TM6 : status as of November 2011

"the first results"

P. Le Sager

Royal Netherlands Meteorological Institute (KNMI)
The Netherlands

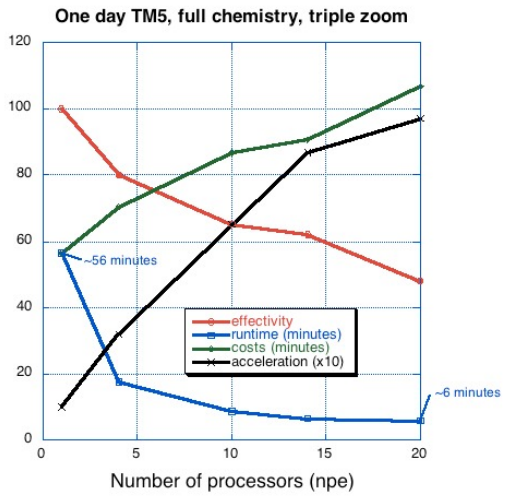
2011-11-29 - TM5 meeting - Wageningen

Outline

Outline

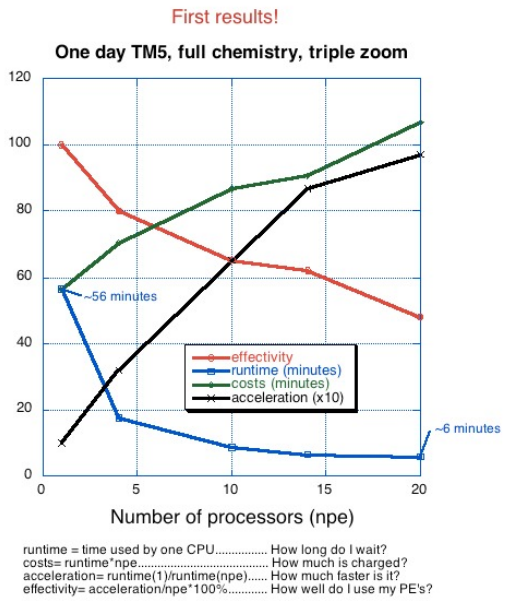
Scalability issue

First results!



runtime = time used by one CPU..... How long do I wait?
costs= runtime*npe..... How much is charged?
acceleration= runtime(1)/runtime(npe)..... How much faster is it?
effectivity= acceleration/npe*100%..... How well do I use my PE's?

Scalability issue



Limits

- EC-Earth
  - IFS and NEMO **scale** way better than TM5
  - a decade or two, no ensemble run
- Hi-Res slower than real time!

Ways to better scalability

- minimize sequential parts
- minimize communication
- load balancing

MPI implementation in TM5

ARRAYS(lon, lat, levels, tracers)
decomposed along levels **AND** tracers (**separately**)

MPI implementation in TM5

ARRAYS(lon, lat, levels, tracers)
decomposed along levels **AND** tracers (**separately**)

THEN

1. processor starvation > **42**
2. large communication (switch decomposition)
3. large memory requirements
4. code complexity (which_par, tracerloc, lmloc, tracerorder)

So, where is the bottleneck? MPI profiling

2-days sim, full-chemistry @ 3x2, 4 MPI tasks

task id	comm(s)	elapsed(s)
0	16.12	470.73
3	259.99	470.73
2	260.47	470.73
1	263.86	470.73

So, where is the bottleneck? MPI profiling

2-days sim, full-chemistry @ 3x2, 4 MPI tasks

task id	comm(s)	elapsed(s)
0	16.12	470.73
3	259.99	470.73
2	260.47	470.73
1	263.86	470.73

MPI communication alone = 55% runtime !

