TM6: coupling to EC-Earth and parallel I/O for meteo

Philippe Le Sager, KNMI

2013-11-13

Outline

What's TM6

EC-Earth

Parallel I/O

Extra

Outline

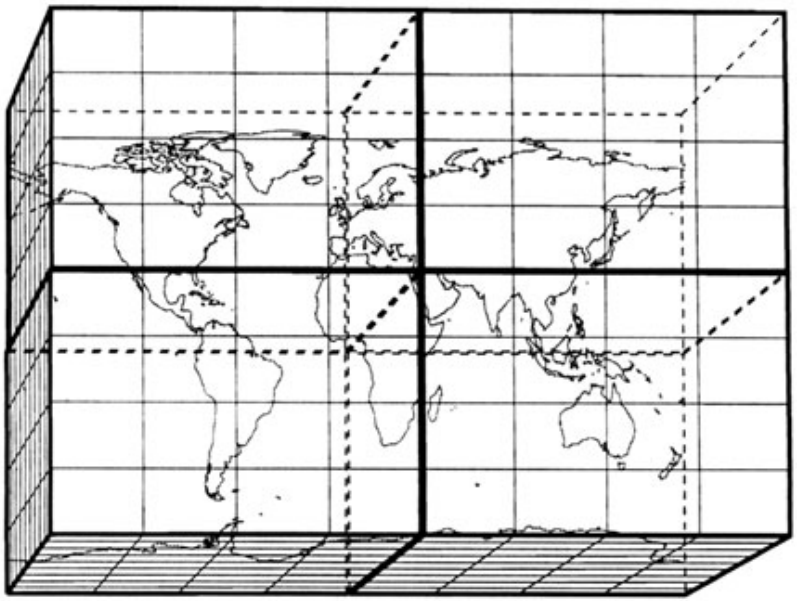
What's TM6

EC-Earth

Parallel I/O

Extra

TM6 = a new MPI implementation in TM5



Arrays are split across processors

- TM6 splits tracers & meteo
- TM5 splits tracers
  - but copies entire meteo fields:
    - **huge** memory
    - **heavy** communication

Theoretical limits - Max number of processes

TM5	TM6	TM6 w/ reduced grid
5 (base)	2700 (@3x2)	45 (@3x2)
27 (chem)	16200 (@1x1)	90 (@1x1)
52 (chem+M7)		

Theoretical limits - Max number of processes

TM5	TM6	TM6 w/ reduced grid
5 (base)	2700 (@3x2)	45 (@3x2)
27 (chem)	16200 (@1x1)	90 (@1x1)
52 (chem+M7)		

but memory at 1x1 for chemistry

- humongous in TM5 => only **3-5 processes**
- not an issue in TM6

Since the last meeting...

debug

- ~~fix buggy M7~~ —> TM6 is 'production ready' for chemistry

feature

- ~~couple to EC-Earth~~

optimization

- optimize reduced grid
- optimize time-series output in chemistry
- ~~read netCFD meteo in parallel~~

Outline

What's TM6

EC-Earth

Parallel I/O

Extra

Implementation in EC-Earth

Two steps in EC-Earth 2.4

- replace TM5_v3 with TM5_v4
  - switch to (and update) pycasso
  - update source code
- replace TM5_v4 with its TM6 branch

Implementation in EC-Earth

Two steps in EC-Earth 2.4

- replace TM5_v3 with TM5_v4
  - switch to (and update) pycasso
  - update source code
- replace TM5_v4 with its TM6 branch

Next : EC-Earth 3.0.1

- newer IFS (atmo) and NEMO (ocean) versions
- coupler OASIS is in charge instead of IFS

Outline

What's TM6

EC-Earth

Parallel I/O

Extra

On coding the parallel reading of meteo

- new SETUP_xyz routines in meteo.F90
- parallel reading (define the tile to read)
- grid_type_ll.F90
  - added '==' operator to compare grids
  - added '=' assignment to copy data instead of pointers
  - initialization of TllGridInfo pointers

On coding the parallel reading of meteo

- new SETUP_xyz routines in meteo.F90
- parallel reading (define the tile to read)
- grid_type_ll.F90
  - added '==' operator to compare grids
  - added '=' assignment to copy data instead of pointers
  - initialization of TllGridInfo pointers

Limitations

- netCDF4 meteo
- strict F95
- evenly decomposed grid: Nb latitude bands / Nb procs
- meteo grid is one of the model grid

