

2.0.

Give a parallel program for adding two vectors x and y . Process 0 lets the user input the order, x and y , and then print x , y and $x+y$. Your program should have `Read_vector` and `Print_vector` functions.

2.1.

`MPI_Type_contiguous` can be used to build a derived datatype from a collection of contiguous elements in an array. Its syntax is

```
int MPI_Type_contiguous(
    int          count      /* in */,
    MPI_Datatype old_mpi_t  /* in */,
    MPI_Datatype* new_mpi_t_p /* out */);
```

Modify the `Read_vector` and `Print_vector` functions so that they use an MPI datatype created by a call to `MPI_Type_contiguous` and a count argument of 1 in the calls to `MPI_Scatter` and `MPI_Gather`.

2.2.

Finding **prefix sums** is a generalization of global sum. Rather than simply finding the sum of n values,

$$x_0 + x_1 + \cdots + x_{n-1},$$

the prefix sums are the n partial sums

$$x_0, x_0 + x_1, x_0 + x_1 + x_2, \dots, x_0 + x_1 + \cdots + x_{n-1}.$$

- a. Devise a serial algorithm for computing the n prefix sums of an array with n elements.
- b. Parallelize your serial algorithm for a system with n processes, each of which is storing one of the x_i s.
- d. MPI provides a collective communication function, `MPI_Scan`, that can be used to compute prefix sums:

```
int MPI_Scan(
    void*      sendbuf_p  /* in */,
    void*      recvbuf_p  /* out */,
    int        count      /* in */,
    MPI_Datatype datatype  /* in */,
    MPI_Op     op         /* in */,
    MPI_Comm   comm       /* in */);
```

It operates on arrays with `count` elements; both `sendbuf_p` and `recvbuf_p` should refer to blocks of `count` elements of type `datatype`. The `op` argument is the same as `op` for `MPI_Reduce`. Write an MPI program that generates a random array of `count` elements on each MPI process, finds the prefix sums, and prints the results.