MPI profiling (part #2) - Data for MPI rank 3 of 4

MPI Routine	#calls	avg. bytes	time(sec)
MPI_Comm_size	1	0.0	0.000
MPI_Comm_rank	11	0.0	0.000
MPI_Bcast	**5537**	289246.1	**235.000**
MPI_Barrier	1080	0.0	2.430
MPI_Gatherv	19	5286110.3	0.460
MPI_Scatterv	61	777600.0	0.012
MPI_Allreduce	321	1206723.9	6.985
MPI_Alltoallv	976	5711440.0	14.740

MPI profiling (part #2) - Data for MPI rank 3 of 4

MPI Routine	#calls	avg. bytes	time(sec)
MPI_Comm_size	1	0.0	0.000
MPI_Comm_rank	11	0.0	0.000
MPI_Bcast	**5537**	289246.1	**235.000**
MPI_Barrier	1080	0.0	2.430
MPI_Gatherv	19	5286110.3	0.460
MPI_Scatterv	61	777600.0	0.012
MPI_Allreduce	321	1206723.9	6.985
MPI_Alltoallv	976	5711440.0	14.740

MPI_Bcast alone = 50% runtime !

MPI profiling (part #3) - MPI_Bcast details

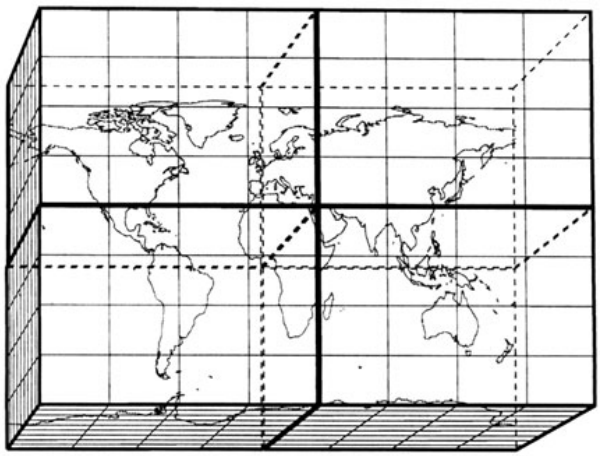
#calls	avg. bytes	time(sec)
363	2776633.8	**194.825**
382	518400.0	8.099
4507	86432.3	31.054
247	24480.0	1.295
2	4096.0	0.045
1	1320.0	0.000
2	720.0	0.006
2	256.0	0.000
3	120.0	0.000
1	56.0	0.000
3	24.0	0.000
3	8.0	0.000
21	4.0	0.039

Lots of..

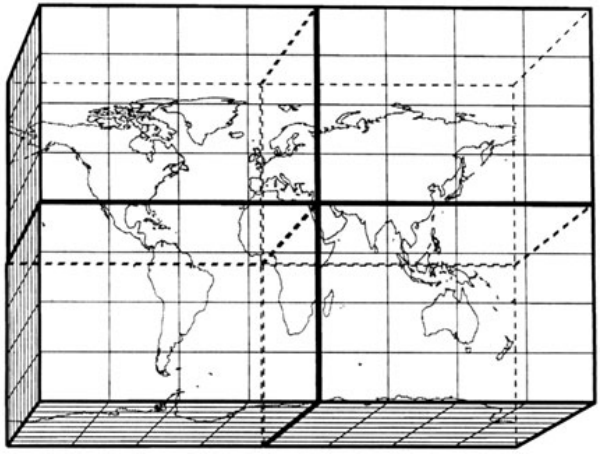
- call
- data

Outline

Revised domain decomposition



Revised domain decomposition



	TM5	TM6
processor starvation	42	10800 (3x2)
communication	swapping	halo update
memory	global arr.	shrinking arr.
code complexity	aware of //-ization	transparent to //-ization

Strategies to set up 2D topology

coarrays (Fortran 2008)

- extension for parallel processing : array(*)[*]
- growing implementation (CRAY, g95, intel, gfortran)
- **BUT** no collective communication

Strategies to set up 2D topology

coarrays (Fortran 2008)

- extension for parallel processing : array(*)[*]
- growing implementation (CRAY, g95, intel, gfortran)
- **BUT** no collective communication