Tests

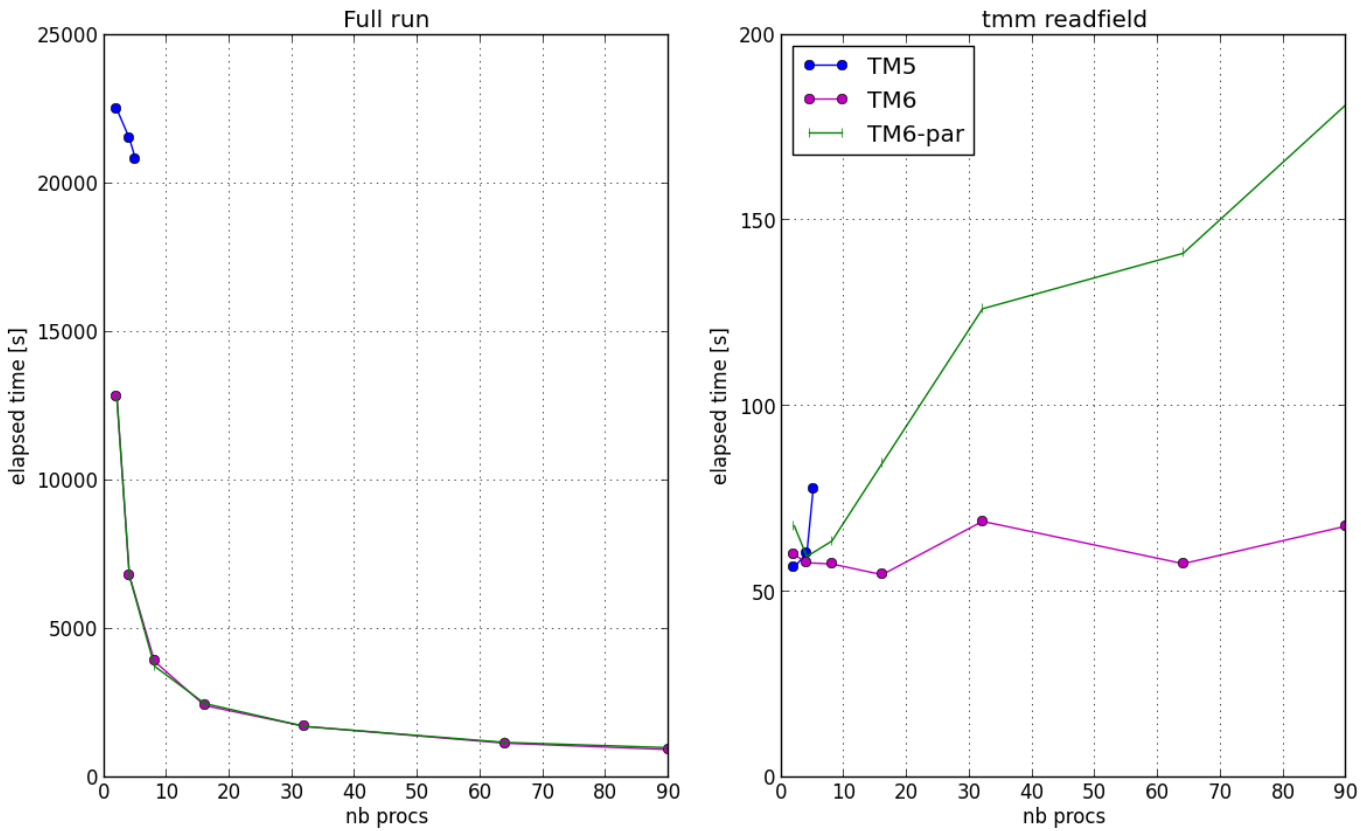
Short runs with Base

- ☒ serial reading is not broken
- serial and parallel i/o give same result, when:
  - ☒ no transform
  - ☒ vertical transform (60 to 34 levels)
  - ☐ horizontal transform

To keep in mind

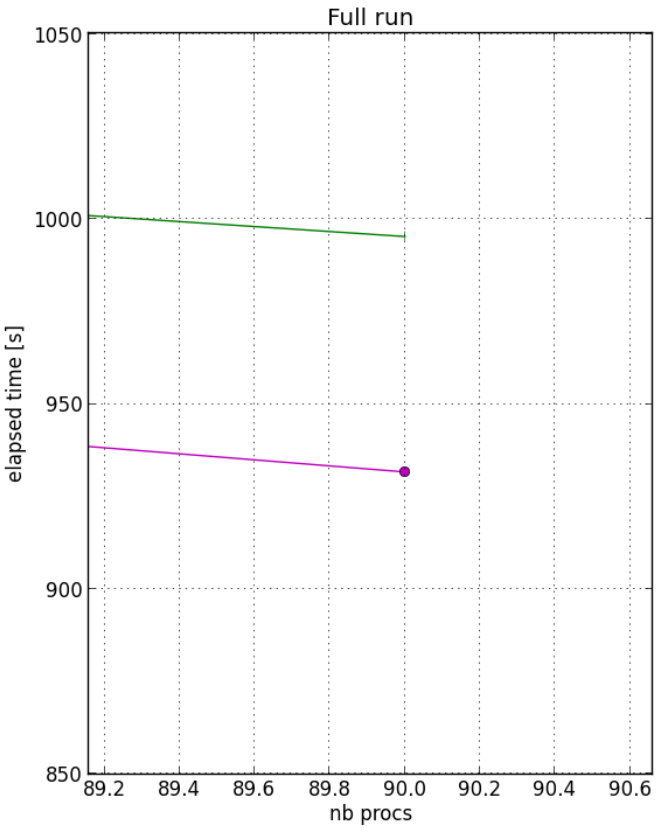
- reading meteo = read file + remap
- but should add scatter (serial I/O only)

