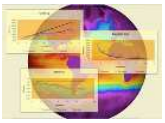
Oasis3: an MPI1/2 per-field parallel approach

I. Epicoco, S. Mocavero, G. Aloisio

Euro-Mediterranean Center for Climate Change (CMCC)
Scientific Computing and Operations (SCO) Division
Director prof. Giovanni Aloisio

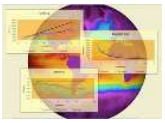
OASIS User Meeting, May 2009, Tolouse

Outcome



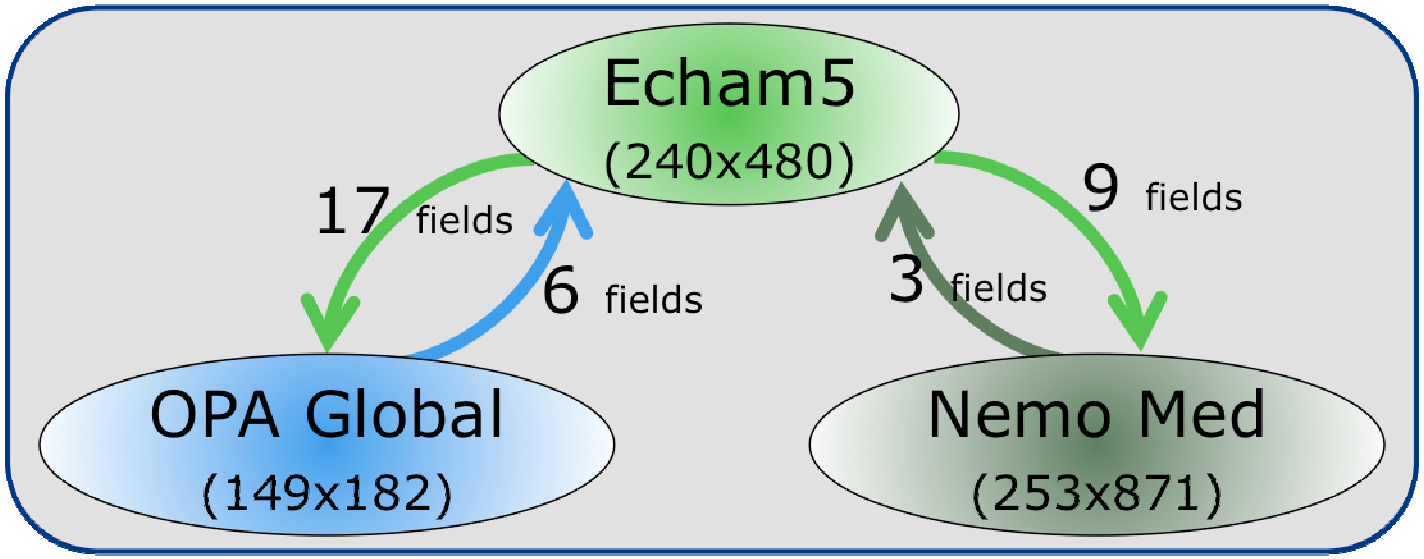
- Goal
- Case study
- CMCC Supercomputing Center
- OASIS3 performance analysis
- OASIS3 Optimization
  - Optimization of EXTRAP transformation
  - EXTRAP numerical displacement
  - EXTRAP performance evaluation
  - Optimization of SCRIPR transformation
  - Performance evaluation
- Parallelization
  - parallel algorithm
  - data dependence issues
  - parallel model
  - parallel OASIS3 performance evaluation
  - MPI1/2 implementation
  - parallel OASIS3 validation
- Pseudo-parallel OASIS3 vs parallel OASIS3
- Next steps

Goal



Reducing the wall clock time for CMCC-MED coupled model currently deployed on NEC SX9

