Centro Euro-Mediterraneo
per i Cambiamenti Climatici

Issue RP0068
July 2009

Scientific Computing and Operation (SCO)

THE OASIS3 MPI1/2 PARALLEL VERSION

By **Italo Epicoco**
University of Salento, Italy
italo.epicoco@unisalento.it

Silvia Mocavero
CMCC
silvia.mocavero@cmcc.it

and **Giovanni Aloisio**
CMCC
University of Salento, Italy
giovanni.aloisio@unisalento.it

SUMMARY This work describes the optimization and parallelization activities performed on the OASIS3 coupler. The test case used for evaluating and profiling the coupler consists on the CMCC-MED coupled model developed by the ANS division of the CMCC and currently in production on the NEC SX9 cluster. The experiments highlighted that the most time consuming transformations are the extrapolation of the fields on the masked points (performed in the *extrap* function) and interpolation (performed in the *scriprmp* function). The optimization has been mainly focused on reducing the time spent for I/O operations this reduced the coupling time of 27%. The parallelization of the OASIS3 has been a further step for reducing the elapsed time of the whole coupled model. The proposed parallel approach is based on the distribution of the fields among the available processes. Each process is in charge to apply the coupling transformations on the assigned fields. With this approach the number of coupling fields represents an upper bound to the parallelization level. However this approach can be fully combined with the parallelization based on the geographical domain distribution. The work concludes with a qualitative comparison of the proposed approach with the OASIS3 pseudo-parallel version developed by CERFACS.

Keywords: Oasis3, Performance evaluation, Parallel Computing

JEL: C63

The work here presented, carried out with support of the CMCC staff member of the ANS Division.

INTRODUCTION

Each stand alone model was born to describe the physical, chemical, biological behavior of complex systems such as the atmosphere, oceans, vegetation, and so on. In most of cases, these models are not complete enough to describe the real behavior of the whole climate system in its complexity, if they are considered decoupled. A more detailed approach is modeling the climate behavior by coupling models each others. Within this scenario, the coupler component becomes one of the key point for the parallel performance of the whole coupled model. The coupler is requested to act as a "collector" among the component models. Its main function is to interpolate, to extrapolate, to regrid and, more in general, to transform the exchanged fields. It must also handle and support different parallel approach in order to be compliant and portable on different parallel architectures. From its nature, the time spent by the coupler during the transformation of the fields, can not be ever overlapped with the execution of the component models. This mean that an optimization and parallelization of the coupler reflects on the wall clock time of the whole coupled model. The main goal of this work is to reduce the wall clock time of the coupled models through the optimization and parallelization of the OASIS3 coupler. The report is organized as follows: the next session introduces the profiling of the coupled model in order to identify the functions to be taken into account; the description of the optimization is described in the further 2 sections; the second half of the paper describes the parallel approach, the definition of the analytic performance model and its analysis; the work concludes with a qualitative comparison of the proposed approach with the OASIS3 pseudo-parallel version.

OASIS3 PROFILING

The OASIS3 coupler [12] has been evaluated and profiled considering the coupled model currently in production on CMCC Supercomputing Center. The CMCC-MED model (S. Gualdi, E. Scoccimarro et al.), is a 3 component coupled model made of the atmospheric model Echam5 [10] T159L31, oceanic global model OPA 8.2 [7] 2°and Nemo [6] 1/16°Mediterranean sea model. As depicted in fig. 1, the atmosphere sends to the coupler 26 fields defined on a grid of dimension 480x240, 17 of them addressed to the ocean global and 9 of them addressed to the Mediterranean sea. The ocean global model sends to the coupler 6 fields, defined on a spatial grid of 182x149, destined to the atmosphere, and the Mediterrenan sea model sends to the coupler 3 fields, defined on a spatial grid of 871x253, addressed to the atmosphere. The coupler has to handle a total of 35 fields, exchanged among the component models, with a coupling period of 2h 40′ for a total of 279 coupling steps in one month.

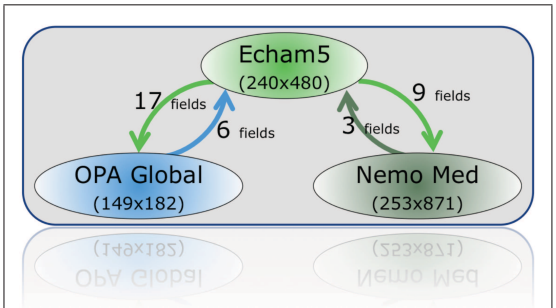


Figure 1:
CMCC-MED model configuration.

The transformations, that the coupler performs on the exchanged fields, are listed on table 1. The most frequent transformations are the extrapolation, performed on 29 fields, and the interpolation applied on 35 fields.