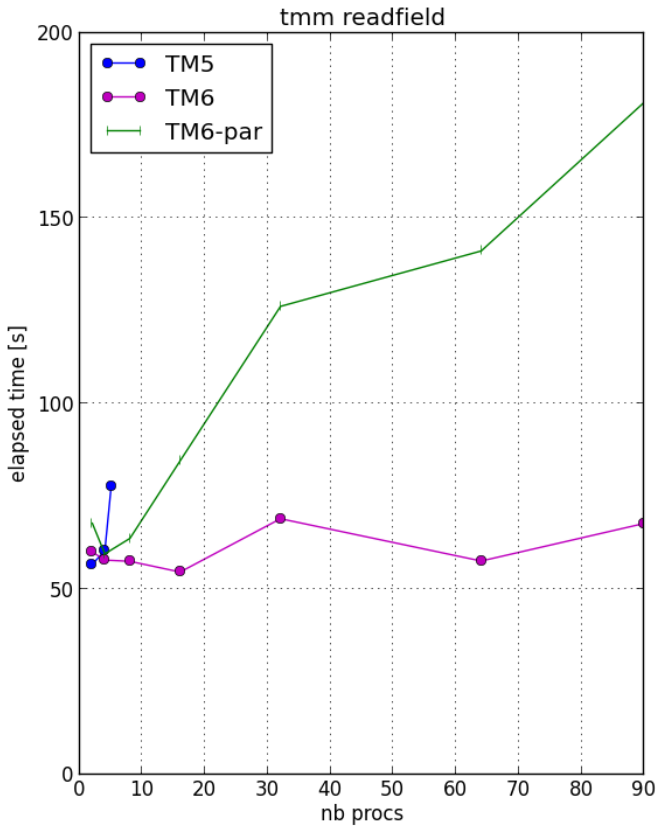
Base @1x1 - one week runs, EI-60 levels, reduced grid



20x faster, but is parallel i/o **broken**?

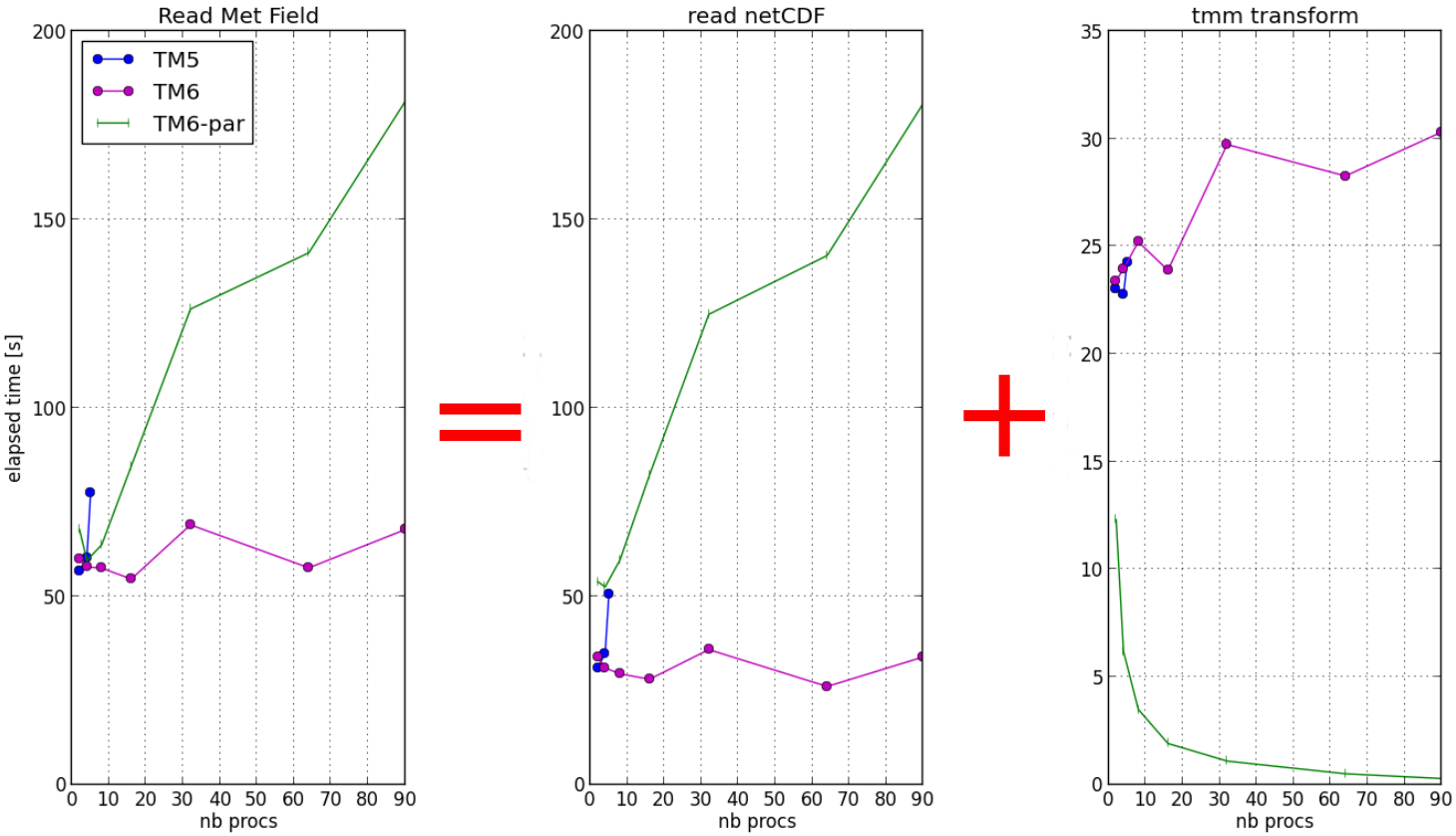
Base @1x1 (2) - scattering & reading costs