ESMF (Earth System Modeling Framework)

- overhead
- worth for modular coupling (but not used in EC-Earth)

Strategies to set up 2D topology

coarrays (Fortran 2008)

- extension for parallel processing : array(*)[*]
- growing implementation (CRAY, g95, intel, gfortran)
- **BUT** no collective communication

ESMF (Earth System Modeling Framework)

- overhead
- worth for modular coupling (but not used in EC-Earth)

classic MPI

- with MPI_CART_xxx tools : create, coords, get, ...
- or **without** [elected]

Development choices

- 42 tracers of chem/base
- up-to-date trunk only (pycasso only)
- no zoom (yet)
- new rc keys : par.nx & par.ny (par.nx * par.ny = par.ntask)
- 3 grids:
  - global one-cell
  - local grid (6x4, 3x2, . . .)
  - extra ~~global~~ local 1x1

TDD : Test Driven Development

- test = **bitwise** agreement b/w TM5 & TM6 final restart file
- fully **automatic** (stress free!)

```
1  : c1a[1048] ~/TM5 > tm5_test.py tm6.rc tm5.rc
   :
3  : Compare tm6.rc and tm5.rc
   :
5  : submitting run for tm6.rc... submit ok
   : submitting run for tm5.rc... submit ok
7  : checking run for tm6.rc
   :     checking last jobstep: /tmp/tm6/run/tm6_001.rc
9  : Restart converted to netCDF-3
   : ...
11 : comparing:
   :     /tmp/tm6/restart/TM5_restart_20060101_0200_glb600x400.nc3
13 :     /tmp/standard/restart/TM5_restart_20060101_0200_glb600x400.n
   :
15 :   SUCCESS
```

refactoring

- improve nomenclature-zoology (trace0, trace1, . . .)
- improve documentation
- remove non-pycasso code
- remove obsolete code (e.g. geomtryh)

Outline

distributed grid object

```
1  TYPE DIST_GRID

3     ! parameters for global domain
      integer :: im_world         ! number of longitudes
5     integer :: jm_world         ! number of latitudes

7     ! parameters for local domain
      integer :: i_strt           ! begin local domain longitude index
9     integer :: i_stop           ! end   local domain longitude index

11    integer :: j_strt           ! begin local domain latitude index
      integer :: j_stop           ! end   local domain latitude index
13
      type(TllGridInfo) :: lli       ! local Lat-Lon grid info
15    type(TllGridInfo) :: glbl_lli ! global Lat-Lon grid info
      type(TllGridInfo) :: lli_z     ! local zonal grid info
17
      logical :: has_south_pole  ! south pole is in local domain
19    logical :: has_north_pole  ! north pole is in local domain

21 END TYPE DIST_GRID
```

Local Indices

j_start, . . . , j_stop = 1, . . . , n

- if global coordinate needed : process-dependent offset

j_start, . . . , j_stop = global indices [elected]

- given value is easily understood independent of process
- debugging made easy
- easy I/O when interface with “start” and “stride”

Price = must specify lower bounds

- procedure dummy variables

```
! before
real, intent(inout) :: arr(:,:,:)
! now
real, intent(inout) :: arr(dgrid%i_start:,dgrid%j_start:,:)
```

- local arrays

```
! before, automatic
real :: arr(im)
! now, still automatic
real :: arr(dgrid%i_start:dgrid%j_stop)
! or allocatable
real, allocatable :: arr(:,:,:)
```

- always be **careful** with pointers

```
p  =>  sp_dat%data( i1:i2 )  ! indices= 1,...,i2-i1+1
p  =>  sp_dat%data( : )      ! indices=i1,...,i2
```

Outline

Decomposition

Module with

- domains **definition** and distributed grid object
- main **communication** methods
  - point-to-point : halo_update
  - collective : gather, scatter
  - 2D, 3D, 4D
  - any halo