CMCC-MED coupler configuration

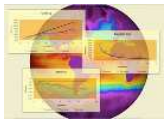


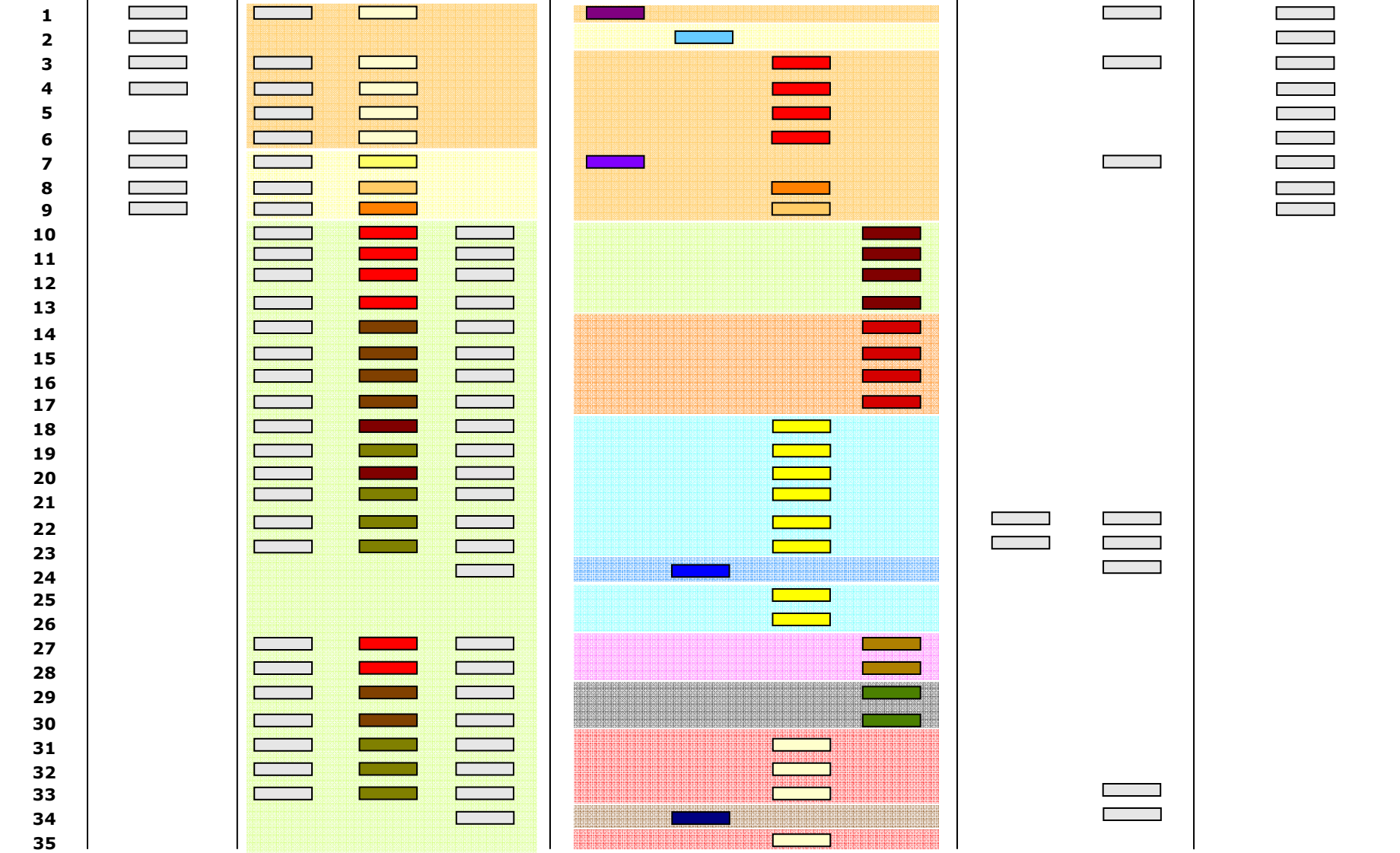
number of fields	35
coupling period	2h40'
coupling steps in one month	279

ANS Division

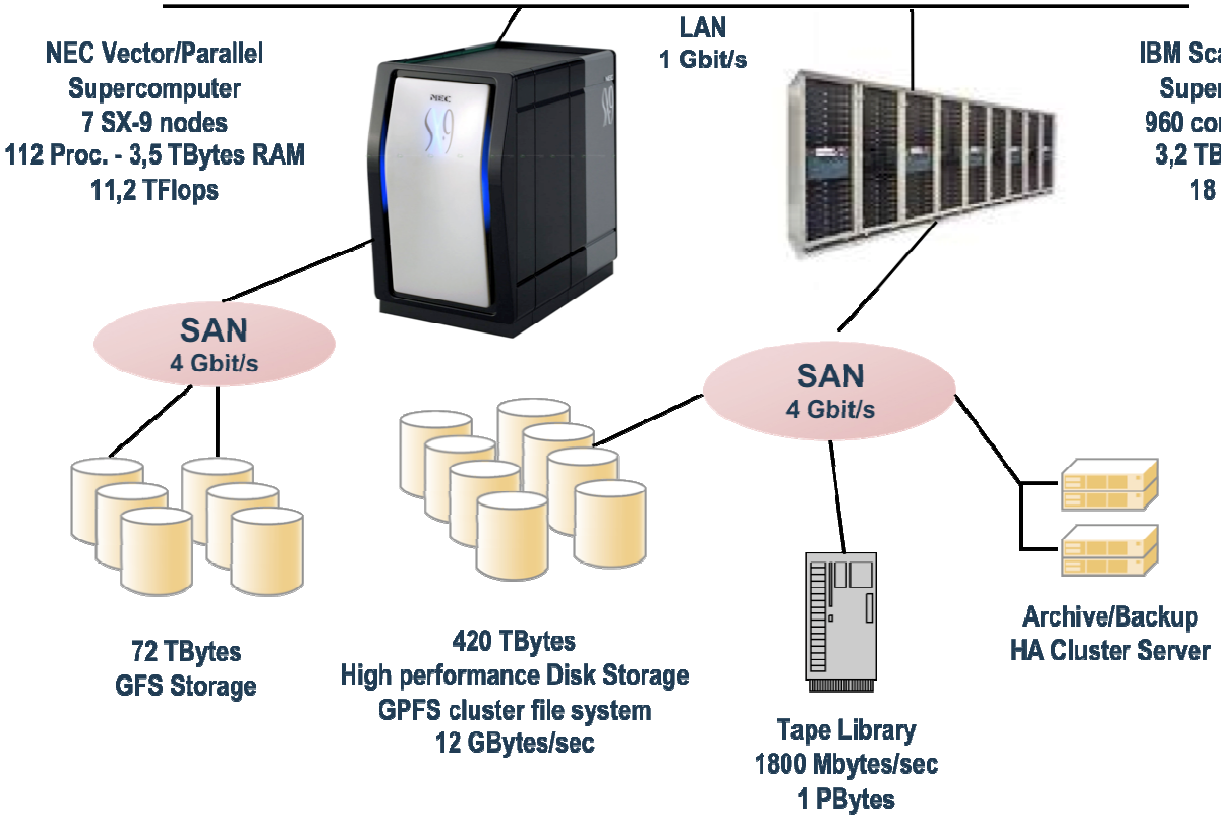
E. Scoccimarro et al.