- scattering = ˜45s
- reading w/ 90 procs : 12% (serial)

Base @1x1 (3) - reading details



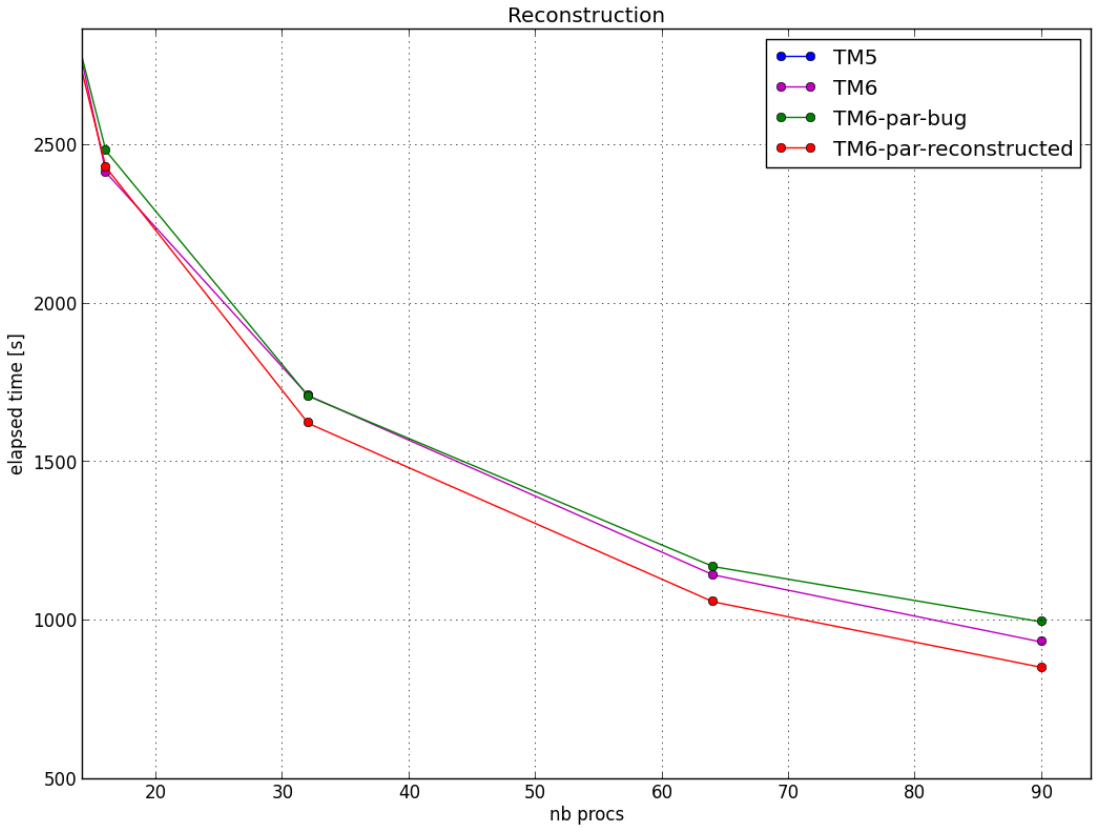
- gain (!!) in transform = ˜ 30s
- bug in netcdf implementation

On parallel I/O with netcdf4

from netcdfgroup:

