A HPC approach to the Boundary Conditions for the Copernicus biogeochemical model of the Mediterranean Sea

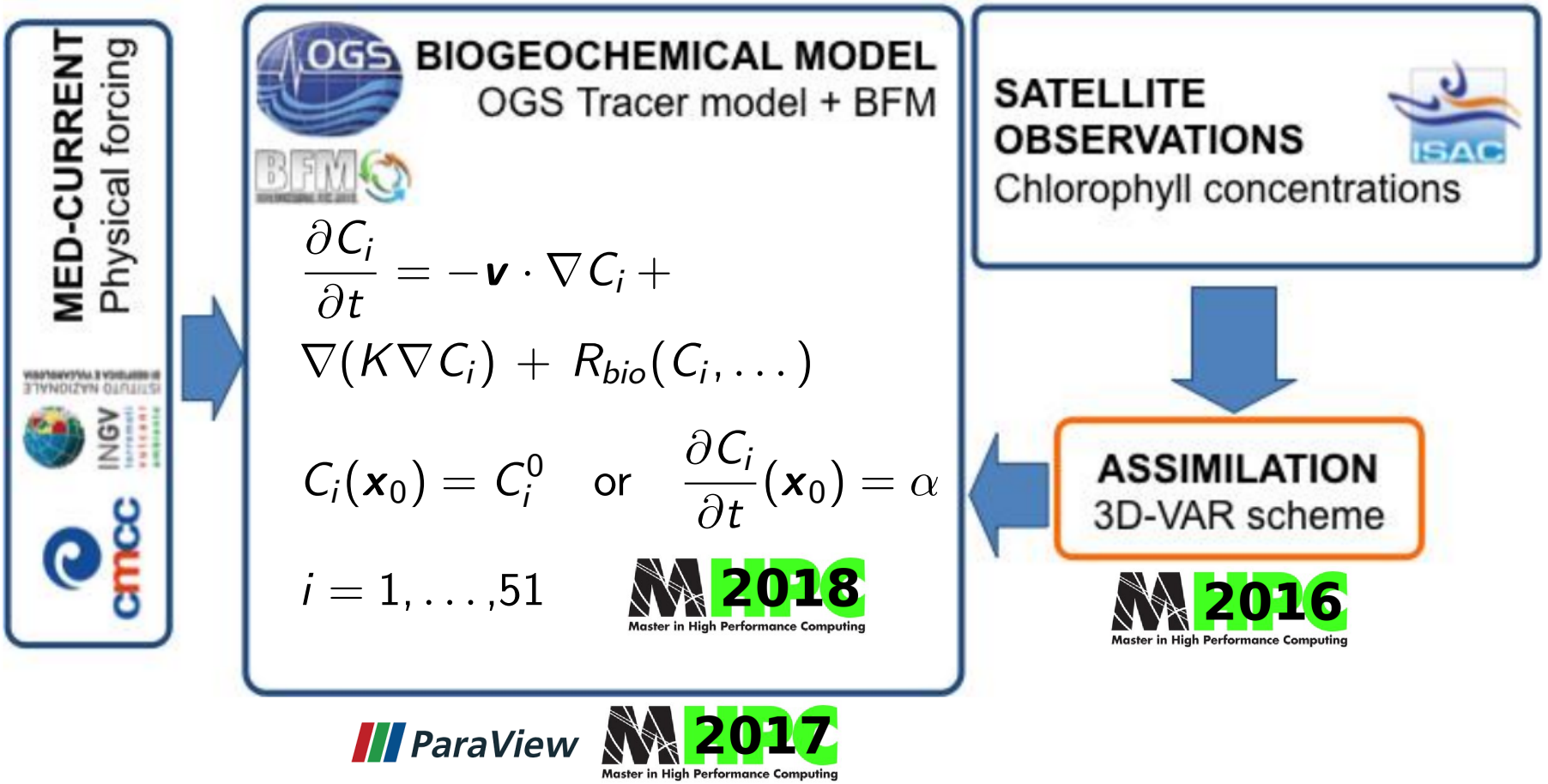
Student: Marco Bettiol[1]
Tutors: Gianpiero Cossarini[1], Alberto Sartori[2]

