/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* FILE: omp\_dotprod\_mpi.c

\* DESCRIPTION:

\* This simple program is the MPI version of a dot product and the third

\* of four codes used to show the progression from a serial program to a

\* hybrid MPI/OpenMP program. The relevant codes are:

\* - omp\_dotprod\_serial.c - Serial version

\* - omp\_dotprod\_openmp.c - OpenMP only version

\* - omp\_dotprod\_mpi.c - MPI only version

\* - omp\_dotprod\_hybrid.c - Hybrid MPI and OpenMP version

\* SOURCE: Blaise Barney

\* LAST REVISED: 06/02/17 Blaise Barney

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <mpi.h>

#include <stdio.h>

#include <stdlib.h>

/\* Define length of dot product vectors \*/

#define VECLEN 100

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

{

int i,myid, numprocs, len=VECLEN;

double \*a, \*b;

double mysum, allsum;

/\* MPI Initialization \*/

MPI\_Init (&argc, &argv);

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

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

/\*

Each MPI task performs the dot product, obtains its partial sum, and then calls

MPI\_Reduce to obtain the global sum. \*/

if (myid == 0)

printf("Starting omp\_dotprod\_mpi. Using %d tasks...\n",numprocs);

/\* Assign storage for dot product vectors \*/

a = (double\*) malloc (len\*sizeof(double));

b = (double\*) malloc (len\*sizeof(double));

/\* Initialize dot product vectors \*/

for (i=0; i<len; i++) {

a[i]=1.0;

b[i]=a[i];

}

/\* Perform the dot product \*/

mysum = 0.0;

for (i=0; i<len; i++)

{

mysum += a[i] \* b[i];

}

printf("Task %d partial sum = %f\n",myid, mysum);

/\* After the dot product, perform a summation of results on each node \*/

MPI\_Reduce (&mysum, &allsum, 1, MPI\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

if (myid == 0)

printf ("Done. MPI version: global sum = %f \n", allsum);

free (a);

free (b);

MPI\_Finalize();

}