Numerical Simulation and Scientific Computing II
Exercise 1

Eva Doloszeski - 11713043
Yannick Ramic - 11771174
Cameron James Black - 12131828
Benjamin Gruber - 11770987

March 30, 2023

1 Task 1 - Questions

a. *Describe the advantages/ disadvantages of a two-dimensional decomposition (compared to a one-dimensional decomposition)*

Assuming an **N X N** grid and a five-point-stencil..

1. Using a 1-D decomposition the smallest partition possible is into **N/3** layers of size **3 X N** (otherwise the stencil update can not even be performed for one row). In the 2-D decomposition approach, however, parcelling the domain into **3 x 3** smallest units is feasible, enabling utilization of up to **N/3 x N/3 =** $N^2/9$ parallel processes. The higher ratio of inner nodes to boundary nodes and a higher partition for the 2D case makes this approach better scalable and more flexible.

2. On the other hand-side, the 2-D decomposition imposes a second communication round in between iterations compared to the 1-D scheme. This is because every process has to communicate not only with the processes performing the update on the two mesh regions horizontally neighbouring its own, but also with the two additional ones performing the update on the two mesh regions vertically neighbouring its own. Ultimately, this consideration brings up questions about the structure and speed of the communication network. Example: If the network is 4-ported, a 2-D decomposition does not impose additional communication cost, in fact, communication time should decrease, since a lesser number of values has to be communicated per node-node connection. However, if the network is (more realistically) 1- or 2-ported, one would need to consider latency effects in order to determine how much extra cost a second communication round imposes.

3. Last, the 1-D decomposition is row-wise contiguous in memory. For the 2D case, when partitioning or combining the partial solutions strides in memory need to be used. This could possible mean a higher time for acesssing the memory.

b. *Discuss if the decomposition of the domain changes the order of computations performed during a single Jacobi iteration (i.e., if you expect a numerically identical result after each iteration, or not)*

The only values which change in each iteration step for the Jacobi stencil update are the values of the knots in the stencil. For the five-point-stencil, it would mean that the value in the middle knot gets updated with the values of the knots in the north, east, south and west direction. We distinguished two cases for the updating procedure:

1. **First Strategy:** The newly computed values are stored in an separate vector from the old values. If here is insured, for example by ghost layers, that each stencil update has all 4 necessary values of the time step before, the result is numerically identical to the sequential calculation. Each stencil update is an independent calculation which only depends on the old values of four other knots. All of those updates can be run in parallel if to solution vectors for two sequential time steps are stored.

2. **Second Strategy:** Old values in each knot are overwritten by newly computed values (no separate storing of u_t and u_{t-1}. This approach requires less memory, but at the same time yields numerically different solutions. If a sequential update of the entire domain is performed and one iterates from left to right and top to bottom, each knot gets updated with values of time step t_n for the northern and western knot, but for the eastern and southern knot with values of time step t_{n-1}. If the domain is partitioned into smaller elements, for the knots at the left boundary only values from the previous time step will be available for the western knots. However, we need to disclaim that this strategy does not align with the strict formulation of the Jacobi method; consider this a theoretical ex-curs.

c. *A generalization of the ghost layer approach would be to set the width of the ghost layer that is exchanged as a parameter W of the decomposition. This allows to perform W independent iterations before a communication of the ghost layers has to happen. Comment in which situation (w.r.t the available bandwidth or latency between MPI-processes) multiple independent iterations are potentially advantageous*

Increasing the ghost layer width W has several implications: While decreasing the periodicity of communication rounds by enabling more independent iterations in-between, it demands increased memory footage for every process. However, a single communication round comprises a larger amount of data.

1. Transferring more data at once would be well-suited if the machine facilitates high bandwidth communication in between processes. Sending extra chunks of data might induce only little extra communication time if the process is still latency bound and the bandwidth is not fully exploited.

2. On the other side, a high-latency communication network is a good candidate for increasing parameter W. It might actually be sensible to accept the larger memory footprint in order to reduce latency-overhead induced by the number of communication rounds. Manifestation of this case is likeliest for large clusters in which physical distance between two nodes can become quite significant.