www.ogs.trieste.it

www.cineca.it

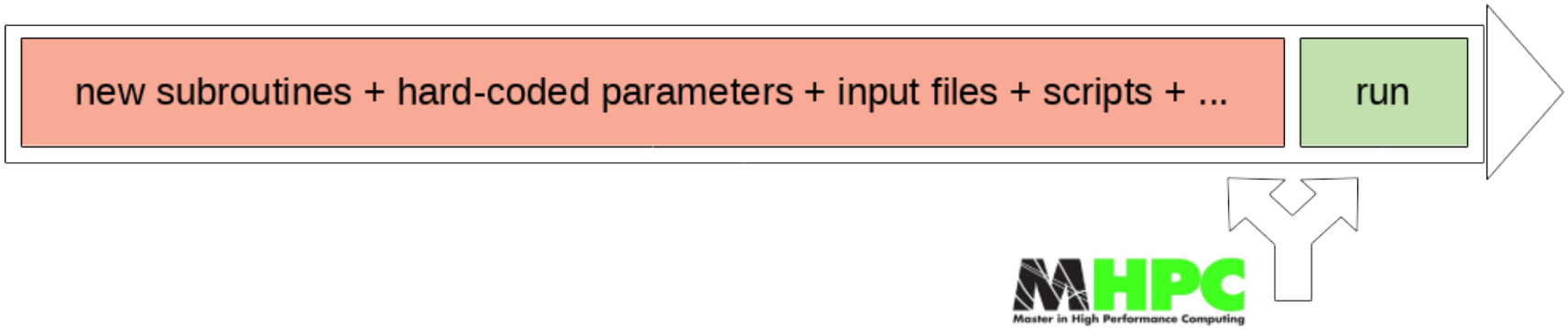
[1]Istituto Nazionale di Oceanografia e di Geofisica Sperimentale - OGS - Trieste
[2]Scuola Internazionale di Studi Superiori Avanzati - SISSA - Trieste

