ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

M. Troyer, P. Koumoutsakos
ETH Zürich, HIT G 31.8
CH-8093 Zürich

Set 6 - MPI Communicators

Issued: April 20, 2015
Hand in: April 27, 2015

Question 1: MPI Communicators

The $N \times N$ Lehmer matrix A is defined by $A_{i,j} = \min(i,j)/\max(i,j)$, where $i, j = 1, 2, \ldots, N$. We will here compute the L_∞ norm

$$L_\infty = \max_i \left\{ \sum_j |a_{ij}| \right\} \tag{1}$$

of this matrix distributed across a **two dimensional** grid of MPI processes.

A serial implementation can be found in `mpi_communicators_serial.cpp`.

a) On each process allocate an array of appropriate size for the local tile of the matrix such that the number of processes along both dimensions is as close as possible.

   Assume perfect tiling and make use of the MPI helper function `MPI_Dims_create`. For example, if we have 12 MPI processes, assume that N is divisible by 12. We should then have a grid with 4 processes along one dimension and 3 processes along the other. If we have 13 MPI processes, then 1 along one dimension and 13 along the other.

b) Fill the local sub-matrix on each process with appropriate values.

c) Compute the L_∞ norm of the distributed matrix using row and column communicators.

The full matrix is split in $N_p^x \times N_p^y$ tiles, each having size $N_{loc}^x \times N_{loc}^y$, with $N_{loc}^x = N/N_p^x$ and $N_{loc}^y = N/N_p^y$, respectively.

To compute the L_∞ norm we have to interleave two local and two collective reduction operations.

1. For each block, we compute the local sum along each row. The result of this operation is a vector for each block/process.
2. A parallel reduction is performed with the row communicators, i.e. there will be N_p^y independent reductions `MPI_Reduce(...,MPI_SUM,0,row_comm)`, which collect the final sum on the master processes of `row_comm`.
3. The master processes of `row_comm` perform the `max()` algorithm on their local chunks. This results in a single number.
4. The partial result is reduced along the column communicator.

Note that the last two parts operate only on the left-most N_p^y process.

Solution code in `mpi_communicators.cpp`.

d) Once your code works for small matrix sizes, analyze the scaling for $N = 2^{20} = 1'048'576$.

Unfortunately on Euler we have only 2 nodes for benchmarks, therefore scaling results are limited to only a maximum $N = 2^{17} = 131'072$. For this calculation it is required to use the big memory queue by adding the additional argument `-R "rusage[mem=8192]"` in the submission command.

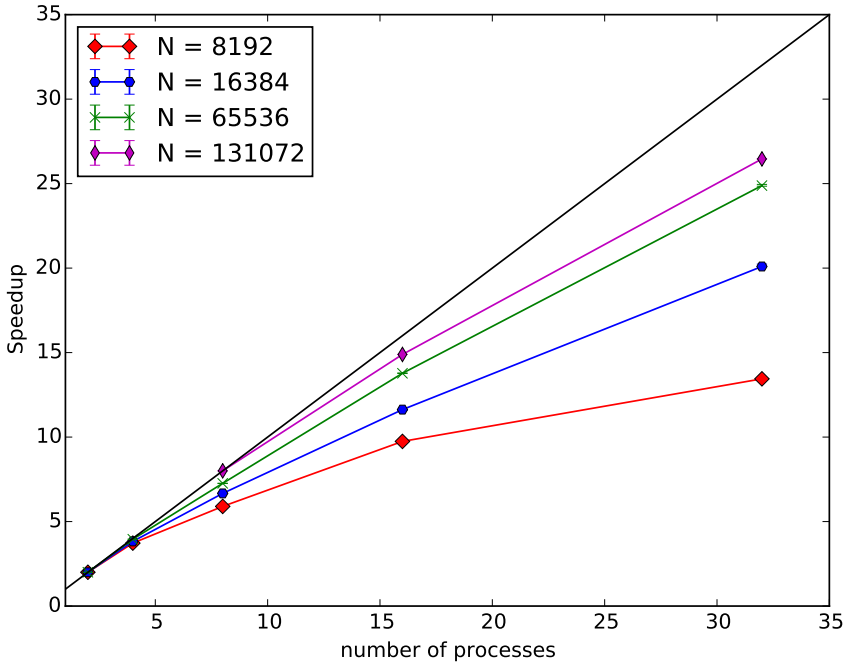

Figure 1: Strong scaling performed on the Euler cluster. The code is compile with OpenMPI and GCC 4.8.2 with options `-Wall -O3 -std=c++11`. As a reference point for each line is the timing with the least number of processes for that system size, e.g. for the largest system only `np` $= 8$ can be computed.

Summary

Summarize your answers, results and plots into a short PDF document. Furthermore, elucidate the main structure of the code and report possible code details that are relevant in terms of accuracy or performance. Send the PDF document and source code to your assigned teaching assistant.