Case study




field
Time transf
Locktrans
(avarage)
Pre-proc transformation
Mask
Extrap
Invert
(ninenn)
Interp transformation
Scripr
(distwgt conserv bilinear bicubic)
Cooking stage
Conserv
Blasnew
(global)
Post-proc transf
Reverse
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35

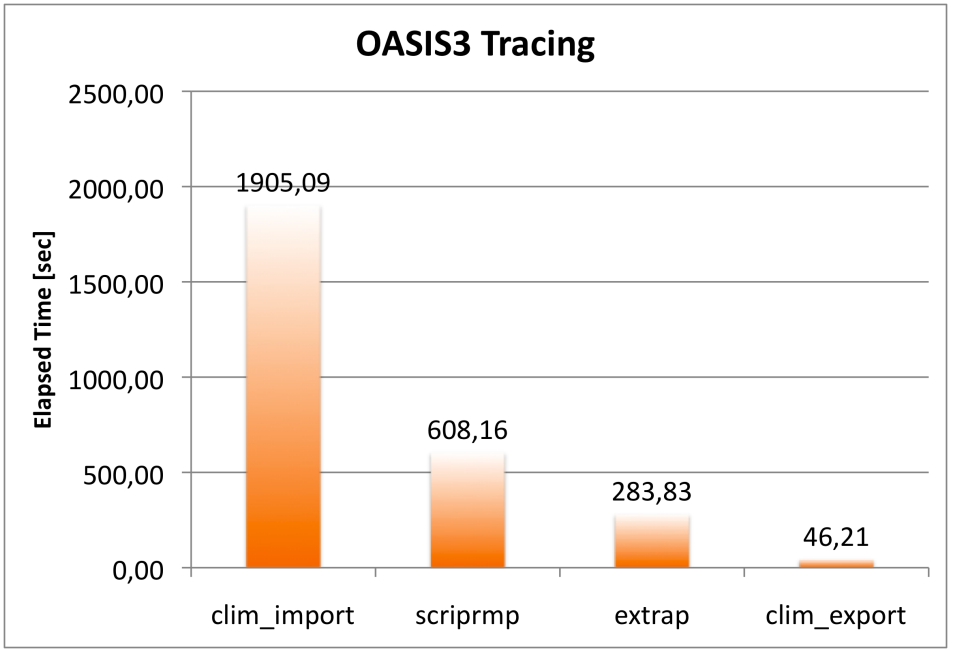
CMCC – Supercomputing Center

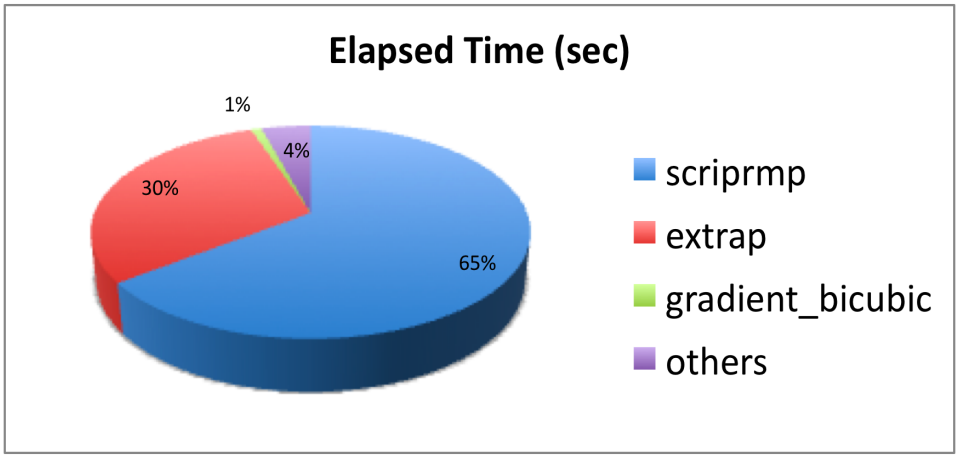


	NEC SX-9
Performance per CPU	Over 100 GF
Machine cycle (clock)	3.2 GHz
Memory bandwidth	4 TB/s
Memory capacity per node	512 GB/1 TB
CPUs per node	16
Performance per node	1.6 TF
I/O Data rate	64 GB/s
Internode bandwidth	128 GB/s x 2

OASIS3 performance analysis

The time spent by *clim_import* routine depends only on the models.
At the moment we take into account only the coupling process.