3. Assuming a large number of processes, the respecting subdomains decrease in size. Latency becomes more relevant in this case, since message size decreases. It might therefore be advantageous to increase W as an attempt to reduce latency-induced overhead.

d. *Assume a ghost layer with width W=1 (this is what you will later implement) and discuss if a data exchange between parts of the domain which are "diagonal neighbors" is required assuming a "5-point star-shaped stencil"*

1. Since the stencil for the computation of the value of a knot only requires the values of its northern, eastern, southern and western neighbours, but not those of the north-east, north-west, ... , this is not necessary.

e. *How big is the sum of all L2 caches for 2 nodes of the IUE-cluster*

1. 1 regular compute node consists of 2x INTEL Xeon Gold 6248, 2.5GHz
2. INTEL Xeon Gold 6248 (20 pyhsical cores); Cache Sizes: L1: 1024 KB, L2: 16.0 MB, L3: 28 MB
3. Assuming that 10 cores share L2 cache, such that both L2 caches are subordinate to the large L3 cache, it follows that the all L2 caches for 2 nodes in the IUE-cluster together can store: $2 * 2 * 16MB = 64MB$

2 Task 2 - One-Dimensional Decomposition

Implementation of a one-dimensional decomposition using a ghost layer and MPI-communication to update the ghost layers.

In the 1D case the processors were rowed after each other and each one received a subdomain. For this subdomain, the initial knot values including the boundary conditions and the right hand side were initialised on each processor. For the number of iterations, a communication round for the updating of the ghost layers was performed as well as the stencil update for the entire subdomain. After all iterations, the solution was gathered at the root processor for further processing. Note that, the total number of rows of the problem are divided among all processors, however the last processor gets the more rows if they cannot be divided evenly.

2.1 Performance Benchmark

For different resolutions and with utilization of 1-40 MPI-processes on one node of the IUE-Cluster.

The following section contains plots showing the parallel speed up and parallel efficiency of our 1D decomposition Jacobi solver for the resolutions 125, 250, 1000 and 2000. These values are for an iteration number of 100.

The parallel speed-up was calculated according to $S = \frac{T_s}{T}$, with T_s being the average serial runtime per iteration and T being the average parallel runtime per iteration. For all four resolutions we observe a noticeable initial speed-up which at some point plateaus before tapering off. While an unbounded linear speed-up would be ideal, Ahmdahl's law shows that there is a theoretical upper limit for speed-up. When we consider a communication overhead that increases with the number of MPI-processses used, the tapering off at high MPI-Processor numbers is plausible. It is also noteworthy that our code captures time around the MPI block, instead of inside it, which could contribute to the tapering off of speed-up.
The parallel efficiency was calculated according to $\epsilon = \frac{S}{n}$, with S being the parallel speed-up and n being the number of processes. Here we see a decline of efficiency with an increase in MPI-processes.