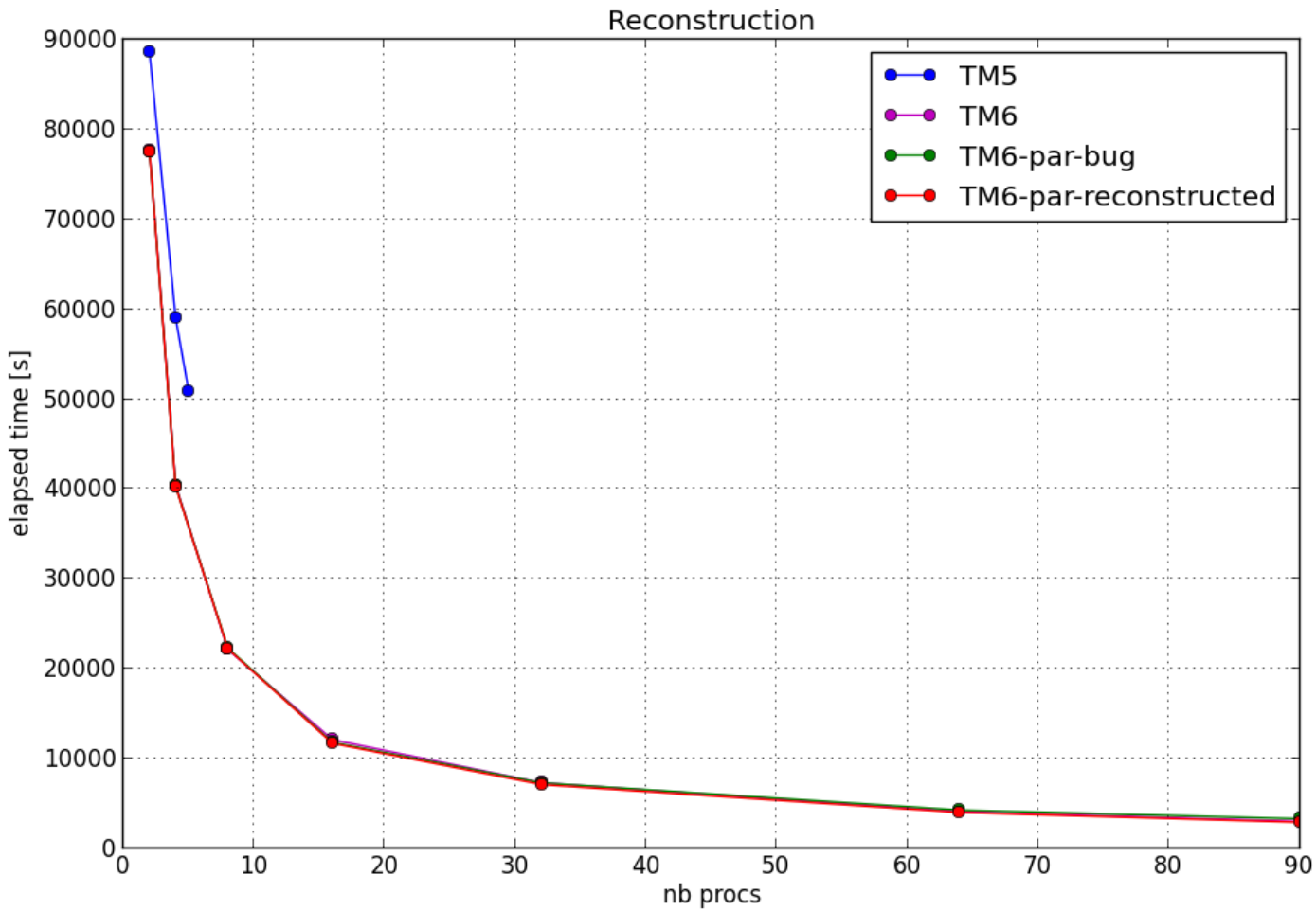
- *Parallel I/O is a very complex topic*
- *I/O scales reasonably linearly for less than about 8 processors*,
- *after your parallel application is saturating your I/O subsystem, and further I/O performance is marginal.*