The coupled model has been profiled on the

Table 1
CMCC-MED namcouple configuration.

Transformation	# of fields
Locktrans	8
Mask	29
Extrap [ninenn]	29
Invert	23
Scripr [distwgt]	2
Scripr [conserv]	3
Scripr [bilinear]	18
Scripr [bicubic]	12
Conserv [global]	2
Blasnew	8
Reverse	9

NEC SX9 cluster using FTRACE [2] analysis tool in order to identify the functions that take most execution time. FTRACE provides flow trace profiling. Unlike statistical profiling, which samples execution to calculate the profile, FTRACE reads the CPU performance counters at the beginning and end of routines, providing also various performance metrics, so it is possible to infer whether the time spent in a routine is due to a performance bottleneck, and if so, what kind of performance bottleneck. The FTRACE region, defined within the OASIS3 code showed in fig 2, highlights that *clim_import* takes about 1900 seconds followed by the *scriprmp* and *extrap* functions. It is worth noting here that the *clim_import* belongs to the CLIM library used for the communication among the coupler and the component models; in particular the *clim_import* is devoted to receive the exchanging fields from models. The elapsed time spent in the function is actually the idle time the coupler has to wait for the component models to simulate the coupling period. For the optimization of the coupler we can thus safely ignore this function because it does not include computing time.

As illustrated in table 2, the most time consuming coupling transformations are the *extrap* and the *scriprmp* functions; they take about 95% of

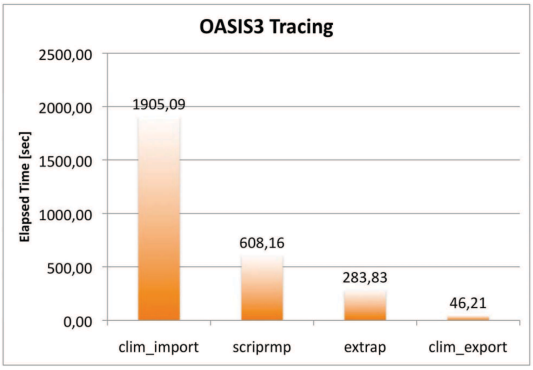


Figure 2:
OASIS3 performance tracing.

the total coupling time.

Table 2
OASIS3 performance analysis

	Elapsed Time (sec)	%
scriprmp	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	

In the following sections we give the detailed description of the optimization performed on the selected functions.

EXTRAP ANALYSIS AND OPTIMIZATION

The *extrap* function performs the extrapolation of the fields over its masked points using the field source grid. Since the weights used for extrapolation depend only on the source grid, it is convenient to group the fields into different datasets characterized by the same source grid and hence with the same weights. Within the *namcouple*, a field is also tagged with a parameter value (named NIO) that defines if the weights must be computed and written to file

(NIO=1) or read from file (NIO=0). It is worth noting here that the NIO parameter is taken into account only for the first field of a given dataset, for all other fields belonging to the same dataset its value is ignored; for these fields, the weights are always read from main memory. The flow chart on figure 3 gives an overview of the algorithm implemented in the original versione of OASIS3. The *wflag* boolean variable is used to establish if the weights and address values for dataset i are available in main memory or not. At beginning the *wflag* is initialized to *FALSE* for all of the datasets; when the coupler encounters the first field of the dataset i, the instruction control flow dependes on NIO value. Following the Branch A, both operations, the definition of the weights and the extrapolation of the field, are performed jointly, after the definition of the weights they are stored in a file. The Branch B is followed when NIO is 0 and the field is extrapolated using the weights read from file. In both cases the weights are stored in main memory and *wflag* is asserted. This implies that for all other fields of the same dataset and for all further coupling steps the extrapolation is performed reading the weights from the main memory. Considering the flow chart depicted on figure 3 it is evident that:

1. the weights and address values are written to file only when *Oasis* is transforming the first field of a given dataset and if its NIO value is equal to 1

2. the weigths and address values are read from file only when *Oasis* is transforming the first field of a given dataset and if its NIO value is equal to 0

3. for all fields, but the first time, the weights and address values are read from main memory

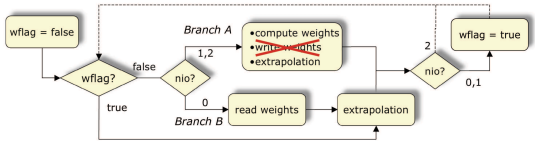


Figure 3:
Flow chart of the *extrap* function.

These assertions reveal that the weights written to file are never read. We thus can optimize the *extrap* function by deleting the writing of the weights. Even though the introduction of this optimization, the performance improvements, shown in table 3, was very limited. This happen because the weights writing is performed only during the first coupling step.