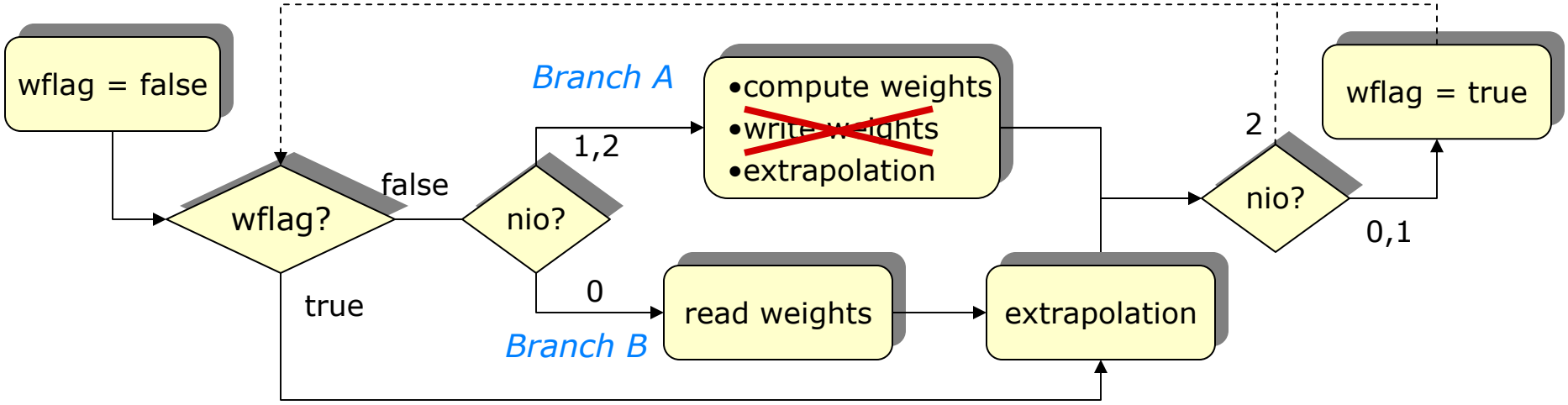


sciprmp and *extrap* functions must be taken into account for optimization.

	Elapsed Time (sec)	%
sciprmp	608,16	64,61%
extrap	283,83	30,15%
clim_export	46,21	4,91%
others	3,14	0,33%
Total Coupling Time	941,35	

Optimization of *EXTRAP* transformation

Upon computation of the first field of a given dataset, the weights for extrapolation are computed (for NIO=1 or 2) or they are read from file (for NIO=0). The weights are then kept in memory for the next fields belonging to the same dataset.



Branch A and Branch B are mutually exclusives. This implies that the weights written in the file during the branch A are never read.

We can optimize the function avoiding the weights writing

EXTRAP numerical displacement

The extrapolation is replicated twice within the code (Branch A and Branch B).

The optimization level of compiler can cause numerical displacement between the extrapolation performed on different branches.

The measured numerical displacement after the extrapolation is of order of **4,05e-16**.

The displacement on some fields, produce an average numerical shifting of **0,25%** on the variables after one simulated month.

VOVERTAKE directive, defined on *count* loop in Branch B, is a further source of numerical displacement.

We canceled it from the loop

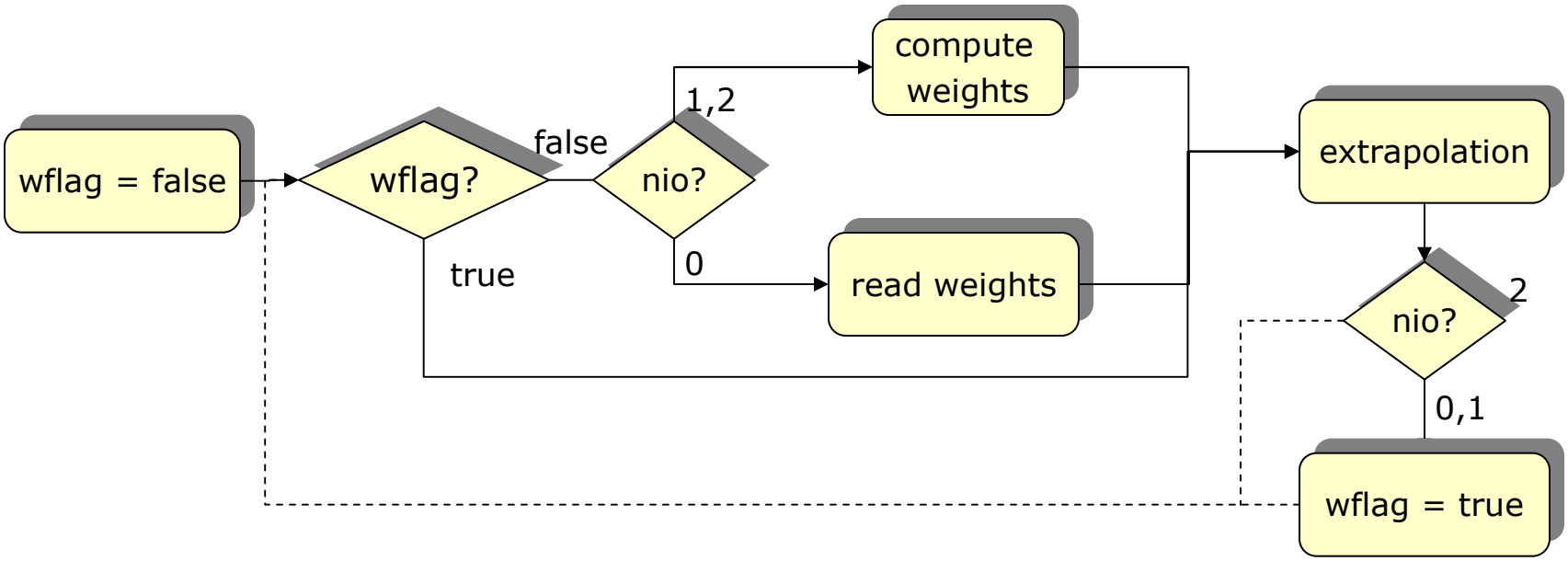
Branch B

```
            DO ind = 1, count
               ji = list_ji(ind)
               jj = list_jj(ind)
               iind=(jj-1)*kxlon + ji
               pwork(ji,jj) = 0.
!cdir unroll=9
               DO 250 jl = 1,9
                 IF (zweights(knb,jl,iind) .NE. 0.) THEN
                   idivi=iaddress(knb, jl, iind)/kxlon
                   imult=idivi*kxlon
                   IF (iaddress(knb,jl, iind) .EQ. imult) THEN
                     ilat=idivi
                     ilon=kxlon
                   ELSE
                     ilat=idivi+1
                     ilon=iaddress(knb,jl, iind) imult
                   ENDIF
                   pwork(ji,jj) = pwork(ji,jj) + pfild(ilon,ilat)
     $                           *zweights(knb,jl,iind)
                 END IF
250            CONTINUE
            END DO
```

