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

send\_counts[i] = elements\_per\_proc + (i < remaining\_elements ? 1 : 0); displs[i] = offset;

offset += send\_counts[i];

}

}

MPI\_Scatterv(array, send\_counts, displs, MPI\_INT, local\_array, local\_size, MPI\_INT, 0, MPI\_COMM\_WORLD);

for (int i = 0; i < local\_size; i++) { local\_sum += local\_array[i];

}

free(local\_array);

printf("Processor %d calculated local sum: %d\n", rank, local\_sum);

MPI\_Reduce(Clocal\_sum, Ctotal\_sum, 1, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

if (rank == 0) {

printf("Total sum of array: %d\n", total\_sum);

}

MPI\_Finalize(); return 0;

}