Table 3
extrap performance evaluation

	Elapsed Time (sec)	Saved Time (sec)	%
original	286.218		
optimized	285.032	1.186	0.41

The performance analysis of the *extrap* function highlighted also some numerical problem due to the replication of the same source code on two different branches (see fig. 4). In particular, the extrapolation of the first fields of the dataset is performed during the evaluation of the weights (Branch A); all other fields are extrapolated using a different branch (Branch B). Unfortunately the compiler optimizes the two branches in different way introducing some optimizing transformation for the floating point operations. The experiments showed that if we change the order position of a field in the *namcouple* configuration file and, in particular, if we swap each other the position of the first field with the second one, its values, after the extrapolation, differs of about $1.6 \cdot 10^{-14}\%$. This dis-

placement can absolutely be negligible. However, if we change the order of more then one field on different datasets, the displacement produced a difference, on the final *netcdf* output files at the end of one simulated month, of 0.25%. It is relevant to underline that this discrepancy is only due to a different order with which the fields appear on the *namcouple* file.

The adopted solution consists in the separation of code for the evaluation of the weights from the extrapolation itself. In this case all the fields, including the first one, will be extrapolated using the same piece of code. The final solution is represented in fig. 5.

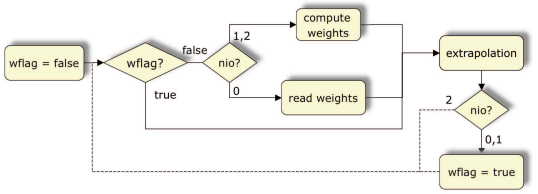


Figure 5:
Optimized *extrap* transformation.

SCRIPR ANALYSIS AND OPTIMIZATION.

The *scriprmp* routine implements the interpolation techniques offered by Los Alamos National Laboratory SCRIP1.4 library. In particular, it performs a remapping of the fields using weights and addresses values that are evaluated taking into account the source grid, the target grid, the type of interpolation to be used and the normalization option. For each field, the *scriprmp* function checks if the file containing the remapping weights exists. If not, they are evaluated first and then written to file for the further coupling steps, as illustrated with the flow chart in fig. 6.

At each coupling step an access to file is performed. The main optimization, introduced

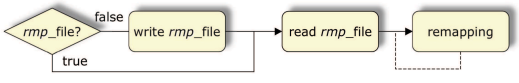


Figure 6:
OASIS3 *scripr* transformation.

within the *scriprmp*, regards the management of the remapping weights into the main memory in order to reduce the time spent for I/O operations. As detailed in table 4, the optimization reduced the elapsed time for the *scriprmp* function of 40%.

Table 4
scripr performance evaluation

	Elapsed Time (sec)	Saved Time (sec)	%
original	617.129		
optimized	367.615	249.514	**40.43**

The overall optimizations of the sequential version of OASIS3, performed on the *scriprmp* and on the *extrap* functions, are reported on table 5. As already described the main contribution on the optimization has been gained in the *scriprmp* function and in general the optimizations were mainly focused on reducing the I/O time. The overall performance improvements are 27% of the whole coupling time.

PER FIELD PARALLELIZATION

In order to further reduce the elapsed time of the coupling transformations, a parallel approach to the algorithm has been developed. The adopted solution consists on the distribution of the fields among the available processes using the MPI library, as shown in fig 7. Each *Oasis* process is then in charge to compute all of the foreseen transformations for all the assigned fields. The design of the parallel algorithm is driven by two main factors:

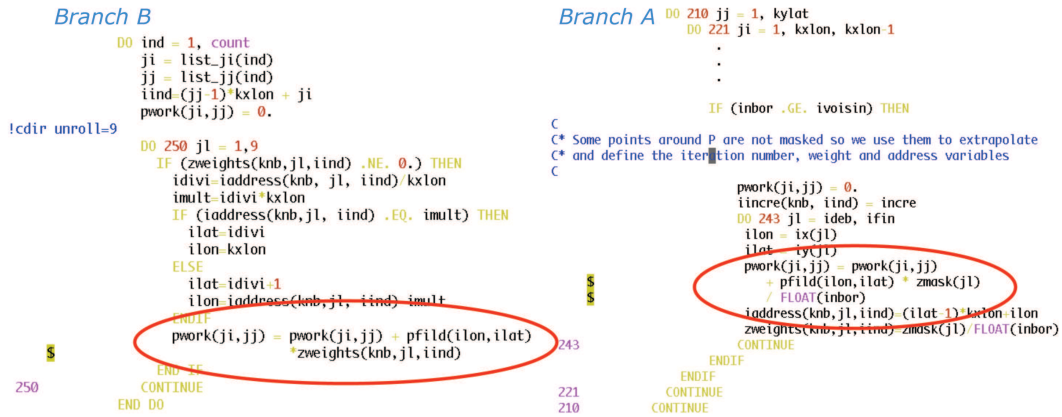


Figure 4:
extrap numerical displacement.