Model overview



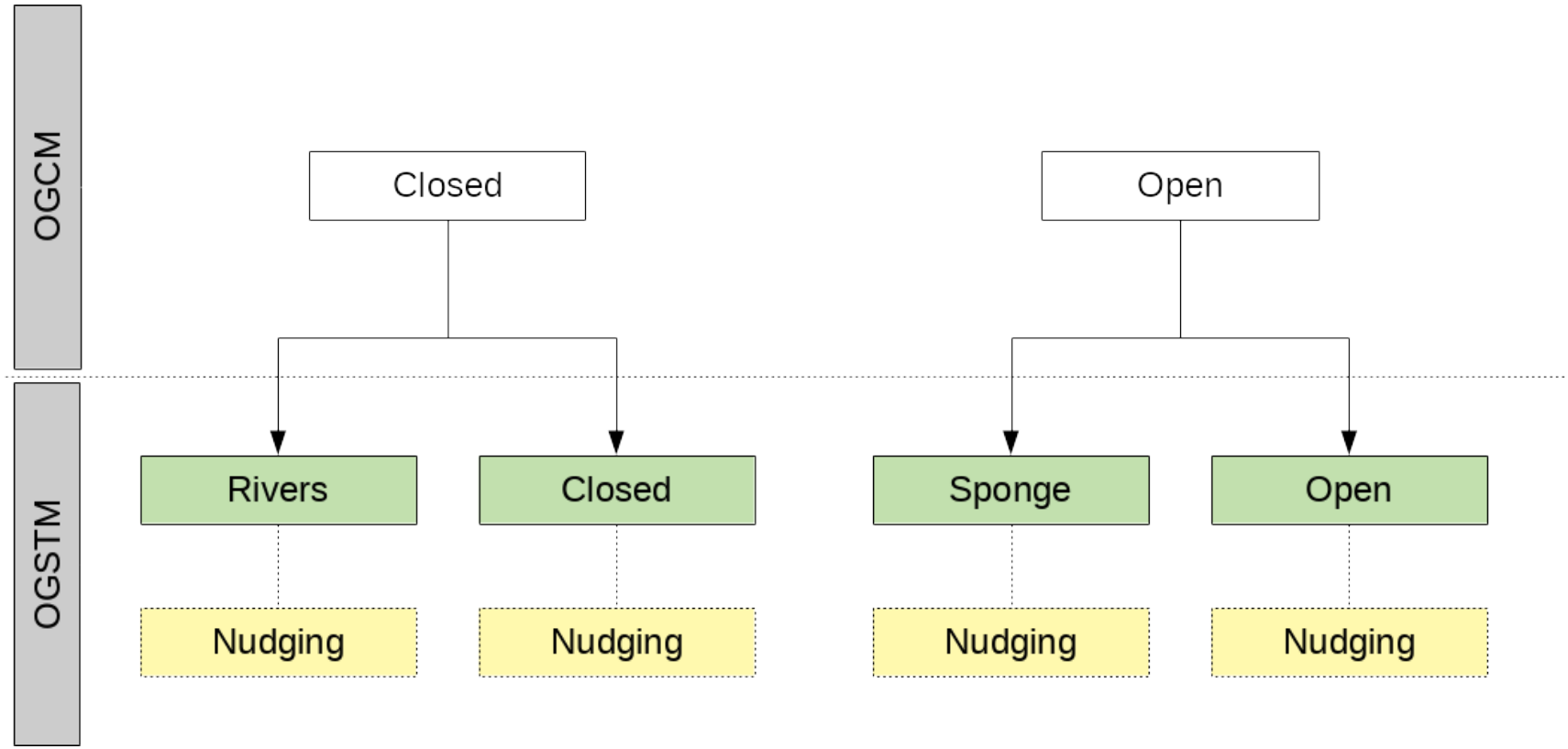
Motivation

- **Different domains** and wider **community** of users;
- unlock the **HPC potential** of the model to run **ensemble simulations**:
  - implement a flexible sensitivity/calibration framework to get more accurate model output;
  - couple the Mediterranean forecast system with global or adjacent CMEMS forecast systems;
  - uncertainty quantification;
  - calibration with a Reduced Basis approach.