Branch A

```
            DO 210 jj = 1, kylat
               DO 221 ji = 1, kxlon, kxlon-1
                  .
                  .
                  .
                IF (inbor .GE. ivoisin) THEN
C
C* Some points around P are not masked so we use them to extrapolate
C* and define the iteration number, weight and address variables
C
                   pwork(ji,jj) = 0.
                   iincre(knb, iind) = incre
                   DO 243 jl = ideb, ifin
                    ilon = ix(jl)
                    ilat = iy(jl)
                    pwork(ji,jj) = pwork(ji,jj)
     $ $               + pfild(ilon,ilat) * zmask(jl)
                      / FLOAT(inbor)
                    iaddress(knb,jl,iind)=(ilat-1)*kxlon+ilon
                    zweights(knb,jl,iind)=zmask(jl)/FLOAT(inbor)
243                 CONTINUE
                ENDIF
             ENDIF
221          CONTINUE
210       CONTINUE
```

EXTRAP performance evaluation



The weights evaluation and extrapolation in *Branch A* have been disjoined

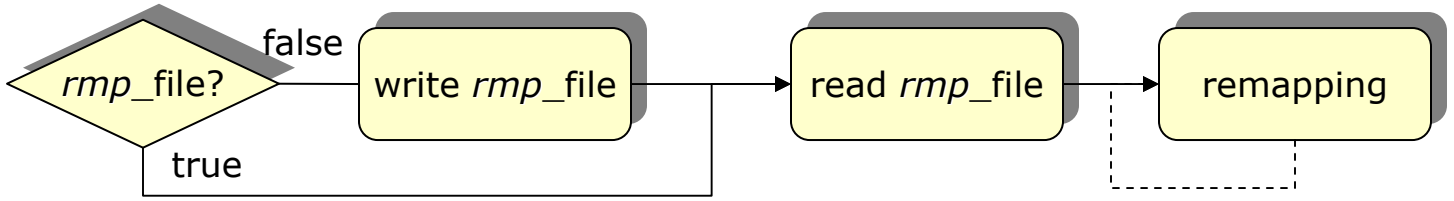
Branch A only computes the weights; the extrapolation is done only in *Branch B* for all fields (also for those ones with NIO=1)

The elimination of weights writing produces a modest performance improvement.

It is done only on the first coupling step and only for a few number of fields

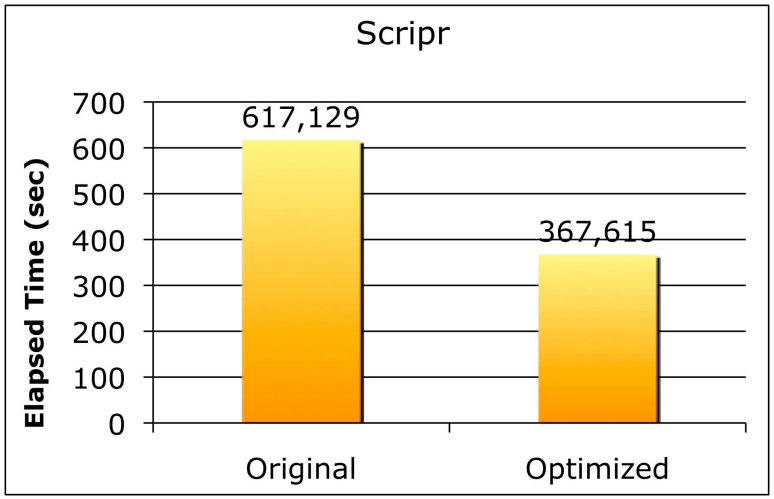
	Extrap		
	Elapsed Time (sec)	Saved Time (sec)	%
Original	286,218		
Optimized	285,032	1,186	**0,41%**

Optimization of *SCRIPR* transformation



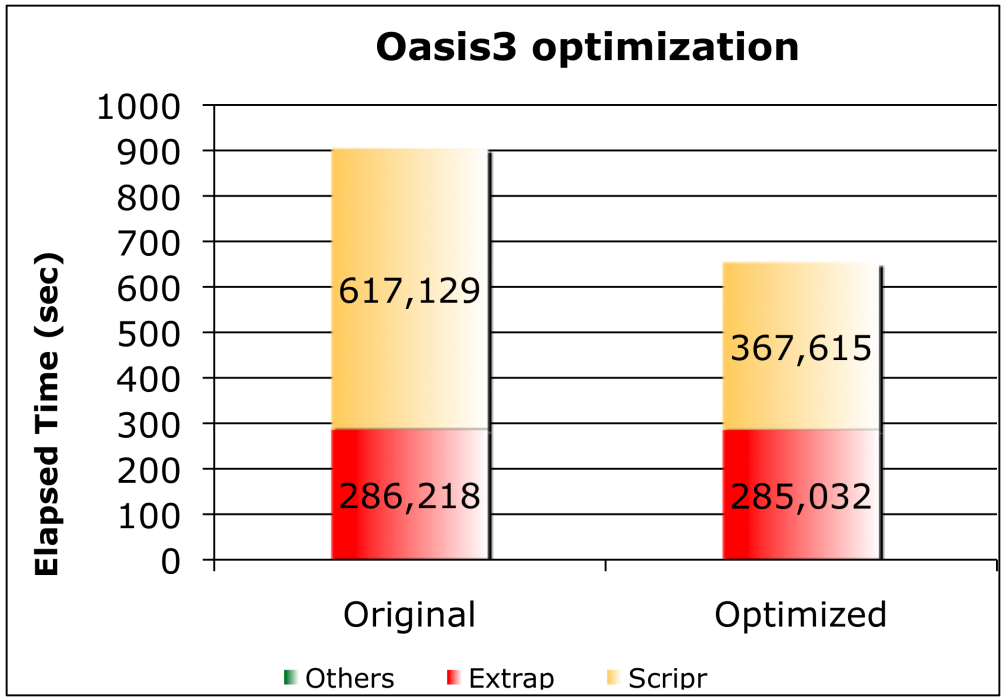
The remapping files are written once (at the first coupling step) and read every step

