DHRUV GARG 16 BCE 1190

HIGH PERFORMANCE COMPUTING

ASSIGNMENT

1. Write a program to sum the numbers from 1 to 10000 in sequential program. To parallelize this procedure using MPI, divide the process into sub-processes and write the code. The code should assign different numbers to be summed to different processes, and outputs the sum and sub-sums in different processes.

Code:

MPI_ques1.c (~/Desktop/HPC_assignment) - gedit

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

int main(int argc, char ** argv){
  int id, nproc;
  int sum,startval,endval,accum;
  MPI_Status status;
  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD, &nproc);
  MPI_Comm_rank(MPI_COMM_WORLD, &id);

  sum = 0; // zero sum for accumulation
  startval = 10000*id/nproc+1;
  endval =   10000*(id+1)/nproc;

  for(int i=startval;i<=endval;++i)
    sum = sum + i;
  printf("\tNODE : %d",id);
  printf("\n\tPartial sum is : %d\n",sum);

  if(id!=0) //the slaves sending back the partial sums
    MPI_Send(&sum,1,MPI_INT,0,1,MPI_COMM_WORLD);
  else //id==0! the master recieves the partial sums
    for(int j=1;j<nproc;j=j+1){
      MPI_Recv(&accum, 1, MPI_INT, j, 1, MPI_COMM_WORLD, &status);
      sum = sum + accum;
      printf("\n\tSum till now: %d\n",sum);
    }

  if(id == 0)
    printf("\n\n\tFINAL SUM (1 TO 10000): %d\n",sum);
  MPI_Finalize();
}
```

Output:

```
dhruv@dhruv-Inspiron-5559:~/Desktop/HPC_assignment$ mpicc MPI_ques1.c
dhruv@dhruv-Inspiron-5559:~/Desktop/HPC_assignment$ mpirun -np 4 ./a.out
        NODE : 0
        Partial sum is : 3126250

        Sum till now: 12502500

        Sum till now: 28128750

        Sum till now: 50005000


        FINAL SUM (1 TO 10000): 50005000
        NODE : 1
        Partial sum is : 9376250
        NODE : 2
        Partial sum is : 15626250
        NODE : 3
        Partial sum is : 21876250
```

2. Study the relative performance of regular strided data in the MPI_Send and MPI_Recv. Send a vector of 1000 elements of type MPI_DOUBLE, with a stride of 24 between each element. Test these three forms:

1. Use MPI_Type_vector
2. Use MPI_Type_struct to form a "striding" structure
3. Use MPI_DOUBLE and a loop to pack and unpack the vector yourself (that is, do not use MPI Datatypes)

Do several iterations to get a good average, and repeat the test 10 times.

Code:

MPI_ques2.c (~/Desktop/HPC_assignment) - gedit

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

#define NUMBER_OF_TESTS 10

int main( argc, argv )
int argc;
char **argv;
{
    MPI_Datatype vec1, vec_n;
    int          blocklens[2];
    MPI_Aint     indices[2];
    MPI_Datatype old_types[2];

    double       *buf, *lbuf;
    register double *in_p, *out_p;
    int          rank;
    int          n, stride;
    double       t1, t2, tmin;
    int          i, j, k, nloop;
    MPI_Status   status;

    MPI_Init( &argc, &argv );

    MPI_Comm_rank( MPI_COMM_WORLD, &rank );

    n    = 1000;
    stride = 24;
    nloop  = 100000/n;

    buf = (double *) malloc( n * stride * sizeof(double) );
    if (!buf) {
    fprintf( stderr, "Could not allocate send/recv buffer of size %d\n", n * stride );
    MPI_Abort( MPI_COMM_WORLD, 1 );
    }
    lbuf = (double *) malloc( n * sizeof(double) );
    if (!lbuf) {
    fprintf( stderr, "Could not allocated send/recv lbuffer of size %d\n", n );
```