Base @1x1 (4) - Reconstruction

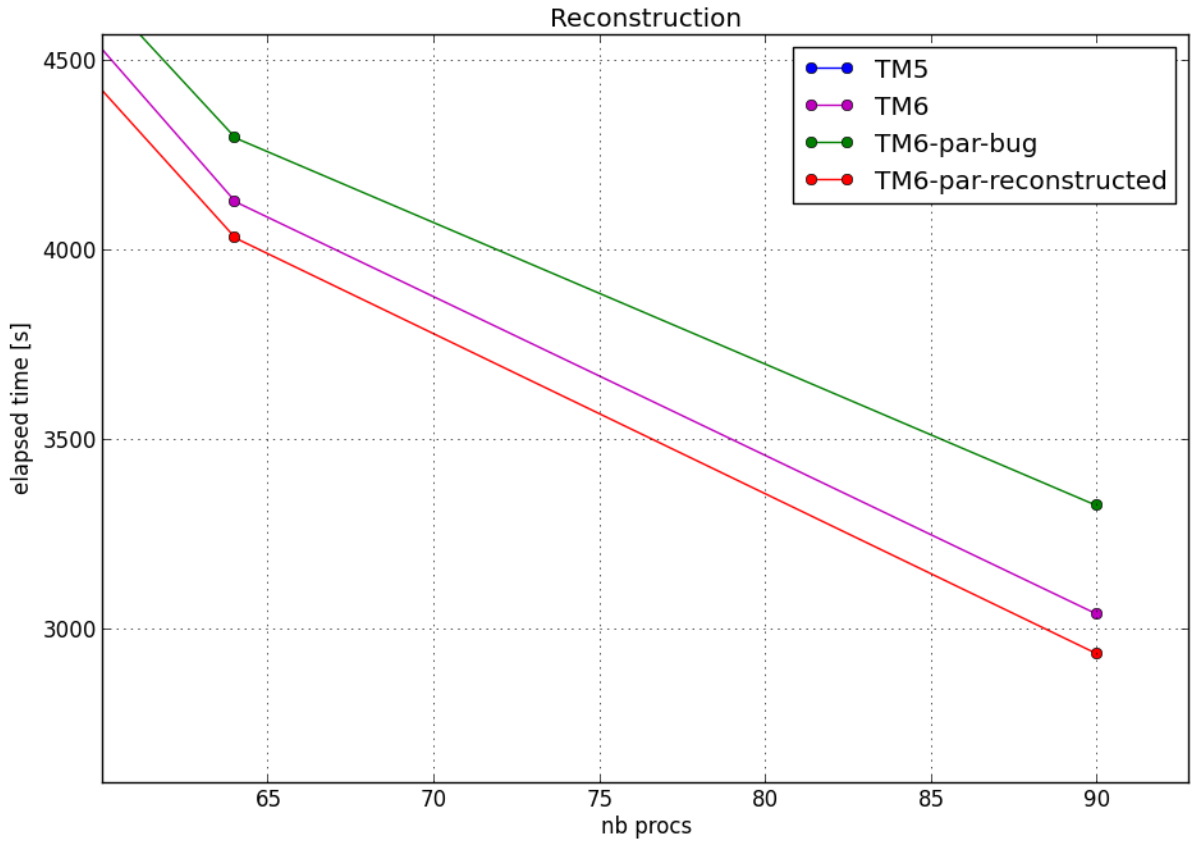


- parallel i/o should save at least 8.5%
- read meteo: from 12% to 4%

Chemistry @1x1 - Overall



Chem @1x1 - Reconstruction zoom

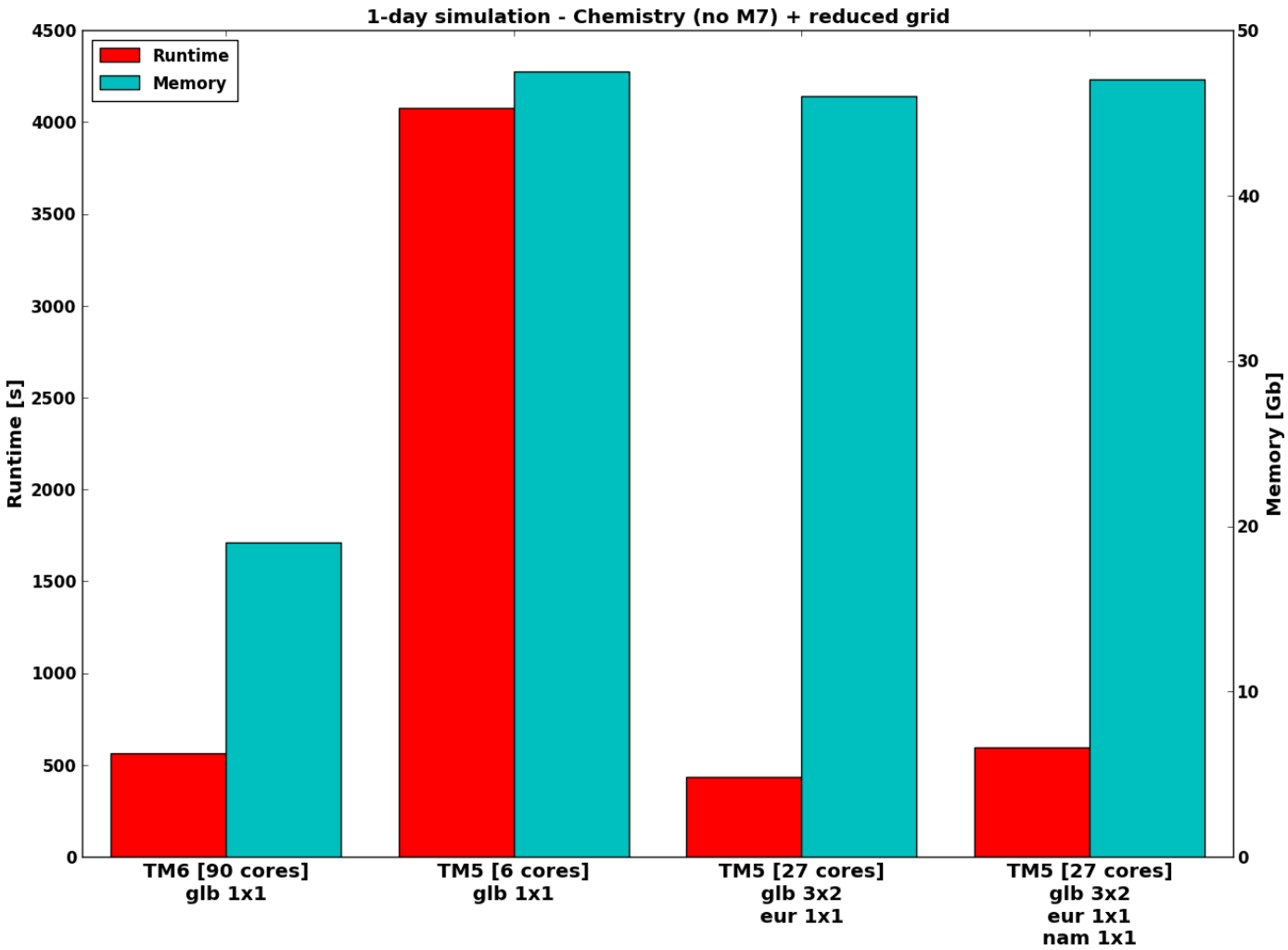


- parallel i/o should save about 3.5%
- read meteo: from >5% to <2%

Summary

- main gain from parallel I/O : no scattering and cheap remap
- the less the number of tracers, the larger the potential gain
- need netcdf 4.1.2 at least, and configured with –enable-parallel-tests

How runs @1x1 compare with runs @3x2?