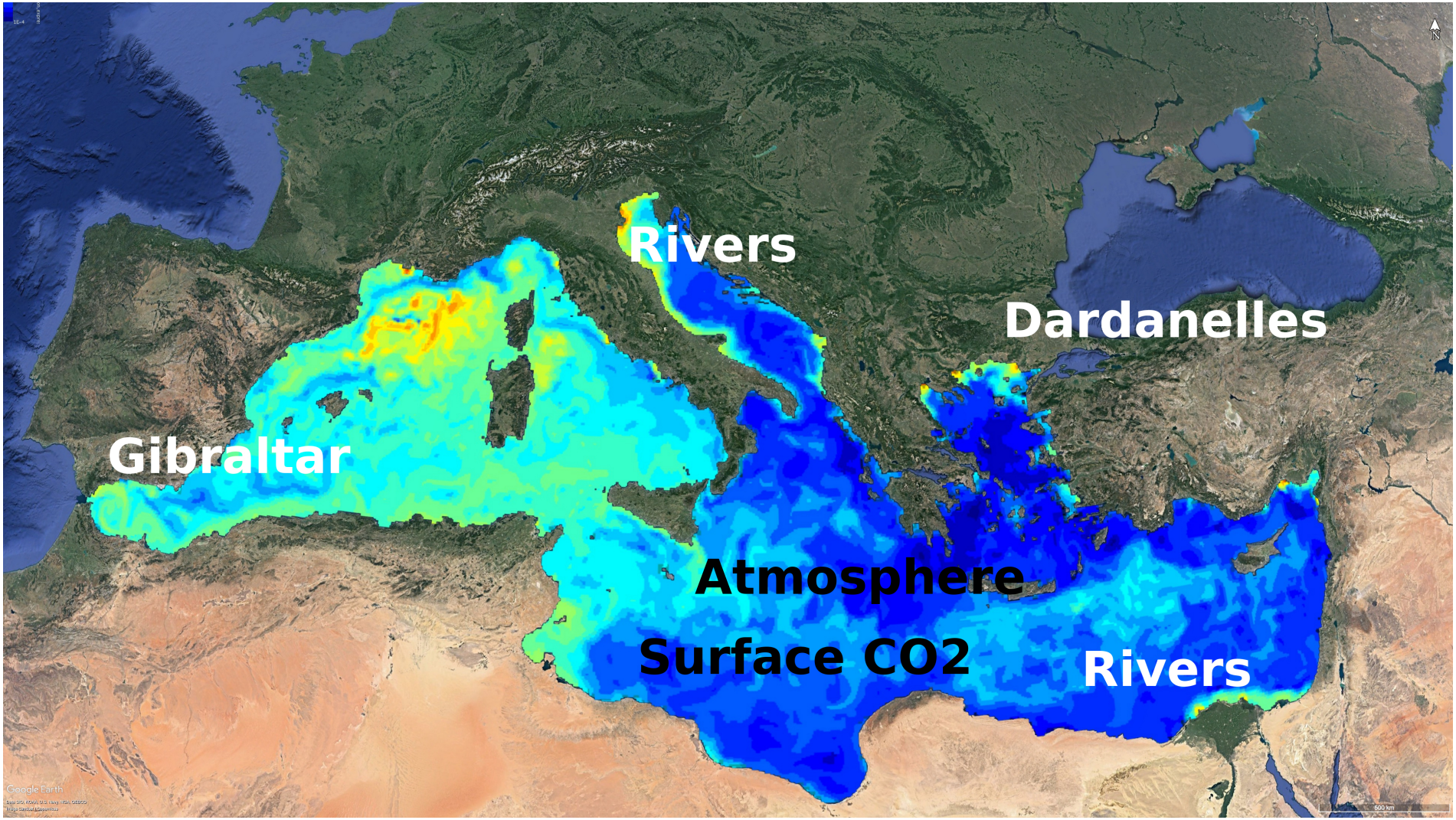


The first step is to build a **versatile interface and infrastructure** to handle the boundary conditions.

BC classification



BC overview in current configuration



New interface

boundaries.nml

```
3
"riv , RIV , riv.nml , files_namelist_riv.dat , T , F"
"gib , SPO , gib.nml , files_namelist_gib.dat , T , T"
"dar , OPE , dar.nml , files_namelist_dar.dat , T , F"
```

Rivers

riv.nml

```
&VARS_DIMENSION
     n_vars = 6
/
&CORE
     vars(1) = "N1p"
     [...]
     vars(6) = "O2o"
     var_names_idx(1) = 2
     [...]
     var_names_idx(6) = 1
/
```

Sponge

gib.nml

```
&VARS_DIMENSION
     n_vars = 7
/
&CORE
     vars(1) = "O2o"
     [...]
     var_names_idx(1) = 1
     [...]
     alpha = 4.0d0
     reduction_value_t = 1.0d-6
     length = -7.5d0
/
```

Nudging

gib.nml

```
&NUDGING_VARS_DIMENSION
     n_vars = 7
/
&NUDGING_CORE
     data_file = "bounmask.nc"
     vars(1) = "O2o"
     [...]
     var_names_idx(1) = 1
     [...]
     rst_corr(1) = 1.0d0
     [...]
/
```