MPI_ques2.c (~/Desktop/HPC_assignment) - gedit

```
    MPI_Abort( MPI_COMM_WORLD, 1 );
    }
    lbuf = (double *) malloc( n * sizeof(double) );
    if (!lbuf) {
    fprintf( stderr, "Could not allocated send/recv lbuffer of size %d\n", n );
    MPI_Abort( MPI_COMM_WORLD, 1 );
    }

    if (rank == 0)
    printf( "Kind\tn\tstride\ttime (sec)\tRate (MB/sec)\n" );

    /* Use a fixed vector type */
    MPI_Type_vector( n, 1, stride, MPI_DOUBLE, &vec1 );
    MPI_Type_commit( &vec1 );

    tmin = 1000;
    for (k=0; k<NUMBER_OF_TESTS; k++) {
    if (rank == 0) {
     /* Make sure both processes are ready */
     MPI_Sendrecv( MPI_BOTTOM, 0, MPI_INT, 1, 14,
     MPI_BOTTOM, 0, MPI_INT, 1, 14, MPI_COMM_WORLD,
     &status );
     t1 = MPI_Wtime();
     for (j=0; j<nloop; j++) {
     MPI_Send( buf, 1, vec1, 1, k, MPI_COMM_WORLD );
     MPI_Recv( buf, 1, vec1, 1, k, MPI_COMM_WORLD, &status );
     }
     t2 = (MPI_Wtime() - t1) / nloop;
     if (t2 < tmin) tmin = t2;
}
else if (rank == 1) {
     /* Make sure both processes are ready */
     MPI_Sendrecv( MPI_BOTTOM, 0, MPI_INT, 0, 14,
     MPI_BOTTOM, 0, MPI_INT, 0, 14, MPI_COMM_WORLD,
     &status );
     for (j=0; j<nloop; j++) {
     MPI_Recv( buf, 1, vec1, 0, k, MPI_COMM_WORLD, &status );
     MPI_Send( buf, 1, vec1, 0, k, MPI_COMM_WORLD );
     }
```

MPI_ques2.c (~/Desktop/HPC_assignment) - gedit

Open

```
}
    }
    /* Convert to half the round-trip time */
    tmin = tmin / 2.0;
    if (rank == 0) {
     printf( "Vector\t%d\t%d\t%f\t%f\n",n, stride, tmin, n * sizeof(double) * 1.0e-6 / tmin );
    }
    MPI_Type_free( &vec1 );

    /* Use a variable vector type */
    blocklens[0] = 1;
    blocklens[1] = 1;
    indices[0]   = 0;
    indices[1]   = stride * sizeof(double);
    old_types[0] = MPI_DOUBLE;
    old_types[1] = MPI_UB;
    MPI_Type_struct( 2, blocklens, indices, old_types, &vec_n );
    MPI_Type_commit( &vec_n );

    tmin = 1000;
    for (k=0; k<NUMBER_OF_TESTS; k++) {
    if (rank == 0) {
     /* Make sure both processes are ready */
     MPI_Sendrecv( MPI_BOTTOM, 0, MPI_INT, 1, 14,
     MPI_BOTTOM, 0, MPI_INT, 1, 14, MPI_COMM_WORLD,
     &status );
     t1 = MPI_Wtime();
     for (j=0; j<nloop; j++) {
     MPI_Send( buf, n, vec_n, 1, k, MPI_COMM_WORLD );
     MPI_Recv( buf, n, vec_n, 1, k, MPI_COMM_WORLD, &status );
     }
     t2 = (MPI_Wtime() - t1) / nloop;
     if (t2 < tmin) tmin = t2;
}
else if (rank == 1) {
     /* Make sure both processes are ready */
     MPI_Sendrecv( MPI_BOTTOM, 0, MPI_INT, 0, 14,
     MPI_BOTTOM, 0, MPI_INT, 0, 14, MPI_COMM_WORLD,
     &status );
```

File Edit View Search Tools Documents Help

Open

```
     for (j=0; j<nloop; j++) {
     MPI_Recv( buf, n, vec_n, 0, k, MPI_COMM_WORLD, &status );
     MPI_Send( buf, n, vec_n, 0, k, MPI_COMM_WORLD );
     }
}
    }
    /* Convert to half the round-trip time */
    tmin = tmin / 2.0;
    if (rank == 0) {
    printf( "Struct\t%d\t%d\t%f\t%f\n", n, stride, tmin, n * sizeof(double) * 1.0e-6 / tmin );
    }

    MPI_Type_free( &vec_n );

    /* Use user-packing with known stride */
    tmin = 1000;
    for (k=0; k<NUMBER_OF_TESTS; k++) {
    if (rank == 0) {
     /* Make sure both processes are ready */
     MPI_Sendrecv( MPI_BOTTOM, 0, MPI_INT, 1, 14,
     MPI_BOTTOM, 0, MPI_INT, 1, 14, MPI_COMM_WORLD,
     &status );
     t1 = MPI_Wtime();
     for (j=0; j<nloop; j++) {
  /* If the compiler isn't good at unrolling and changing
     multiplication to indexing, this won't be as good as
     it could be */
      for (i=0; i<n; i++)
      lbuf[i] = buf[i*stride];
      MPI_Send( lbuf, n, MPI_DOUBLE, 1, k, MPI_COMM_WORLD );
      MPI_Recv( lbuf, n, MPI_DOUBLE, 1, k, MPI_COMM_WORLD, &status );
      for (i=0; i<n; i++)
      buf[i*stride] = lbuf[i];
     }
     t2 = (MPI_Wtime() - t1) / nloop;
     if (t2 < tmin) tmin = t2;
}
else if (rank == 1) {
     /* Make sure both processes are ready */
     MPI_Sendrecv( MPI_BOTTOM, 0, MPI_INT, 0, 14,
```

