Iowa High Performance Computing Summer School 2013

MPI Programming Exercise Set

May 20, 2013

Problem 1 *Parallel "Hello, World!"* Write a parallel version of the "Hello, World!" program using MPI and run it to observe the output. You will need to use the Environment Management routines `MPI_Init`, `MPI_Comm_rank`, `MPI_Comm_size`, and `MPI_Finalize`. Write the program such that the output appears on the screen as `rank 0: ''Hello, World!''`. Run the program using 16 processors. Does the output appear in order of ascending rank? If not, modify your program so that it does.

Problem 2

(a) *Send/Recv Example.* Write a MPI program that passes one integer from process 0 to process `numprocs-1` through each process in between and adds one to it after each `MPI_Recv`. Run your program using 16 processors. Choose 100 for the starting integer.

(b) *Blocking vs. non-blocking Send/Recv.* Compile and run Blaise Barney's `mpi_bandwidth.[c,f]` and `mpi_bandwidth_nonblock.[c,f]` to observe the differences between blocking and non-blocking communications. What do the programs tell you about MPI and its use of the underlying hardware?

file	**url**
mpi_bandwidth.c	`https://computing.llnl.gov/tutorials/mpi/samples/C/mpi_bandwidth.c`
mpi_bandwidth_nonblock.c	`https://computing.llnl.gov/tutorials/mpi/samples/C/mpi_bandwidth_nonblock.c`
mpi_bandwidth.f	`https://computing.llnl.gov/tutorials/mpi/samples/Fortran/mpi_bandwidth.f`
mpi_bandwidth_nonblock.f	`https://computing.llnl.gov/tutorials/mpi/samples/Fortran/mpi_bandwidth_nonblock.f`

Problem 3 *Parallel Dot-Product.* Develop a parallel code to take the scalar product of two $N \times 1$ vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, *i.e.*, $\boldsymbol{x}^T \boldsymbol{y}$. Choose $N = 5N_p$, where N_p is the number of MPI processes. Initialize the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ as $[1, 2, \ldots, N]$. Have the scalar answer stored on all processors. Use $N_p = 16$ processors.

Problem 4 *Parallel Matrix Transpose.* Use MPI to write a parallel program that uses `MPI_Alltoall` of a matrix A. The size of the global matrix A is $10N_p \times 10N_p$, where N_p is the number of MPI processes. Initialize the matrix A as $A[i, j] = i * j$.

(a) First write a serial code that performs the same task on a 160×160 size matrix. Determine the required run time.

(b) Write a parallel version using $N_p = 16$ processors that does the same task. Verify that your transposed matrix is correct. Use `MPI_WTime` to determine the required run time and compare it to the serial case. Is it 16 times faster?