Parallel computing : example of a 2D heated back facing step flow

Here, the namelist “Domain_Features” related to the computational domain of our example is completed with data for OpenMP or MPI computing.

For an OpenMP parallel computing

The variable “Number_OMP_Threads” must be added to the namelist and set to a value corresponding to number of threads wanted.

```
  &Domain_Features Geometric_Layout              = 0,         !--- Option
value for a cartesian geometry
                   Start_Coordinate_I_Direction   = 0.0  ,     !--- Start
coordinate along the I-direction
                   End_Coordinate_I_Direction     = 12.  ,     !--- End
coordinate along the I-direction
                   Start_Coordinate_J_Direction   = 0.0  ,     !--- Start
coordinate along the J-direction
                   End_Coordinate_J_Direction     = 2.0  ,     !--- End
coordinate along the J-direction
                   Start_Coordinate_K_Direction   = 0.00 ,     !--- Start
coordinate along the K-direction
                   End_Coordinate_K_Direction     = 0.00 ,     !--- End of
the domain along the K-direction
                   Cells_Number_I_Direction       = 384 ,      !--- Number
of cells along the I-direction (not counting the ghost cells)
                   Cells_Number_J_Direction       = 128 ,      !--- Number
of cells along the J-direction (not counting the ghost cells)
                   Cells_Number_K_Direction       = 1,         !--- Number
of cells along the K-direction (not counting the ghost cells)
                   Regular_Mesh                   = .true. ,   !--- The grid
is regular for any direction
                   Number_OMP_Threads             = 4      /   !---
Computation performed by 4 threads
```

In this case, the code automatically split the loops associated to the computational grid in parts along the last direction preferentially. Each part is linked to a thread.

For a MPI parallel computing

We must set the number of MPI processes used and describe how each MPI process is linked to a specific subdomain. Two ways are possible :

- Using the MPI cartesian topology : the MPI processes are ordered as a regular cartesian grid on which the domain is divided.
- Using the MPI graph topology : The connections between MPI processes are not necessary regular as for the the MPI cartesian topology. This allows the user to define more complex domain decompositions in order to remove the solid parts of the physical domain out of the computational domain.

For any MPI parallel computing, the mesh size data set is applied for each subdomain, not the entire domain. In this way, the load balancing between MPI processes is better ensured. Note that for irregular cell sizes, subdomain sizes can also differ even though the subdomain cell number is identical.

Example for a MPI cartesian topology

We here consider 4 MPI processes, 2 along each direction. The cell size is the same as previously reported.

```
  &Domain_Features Geometric_Layout               = 0,         !--- Option
value for a cartesian geometry
                   Start_Coordinate_I_Direction   = 0.0  ,     !--- Start
coordinate along the I-direction
                   End_Coordinate_I_Direction     = 12.  ,     !--- End
coordinate along the I-direction
                   Start_Coordinate_J_Direction   = 0.0  ,     !--- Start
coordinate along the J-direction
                   End_Coordinate_J_Direction     = 2.0  ,     !--- End
coordinate along the J-direction
                   Start_Coordinate_K_Direction   = 0.00 ,     !--- Start
coordinate along the K-direction
                   End_Coordinate_K_Direction     = 0.00 ,     !--- End of
the domain along the K-direction
                   Cells_Number_I_Direction       = 192 ,      !--- Number
of cells along the I-direction (not counting the ghost cells)
                   Cells_Number_J_Direction       = 64  ,      !--- Number
of cells along the J-direction (not counting the ghost cells)
                   Cells_Number_K_Direction       = 1,         !--- Number
of cells along the K-direction (not counting the ghost cells)
                   Regular_Mesh                   = .true. ,   !--- The grid
is regular for any direction
                   MPI_Cartesian_Topology         = .true. ,   !--- MPI
cartesian topology enabled
                   Total_Number_MPI_Processes     = 4,         !--- Total
number of MPI processes
                 Max_Number_MPI_Proc_I_Direction= 2,          !--- Number
of MPI processes along the I-direction
                 Max_Number_MPI_Proc_J_Direction= 2,          !--- Number
```

```
of MPI processes along the J-direction
                Max_Number_MPI_Proc_K_Direction= 1 /          !--- Number
of MPI processes along the K-direction
```

Example for a MPI graph topology

We here consider 48 MPI processes, 12 along the I-direction and 4 along the J-direction. The cell size is the same as previously reported.
If we use the MPI cartesian topology, we notice that 4 MPI-processes are associated to the solid part of the step. In this case, it is useful to use the MPI graph topology for excluding the 4 pointless processes. The number of MPI processes is reduced from 48 to 44. The corresponding data setup is :

```
  &Domain_Features Geometric_Layout              = 0,         !--- Option
value for a cartesian geometry
                   Start_Coordinate_I_Direction   = 0.0  ,     !--- Start
coordinate along the I-direction
                   End_Coordinate_I_Direction     = 12.  ,     !--- End
coordinate along the I-direction
                   Start_Coordinate_J_Direction   = 0.0  ,     !--- Start
coordinate along the J-direction
                   End_Coordinate_J_Direction     = 2.0  ,     !--- End
coordinate along the J-direction
                   Start_Coordinate_K_Direction   = 0.00 ,     !--- Start
coordinate along the K-direction
                   End_Coordinate_K_Direction     = 0.00 ,     !--- End of
the domain along the K-direction
                   Cells_Number_I_Direction       = 32  ,      !--- Number
of cells along the I-direction (not counting the ghost cells)
                   Cells_Number_J_Direction       = 32  ,      !--- Number
of cells along the J-direction (not counting the ghost cells)
                   Cells_Number_K_Direction       = 1,         !--- Number
of cells along the K-direction (not counting the ghost cells)
                   Regular_Mesh                   = .true. ,   !--- The grid
is regular for any direction
                   MPI_Graphic_Topology           = .true. ,   !--- MPI
graph topology enabled
                   Total_Number_MPI_Processes     = 44,         !--- Total
number of MPI processes
                 Max_Number_MPI_Proc_I_Direction= 12,          !--- Maximum
number of MPI processes along the I-direction
                 Max_Number_MPI_Proc_J_Direction= 4,           !--- Maximum
number of MPI processes along the J-direction
                 Max_Number_MPI_Proc_K_Direction= 1 /          !--- Maximum
number of MPI processes along the K-direction
```

- Note that the product of “Maximum number of MPI processes” per direction corresponds to the number of total MPI processes that would be used in MPI cartesian topolgy.

- Using the MPI graph topology is not so easy. The information about the connection between the MPI processes must be provided to the code SUNFLUIDH. In other words, this file explains how is defined the spatial distribution of subdomains. This information is contained in a data file named “data_mpi_subdomain_layout.dat” (described hereafter).

Description of the file “data_mpi_subdomain_layout.dat”

This ASCII file must be read by the code SUNFLUIDH to get information on the MPI process connection when the MPI graph topology is used. In our example it reads as :

```
44      Number of enabled process
 -
01      MPI K-plane :
 -
1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1
0 0 1 1 1 1 1 1 1 1 1 1
0 0 1 1 1 1 1 1 1 1 1 1
```

The file provides a map of enabled processes (1) and disabled processes (0) in regard to the maximum number of MPI processes that will be used in MPI cartesian topology (here 48). This map is defined for each plan of MPI processes along the K-direction given by the variable “Max_Number_MPI_Proc_K_Direction”(here 1).
This file can be automatically created by using the homemade software “mpi_subdomain_decomposition”. It analyses the sunfluidh data file initially defined for a MPI cartesian topology and provides the output file “data_mpi_subdomain_layout.dat”. Click here for more details.