Homework - Butterfly Allreduce

Overview

- Write your own implementation of MPI_Allreduce using the log P butterfly allreduce algorithm.
- Compare the performance of your implementation with the built-in MPI_Allreduce function.

Theoretical Development

Draw node communication diagrams to show how data is transmitted in a 8-node network for a low-to-high butterfly allreduce and a high-to-low butterfly allreduce.

Software Development

Write a subroutine that uses MPI_Send and MPI_Recv to perform a tree-staged butterfly allreduce (see PPMPI page 77).

- Your routine should support the usual MPI_Allreduce parameters, including a pointer to the data, the number of elements, and the communicator.
- Your routine should also accept a parameter specifying whether it should use a high-to-low or a low-to-high bit traversal sequence.
- For simplicity, hardcode the *operation* and the *data type*. Make your reduce operation MPI_SUM and MPI_DOUBLE.

Integrate your routine into a driver program that uses MPI_Allreduce and your allreduce to sum vectors of different sizes on different processor counts. You may constrain execution to powers of two.

Testing and Analysis

Verify the correctness of your implementation:

- Instrument your program to produce output showing the communications occurring at each step of your allreduce operation. Run for both the low-to-high and high-to-low allreduce operations, and verify that the steps performed match your conceptual communication diagrams. (Provide a clear example of your program executing both traversal pattern.)
- Verify that your program does, in fact, properly allreduce numbers for the operation you have selected.

Perform tests examining the performance of your algorithms.

- Instrument your program with MPI_Wtime to produce timing data for your implementations. Consider using a warm-up iteration (either discard the first timing result or run an untimed iteration).
- Examine how increasing the amount of data and increasing the processor count impacts each algorithm. Produce plots showing the following data:
    - Increasing data for fixed processor count (network throughput analysis): Provide a graph showing allreduce time at np=128 by increasing message size, from 1 double (8 bytes) to 1 MB,

for your allreduce variants as well as the MPI_Allreduce function. (Remember, 1 double = 8 bytes.)

  - Increasing processor count for fixed data size (network latency analysis): Provide a graph showing allreduce time for 1 double at np=2,4,8,16,32,64,128.

Then, answer the following questions:

- How does the performance of your implementations compare with MPI_Allreduce?
- Does the bit traversal order matter for your tests?
- How do the vector size (e.g., the network latency and network throughput), and number of processes influence allreduce performance?

Assignment Submission

Gather the following materials:

- the allreduce communication diagrams
- the materials requested in the Testing and Analysis section (plots, comments, answers)

Create a file username-hw4.tar.gz file of your homework directory, including all code and makefiles, but no object files or executables.

Submit the .tar file to the d2L site by the due date, and bring a printed copy of report to lab.