Table 5
OASIS3 optimization

	Extrap	Scripr	Others	Coupling	Saved Time (sec)	%
original	286.218	617.130	1.008	904.360		
optimized	285.032	367.620	1.018	653.670	250.690	**27.72**

1. balance the load among the *Oasis* processes;
2. reduce the communications at minimum.

For the first point, it is necessary to consider that different fields could require a different number and kind of transformations; moreover the fields are also defined on different grids at different resolutions, this implies that the coupling time can not be considered constant for all of the fields.

Even if the best scheduling approach is a dynamic allocation of the fields to the available processes [9], this choice should introduce an overhead of the same order of magnitude of the computing time; for this reason, a static scheduling algorithm has been implemented. The fields are allocated to the processes taking into account the sequencing index (*SEQ*) and the correlation among the fields. This can happen when a field is a linear combination of other fields using the *BLASNEW* and *BLASOLD* transformations. The sequencing index defines an order for fields to be transformed. It has been introduced to allow the overlapping of the coupling time with the computing time of the models. Indeed, fields sent to the coupler from faster models must be tagged, in the *namcouple* file, with smaller values of *SEQ*; in this way the *Oasis* coupling time spent over those fields is overlapped with the computing time of the slower models. This constraint introduces a temporal dependence among processes, because the process transforming a field with a high *SEQ* value should wait for those processes with fields that have a smaller sequencing index. To avoid some processes idle time, the scheduling policy must take into ac-

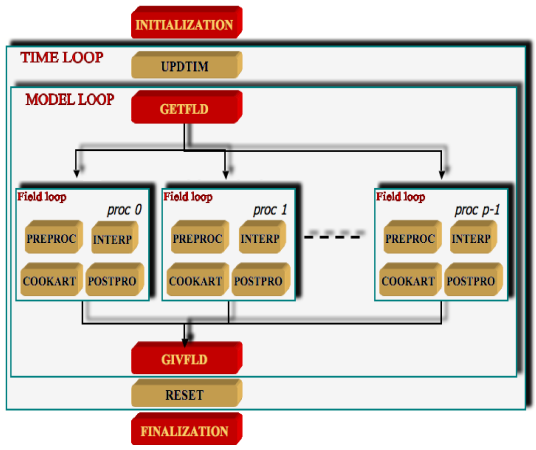


Figure 7:
OASIS3 parallel approach.

count the *SEQ* values of the fields considering that fields with the same *SEQ* must be uniformly distributed over the available processes. In this case, the maximum level of parallelism is given by the maximum number of fields that have the same sequencing index. Since the relationship among fields introduced by the use of *SEQ* is not a functional dependence, a field with a high *SEQ* value does not need of the results of the fields with smaller *SEQ*; this implies that the sequencing order does not introduce communication among processes.

The *BLASNEW* and *BLASOLD* transformations introduce, instead, functional dependence among fields. If a given field A is a linear combination of one or more other fields B, C, the process with A must wait the completion of the transformation on B and C and must communicate with the respective processes. In order to avoid communications, the scheduling algorithm aggregates those fields that depend each others and assign them to the same process.

The parallel algorithm is then structured as follow:

1. at the beginning of the simulation, the scheduling algorithm defines the sets of fields to be assigned to each available process, according to the actual configuration and taking into account the *SEQ*, *BLASNEW* and *BLASOLD* transformations;

2. at each coupling step, the master process of *Oasis* gets the fields from the models and scatters them to the slaves according to the distribution policy established by the scheduling algorithm;

3. each slave process performs the coupling transformation on the assigned fields and sends them to the master

4. the master process exports them to the models.