Outline

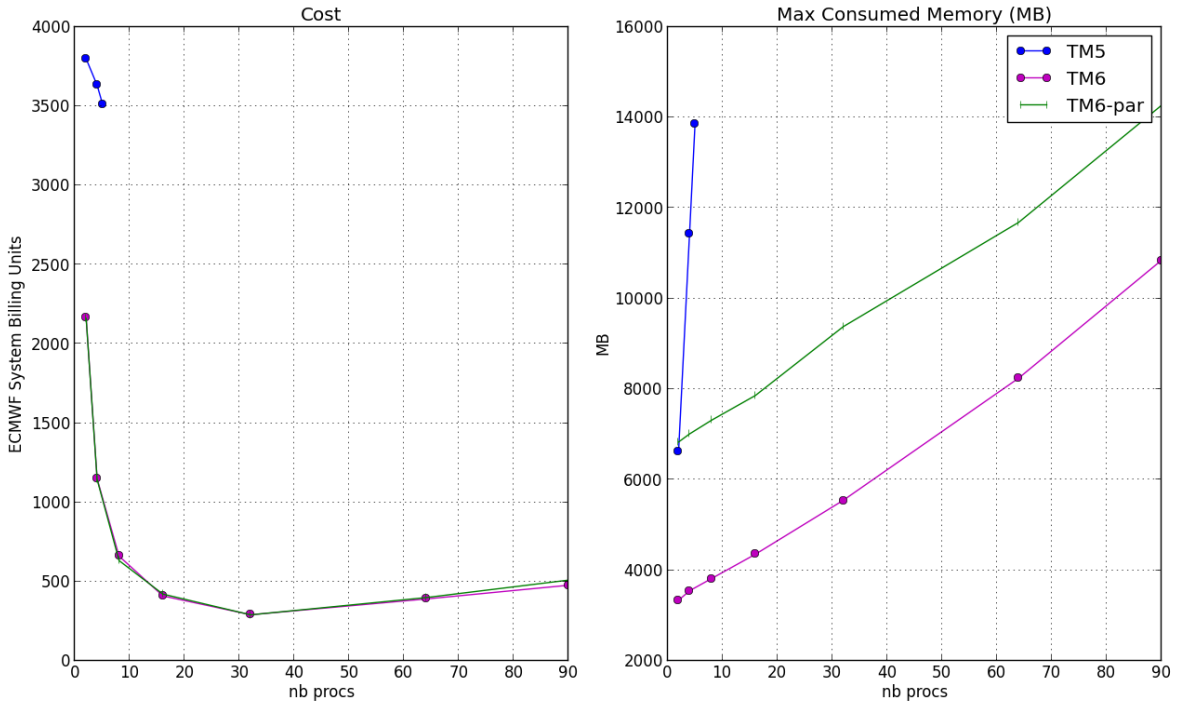
What's TM6

EC-Earth

Parallel I/O

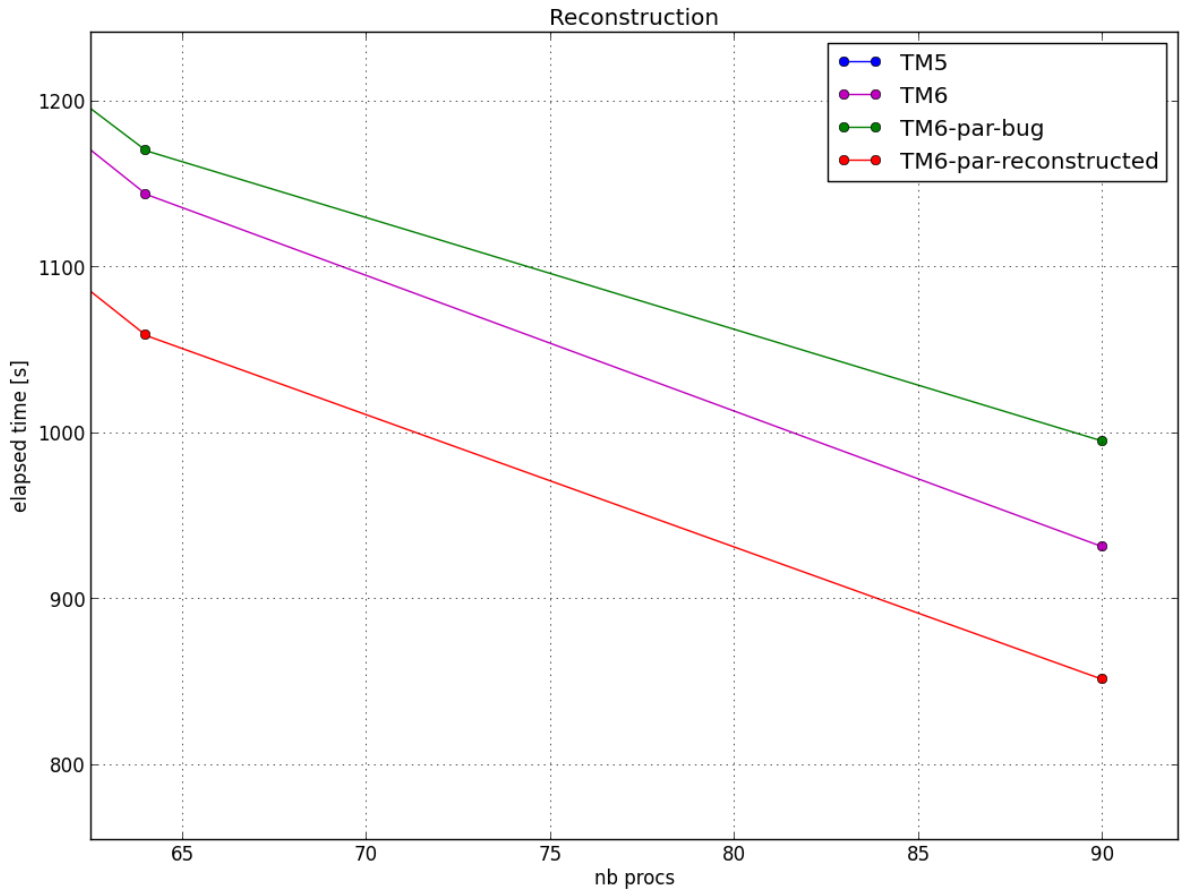
Extra

Base @1x1 - memory and cost

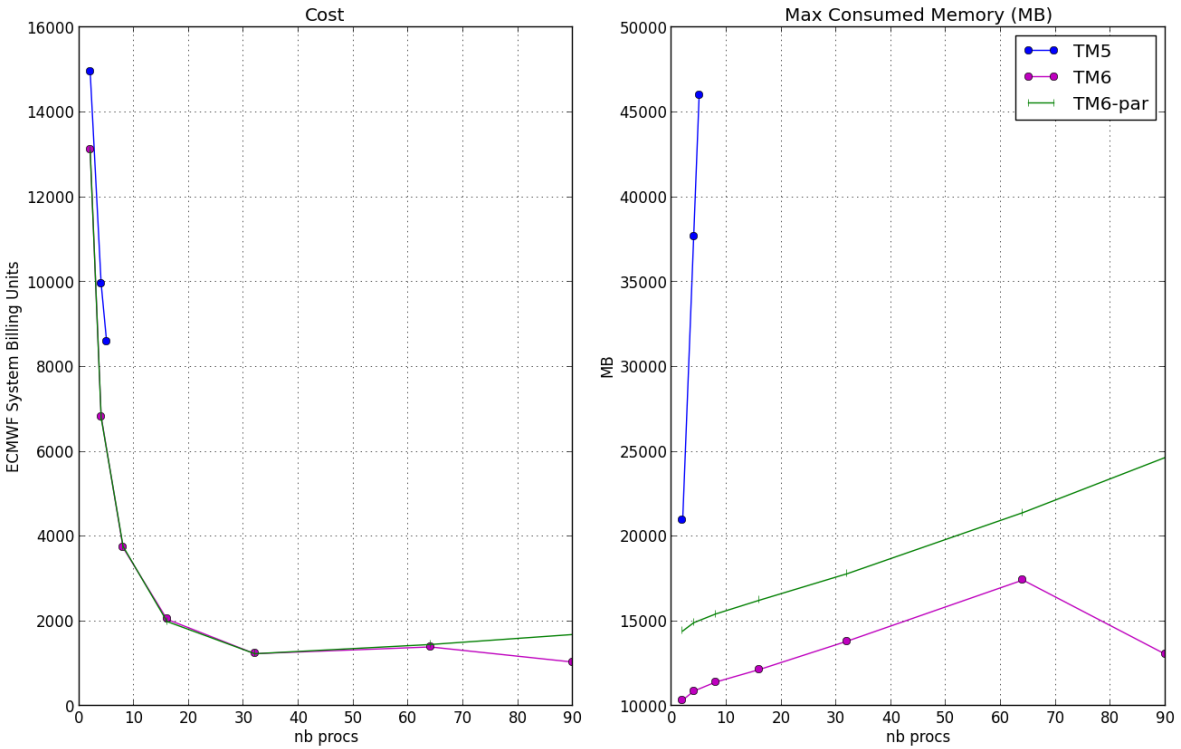


- 7 x cheaper
- more memory w/ parallel I/O (little value here)

Base @1x1 - Reconstruction zoom



Chem @1x1 - memory and cost



- 5-8 x cheaper
- more memory w/ parallel I/O (little value here)