Parallel Programming 2015, Assignment 3: Parallel Sparse LU Factorization

Deadline: Monday, June 29 before 23:59 hours.

The task in the third and final assignment is to implement Parallel Sparse LU Factorization. After completing and testing your implementation, you will perform a thorough experimental evaluation of your implementation.

The deadline for the assignment is Monday, June 29. The assignment has to be completed individually. You are expected to hand in a tarball containing your source code and an extensive report in PDF format that describes your implementation and contains a thorough experimental evaluation. The assignments can be handed in by e-mail to *krietvel (at) liacs (dot) nl*.

1 Implementation

You need to implement a Parallel Sparse LU Factorization kernel that applies *partial* pivoting. Parallelization must be carried out using the MPI framework. You can assume that only one process will be running per node, so within your parallelized program you do not have to account for multiple processes that execute within the same address space (threading). The program must be written in C/C++.

Describe in your report how you have parallelized the LU Factorization kernel and how you solved the problems you encountered.

2 Experimental Evaluation

The final part of the assignment is to perform a thorough experimental evaluation of your implementation. The target machine will be the DAS-4 cluster at Leiden University. Think about *what* you want to show and *how* you need to show this: so, what experiments do you need to carry out. Examples are: performance / execution time, amount of communication between nodes, memory usage, norm of the result, impact on addition of compute nodes, etc., etc.

As test data, you can again use matrices from the University of Florida Sparse Matrix Collection (`http://www.cise.ufl.edu/research/sparse/matrices/`). We selected the following test matrices:

Simon/raefsky6
Garon/garon1
Grund/poli4
Schenk IBMNA/c-25
Bai/cryg10000

To test your implementation you can generate solution vectors as follows: Given a matrix A, you need to construct the B vector as follows: $B(i) = \text{sum}(i^{th} \text{ row of } A)$. This will cause the solution vector to consists of all ones.