Homework 3: Network and MPI, Due 10/26 11:59 pm

Instructions:

Please create a single PDF for your write up. Please also provide a run.sh script that compiles and runs the code. These are to make it easier for grading.

Please indicate the log files for your program.

Please submit to GitHub Classroom Assignment: https://classroom.github.com/a/I3SdvJz-

On-Paper

1. Network properties. (10pt)

1.1. Consider a k-ary d-cube **without** wrap-around. Show its degree, cost, diameter, bisection bandwidth, arc connectivity in general, and for k=16 and d = 5.

1.2. show that a hypercube is a k-ary d-cube without wrap-around and its degree, cost, diameter, bisection bandwidth, arc connectivity. Justify that wrap-around is not needed.

2. (10 pt) Design an algorithm for all-gather for the 2-D mesh topology, and write the pseudocode for the algorithm. Analyze its complexity (in terms of latency Ts, bandwidth Tw, total processor count p, and message size m.) Assume each node has 1 message of size m.

Programming

3. **MPI Quicksort (10pts, 5 pts for write up)**

Implement the quicksort algorithm described in class as an MPI program. Do not use SIMD or OpenMP. Evaluate its strong scaling speedup and parallel efficiency for an array of 1000000 randomly generated double precision floating point numbers, for nodes from 1 to 8. To increase the computational intensity, re-run the evaluation by comparing log(x_i + 1.0), where x_i is the array element value at position i.

4. MPI Simple matrix-vector multiplication (C[N] = A[N][N] x B[N]) (10points, 5 points for write-up)
Starting with the simple matrix multiplication code from HW 0, re-implement using MPI. Do not use SIMD or OpenMP. Evaluate its strong scaling speedup and parallel efficiency for sizes of A = 4000x4000 and B = 4000x1, both as randomly generated double precision floating point numbers, for nodes from 1 to 8. Evaluate weak scaling performance sizes of A = (p * 1000) x 4000 and B = 4000x1. Present a table as well as a plot for the scaling results. Describe your algorithm in the write up, including partitioning and data movement/communication strategy.

5. PageRank: **(10points, 5 points for write-up)** Matrix-vector multiply is a core operation for PageRank computation using the iterative method, and is invoked iteratively until convergence.

https://en.wikipedia.org/wiki/PageRank (see the iterative method section under “implementation”). Here the matrix A (M on the Wikipedia page) is fixed, and the page range vector B (R on the Wikipedia page) is updated repeatedly, so efficient communication of B is important. Extend your implementation from 4, implement the core component of PageRank as :

$$B(t+1) = A*B(t)$$

Evaluate the strong scaling performance of your algorithm for A = 4000x4000, and B = 4000x1, for 100 iterations, for 1 to 8 nodes, and present a table as well as a plot for the strong scaling results.