Università della Svizzera italiana | **Institute of Computing CI**

High-Performance Computing **2022**

Student: Nicolai Hermann Discussed with: FULL NAME

Solution for Project 5

Due date: 30.11.2022, 23:59

This project will introduce you a parallel space solution of a nonlinear PDE using MPI.

1. Task 1 - Initialize and finalize MPI [5 Points]

The classical `MPI_Init(&argc, &argv)`, `MPI_Comm_rank` and `MPI_Comm_size` were used for the initialization. To finalize, the `domain.comm_cart` was freed and `MPI_Finalize` was called afterwards.

2. Task 2 - Create a Cartesian topology [10 Points]

The dimension decomposition was achieved with the helper method `MPI_Dims_create` analogous to the mandel task from the last assignment. To create the Cartesian communicator the respective create method was called and the coords of the current rank were obtained with `MPI_Cart_coords`. The coordinates of the neighbours were returned by the `MPI_Cart_shift` function. Counter intuitively, the north boundary accesses the last row of the array which requires the shift to move `-1` instead of `+1` along the x-axis.

3. Task 3 - Change linear algebra functions [5 Points]

In both cases `MPI_Allreduce` was chosen where the single doubles are reduced by summing over all of them. The result is back propagated to all ranks. We also only need to gather the results for those two functions since they are the only ones that need to access data from other ranks. For example when scaling a vector by a constant we can alter every entry without knowledge of any other entries of the vector. Therefore they can be done individually without synchronization.

4. Task 4 - Exchange ghost cells [45 Points]

The exchange for the remaining borders is pretty much a continuation of the given implementation for the north boundary, with the adjustment of the correct variables and accesses for the buffer. To receive all boundaries we can wait until the main for loop for the inner grid cells is completed and call `MPI_Allwait` afterwards. After the inner grid cells are computed we need the data to do the computations that require the ghost cells.

5. Task 5 - Testing [20 Points]

In Fig. 1 the resulting image can be seen after implementing and debugging everything. After that the computation time for multiple combinations of grid sizes, number of threads, and number of ranks was measured. It must be noted that the cluster was very inconsistent. Rerunning the same experiment on different days returned different results and outliers were often observed throughout the experiment.

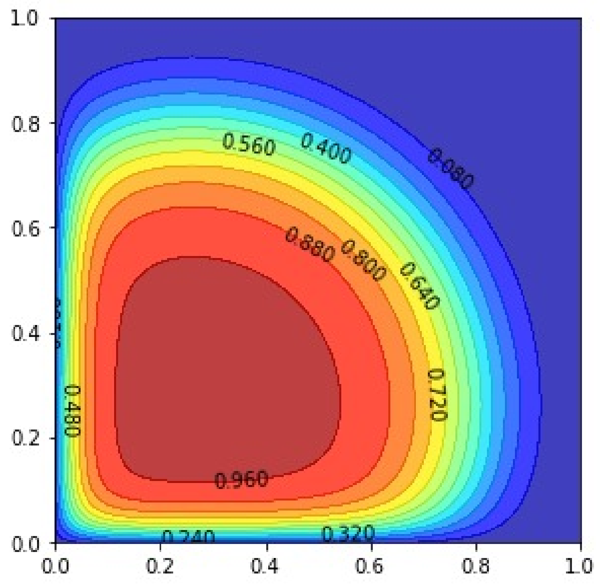

Figure 1: The resulting output image after running the openmpi parallel code.

In Fig. 2 the results of the experiment are shown where each subplot depicts the run time for different grid sizes and numbers of threads per process for a fixed number of processes in each subplot. When there are one or four nodes in use a clear boost of performance can be observed as the number of threads increases. Unsurprisingly the larger grid size correlates with the execution time. Using more nodes had an overall negative impact on the execution time which was surprising. But maybe there wasn't enough computation between the synchronization steps to really get a benefit from distributing the computations over multiple nodes. Especially the plots for the two and four processes are very incoherent. The plot for two processes looks like parallelization overheads accumulate as the number of threads increase and slow down the computation. But this is not resembled at all in the plot for four processes. The best explanation for those results is indeterministic latencies in the communication between the cluster nodes.

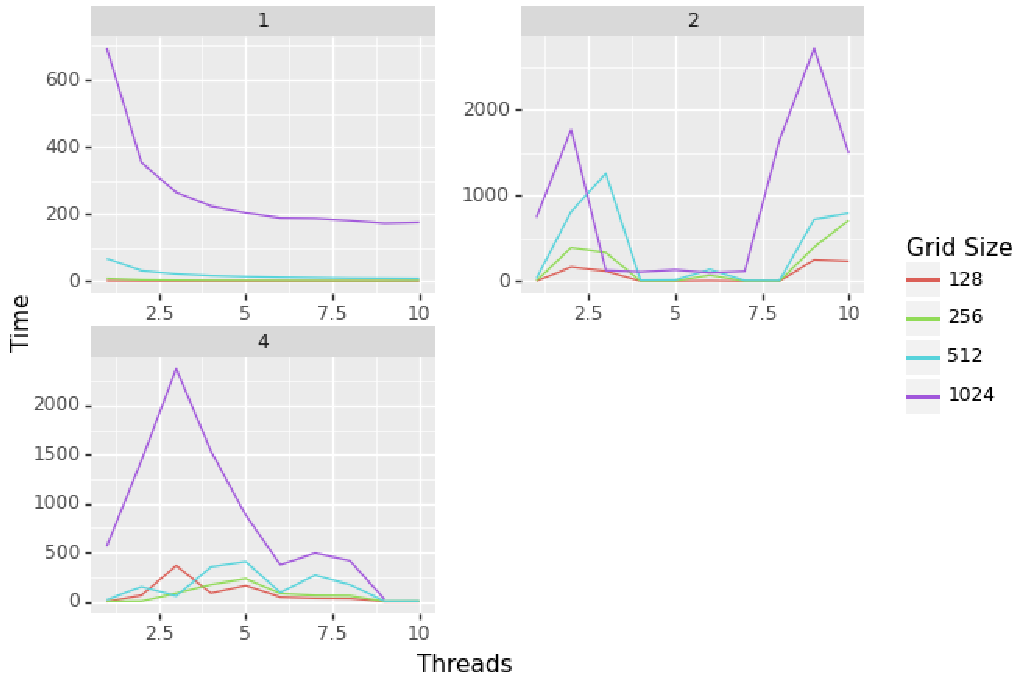

Figure 2: The figure shows plots for different numbers of processes where the computation time was measured with additionally varying numbers of threads per process and different grid sizes.

6. Task 6 - Quality of the Report [15 Points]

Additional notes and submission details

Submit the source code files (together with your used `Makefile`) in an archive file (tar, zip, etc.) and summarize your results and the observations for all exercises by writing an extended Latex report. Use the Latex template from the webpage and upload the Latex summary as a PDF to iCorsi.

- Your submission should be a gzipped tar archive, formatted like project_number_lastname_firstname.zip or project_number_lastname_firstname.tgz. It should contain:
  - all the source codes of your MPI solutions.
  - your write-up with your name project_number_lastname_firstname.pdf,
- Submit your .tgz through Icorsi.