Parallel implementation

Hendrik Tolman & Mickael Accensi

The friends of WAVEWATCH III Team
Marine Modeling and Analysis Branch
NOAA / NWS / NCEP / EMC

Mickael.Accensi@ifremer.fr
NCEP.list.waves@NOAA.gov

Outline

Covered in this lecture:

- Compiling the code.
- Running the code.
- Optimizing parallel model implementations.
  - Parallel implementation of individual grids (*ww3_shel*).
  - Additional options in *ww3_multi*.
    - Hybrid parallelization.
    - Profiling.
    - Memory use (+IO).
  - Considerations and pitfalls.
  - Future ….

Parallel implementation of WAVEWATCH III

- Using MPI, but code allow for other type of parallel architecture.
  - IntelMPI (*mpiifort*) was the standard before, now is even more so.
  - Exists also MPT (*ifort –lmpi*) and OpenMP (*ifort –openmp*)
  - Future use of MPI-OpenMP hybrid ?
    - Using MPT (*ifort –openmp –lmpi*)
    - Using IntelMPI (*mpicc –openmp*)

- Not all codes use / can benefit from parallel implementation:
  - Actual wave model codes *ww3_shel*, *ww3_multi*, *ww3_prnc* will run much more efficient.
  - Thinking about parallelize *ww3_ounp* and *ww3_ounf*

Compile in several steps:

- Set switches for serial code (SHRD switch)
- Compile serial codes from scratch:
  - Call *w3_new* to force complete compile of all routines (not strictly necessary).
  - Call *w3_make* without program names to get base set of serial codes.
- Set switches for parallel code (DIST and MPI switch).
- Compile parallel codes:
  - Compile selected codes only
    - *w3_make ww3_shel ww3_multi* ww3_prnc
  - This will automatically recompile all used subroutines.

All this is done automatically in *make_MPI*.

Running parallel

Running code in parallel depends largely on hardware and software on your computer.

- Generally there is a parallel operations environment:
  - "poe" on IBM systems
  - "mpirun" on Linux systems.
  - Sometimes, environment needs to be started separately.
    - *mpdboot* combined to *mpiexec*.

- There are some examples in the test scripts, particularly the *ww3_multi* and real-world test cases *mww3_test_NN* and *mww3_case_NN*.

- Pitfall: many duplicate output lines: you are running a serial code in a parallel environment …..

