Portable I/O Data Format Tutorial

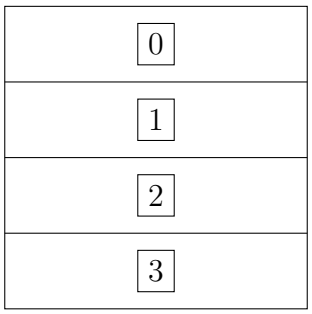
Date: 06.11.2024

Diffusion equation

You'll be working with the provided code in:
`/p/project1/training2403/ParIO_course_material/exercises/Tutorial/`
The code solves the heat diffusion equation $\partial_t f = \kappa \left(\partial_{xx} + \partial_{yy}\right) f$ in two dimensions. Your task is to write the subroutines for creating checkpoint files, such that it enables the simulation to be resumed at a later time point.

Checkpoint/restart file is a common strategy to allow simulations to run for an arbitrarily long time, often a workaround to the imposed hard time limit of continuous computations on HPC systems. It is also crucial to avoid data losses due to unexpected crashes. This works by writing the computed data onto the file system as a file, which can be read into memory at a later time in order to resume the simulation. Checkpoint file is also vital to prevent data loss due to unexpected termination of simulation.

The template for the 2D heat diffusion simulation code is provided in C & Fortran language. The code itself is already MPI parallelized, and you can focus purely on structuring the data on every MPI process to be written into a single file. The compiled program runs on 4 (8) MPI processes. The dimension of the 2D grid is 1000×1000 points. Each process updates the temperature values of their own 125 (250) $\times$ 1000 grid extents, and their ranks are distributed following the diagram below.



The simulation of the provided code reaches convergence in approximately 3300 steps. Correct implementation of the write/read subroutines should yield the converged results after similar total number of iterations are completed, regardless if you pause (write simulation data output) and resume (read in the prior written results) the simulation at an arbitrary iteration step.

Your specific task is to code up the writing subroutine, storing the temperature data of every gridpoint of every MPI processes in a single file called `checkpoint_<steps>.[nc/h5]` at every 1000 steps. As the file extension suggests, you can choose either NetCDF or HDF5 file format for your implementation. A template for hdf5 implementation is provided. Then, code up the reading subroutine such that the generated files can be read into memory, and resume the simulation from the specific chosen file. You can check the correctness of your write/read subroutines by comparing the globally averaged temperature at convergence.