WG6 : Bringing back POMPA developments to the COSMO code

Oliver Fuhrer and Xavier Lapillonne

Overview

- ICON-COSMO Physics
- New permanent arrays
- Serialization (C++ validation)

ICON-COSMO Physics

- ICON and COSMO have different grids/data structures
- In order to shared the physics package new interfaces need to be introduce : data field will be passed by arguments
- ICON developers proposed to have only one horizontal index inside the physics packages (block data)

- What are the implications for COSMO ?

Block structure in the physics

- Data fields are copied from the format f(ie,je,ke) to the block structure f (nproma,ke), with nproma = (ie-2 nbound) x (je-2 nbound) / nblock.
- physics parametrization could be computed while data remains in the cache (on CPU by selecting an appropiate value for nproma)
- organize_physics could be structured as follows:

```
call prepare_radiation
call prepare_turbulence
…
 do ib=1,nblock
     call copy_to block
     call organize_gscp
     call organize_radiation
     call organize_turbulence
     call copy_back
 end do
```

Operations requiring neighbouring information : ex averaging

where data inside organise_scheme is in block form t_b(nproma,ke)

Routines below organize_scheme will be shared with ICON. Fields are passed via argument list:

```
call fesft(t_b(:,:), …
```

- Note : an omp parallelization could be easily introduced around the block loop
- This is the current implementation in the OPCODE branch

COSMO-ICON Physics : 3 possible approaches

1. Keep i,j indices

- ICON would run with j loop from 1 to 1

Implications:

- Only need to adapt shared physics (only interface)
- Keep original organize_physics
- Lower performance on GPUs
- May decrease ICON's performance

2. Full block physics (single horiz. index)

Implications:

- Need to adapt (or remove) all COSMO schemes (index + interface).
- Unique computation domain for all physics (istarpar:iendpar ?)
- Some options need to be adapted: (ex nradcoarse)
- Straightforward for OpenMP
- Good for GPU
- All physics have the same interface
- Need to deal with the copy to block

POMPA RECOMANDATION

3. Mixed block/nonblock (single horiz. index)

- Keep two versions of organize_physics

Implications:

- One block version with only physics which are shared with ICON
- One "original" where the block physics are called inside a j loop.
- Increase complexity
- Less work

COSMO-ICON Physics

- Whatever decision is made for the physics we need to have some time schedule concerning its implementation in the official COSMO

- This is a critical aspect for the reintegration of the POMPA work

Other

- Move microphysics at the beginning of the physics

- Could we remove some of the NEC optimization ?

New permanent arrays

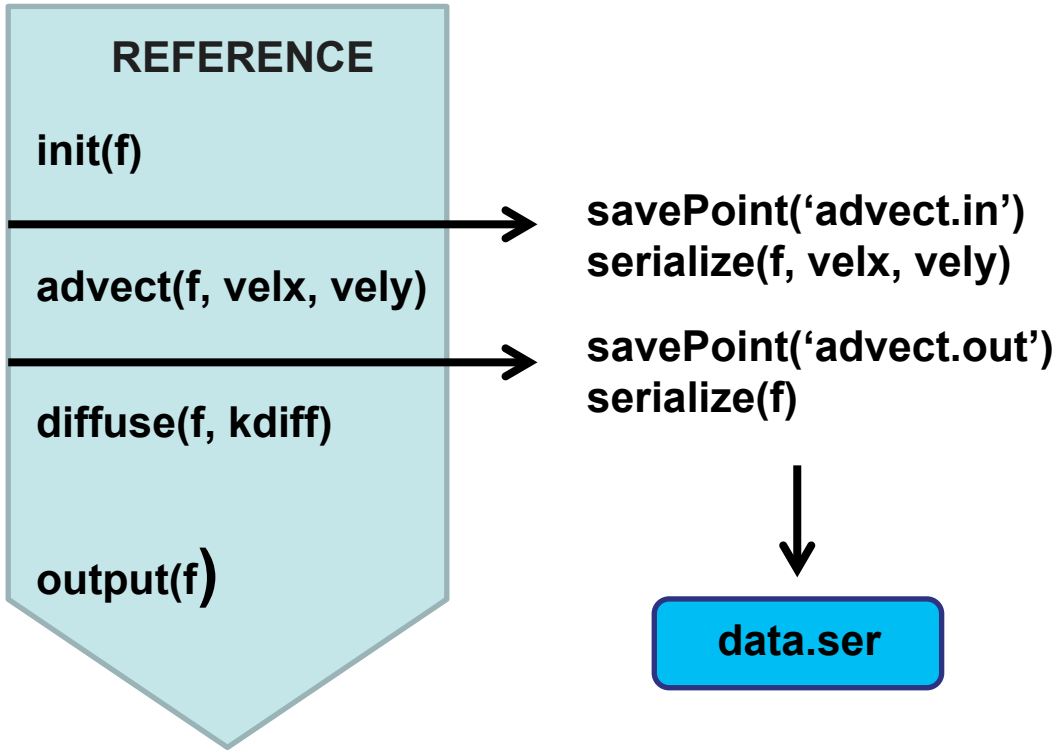
- Problem : memory allocation is very costly on GPU
- We've replaced all automatic arrays in parts of the code which runs on GPUs with allocable arrays. The arrays are allocated for the full model run
- Implication : significant increase of memory usage
- COSMO has however a low memory footprint considering nowadays hardware:

  E.g. COSMO-2 Opr uses only 13 GB when run on a single node

- Practical implementation:
- Added specific modules for each parts (e.g. each physics), containing the local arrays together with an allocate and deallocate routines.

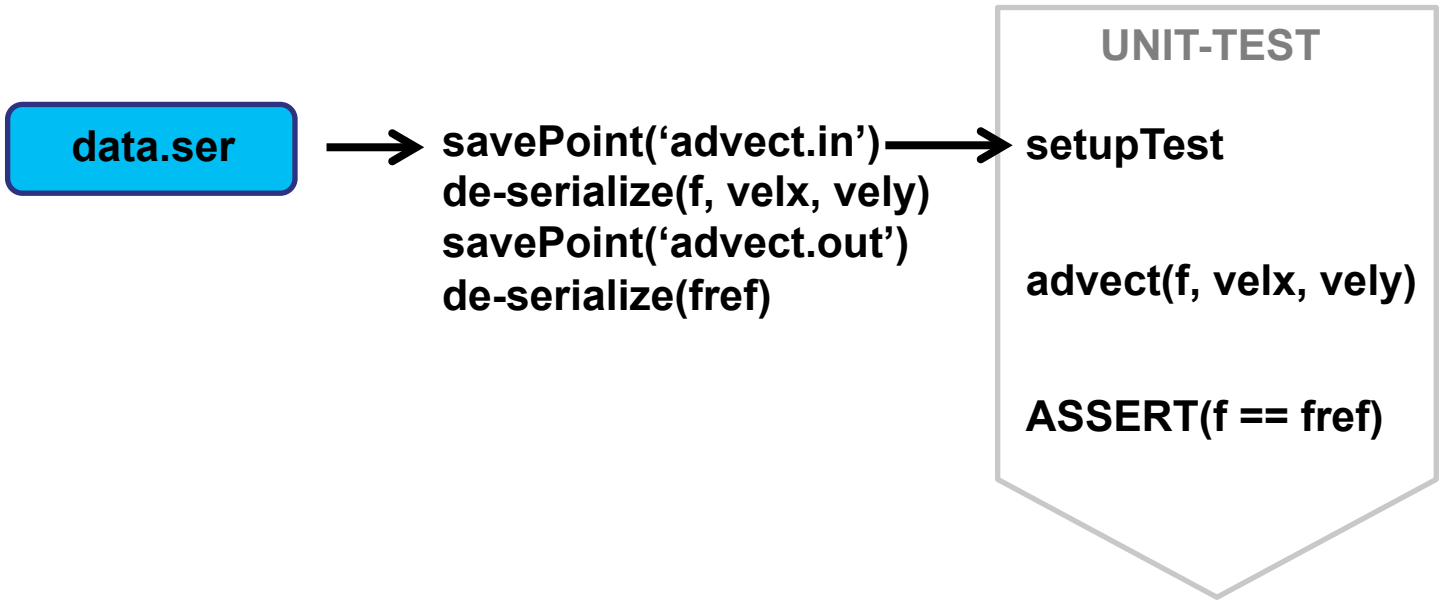
Serialization Framework

- Used to validate the C++ dycore
- Serialization means converting data structures or objects into a format that can be stored (in memory or on a **file**)
- We use serialization for unit-testing stencils against their **reference version** (e.g. in Fortran)



Unit-Testing

- The de-serialized fields can be read and used to check a new implementation
- This can be done for a single stencil (unit-testing)




```
[----------] 3 tests from AdvectionUnittest
[ RUN      ] AdvectionUnittest.Do
initializing data field demanded_f_in
initializing data field demanded_vel_in
Initializing data_field demanded_f_out
[       OK ] AdvectionUnittest.Do (1734 ms)
```

Serialization Framework

- Would it be possible to include the serialization calls in the trunk ?
- Calls would be embedded in a new module

Other

- Invert Relaxation and Assimilation