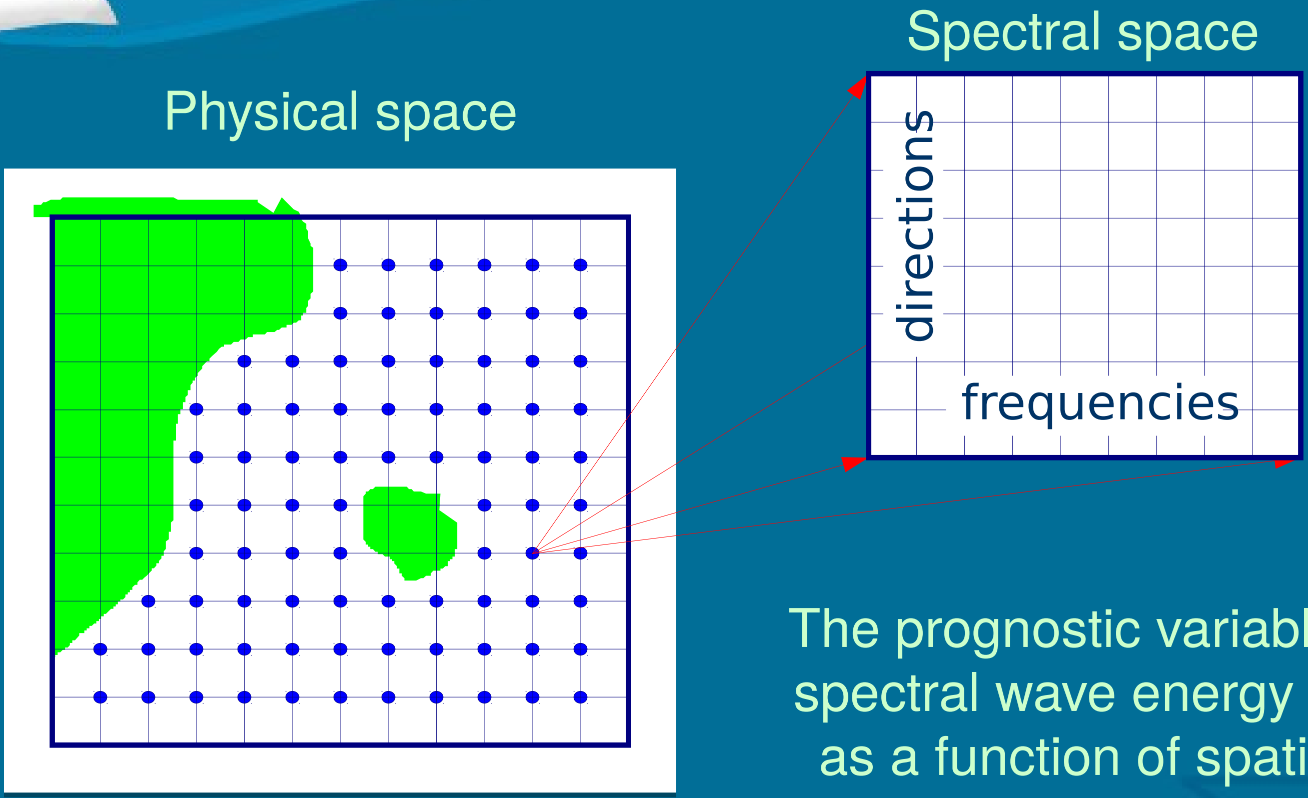
Optimization

Three things to consider while optimizing the implementation.

- General code optimization (no further discussion):
  - Compiler options.
  - Switches (2nd order versus 3rd order propagation, etc.)
  - Spectral resolution.
  - Time stepping.
- MPI optimization (no further discussion).
  - Often overlooked, but can be very important on Linux systems.
- Application optimization (see below):
  - Cannot do too much with *ww3_shel*, but will show techniques used here.
  - Many additional options in *ww3_multi*.

Tolman, H. L., 2002: *Parallel Computing,* **28**, 35-52.
Tolman, H. L., 2003: MMAB Tech. Note **228**, 27 pp.

ww3_shel



The prognostic variable is the spectral wave energy density as a function of spatial and spectral coordinates and of time.

Propagation :

- By definition linear, nonlinear corrections possible.
- Covers all dimensions.

Physics :

- Wave growth and decay due to external factors :
  - wind-wave interactions,
  - wave-wave interactions,
  - dissipation.
- Local in physical space.

Time splitting / Fractional steps. Separate treatment of :

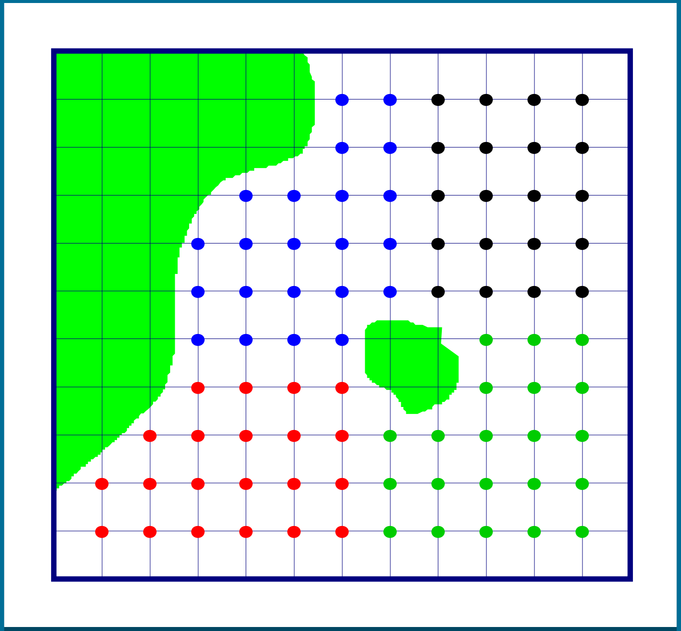
- physics (local),
- local propagation effects (change of direction or frequency),
- spatial propagation.