The IO operations can be optimized keeping in memory the remapping data



	Scripr		
	Elapsed Time (sec)	Saved Time (sec)	%
Original	617,129		
Optimized	367,615	249,514	**40,43%**

OASIS3 Optimization



The optimization of the *extrap* and *scripr* routines gets a reduction of elapsed time on the whole coupling operation of about 27%

	Extrap	Scripr	Others	Coupling	Saved Time (sec)	%
Original	286,218	617,13	1,008	904,36		
Optimized	285,032	367,62	1,018	653,67	250,690	**27,72%**

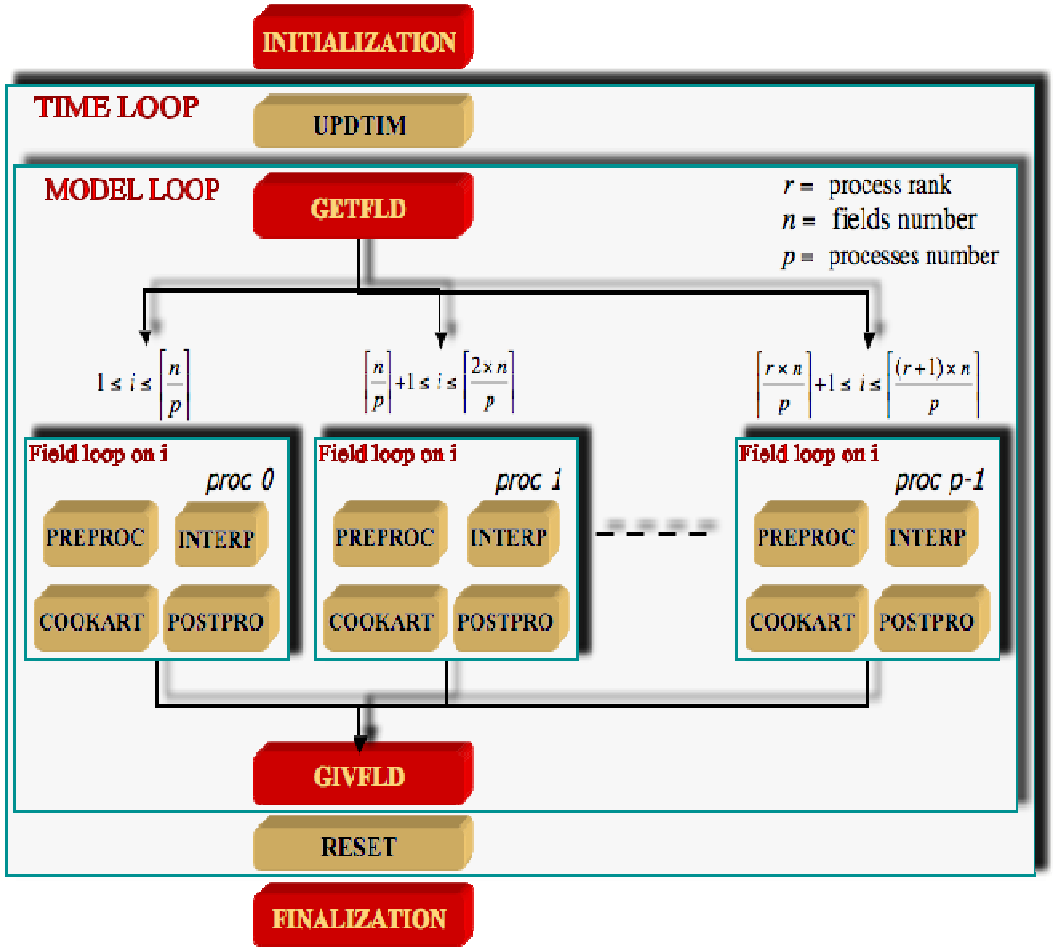
Parallel algorithm

The parallelization is based on the distribution of the fields among the available processes

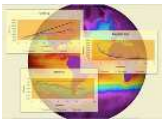
The OASIS master process gets fields from the models and scatters them among the OASIS slave processes

Each slave executes the coupling transformations for the assigned fields

The master gathers fields from the slaves and exports them to the models



Data dependence issues



Extrapolation dataset management

the weights defined for the first field of a given dataset (NIO=1), are used for all the others fields belonging to the same dataset.

The semantics of NIO=0 has been extended

If weights are not stored in memory and a field with NIO=0 has to be extrapolated, the weights are evaluated unless the *nweight* file exists.