PARALLEL MODEL

In this section, the definition of an analytic model of the execution time for the parallel algorithm is given. The total elapsed time of the coupled model depends by many factors: the number of processes assigned to the single component model; the overhead introduced by the communication among the processes of a model (intra-model communication overhead); the coupling transformations and so on. Even tough, in this paper we focused only on the aspects that governs the behavior of the coupler. The elapsed time of the CMCC-MED model can be though composed by the following operations:

1. time for initializing the computing environment;

2. time spent by the component models: this includes also the intra-model communications;

3. computational time for coupling transformations: it is important to properly evaluate this time since it could be partially overlapped with the computing time of the component models;

4. communication overhead within the coupler: also this time could be partially overlapped with the computing time of the models;

5. operation for finalizing the simulation;

In order to focus the analysis only on the coupler parallel behavior, we fixed the number of processes assigned to the component models and considered the time for computing the models as intrinsically sequential, constant and independent form the number of processes assigned to the coupler. This choice is also justified considering that the parallelization effort concerned only the coupler and not the whole coupled model. The analysis of the coupler showed also that the operations for the initialization and finalization of the simulation can not be parallelized. The intrinsically sequential time, T_{seq}, can be expressed as:

$$T_{seq} = T_{init} + T_{models} + T_{end} \quad (1)$$

hence the parallel time is given by:

$$T_{par} = T_{seq} + num_{couple} \cdot (T_{couple} + T_{com}) \quad (2)$$

where num_{couple} represents the total number of coupling steps occurring during the simulation; T_{couple} is the elapsed time for the slowest process to transform all of the fields assigned to it; and T_{com} represents the communication overhead that occurs in a coupling step for transferring the fields from the master process of *Oasis* to the slaves and back from the slaves to the master. The aforementioned SEQ values can be used for partially overlapping the coupling time with the computing time spent by the component models. Let define $\mathcal{F}_i$ the set of fields assigned to the process i. This set may contain fields with different SEQ values, it can also be given by the union of disjoined subsets $\mathcal{G}_{i,j}$ that contain the fields with SEQ j assigned to process i. If we define s^* the maximum value of SEQ, we have:

$$\mathcal{F}_i = \bigcup_{j=1}^{s^*} \mathcal{G}_{i,j} \quad (3)$$

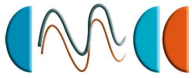
It is worth noting here that the coupling time depends only on the transformations applied on those fields with the maximum value of SEQ and hence the T_{couple} can be expressed as:

$$T_{couple} = max_i \sum_{k \in \mathcal{G}_{i,s^*}} T_{tr_k} \quad (4)$$

where index i represents the process and T_{tr_k} the elapsed time for coupling transformations applied on field k. The communication overhead has been modeled according to the standard linear communication model [4],[8]. At each coupling step, the *Oasis* takes T_{pp} time for point-to-point communication and T_{broad} time for broadcast.

$$\begin{aligned} T_{com} &= T_{pp} + T_{broad} \\ &= T_s(1 + \log 2p) \cdot n^* \\ &\quad + T_b \cdot \textstyle\sum_{j \in \mathcal{G}_{i,s^*}} (L_{im_j} + L_{ex_j} \cdot \log_2 p) \end{aligned} \quad (5)$$

where T_s and T_b are machine dependent parameters and represent respectively the communication latency and the inverse of the effective throughput of the communication channel. n^* is the highest cardinality (over the processes i) of the sets containing fields with the s^* value of SEQ, given by:



$$n^* = max_i\{|\mathcal{G}_{i,s^*}|\} \quad (6)$$

PARALLEL PERFORMANCE ANALYSIS

The model described in the previous section has been validated with several tests on NEC SX9 cluster available at the CMCC Supercomputing Center. Some preliminary tests have been conducted in order to experimentally evaluate the latency and throughput of the communication channel. Since the architecture is composed by 7 nodes with 16 processors for each node, the communication may happen among nodes or within one node. However, we can consider that the communication within the coupler will happen only inter-nodes . Indeed, the best configuration for the CMCC-MED foresees that the master process of *Oasis* must be mapped on the same node with the master process of the slowest component model (in order to minimize the communication among the models and coupler) while the *Oasis* slaves processes must be mapped on different nodes. The features of the SX9 node are reported on table 6

Table 6
NEC-SX9

	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
CPUs per node	16
Peak performance per node	1.6 TF
I/O Data rate	64 GB/s
Internode bandwidth (peak)	128 GB/s x 2
T_s	$3.40 \cdot 10^{-06}$ sec
T_b	$2.30 \cdot 10^{-11}$ sec/Byte

It is relevant to remind that the performance analysis is mainly focused on the evaluation of the coupler parallelization; then fixed the number of processes assigned to the component models has been fixed changing the number of processes assigned to the coupler. The configuration we used is as follow:

- Ocean global: 1 processor on node A
- Mediterranean sea: 6 processors on node A
- Atmosphere: 8 processors on node A
- Coupler: 1 processor on node A and $(p-1)/2$ processors on nodes B and C

With this configuration, the T_{seq} time has been evaluated as reported on table 7. In table 8, the coupling time for each field is illustrated.