Each step consecutively operates on small subsets of data.

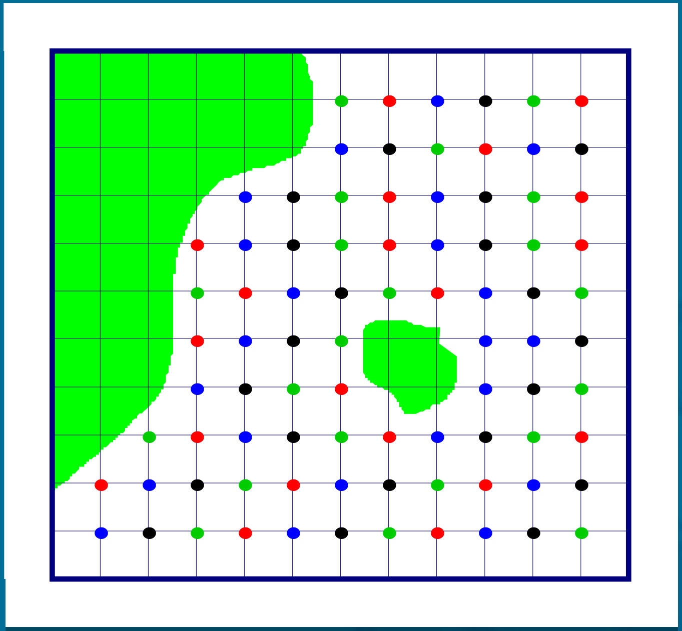
Entire model in core, memory requirements less than twice that of storing single state.

Physics involved suggest that grid points are distributed over processors rather than spectral components, particularly for the time splitting and source term integration techniques used in WAVEWATCH III.

"blocking"



"Scattering"



ww3_shel

Blocking :

- Only data at block bound. needed.
- Total amount of data comm. is a function of # of processes.
- Algorithm depends on actual prop. scheme.

Scattering :

- Full data transpose needed.
- Total amount of data comm. nearly constant.
- Algorithm independent of prop. scheme.
- Load balancing easier.