Restart

OPTION	istart	TM6	TESTED	COMMENT
zeroed fields	1	X		
“random” values	2	X		new set of values
read save file	30	X		read HDF4 on 1 proc
read save file	31	X	X(*)	read HDF4 on 1 proc
read restart	33	X	X	parallel I/O
read “saveoldfile”	4			
read mmix	5			
user_input	9	X		dummy
write restart		X	X	parallel I/O
write save file		X		

- none deals with tendencies yet
- (*) overwrite with TM4 fields implemented but not tested

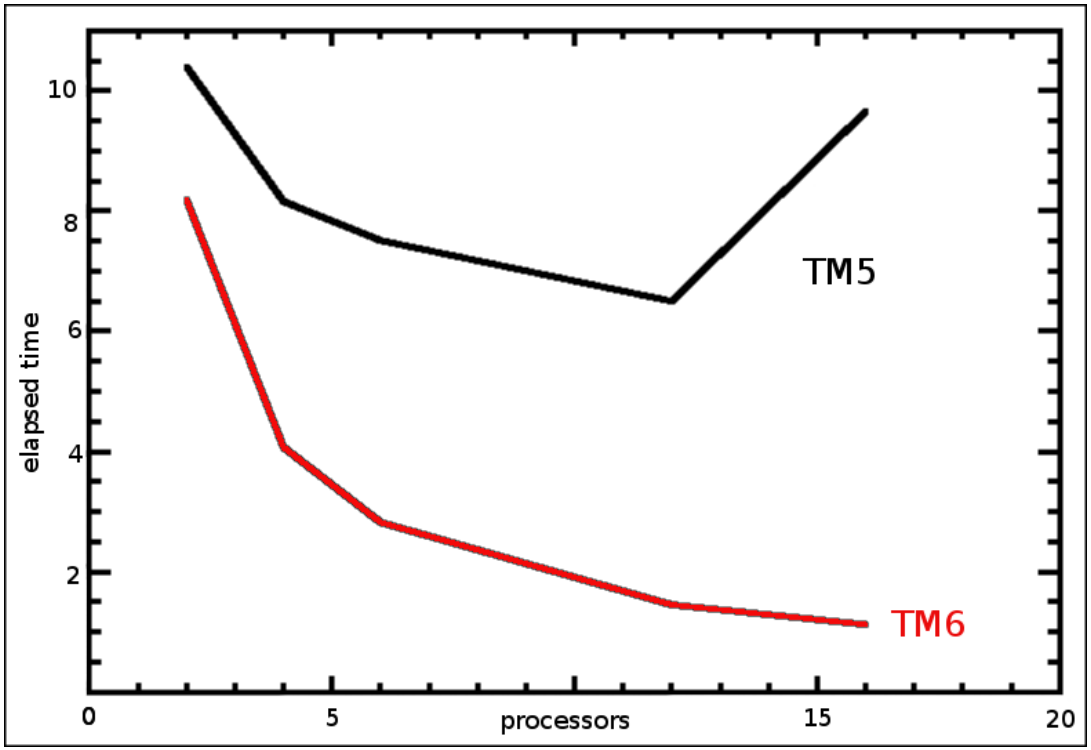
Meteo

- declaration / allocation
- reading on root, then scatter
  - work with all current formats
- I/O more and more of a bottle neck

Other Processes

- **output_mmix** : implemented, tested (-O2 instead of -O3)
- **convection** : implemented, tested

THE FIRST RESULT - Runtime for "RUN step" (convection, accum. mmix)



w/ 12 procs : 4.5 times FASTER !

Gain from 4 to 6 procs : 8% (TM5), 30% (TM6)

NEXT

- TM6-ize remaining of the code (inc. EC-Earth)
- load balancing : issues occur mostly due to complicated geometry (poles, day/night?)

I/O... more and more the bottleneck

- parallel : optimal number of PEs (too many => large overhead)
- non-parallel : move the burden to mpi communication
- solution : using MPI I/O (full exploited : datatype & data sieving)
- hardware issues (file system)