Open

dar.nml

```
&VARS_DIMENSION
     n_vars = 5
/
&CORE
     vars(1) = "N1p"
     [...]
     var_names_idx(1) = 2
     [...]
     geometry = 1
     damping_coeff = 600.0d0
/
```

Files

files_namelist.nml

```
12
'BC/TIN_yyyy0115-00:00:00.nc'
'BC/TIN_yyyy0215-00:00:00.nc'
[...]
'BC/TIN_yyyy1215-00:00:00.nc'
```

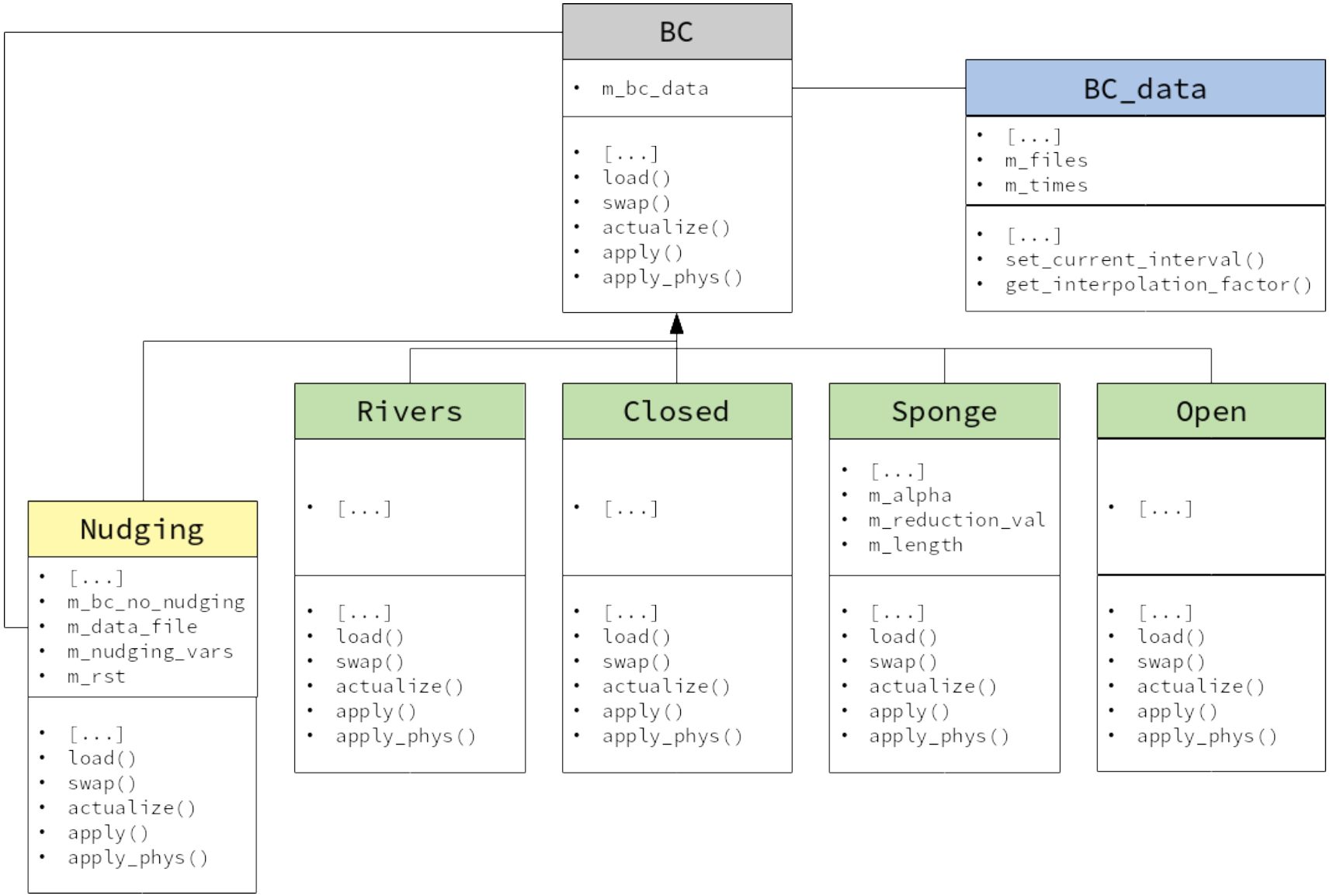
Files namelists are automatically generated by the `genInputsDatelists.sh` script, accordingly to the boundary namelists.