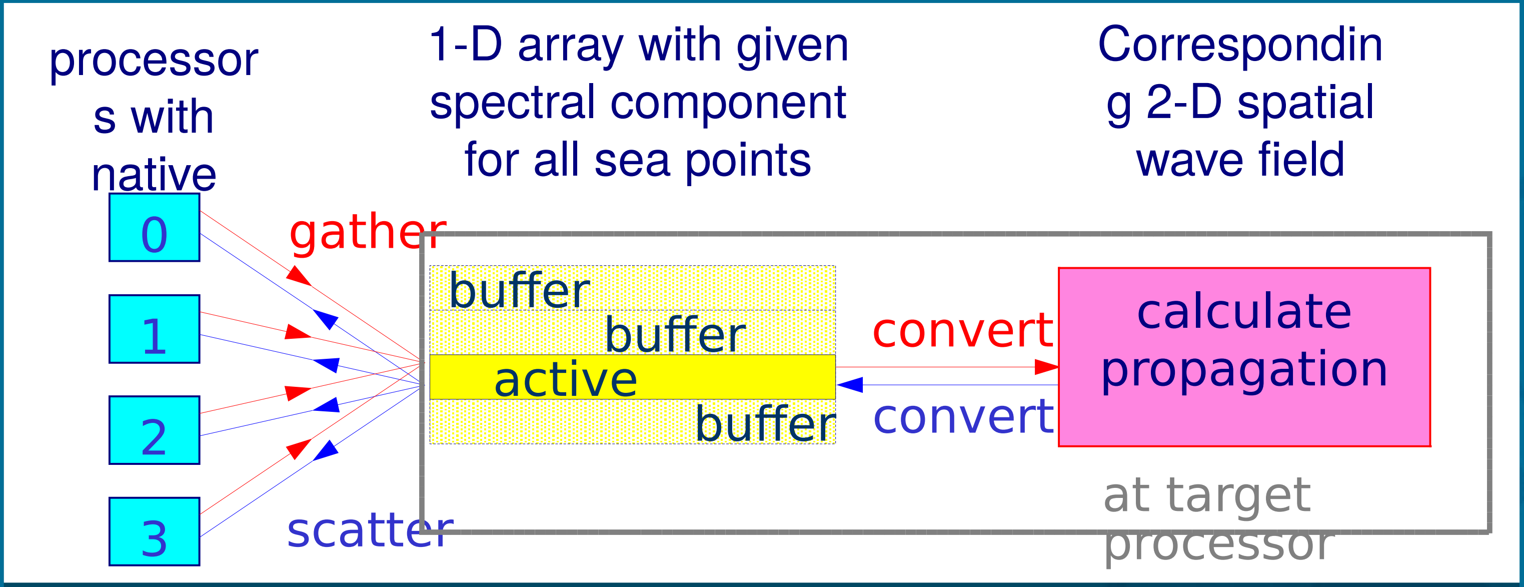
WAVEWATCH

For WAVEWATCH III (ww3_shel) the scattering method is used because :

- Compatibility with previous versions.
- Maximum flexibility and transparency of code (future physics and numerics developments).
- Feasibility based on estimates of amount of communication needed.
- MPI used for portability.

Standard optimization techniques :

- Non-blocking communication
  - overlaps communication and computation operations
  - Use of a buffer and “wait” routine
- Persistent communication
  - transparency of code
  - Security of object manipulations (communicator, group)



For mosaic approach, there are other optimization options:

- Splitting grid in overlapping domains:
  - Better local CFL time steps.
  - Hybrid domain decomposition.

- Running grids with same rank side-by-side on parts of communicator:
  - Localizing communications.
  - Amdahl's law generally favors running grids side-by-side on smaller number of processors over running in sequence over larger number of processors.

Splitting the communicator:

- Example from mww3_test_03, running three overlapping low resolution grids, with overlay of three overlapping high resolution grids.
  - Example runs low1-3 serially on entire communicator.
  - Example runs hgh1-3 side-by-side on fractions of communicator.
  - Output can also go to dedicated processors.

```
from mww3_test_03 ww3_multi input file .....
$
   'low1'   'no' 'no' 'no' 'no' 'no' 'no' 'no'    1  1  0.00 1.00  F
   'low2'   'no' 'no' 'no' 'no' 'no' 'no' 'no'    1  1  0.00 1.00  F
   'low3'   'no' 'no' 'no' 'no' 'no' 'no' 'no'    1  1  0.00 1.00  F
$
   'hgh1'   'no' 'no' 'no' 'no' 'no' 'no' 'no'    2  1  0.00 0.33  F
   'hgh2'   'no' 'no' 'no' 'no' 'no' 'no' 'no'    2  1  0.33 0.67  F
   'hgh3'   'no' 'no' 'no' 'no' 'no' 'no' 'no'    2  1  0.67 1.00  F
$
```

