CSCI 351 – PARALLEL PROGRAMMING

FALL, 2015

Assignment 4 Key

Q1. [PACHECO ex3.9] mpi_vect_mult.c

Write an MPI program that implements multiplication of a vector by a scalar and dot product. The user should enter two vectors and a scalar, all of which are read in by process 0 and distributed among the processes. The results are calculated and collected onto process 0, which prints them. You can assume that *n*, the order of the vectors, is evenly divisible by comm_sz.

Download code template from class website.

Example:

Assume you have:

two vectors:

v1 = (1, -1, 3)

v2 = (2, 1, 1)

and scalar = 2

dot product:

dot product of v1 and v2 is a single number which is the sum of multiplication of corresponding elements in two vectors

v1.v2 = 1x2 + -1x1 + 3x1

= 2 + -1 + 3

= 4

Scalar multiplication:

Multiplying a vector by a scalar is called scalar multiplication. To perform scalar multiplication, you need to multiply the scalar by each component of the vector. (A scalar is just a fancy word for a real number.)

Scalar.v1 = 2v1 = (2x1, 2x-1, 2x3) = (2,-2, 6)

Answer:

```
/* File:
 *    mpi_vect_mult.c
 *
 * Purpose:
 *    Multiply a scalar by a vector and implement a dot product
 *
 * Compile:
 *    mpicc -g -Wall -o ex3.9_mpi_vect_mult ex3.9_mpi_vect_mult.c
 * Run:
 *    mpiexec -n <number of processes> ./ex3.4_mpi_vect_mult
 *
 * Input:
```

```
 *    The order of the vectors
 *    Two vectors and a scalar
 *
 * Output:
 *    The dot product of the two vectors
 *    The product of each vector with the scalar
 *
 * IPP:  Exercise 3.9
 */

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

void Read_n(int* n_p, int* local_n_p, int my_rank, int comm_sz,
      MPI_Comm comm);
void Check_for_error(int local_ok, char fname[], char message[],
      MPI_Comm comm);
void Read_data(double local_vec1[], double local_vec2[], double* scalar_p,
      int local_n, int my_rank, int comm_sz, MPI_Comm comm);
void Print_vector(double local_vec[], int local_n, int n, char title[],
      int my_rank, MPI_Comm comm);
double Par_dot_product(double local_vec1[], double local_vec2[],
      int local_n, MPI_Comm comm);
void Par_vector_scalar_mult(double local_vec[], double scalar,
      double local_result[], int local_n);

int main(void) {
   int n, local_n;                                      /*original vector size and local vector size*/
   double *local_vec1, *local_vec2;                     /*local vectors 1,2 of lemngth local_n*/
   double scalar;                                  /*scalar for scalar multiplication*/
   double *local_scalar_mult1, *local_scalar_mult2; /*resultant vectors of scalar multiplication*/
   double dot_product;                                  /*dot product of vectors of length n*/
   int comm_sz, my_rank;
   MPI_Comm comm;

   MPI_Init(NULL, NULL);
   comm = MPI_COMM_WORLD;
   MPI_Comm_size(comm, &comm_sz);
   MPI_Comm_rank(comm, &my_rank);

   /* Read n from user and initialize local_n*/
   Read_n(&n, &local_n, my_rank, comm_sz, comm);

   /* Allocate memory for loca_vec1, localvec2, local_scalar_mult1, local_scalar_mult2*/
   local_vec1 = malloc(local_n*sizeof(double));
   local_vec2 = malloc(local_n*sizeof(double));
   local_scalar_mult1 = malloc(local_n*sizeof(double));
   local_scalar_mult2 = malloc(local_n*sizeof(double));

   /* Read vectors from user and scatter among processes*/
   Read_data(local_vec1, local_vec2, &scalar, local_n, my_rank, comm_sz,  comm);

   /* Print input data */
   if (my_rank == 0)
      printf("\n\n ===== input data =====\n");
   Print_vector(local_vec1, local_n, n, "first vector is", my_rank, comm);
   Print_vector(local_vec2, local_n, n, "second vector is", my_rank, comm);
   if (my_rank == 0){
      printf("scalar is %f\n",scalar);
   }

   /* Print results */
   if (my_rank ==0)
      printf("\n\n ===== result =====\n");

   /* Compute and print dot product */
   dot_product = Par_dot_product(local_vec1, local_vec2, local_n, comm);
   if (my_rank == 0) {
      printf("Dot product is %f\n", dot_product);
   }

   /* Compute scalar multiplication and print out result */
   Par_vector_scalar_mult(local_vec1, scalar, local_scalar_mult1, local_n);
   Par_vector_scalar_mult(local_vec2, scalar, local_scalar_mult2, local_n);
   Print_vector(local_scalar_mult1, local_n, n,
         "product of the first vector with scalar is",
         my_rank, comm);
   Print_vector(local_scalar_mult2, local_n, n,
         "product of the second vector with scalar is",
```

```
         my_rank, comm);

   free(local_scalar_mult2);
   free(local_scalar_mult1);
   free(local_vec2);
   free(local_vec1);

   MPI_Finalize();
   return 0;
}

/*--------------------------------------------------------------------*/
void Check_for_error(
                int       local_ok   /* in */,
                char      fname[]    /* in */,
                char      message[]  /* in */,
                MPI_Comm  comm       /* in */) {
