RegCM
Climate Model Refactoring for HPC

Graziano Giuliani
ggiulian@ictp.it

International Centre for Theorethical Physics
Earth System Physics Section

Introductory School on Parallel Programming and Parallel Architecture for High-Performance Computing
Monday 10 October 2016

RegCM

- Regional Climate Model
- Physical downscaling of Global Climate Models
  - Regional climate change impacts
  - Impact studies for delicate ecosystems
  - Seasonal to climate prediction
- Developed by the ICTP ESP section
- 1000+ users around the world
- FORTRAN language

Paralell coding from scratch - the stylist

- Examine problem
- Choose programming language
- Chose optimized parallel libraries
- Design parallel data structures
- Implementation part
  - implement communication part
  - implement computing part
- Performance evaluation and optimization
- New users jump in when all is ready

Code refactoring for HPC - the mender

- Esisting bounds
  - fixed programming language
  - fixed data structures
  - existing user base
  - solution optimized for old platforms
- Patchwork frankenstein codes
- Users are a conservative lot

RegCM in 2010

- Atmosphere is fast part = fluid dynamic
- advection code = stencil code
- a lot of physics inside
- code from the '80 with some mpi 1d added
- pre-processing based
- multiple developers with no common style
- flat matrix memory layout

New development phase - 2010 $\rightarrow$ 2014

- Code rescue
- Code shuffle
- Code upgrade

Code rescue

- Code repository
- Shake the rust
    - text refactory to have uniform coding
    - improve code readability
    - remove obsolescent language features
        - plusFORT + sed, awk, perl + manual edit
    - insert new language features
        - dynamic allocation
        - I/O libraries

Code shuffle

- FORTRAN modules
- Identify common subsystems
  - reduce code duplications
  - centralize parameters and control vars
  - reduce complexity
- New code extensions inserted
- Improve User experience

Code upgrade

- Memory management centralization
- Extensive indexing change
- 2D parallelization
  - Identify exchange sections
  - Ad-hoc patch-based library

Strategy

- Ghost (halo) points allocation
- Cartesian topology
- Collectives
- Exchanges
- Communication patterns
- FORTRAN interfaces

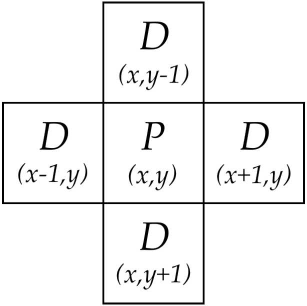
Stencil code

Atmospheric models in their dynamical core are stencil codes:

- A class of iterative kernels which update array elements according to some fixed pattern, called stencil.
- Stencil codes perform a sequence of sweeps (called timesteps) through the simulation space, a 2- or 3-dimensional regular grid whose elements are referred to as nodes, cells or elements.
- In each timestep, the stencil code updates all array elements.
- Using neighboring array elements in a fixed pattern (called the stencil), each cell's new value is computed.
- Boundary values need to be adjusted during the course of the computation as well.
- Since the stencil is the same for each element, the pattern of data accesses is repeated.

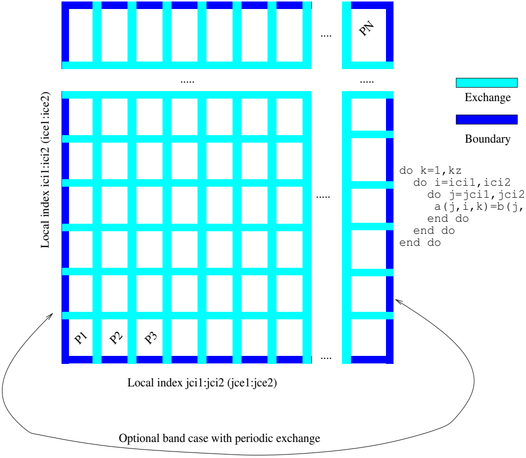
Ghost Points

The algorithm work on a patch library, which handles the synchronization of the ghost zone or halo and the boundaries.