Note: identical fractions = non-overlapping communicators

Running with or without output processor

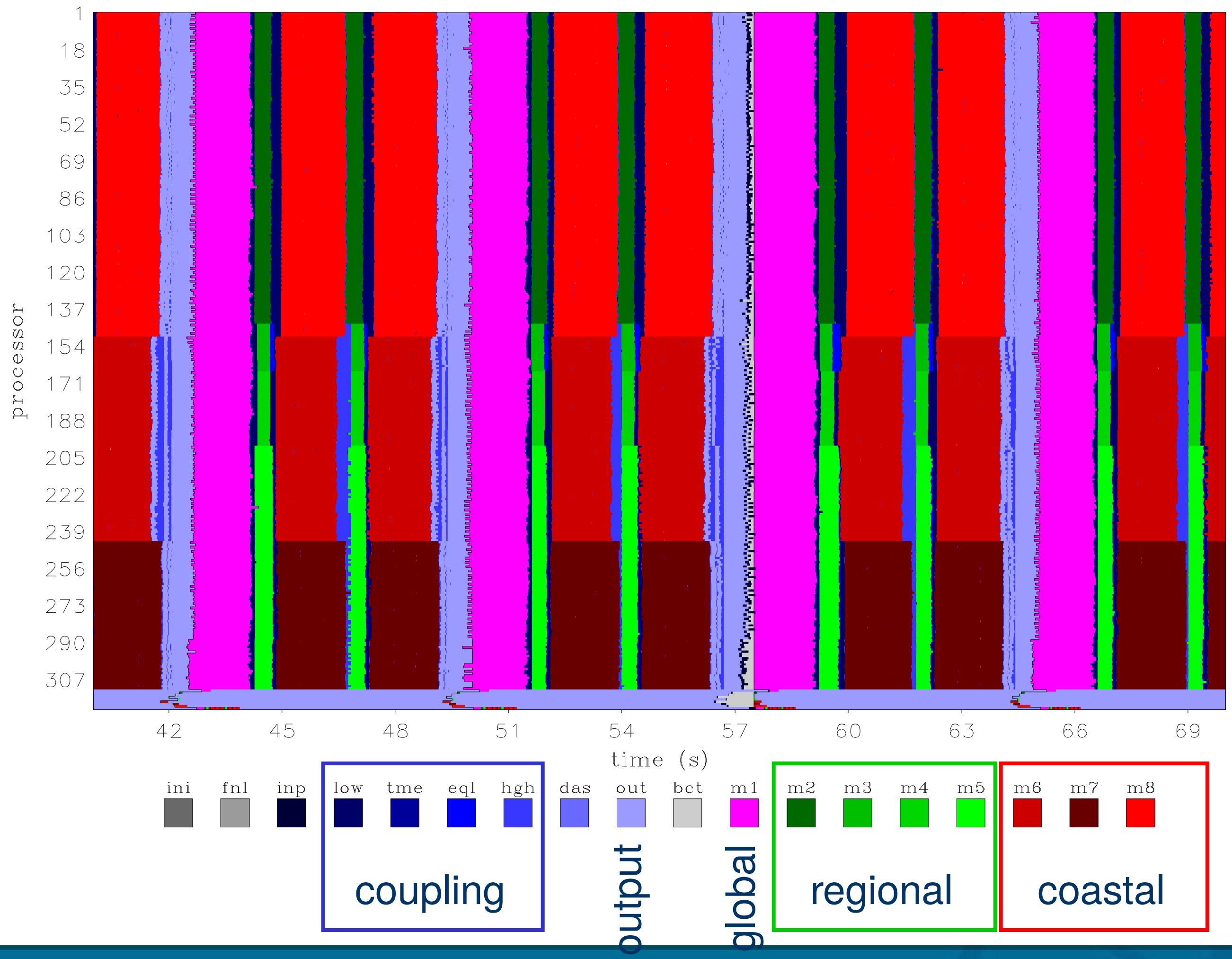
```
 WAVEWATCH III log file                                 version 4.08
 =====================================================================
 multi-grid model driver                           date : 2013/01/04
                                                   time :  11:05:27
. . . . .

  Group information :
  nr   grids (part of comm.)
 -------------------------------------------------------------------
   1     1 (0.00-1.00)
   2     2 (0.00-0.33)  3 (0.33-0.66)  4 (0.66-1.00)
 -------------------------------------------------------------------

  Resource assignment (processes) :
  grid          comp.    grd  pnt  trk  rst  bpt  prt
 ---------------------------------------------------
  low0          001-011  012  ---  ---  ---  ---  ---
  hgh1          001-004  012  ---  ---  ---  ---  ---
  hgh2          005-007  012  ---  ---  ---  ---  ---
  hgh3          008-011  012  ---  ---  ---  ---  ---
 ---------------------------------------------------
```

Grid-level profiling:

- Compile under MPI with MPRF switch on.
- Run short piece of model, generating profiling data sets.
- Run GrADS script *profile.gs* to visualize:

- Example of NCEP’s original multi-grid wave model on next slide.
  - 8 grids.
  - 360 processors.
  - Dedicated I/O processors.