Current code profiling

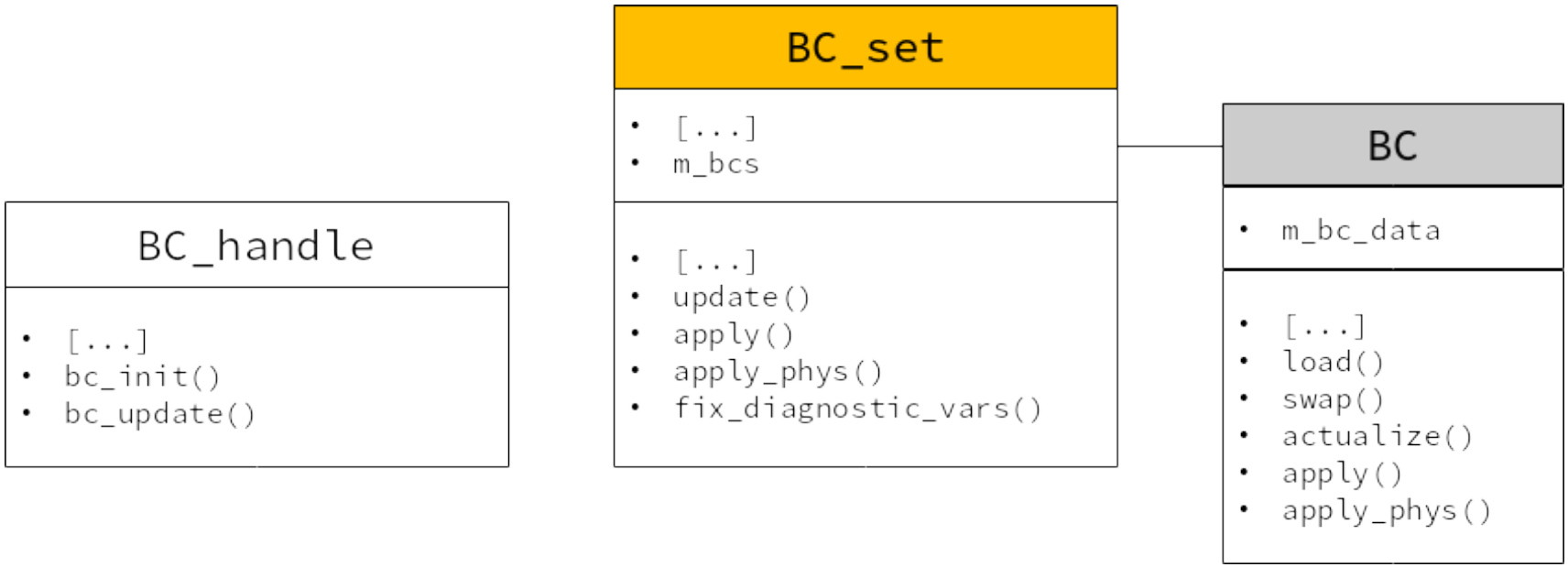
Subroutine	Overall time (s)	Single call time (s)
bcTIN	1.97 e-01	1.97 e-05
bcGIB	1.13 e-01	1.13 e-05
bcATM	1.71 e-01	1.71 e-05
bcCO2	1.08 e-01	1.08 e-05
trcstp	6.29 e+03	6.29 e-01
stp	1.22 e+04	1.22 e+00

Table: operative chain subroutines elapsed times for current version: boundaries and main integration step

