Homework 2

Parallel programming with MPI

Deadline: March 1 2017
Broadcast

- A broadcast collective communication is a rooted communication pattern where a data is transferred from a single location toward multiple others locations. Upon completion all the locations hold identical content.
- Such a collective communication can be implemented using different communication patterns. We are interested in their properties, based on the underlying network characteristics.
- Let’s take 3 patterns: a chain, a binomial tree and a two binary tree (depicted on the next page).
Three popular collective topologies

Chain

Binomial tree

Two binary tree

Two-tree algorithms for full bandwidth broadcast, reduction and scan
Peter Sandersa, Jochen Specka, Jesper Larsson Träff
Q1: Implement a pipelined broadcast operation similar to the MPI_Bcast using the 3 logical topologies from the previous slide.

Q2: Varying the number of processors and the size of the data, assess the performance of your implementation (careful on how you measure the performance).

Q3: Are these results close to the model you have built during the first homework?

Request: Implement a collective function that must be called by all processes, and that will not return until the data is completely broadcasted. This function should have a prototype similar to MPI_Bcast. Please name it CS594_mpi_bcast.
\[ U_{i,j}^{n+1} = \frac{1}{4} \left( U_{i-1,j}^n + U_{i+1,j}^n + U_{i,j-1}^n + U_{i,j+1}^n \right) \]

Laplace’s equation - MPI

for \( j = 1 \) to \( j_{\text{max}} \)

for \( i = 1 \) to \( i_{\text{max}} \)

\[ U_{\text{new}}(i,j) = 0.25 \times ( U(i-1,j) + U(i+1,j) + U(i,j-1) + U(i,j+1) ) \]

end for

end for
• Assuming you have a 2 dimensional matrix distributed in row-major format (also called 2d block-cyclic format familiar to ScaLAPACK users), compute the 1000 iterations of the computation of the Laplace equation using multiple MPI processes.
  – Special attention should be paid to minimize the extra memory requirements
  – Hint: the algorithm is highly parallelizable (it should be visible from your performance graphs)
• Originally the matrix is initialized with 0 everywhere except the boundaries (first and last row and first and last column) which are randomly initialized.
• Highlight the performance of your implementation by providing weak (fixed size per process) and strong scaling (fixed problem size independent of the number of processors) results.
• Benchmarking of the Laplace algorithm should be measured excluding the data and MPI initialization.