Table 7
Sequential time

T_{init}	$2.08 \cdot 10$
T_{models}	$3.67 \cdot 10^3$
T_{end}	$3.73 \cdot 10^{-5}$

In order to have a wider range of analysis, we have imposed $SEQ = 1$ for all of the fields, regardless the speed of the component models; in this way the number of processors ranged from 1 to 35, that is the total number of fields exchanged through the coupler.

The performance model demonstrated that the scalability is heavily limited by the coarse grained parallelization based on the distribution of the fields among the processors and the different kind and number of transformations performed on the fields. The analysis of scalability showed that the algorithm reach a 50% of efficiency with 13 processors, corresponding to a computational load of about 3 fields per process. The developed parallel approach heavily influences the balancing of the load among the processors. The communication overhead

Table 8
Parallel time

# field (k)	T_{tr_k} (sec)	L_{ex_k} (byte)	L_{im_k} (byte)
1	$4.56 \cdot 10^{-2}$	921600	216944
2	$4.15 \cdot 10^{-2}$	921600	216944
3	$4.30 \cdot 10^{-2}$	921600	216944
4	$3.92 \cdot 10^{-2}$	921600	216944
5	$3.93 \cdot 10^{-2}$	921600	216944
6	$4.04 \cdot 10^{-2}$	921600	216944
7	$1.27 \cdot 10^{-1}$	921600	1762904
8	$1.25 \cdot 10^{-1}$	921600	1762904
9	$1.26 \cdot 10^{-1}$	921600	1762904
10	$4.82 \cdot 10^{-2}$	216944	921600
11	$4.63 \cdot 10^{-2}$	216944	921600
12	$4.61 \cdot 10^{-2}$	216944	921600
13	$4.63 \cdot 10^{-2}$	216944	921600
14	$4.84 \cdot 10^{-2}$	216944	921600
15	$4.57 \cdot 10^{-2}$	216944	921600
16	$4.66 \cdot 10^{-2}$	216944	921600
17	$4.62 \cdot 10^{-2}$	216944	921600
18	$2.67 \cdot 10^{-1}$	216944	921600
19	$3.95 \cdot 10^{-2}$	216944	921600
20	$2.64 \cdot 10^{-1}$	216944	921600
21	$3.93 \cdot 10^{-2}$	216944	921600
22	$3.93 \cdot 10^{-2}$	216944	921600
23	$3.93 \cdot 10^{-2}$	216944	921600
24	$4.26 \cdot 10^{-2}$	216944	921600
25	$2.69 \cdot 10^{-2}$	216944	921600
26	$2.68 \cdot 10^{-2}$	216944	921600
27	$8.07 \cdot 10^{-2}$	1762904	921600
28	$7.52 \cdot 10^{-2}$	1762904	921600
29	$8.00 \cdot 10^{-2}$	1762904	921600
30	$7.70 \cdot 10^{-2}$	1762904	921600
31	$5.63 \cdot 10^{-2}$	1762904	921600
32	$5.49 \cdot 10^{-2}$	1762904	921600
33	$5.53 \cdot 10^{-2}$	1762904	921600
34	$4.06 \cdot 10^{-2}$	1762904	921600
35	$4.08 \cdot 10^{-2}$	1762904	921600

takes just almost the 2% of the coupling time and it can not be considered the limitation factor.

Figures 8-10 depict the coupling time (on one month simulation $num_{couple} = 279$), the speed-up and efficiency of the parallel algorithm with a number of processors ranging from 1 to 35. The analytic model of performance approximates the real behavior of the algorithm with a standard deviation of $2,4\%$, hence it can be considered reliable. As confirmed by the swing trend of the speep-up and efficiency functions, the coarse grained parallelization produces worst performance when the number of fields is not perfectly divisible by the number of processes, whereas the different number and kind of transformations deteriorate performance even if the number of fields is divisible by the number of processes (i.e. $p = 5, 7$). The experimental data obtained analyzing the parallel performance are also reported in table 9.

Table 9
Parallel OASIS3 performance evaluation

# of procs	Execution Time (sec)	Efficiency	Speed up
1	645.13	1.00	1.00
2	351.80	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	90.01	0.28	7.16
33	89.59	0.22	7.20

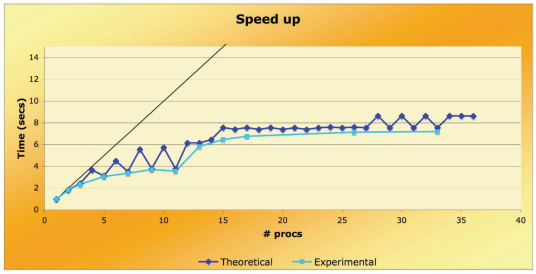


Figure 8:
Parallel OASIS3 speedup.