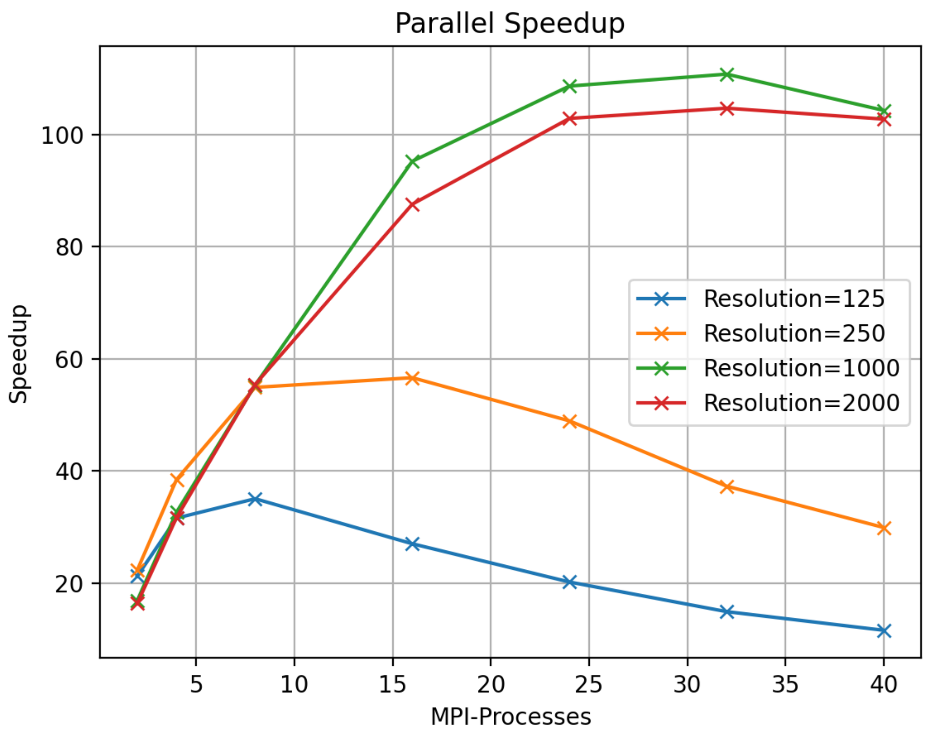

Figure 1: 1D Parallel speedup for different resolutions

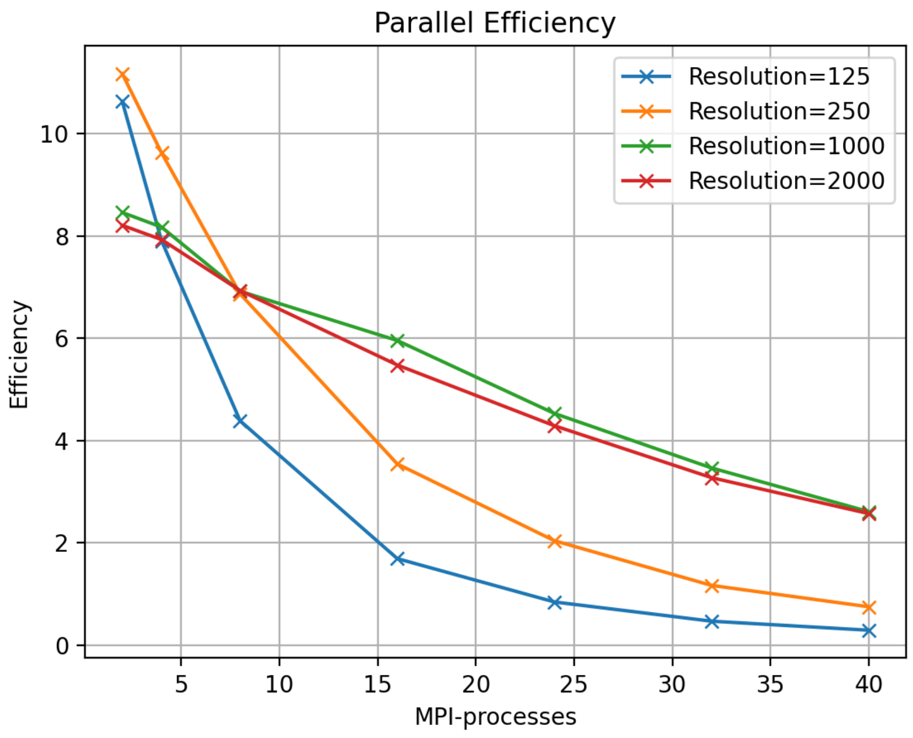

Figure 2: 1D Parallel efficiency for different resolutions

3 Task 3 - Two-Dimensional Decomposition

Implementation of a two-dimensional decomposition using a ghost layer and MPI-communication to update the ghost layers.

For the 2D decomposition the processors are organised in a Cartesian grid that allows slicing along the vertical axis. This difference compared to the 1D decomposition requires slightly more sophisticated approaches towards..

1. **Gathering operations to root:** The solution is first gathered row-wise (with respect to the Cartesian communicator). After this step the $coord = (0, y0)$ process holds all information from $coord = (., y0)$. In order to gather along the $coord = (0, .)$ processes, we create new communicator and use MPI-Gatherv.

2. **The stencil update:** For each processor it must be checked if there is a neighbour in one of the directions (north, east, south, west) to know whether data must be used from the ghost layers. For the 1D case this check was only done in northern and southern direction. Additionally, for the 2D case, special corner cases exist, where the data from two ghost layers is needed. For example the knot in the left upper corner needs values of the norther and western ghost layer (of course only if there is no pouter boundary of the domain).