   int ok;

   MPI_Allreduce(&local_ok, &ok, 1, MPI_INT, MPI_MIN, comm);
   if (ok == 0) {
      int my_rank;
      MPI_Comm_rank(comm, &my_rank);
      if (my_rank == 0) {
         fprintf(stderr, "Proc %d > In %s, %s\n", my_rank, fname,
               message);
         fflush(stderr);
      }
      MPI_Finalize();
      exit(-1);
   }
}  /* Check_for_error */


/*--------------------------------------------------------------------*/
void Read_n(int* n_p, int* local_n_p, int my_rank, int comm_sz,
      MPI_Comm comm) {
   int local_ok = 1;

   if (my_rank == 0){
      /* Read n from user */
      printf("What is the order of the vector?\n");
      scanf("%d", n_p);
   }

   /* Broadcast n*/
   MPI_Bcast(n_p, 1, MPI_INT, 0, comm);

   /* Error check*/
   if (*n_p < 0 || *n_p % comm_sz != 0) local_ok =0;
   Check_for_error(local_ok, "Read_n",
         "n should be > 0 and evenly divisible by comm_sz", comm);

   /* Calculate n_p*/
   *local_n_p = *n_p / comm_sz;
}  /* Read_n */


/*--------------------------------------------------------------------*/
void Read_data(double local_vec1[], double local_vec2[], double* scalar_p,
      int local_n, int my_rank, int comm_sz, MPI_Comm comm) {
   double* a = NULL;
   int i;
   if (my_rank == 0){
      /* Read in scalar*/
      printf("What is the scalar?\n");
      scanf("%lf", scalar_p);
   }

   /* Broadcast scalar*/
   MPI_Bcast(scalar_p, 1, MPI_DOUBLE, 0, comm);

   if (my_rank == 0){
      /* Allocate memory for a for local_n*comm_sz double elements*/
      a = malloc(local_n * comm_sz * sizeof(double));

      /* Read in first vector to a*/
      printf("Enter the first vector\n");
      for (i = 0; i < local_n * comm_sz; i++)
         scanf("%lf", &a[i]);
```

```
      /* Scatter vector1 local_n elements per process*/
      MPI_Scatter(a, local_n, MPI_DOUBLE, local_vec1, local_n,
            MPI_DOUBLE, 0, comm);

      /* Read in second vector to a*/
      printf("Enter the second vector\n");
      for (i = 0; i < local_n * comm_sz; i++)
         scanf("%lf", &a[i]);

      /* Scatter vector2, local_n elements per process*/
      MPI_Scatter(a, local_n, MPI_DOUBLE, local_vec2, local_n,
            MPI_DOUBLE, 0, comm);

      free(a);
   } else {
      MPI_Scatter(a, local_n, MPI_DOUBLE, local_vec1, local_n,
            MPI_DOUBLE, 0, comm);
      MPI_Scatter(a, local_n, MPI_DOUBLE, local_vec2, local_n,
            MPI_DOUBLE, 0, comm);
   }
}  /* Read_data */

/*--------------------------------------------------------------------*/
void Print_vector(double local_vec[], int local_n, int n, char title[],
      int my_rank, MPI_Comm comm) {
   double* a = NULL;
   int i;

   if (my_rank == 0) {
      a = malloc(n * sizeof(double));
      MPI_Gather(local_vec, local_n, MPI_DOUBLE, a, local_n,
            MPI_DOUBLE, 0, comm);
      printf("%s\n", title);
      for (i = 0; i < n; i++)
         printf("%.2f ", a[i]);
      printf("\n");
      free(a);
   } else {
      MPI_Gather(local_vec, local_n, MPI_DOUBLE, a, local_n,
            MPI_DOUBLE, 0, comm);
   }

}  /* Print_vector */


/*--------------------------------------------------------------------*/
double Par_dot_product(double local_vec1[], double local_vec2[], int local_n, MPI_Comm comm) {
   int local_i;
   double dot_product, local_dot_product = 0;

   for (local_i = 0; local_i < local_n; local_i++)
      local_dot_product += local_vec1[local_i] * local_vec2[local_i];

   MPI_Reduce(&local_dot_product, &dot_product, 1, MPI_DOUBLE, MPI_SUM, 0, comm);
   return dot_product;
}  /* Par_dot_product */


/*--------------------------------------------------------------------*/
void Par_vector_scalar_mult(double local_vec[], double scalar, double local_result[], int local_n) {
   int local_i;

   for (local_i = 0; local_i < local_n; local_i++)
      local_result[local_i] = local_vec[local_i] * scalar;
}  /* Par_vector_scalar_mult */

/*********************************************output**************************
[rdissanayaka@hpc0 243 ~/CS351/mpi]$ mpicc -o mpi_vect_mult mpi_vect_mult.c
[rdissanayaka@hpc0 244 ~/CS351/mpi]$ mpirun -np 4 mpi_vect_mult
What is the order of the vector?
8
What is the scalar?
2
Enter the first vector
1 1 1 1 1 1 1 1
Enter the second vector
2 2 2 2 2 2 2 2


 ===== input data =====
first vector is
```

```
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00
second vector is
2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00
scalar is 2.000000


 ===== result =====
Dot product is 16.000000
product of the first vector with scalar is
2.00 2.00 2.00 2.00 2.00 2.00 2.00 2.00
product of the second vector with scalar is
4.00 4.00 4.00 4.00 4.00 4.00 4.00 4.00
[rdissanayaka@hpc0 245 ~/CS351/mpi]$
*****************************************************************************/
```

Online Submission Instructions:

Copy the file <<yourname>>Asg1.pdf (e.g. nto /usr/people/handin/CS160 before the deadline posted in the class website.