```
#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(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int elements_per_proc = N / size;
    int local_array[elements_per_proc];

    if (rank == 0) {
        // Initialize the array
        for (int i = 0; i < N; i++) {
            array[i] = i + 1;  // Array contains values 1 to N
        }
        printf("Original array: ");
        for (int i = 0; i < N; i++) {
            printf("%d ", array[i]);
        }
        printf("\n");
    }

    // Scatter the array to all processes
```

```
    MPI_Scatter(array, elements_per_proc, MPI_INT, local_array, elements_per_proc, MPI_INT, 0,
MPI_COMM_WORLD);

    // Compute local sum
    for (int i = 0; i < elements_per_proc; i++) {
        local_sum += local_array[i];
    }

    // Print intermediate sum calculated by each processor
    printf("Processor %d calculated local sum: %d\n", rank, local_sum);

    // Reduce all local sums to the root process
    MPI_Reduce(&local_sum, &total_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;
}
```