Table 10
Parallel OASIS3 improvements

	Coupling Time (sec)	Saved Time (sec)	%
original	904		
parallel (13 proc)	110	794	**87.83**

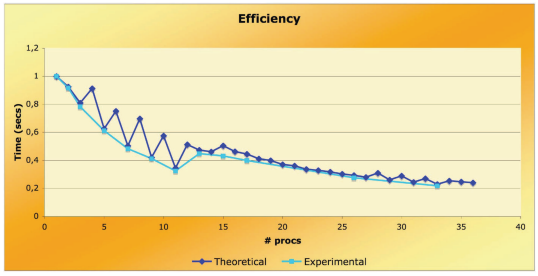


Figure 9:
Parallel OASIS3 efficiency.

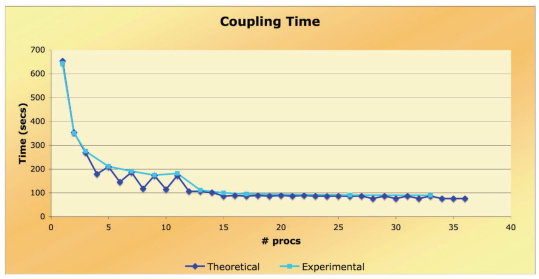


Figure 10:
Parallel OASIS3 execution time.

As already previously underlined, one of the limit of the proposed approach is that the scheduling policy considers the time taken by each fields for coupling transformations constant and independent by the fields. Better performance could be achieved taking into account the different computational load required by applying the transformation on the fields and trying to better balance the load among the processors. Even though, a per-field parallelization keep still limited by the total number of fields. The highest level of parallelism can be achieved by combining the proposed approach with a parallelization based on the spatial domain decomposition.

IMPLEMENTATION DETAILS

The implementation of the parallel algorithm has been completely integrated with the official version of OASIS3 coupler distributed by CERFACS. The code modification has been made in order to have the minimal impact on the structure of the existent code structure. Taking into account that the CLIM libraries, used by the coupler for communicating with the component models, supports both MPI1 and MPI2, the parallel model has been accordingly implemented. More in detail, with MPI1 [11] implementation, a MPMD [3] approach is used; each component models and coupler are executed launching different executables. The "specialization" of each process is then known only to the process; an initialization phase with the exchanging of the colour of models allows each process to know the masters and their slaves of the others models . A communicator for each model, including the coupler, is created using the *MPI_Comm_split* function.

The MPI2 [5] implementation follows a different approach: with the *mpirun* command, only the coupler processes are instantiated. The executable names and the number of processes to be spawn for each component models is also passed through the command line to the *Oasis* executable. In this case the *Oasis* communicator is duplicated from the *MPI_COMM_WORLD* at the beginning and then the other communicators are created during the spawn of the corresponding processes.

The two implementations differ only on the management of the communicators. Once the coupler communicator has been created, the communications are ever executed within it.

Moreover, the parallel implementation has been verified with a bit-to-bit comparison against the output got from the original OASIS3 version, after 3 simulated months with the use of restart files. The current version has been tested only on a subset of the whole available *Oasis* transformations. Namely, with those ones used within the CMCC-MED model:

- Time transformations: *LOCTRANS*, *AVERAGE*
- Pre-processing transformations: *MASK*, *EXTRAP*, *NINENN*, *INVERT*
- Interpolation transformations: *SCRIPR*, *DISTWGT*, *CONSERV*, *BILINEAR*, *BICUBIC*
- Cooking stage: *CONSERV*, *GLOBAL*, *BLASNEW*
- Post-processing transformations: *REVERSE*

COMPARISON WITH THE PSEUDO PARALLEL VERSION

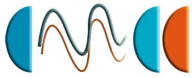
A qualitative comparison between the proposed approach with respect to the pseudo-parallel implementation of OASIS3 implemented by CERFACS has been analyzed. In the pseudo-parallel approach, each *Oasis* process must have its own *namcouple* file carefully created by the modeler. Each process is then independent and unaware of the existence of others and it communicates directly to the models exchanging the fields included into its *namcouple* file. Such an approach implements a distributed communication with the models avoiding the bottleneck represented by a single master process in charge to exchange the fields with the models and to coordinate the slaves. The manual definition of the *namcouple* file allows a more accurate distribution of the fields among the processes taking also into account the computational load required by each field. The main disadvantage of the pseudo-parallel approach regards the configuration, indeed, the user is charged with the burden of creating *namcouple* files each time the number of *Oasis* processes changes. Moreover, the parallel version of OASIS3 provides MPI1 and MPI2 CLIM communication technique, whereas the pseudo- parallel version only supports MPI1.