BLASNEW, BLASOLD

They could introduce dependences among fields

A slight overhead has been introduced. Let *F2* requires a linear combination with *F1*. If process *i* has to transform *F2*, then also *F1* should be assigned to process *i*

Parallel model

Parallel time is given by:

$$T_{par} = T_{init} + T_{couple} + T_{\mathrm{mod}\,els} + T_{com} + T_{end}$$

Intrinsic sequential time is given by:

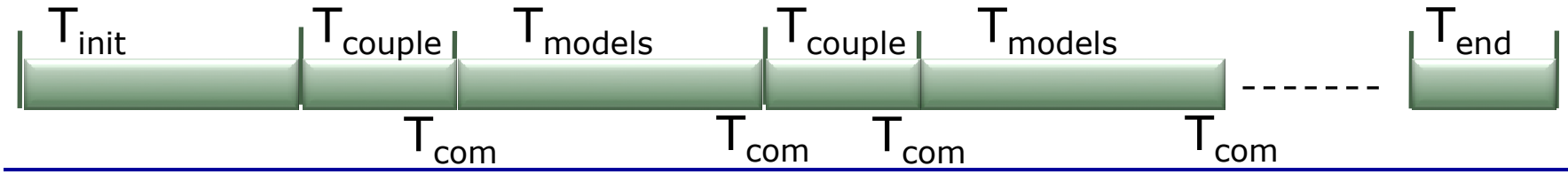
$$T_{seq} = T_{init} + T_{\mathrm{mod}\,els} + T_{end}$$

Let *i* the process number, *j* the field number and *nfield* the total number of fields, then:

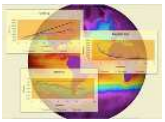
$$T_{couple} = \max_i \sum_{j=\left\lceil \frac{nfield \times i}{p} \right\rceil + 1}^{\left\lceil \frac{nfield \times (i+1)}{p} \right\rceil} T_{couple_j}$$

Let *p* the total number of processes, num_{couple} the number of coupling steps, T_s the communication startup time and T_B the time needed to transfer 1 Byte, so:

$$T_{com} = T_{send/recv} + T_{broad} = \sum_{j=1}^{nfield} (T_s + T_B L_i)(p + \log_2 p) \times num_{couple}$$



Parallel OASIS3 performance evaluation



$$T_{com} = T_{send/recv} + T_{broad} =$$

$$\sum_{j=1}^{nfield} (T_s + T_w L_i)(p + \log_2 p) \times num_{couple}$$

givfld field dimension (byte)	getfld field dimension (byte)
921600	216944
921600	216944
921600	216944
921600	216944
921600	216944
921600	216944
921600	1762904
921600	1762904
921600	1762904
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
216944	921600
1762904	921600
1762904	921600
1762904	921600
1762904	921600
1762904	921600
1762904	921600
1762904	921600
1762904	921600
1762904	921600

Ts	3,40E-06
Tw (byte)	2,30E-11
num_couple	31x9

$$T_{couple} = \max_i \sum_{j=\left\lceil \frac{nfield \times i}{p} \right\rceil + 1}^{\left\lceil \frac{nfield \times (i+1)}{p} \right\rceil} T_{couple_j}$$

$$T_{seq} = T_{init} + T_{models} + T_{end}$$

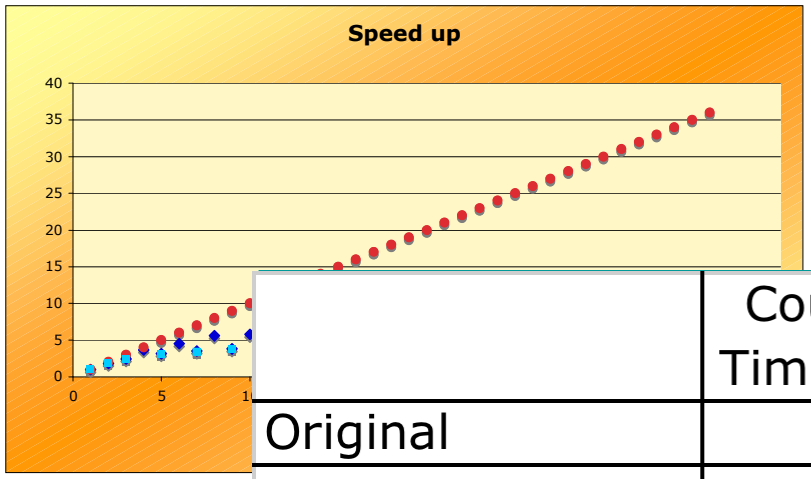
Init Time	2,08E+01
Models Time	3,67E+03
End Time	3,73E-05

couple time
1,27E+01
1,16E+01
1,20E+01
1,09E+01
1,10E+01
1,13E+01
3,53E+01
3,48E+01
3,52E+01
1,34E+01
1,29E+01
1,29E+01
1,29E+01
1,35E+01
1,28E+01
1,30E+01
1,29E+01
7,44E+01
1,10E+01
7,37E+01
1,10E+01
1,10E+01
1,10E+01
1,19E+01
7,49E+00
7,47E+00
2,25E+01
2,10E+01
2,23E+01
2,15E+01
1,57E+01
1,53E+01
1,54E+01
1,13E+01
1,14E+01