NCEP “multi_1” global model on IBM ca. 2008

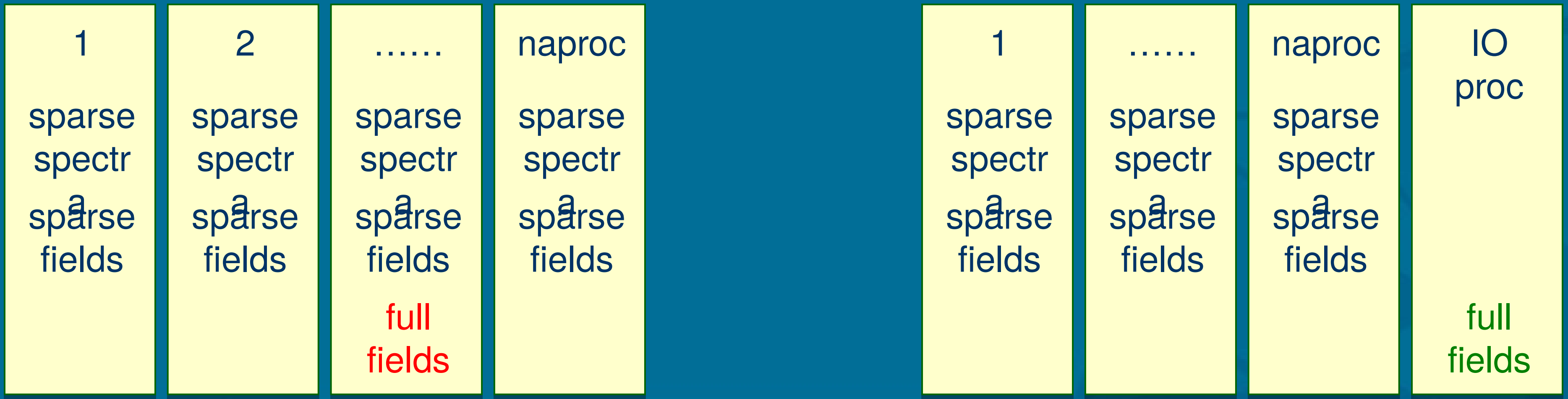


When running a grid on NAPROC processors, each processor stores:

- NSEA/NAPROC spectra. (scaling)
- Output fields:
  - Sparse output fields (NSEA/NAPROC). (scaling)
  - Full output fields (NSEA) in 1 processor only, only fields for selected output. (not sc.)
- Other outputs:
  - Gathered in one processor, with some buffering to limit local memory use. (“scaling”)
- Work arrays, interpolation tables, … (not sc.)

IO considerations: IO server type IOSTYP in *ww3_shel* and *ww3_multi*.

- In ww3_multi, all point output can go to dedicated processor
- In ww3_shel,
  - 0: No IO server process, parallel direct write to output files
  - 1: No IO server process, assigned process for each output file.
  - 2: Single dedicated IO process for all output files.
  - 3: Multiple dedicated IO process for each output file.



1 sparse spectra sparse fields
2 sparse spectra sparse fields
…… sparse spectra sparse fields full fields
naproc sparse spectra sparse fields
1 sparse spectra sparse fields
…… sparse spectra sparse fields
naproc sparse spectra sparse fields
IO proc full fields

Memory use + IO

IO considerations:

- Use IO server to manage memory use as well as faster IO.
  - Combine with smart placement on nodes (e.g., less processes on node that does IO) leaves much flexibility for efficient loading of large grids.
- Use overlapping grids:
  - Each grid has much smaller full field arrays.
  - Stitch together later with *ww3_gint*.

Considerations and pitfalls:

- Intra-node and across-node communications are very different.
  - Keeping grid on node may be important.
- Scaling on different systems is very different:
  - IBM-SP versus Linux.
  - Impact of file system.
  - Optimization of MPI.
    - Data transpose is many small messages, MPI needs to be tuned for this …
- For operational models, dimension for worst case:
  - Profile without ice.
  - Consider smallest grids with largest storms in consideration of load balancing.

Debugging and optimization tools:

- Debugging
  - *valgrind* : to find memory leak
  - *ddt* : to run the compiled code step by step
- Optimization
  - *time* : to check that system time is small
  - *strace -c :* to show the time spent by function
  - *gprof* : combined to *gprof2dot.py* to have a nice visualization
  - *MPIinside (mpi)*, *Vtune (mpt)* or *bandela* : for process performance distribution

Optimization future

Working on the following:

- Hybrid domain decomposition (under development).
- Provide some assessment of optimization for both climate (low-res, high-speed), and deterministic (high-res, high-speed) implementations.

Stay tuned !

The end

End of lecture