# include &lt;mpi.h&gt;

# include &lt;stdlib.h&gt;

# include &lt;stdio.h&gt;

void ring\_io ( int p, int id );

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

{

int error;

int id;

int p;

/\*

Initialize MPI.

\*/

MPI\_Init ( &amp;argc, &amp;argv );

/\*

Get the number of processes.

\*/

MPI\_Comm\_size ( MPI\_COMM\_WORLD, &amp;p );

/\*

Get the individual process ID.

\*/

MPI\_Comm\_rank ( MPI\_COMM\_WORLD, &amp;id );

/\*

Print a message.

\*/

if ( id == 0 )

{

printf ( &quot;\n&quot; );

printf ( &quot;RING\_MPI:\n&quot; );

printf ( &quot; C/MPI version\n&quot; );

printf ( &quot; Measure time required to transmit data around\n&quot; );

printf ( &quot; a ring of processes\n&quot; );

printf ( &quot;\n&quot; );

printf ( &quot; The number of processes is %d\n&quot;, p );

}

ring\_io ( p, id );

/\*

Shut down MPI.

\*/

MPI\_Finalize ( );

/\*

Terminate.

\*/

if ( id == 0 )

{

printf ( &quot;\n&quot; );

printf ( &quot;RING\_MPI:\n&quot; );

printf ( &quot; Normal end of execution.\n&quot; );

}

return 0;

}

void ring\_io ( int p, int id )

{

int dest;

int i;

int j;

int n;

int n\_test[5] = { 100, 1000, 10000, 100000, 1000000 };

int n\_test\_num = 5;

int source;

MPI\_Status status;

double tave;

int test;

int test\_num = 10;

double tmax;

double tmin;

double wtime;

double \*x;

if ( id == 0 )

{

printf ( &quot;\n&quot; );

printf ( &quot; Timings based on %d experiments\n&quot;, test\_num );

printf ( &quot; N double precision values were sent\n&quot; );

printf ( &quot; in a ring transmission starting and ending at process 0\n&quot;

);

printf ( &quot; and using a total of %d processes.\n&quot;, p );

printf ( &quot;\n&quot; );

printf ( &quot; N T min T ave T max\n&quot;

);

printf ( &quot;\n&quot; );

}

/\*

Choose message size.

\*/

for ( i = 0; i &lt; n\_test\_num; i++ )

{

n = n\_test[i];

x = ( double \* ) malloc ( n \* sizeof ( double ) );

/\*

Process 0 sends very first message,

then waits to receive the &quot;echo&quot; that has gone around the world.

\*/

if ( id == 0 )

{

dest = 1;

source = p - 1;

tave = 0.0;

tmin = 1.0E+30;

tmax = 0.0;

for ( test = 1; test &lt;= test\_num; test++ )

{

/\*

Just in case, set the entries of X in a way that identifies

which iteration of the test is being carried out.

\*/

for ( j = 0; j &lt; n; j++ )

{

x[j] = ( double ) ( test + j );

}

wtime = MPI\_Wtime ( );

MPI\_Send ( x, n, MPI\_DOUBLE, dest, 0, MPI\_COMM\_WORLD );

MPI\_Recv ( x, n, MPI\_DOUBLE, source, 0, MPI\_COMM\_WORLD, &amp;status );

wtime = MPI\_Wtime ( ) - wtime;

/\*

Record the time it took.

\*/

tave = tave + wtime;

if ( wtime &lt; tmin )

{

tmin = wtime;

}

if ( tmax &lt; wtime )

{

tmax = wtime;

}

}

tave = tave / ( double ) ( test\_num );

printf ( &quot; %8d %14.6g %14.6g %14.6g\n&quot;, n, tmin, tave, tmax );

}

/\*

Worker ID must receive first from ID-1, then send to ID+1.

\*/

else

{

source = id - 1;

dest = ( ( id + 1 ) % p );

for ( test = 1; test &lt;= test\_num; test++ )

{

MPI\_Recv ( x, n, MPI\_DOUBLE, source, 0, MPI\_COMM\_WORLD, &amp;status );

MPI\_Send ( x, n, MPI\_DOUBLE, dest, 0, MPI\_COMM\_WORLD );

}

}

free ( x );

}

return;

}

Output:

RING\_MPI:

C++/MPI version

Measure time required to transmit data around

a ring of processes

The number of processes is 8

Timings based on 10 experiments

N double precision values were sent

in a ring transmission starting and ending at process 0

and using a total of 8 processes.

N T min T ave T max

100 9.509e-06 3.31515e-05 0.000228655

1000 3.2032e-05 5.20968e-05 0.000118246

10000 0.000100839 0.000122524 0.000256236

100000 0.000738693 0.000891536 0.00208874

1000000 0.0100578 0.0112339 0.0211263

RING\_MPI:

Normal end of execution.