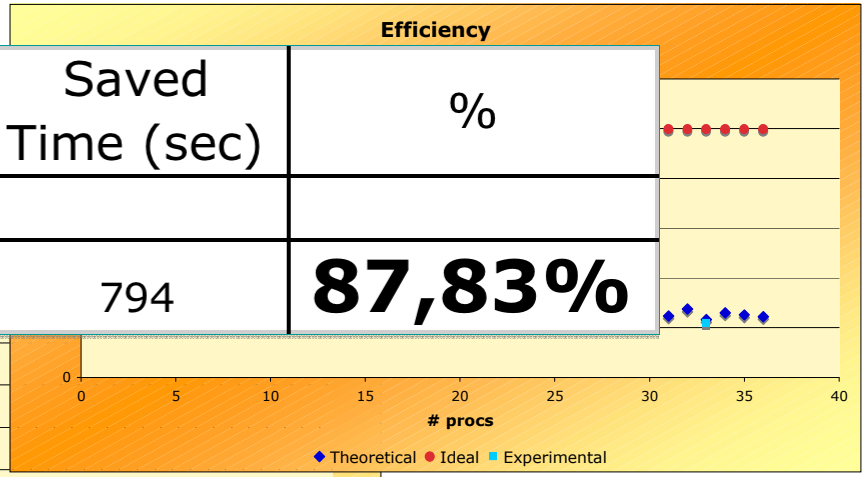
Parallel OASIS3 performance evaluation

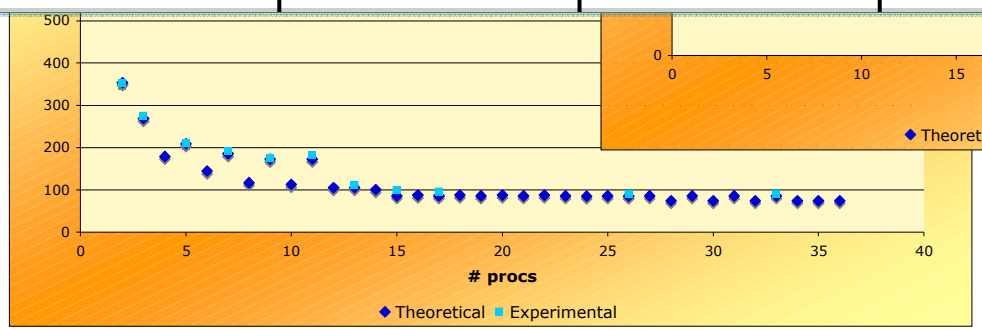
The parallel model shows that the coarse grained approach produces low efficiency with 15 procs. This is mainly due to the bad load balancing

# procs	Execution Time (sec)	Efficiency	Speed up
1	645,13	1,00	1,00
2	351,8	0,92	1,83
3	274,86	0,78	2,35
5	210,83	0,61	3,06
7	191,12	0,48	3,38
9	174,17	0,41	3,70
11	181,22	0,32	3,56
13	110,77	0,45	5,82
15	99,71	0,43	6,47
17	95,28	0,40	6,77
26	90,1	0,28	7,16
33	89,59	0,22	7,20

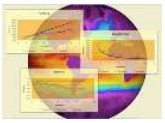


	Coupling Time (sec)	Saved Time (sec)	%
Original	904		
Parallel (13 proc)	110	794	**87,83%**





MPI1/2 implementation

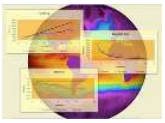


- MPI 2 Implementation
  - The number of processes for OASIS is defined as argument of the *mpiexec* command
  - An ad hoc communicator is created to collect all of the OASIS processes
  - The process with rank 0 is identified as OASIS master process

  *mpiexec –n **13** oasis.x –maxnp 48 …*
- MPI 1 Implementation
  - The number of processes for OASIS is still defined during the *mpiexec* execution
  - The ranks of the master processes for each model involved in the coupling are established taking into account the number of processes for OASIS
  - No new communicator is created

  *mpiexec –n **13** oasis.x : -n 20 echam5 : -n 1 opa.xx : –n 14 opa9m*

Model Validation

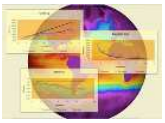


The parallel implementation has been verified with a bit-to-bit comparison against the output got from the original OASIS3 version after a 2 month simulation with restart file.

The current version has been tested only on a subset of the whole
available transformations. Namely:

- Time transformations:
  - LOCTRANS
    - ✓ AVERAGE
- Pre-processing transformations:
  - MASK
  - EXTRAP
    - ✓ NINENN
  - INVERT
- Interpolation transformations:
  - SCRIPR
    - ✓ DISTWGT
    - ✓ CONSERV
    - ✓ BILINEAR
    - ✓ BICUBIC
- Cooking stage:
  - CONSERV
    - ✓ GLOBAL
  - BLASNEW (only CONSTANT)
- Post-processing transformation:
  - REVERSE

Pseudo-parallel OASIS3 vs parallel OASIS3



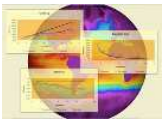
Pseudo-parallel

- allows *ad hoc* fields distribution
- distributed management of coupler communication with models
- different configuration and auxiliary files have to be created by the user
- available only with the MPI1 CLIM communication technique

Parallel

- a single instance of namcouple and auxiliary files is needed
- available with MPI1 and MPI2 CLIM communication techniques
- allows the user to change the number of oasis processes simply modify the mpiexec command line
- master process can represent a bottleneck (due to communication or memory issues)
- load balance is not optimized due to a coarse grained parallelization

Next steps

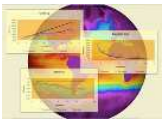


- Further optimization of the OASIS on the vector machine

  we still have almost 40% of memory bank conflicts

- Evaluation of Oasis4 and integration of per-field parallel approach
- Performance evaluation of parallel OASIS3 on scalar architecture IBM power6
- Parallel coupler validation on the whole set of available transformations
- Comparison with other couplers such as the NCAR csm Flux coupler

Further information



- More information about the implementation are available on the following Research papers:

OASIS3 : Analysis and Parallelization

http://www.cmcc.it/publications-meetings/publications/research-papers/rp0052-sco-01-2009

Oasis3 parallel version: Performance Analysis

(work in progress)

- Contact us:

Euro-Mediterranean Center for Climate Change (CMCC)
Scientific Computing and Operations (SCO) Division
Director prof. Giovanni Aloisio

e-mail: sco-hec@cmcc.it