MPI_ques2.c (~/Desktop/HPC_assignment) - gedit

```
else if (rank == 1) {
    /* Make sure both processes are ready */
    MPI_Sendrecv( MPI_BOTTOM, 0, MPI_INT, 0, 14,
    MPI_BOTTOM, 0, MPI_INT, 0, 14, MPI_COMM_WORLD,
    &status );
    for (j=0; j<nloop; j++) {
    MPI_Recv( lbuf, n, MPI_DOUBLE, 0, k, MPI_COMM_WORLD, &status );
    for (i=0; i<n; i++)
     buf[i*stride] = lbuf[i];
    for (i=0; i<n; i++)
     lbuf[i] = buf[i*stride];
    MPI_Send( lbuf, n, MPI_DOUBLE, 0, k, MPI_COMM_WORLD );
    }
}
   }
   /* Convert to half the round-trip time */
   tmin = tmin / 2.0;
   if (rank == 0) {
   printf( "User\t%d\t%d\t%f\t%f\n", n, stride, tmin, n * sizeof(double) * 1.0e-6 / tmin );
   }

   /* Use user-packing with known stride, using addition in the user
      copy code */
   tmin = 1000;
   for (k=0; k<NUMBER_OF_TESTS; k++) {
   if (rank == 0) {
    /* Make sure both processes are ready */
    MPI_Sendrecv( MPI_BOTTOM, 0, MPI_INT, 1, 14,
    MPI_BOTTOM, 0, MPI_INT, 1, 14, MPI_COMM_WORLD,
    &status );
    t1 = MPI_Wtime();
    for (j=0; j<nloop; j++) {
 /* If the compiler isn't good at unrolling and changing
    multiplication to indexing, this won't be as good as
    it could be */
    in_p = buf; out_p = lbuf;
    for (i=0; i<n; i++) {
     out_p[i] = *in_p; in_p += stride;
 }
```

MPI_ques2.c (~/Desktop/HPC_assignment) - gedit

```

  MPI_Send( lbuf, n, MPI_DOUBLE, 1, k, MPI_COMM_WORLD );
  MPI_Recv( lbuf, n, MPI_DOUBLE, 1, k, MPI_COMM_WORLD, &status );
  out_p = buf; in_p = lbuf;
  for (i=0; i<n; i++) {
      *out_p = in_p[i]; out_p += stride;
  }
     }
     t2 = (MPI_Wtime() - t1) / nloop;
     if (t2 < tmin) tmin = t2;
 }
 else if (rank == 1) {
     /* Make sure both processes are ready */
     MPI_Sendrecv( MPI_BOTTOM, 0, MPI_INT, 0, 14,
     MPI_BOTTOM, 0, MPI_INT, 0, 14, MPI_COMM_WORLD,
     &status );
     for (j=0; j<nloop; j++) {
     MPI_Recv( lbuf, n, MPI_DOUBLE, 0, k, MPI_COMM_WORLD, &status );
     in_p = lbuf; out_p = buf;
     for (i=0; i<n; i++) {
      *out_p = in_p[i]; out_p += stride;
     }
     out_p = lbuf; in_p = buf;
     for (i=0; i<n; i++) {
      out_p[i] = *in_p; in_p += stride;
  }
  MPI_Send( lbuf, n, MPI_DOUBLE, 0, k, MPI_COMM_WORLD );
     }
 }
    }
    /* Convert to half the round-trip time */
    tmin = tmin / 2.0;
    if (rank == 0) {
 printf( "User(add)\t%d\t%d\t%f\t%f\n",
  n, stride, tmin, n * sizeof(double) * 1.0e-6 / tmin );
    }

    MPI_Finalize( );
    return 0;
}
```

Output:

```
dhruv@dhruv-Inspiron-5559: ~/Desktop/HPC_assignment
dhruv@dhruv-Inspiron-5559:~/Desktop/HPC_assignment$ mpicc MPI_ques2.c
dhruv@dhruv-Inspiron-5559:~/Desktop/HPC_assignment$ mpirun -np 4 ./a.out
Kind     n        stride   time (sec)       Rate (MB/sec)
Vector   1000     24       0.000005         1673.955201
Struct   1000     24       0.000005         1751.732289
User     1000     24       0.000012         683.389654
User(add)         1000     24       0.000009         867.151622
dhruv@dhruv-Inspiron-5559:~/Desktop/HPC_assignment$
```