The code loops over big arrays, and for this reason usually cannot perform efficient cache blocking or wrapping of the code for accelerators.

RegCM4 2D scheme




```
do k=1,kz
  do i=ici1,ici2
    do j=jci1,jci2
     a(j,i,k)=b(j,i,k)
    end do
  end do
end do
```

Cartesian Grid

RegCM4 works on a Cartesian 2D Grid

- Each processor has a 2D patch of the model domain
- Each processor may have exchange of ghost points
- Each processor may have boundary value area points

Cartesian Grid comunication library interface

- Different data types up to 4 dimensional
- One to all (mpi_bcast, mpi_send, mpi_recv)

```
call bcast(a)
call grid_distribute(a_glob,a_loc,j1)
call subgrid_distribute(a_glob,a_loc)
```

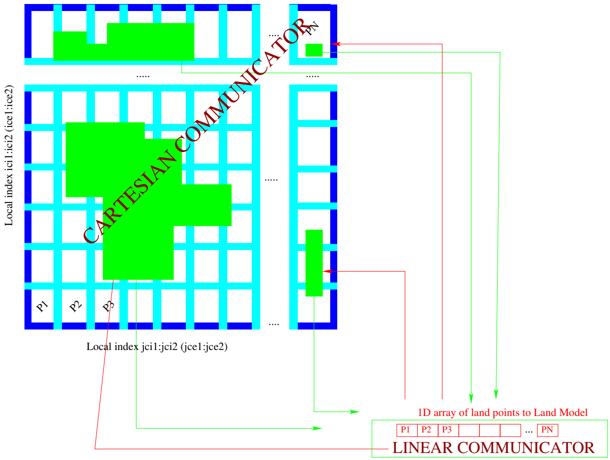
- All to one (mpi_send, mpi_recv)

```
call grid_collect(a_loc,a_glob)
call subgrid_collect(a_loc,a_glob)
```

- Ghost point exchange (mpi_irecv, mpi_send, mpi_wait, mpi_sendrecv)

```
call exchange(a)
```

RegCM4 MPI 1D surface masked scheme



Cartesian To Linear comunication interface

- Different data types up to 4 dimensional
- type(masked_comm), uses mpi_scatterv, mpi_gatherv
- Each processors gives its internal 2D grid, gets 1D land point vector. We have also the option to order up things on the global grid first.

```
call c2l_ss(masked_comm,local_matrix,vector)
call c2l_gs(masked_comm,local_matrix,vector)
call glb_c2l_ss(masked_comm,global_matrix,vector)
call glb_c2l_gs(masked_comm,global_matrix,vector)
```

- Each processor gives 1D land point vector, gets its internal 2D grid

```
call l2c_ss(masked_comm,vector,local_matrix)
call glb_l2c_ss(masked_comm,vector,global_matrix)
```

Parallel I/O

Implemented using netCDF (on top of HDF5, on top of MPI-IO)

- Charming but not scalable
- Library based : use HDF5 + netCDF
- Does not work without parallel FS

Nevertheless the netCDF library is GOOD for a number of other different motivations.

Trust based

- Massive code changes
  - Confidence needed
  - DO NOT PANIC
    - you have introduced bugs
    - you have not full picture
    - you got things wrong
    - you got the blame for all
  - learn from errors
  - be bold and sincere
- Trust the others
  - old guys are not just ancient
    - wisdom comes from errors
    - learn from someone else errors

Caveats

- Must have stomach
  - poor coding technique
  - ugly coding standards
  - multiple voices
  - past limitations
  - you hate the language
  - you know that it can be done better
- Boredom
  - menial tasks repeated
  - use tools
  - cannot use tools
  - testing to the power of testing
- Overconfidence
  - i will fix it just here...
  - nothing works anymore - bisection time again

Different approaches ex-post

- Do not re-invent the wheel
  - increase code dependencies
  - use external libraries is not bad
- Addiction to the codebase

What next - gotta move on

- Stop having nightmare about RegCM
- Try the from-scratch approach
  - Cell-based library on the market
  - template meta-programming

Thanks for your attention! Any questions?

Ciao!