New BC OO structure I

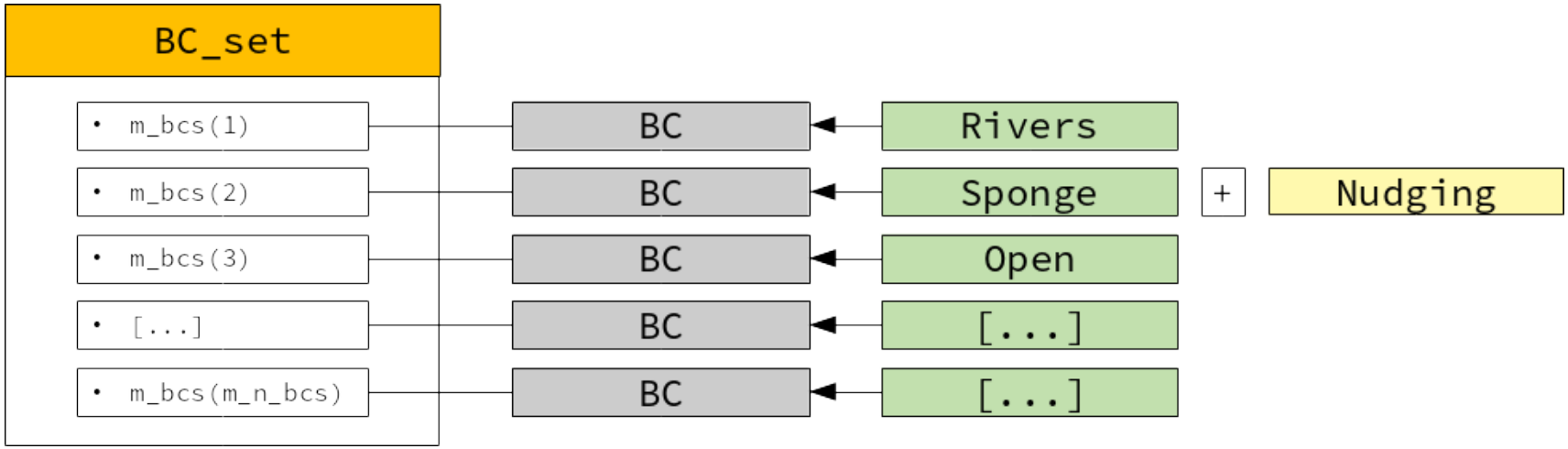


New BC OO structure II




```
class(bc), pointer :: bc_ptr => null()
type(typename), pointer :: typename_ptr => null()
allocate(typename_ptr)
typename_ptr = typename(...)
bc_ptr => typename_ptr
```

New BC OO structure III



boundaries.nml

```
3
"riv , RIV , riv.nml , files_namelist_riv.dat , T , F"
"gib , SPO , gib.nml , files_namelist_gib.dat , T , T"
"dar , OPE , dar.nml , files_namelist_dar.dat , T , F"
```

OO features

- Inheritance;

OO features

- Inheritance;
- Resource Acquisition Is Initialization (*RAII*): all the resources that are needed by the object are bound to its lifetime;

OO features

- Inheritance;
- Resource Acquisition Is Initialization (*RAII*): all the resources that are needed by the object are bound to its lifetime;
- constructor overloading;

OO features

- Inheritance;
- Resource Acquisition Is Initialization (*RAII*): all the resources that are needed by the object are bound to its lifetime;
- constructor overloading;
- procedure overriding;

OO features

- Inheritance;
- Resource Acquisition Is Initialization (*RAII*): all the resources that are needed by the object are bound to its lifetime;
- constructor overloading;
- procedure overriding;
- Pointer to IMPLementation (*PIMPL*);

OO features

- Inheritance;
- Resource Acquisition Is Initialization (*RAII*): all the resources that are needed by the object are bound to its lifetime;
- constructor overloading;
- procedure overriding;
- Pointer to IMPLementation (*PIMPL*);
- objects polymorphism:

```
class(bc), pointer :: m_bc => null()
```