CONCLUSIONS

In the present work, the optimization and parallelization activity of one of the most deployed coupler, has been presented. Followed methodology stated that, before proceeding with the parallelization of an existent code, it is necessary to well understand why it performs bad on the target architecture; it allows optimizations. The profiling phase is mandatory to identify the hot-spot functions and to guide the optimization. Further level of improvement can be reach by the parallelization, after a deep analysis of the algorithm and the identification of the data and functional dependencies. In the case here discussed, with just the optimization and elimination of useless I/O operation, the coupling time has been reduced of 27%. Even if the strategy for parallelization is coarse grained, it allowed a reduction of the coupling time up to 80% of the original sequential version, using 13 processors. As expected, the coarse grained parallel approach can not guarantee a good load balancing and it limits the level of parallelism. The counterpart aspect is that the communication overhead is kept at minimum.

In order to enhance the parallel performance some improvements can be adopted:

- the scheduling algorithm can be modified in order to self adapt to the computing requirements of field and to take into account the coupling time of each field to better balance the load. If the scheduling algorithm could know the coupling time for each field, it should be able to better distribute the load among the processes. This information can be given to the scheduler after a profiling phase of the coupled model; otherwise, the scheduler could self adapt keeping track of the time taken by each field in a month and using this information for the scheduling policy



in the next month

- the presence of memory bank conflicts (about 40%) during OASIS execution on the vector machine could be resolved due to a further optimization step. The bank conflicts occure when two or more processes try to simultaneously access to the same bank of memory. The code can be suitable modified avoiding bank conflicts to happen

- the *Oasis4* [13] is the new parallel version of the coupler, developed at CERFACS and based on a geographical domain decomposition of fields among processes. Performance evaluation of the new coupler can be performed using the CMCC-MED couple model. The two parallel approach can be integrated within a unique solution

- the CMCC Supercomputing Center also stores an IBM supercomputer with 10 power6 nodes for a total number of 960 cores. The performance evaluation of parallel OASIS3 on the scalar architecture can be performed in order to evaluate the behaviour of the code on a many core compared with a vector system

- the parallel coupler has been validate on a set of available transformations. A complete test on the other transformations is need in order to standardize the code

- climate change studies involve several coupled models. They are obtained using different climate models but also different coupler. The performance comparison of parallel OASIS3 with other couplers such as the NCAR csm Flux coupler [1] represents a further step to evaluate pro and contro.

Bibliography

[1] F.O. Bryan, B.G. Kauman, W.G. Large, and P.R. Gent. The ncar csm fluxcoupler. Technical Report NCAR Technical Note NCAR/TN-424+STR, National Center for AtmosphericResearch, 1996.

[2] NEC Corporation. Super-ux performance tuning guide. Technical report, NEC Corporation, 2006.

[3] I. Foster, J. Geisler, S. Tuecke, and C. Kesselman. Multimethod communication for high-performance metacomputing. *Proceedings of ACM/IEEE Supercomputing, Pittsburgh, PA*, pages 1–10, 1997.

[4] I.T. Foster. *Designing and Building Parallel Programs : Concepts and Tools for Parallel Software Engineering*. Addison-Wesley, 1995.

[5] W. Gropp, S. Huss-Lederman, A. Lumsdaine, E. Lusk, B. Nitzberg, W. Saphir, and M. Snir. *MPI The complete reference: Volume 2, the MPI-2 extensions*. The MIT Press, Cambridge, Massachusetts, London, England, 1998.

[6] G. Madec. Nemo ocean engine. Technical Report 27 ISSN No 1288-1619, Institut Pierre-Simon Laplace (IPSL), 2008.

[7] G. Madec, P. Delecluse, M. Imbard, and C. Levy. Opa 8.1 ocean general circulation model reference manual. Technical report, Institut Pierre-Simon Laplace (IPSL), 1998.

[8] N. Nupairoj and L.M. Ni. Performance evaluation of some mpi implementations on workstation clusters. *Proceedings of the 1994 Scalable Parallel Libraries Conference (SPLC94)*, pages 98–105, 1994.

[9] M. Quinn. *Parallel programming in c with mpi and openmp*. McGraw Hill, 2004.

[10] E. Roeckner, R. Brokopf, M. Esch, M. Giorgetta, S. Hagemann, L. Kornblueh, E. Manzini, U. Schlese, and U. Schulzweida. The atmospheric general circulation model echam5. Technical Report 354, Max Planck Institute (MPI), 2004.

[11] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI: The Complete Reference*. The MIT Press, Cambridge, Massachusetts, London, England, 1996.

[12] S. Valcke. Oasis3 user guide. Technical report, CERFACS, 2006.

[13] S. Valcke and R. Redler. Oasis4 user guide. Technical report, CERFACS, 2007.