OO features

- Inheritance;
- Resource Acquisition Is Initialization (*RAII*): all the resources that are needed by the object are bound to its lifetime;
- constructor overloading;
- procedure overriding;
- Pointer to IMPLementation (*PIMPL*);
- objects polymorphism:

```
class(bc), pointer :: m_bc => null()
```

- decorator design pattern;

OO features

- Inheritance;
- Resource Acquisition Is Initialization (*RAII*): all the resources that are needed by the object are bound to its lifetime;
- constructor overloading;
- procedure overriding;
- Pointer to IMPLementation (*PIMPL*);
- objects polymorphism:

```
class(bc), pointer :: m_bc => null()
```

- decorator design pattern;
- factory design pattern.

Benchmarks

Recalling the time to solution:

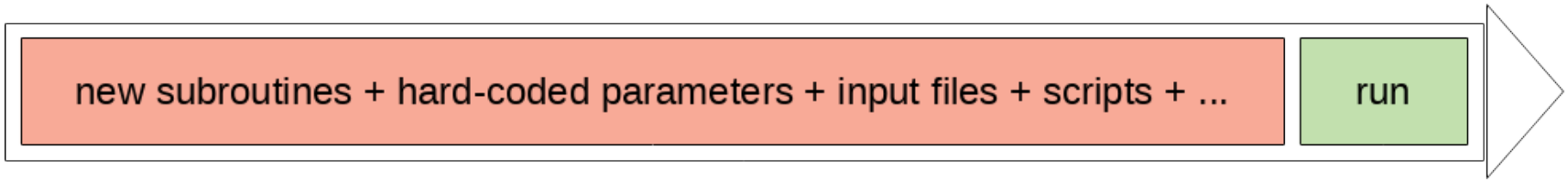


Table: Comparison of the effort required to modify the boundary conditions for two versions of the code: current one and new with additional namelist files and for loops over the boundaries.

	Current	**New ++**
Modify a BC	$\leq$ 30 lines	param. file
Add a new BC	$\simeq$ 300 to 400 lines	param. file
Add / remove *nudging*	not allowed	param. file

Profiling

Table: current version.

Subroutine	**Overall time (s)**	**Single call time (s)**
bcTIN	2.78 e-01	2.90 e-03
bcGIB	**2.39 e+00**	2.49 e-02
stp	3.14 e+02	3.27 e+00

Table: new version.

Subroutine	**Overall time (s)**	**Single call time (s)**
bcTIN	2.36 e-01	2.45 e-03
bcGIB	**3.61 e-01**	3.76 e-03
stp	3.08 e+02	3.21 e+00

Results

- Full re-engineering of the boundary condition modules;
- High Performance Computing: much smaller time to solution:
  - no more hard coding on the main code;
  - better maintenance and portability;
- no bottlenecks added; instead new update subroutine requires one order of magnitude less time;
- further developments will be much easier;
- more flexible management of data files;
- best practices (code documentation and versioning, unit tests with pFunit-3)

Thanks!

www.ogs.trieste.it

www.cineca.it

BC classification

- Rivers

$$\left.\frac{\partial \boldsymbol{C}}{\partial t}(\boldsymbol{x})\right|_{(x_r,y_r,0)} = \boldsymbol{\alpha};$$

- Closed

$$\left.\boldsymbol{C}(\boldsymbol{x})\right|_{D} = \boldsymbol{C}_0;$$

- Sponge

$$\left.\boldsymbol{C}(\boldsymbol{x})\right|_{D} = \boldsymbol{C}_0; \quad \left.\boldsymbol{v}(\boldsymbol{x})\right|_{\partial D} = 0;$$

- Open

$$\left.\boldsymbol{C}(\boldsymbol{x})\right|_{\partial D} = \boldsymbol{C}_0;$$