Experience on the parallelization of the OASIS3 coupler

Italo Epicoco[1] **Silvia Mocavero** [2] **Giovanni Aloisio**[1,2]

[1] Department of Engineering for Innovation
University of Salento,
via per Monteroni, 73100 Lecce, Italy,
Email: {italo.epicoco,giovanni.aloisio}@unisalento.it

[2] Euro-Mediterranean Centre for Climate Change
Via Augusto Imperatore 16, 73100 Lecce, Italy
Email: silvia.mocavero@cmcc.it

Abstract

This work describes the optimization and parallelization of the OASIS3 coupler. Performance evaluation and profiling have been carried out by means of the CMCC-MED coupled model, developed at the Euro-Mediterranean Centre for Climate Change (CMCC) and currently running on a NEC SX9 cluster. Our experiments highlighted that extrapolation (accomplished by the *extrap* function) and interpolation (implemented from the *scriprmp* function) transformations take the most time. Optimization concerned I/O operations reducing coupling time by 27%. Parallelization of OASIS3 represents a further step towards overall improvement of the whole coupled model. Our proposed parallel approach distributes fields over a pool of available processes. Each process applies coupling transformations to its assigned fields. This approach restricts parallelization level to the number of coupling fields. However, it can be fully combined with a parallelization approach considering the geographical domain distribution. Finally a quantitative comparison of the parallel coupler with the OASIS3 pseudo-parallel version is proposed.

Keywords: OASIS3, climate models, coupled models, performance analysis, parallel modeling

1 Introduction

Climate change models describe complex subsystems such as oceans dynamics; atmospheric, chemical and physical processes; vegetation and land use transformations. Historically, these models have always been stand alone applications. They are not complete enough to describe the complexity of the whole climate system, unless we consider the chance to infer new knowledge from their coupling. A more detailed approach is to model the climate behavior by coupling models each others. In this context a coupler component is a key performance factor of the overall coupled model. The coupler acts as a "collector" amid component models. Its main function is to interpolate, extrapolate, re-grid and, more in general, transform exchanged fields. As for modeling, the coupler should support different parallel approaches in order to be compliant with and portable on heterogeneous parallel architectures. To this aim, the OASIS3 coupler is both OpenMP and MPI parallelized; it is possible to select a hybrid approach or just one of the available parallelization methods at compile time. Given the nature of the operations performed by the coupler, its execution time cannot overlap with component models one. Thus optimization and parallelization of the coupler have strong impact on the wall clock time of the overall model. Both work and results described in this paper refer to an optimized and parallelized version of OASIS3 adopted at the Euro-Mediterranean Centre for Climate Change (CMCC). In particular, the CMCC-MED (S. Gualdi, E. Scoccimarro et al.) coupled model has been taken into account to test the parallel coupler.
The paper is organized as follows: section 2 describes the OASIS3 coupler; its performance profiling on the target machine (NEC-SX9) and performed optimization are detailed in section 3. We then detail our parallel approach in section 4, performance model and its analysis in section 5. Finally, we give a qualitative comparison of our proposed approach with the OASIS3 pseudo-parallel version in section 6, an evaluation of alternative scheduling solutions in section 7, and draw our conclusions.

Copyright ©2010, Australian Computer Society, Inc. This paper appeared at 8th Australasian Symposium on Parallel and Distributed Computing (AusPDC 2010), Brisbane, Australia. Conferences in Research and Practice in Information Technology, Vol. 107. Editors Jinjun Chen (Swinburne University of Technology) and Rajiv Ranjan (University of New South Wales), Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

2 The OASIS3 coupler

The coupler OASIS3 (Valcke 2006) consists of a set of Fortran 77, Fortran 90 and C routines. At runtime, OASIS3 acts both as a separate single process executable, whose main aim is to interpolate coupling fields exchanged among the component models, and as a library (OASIS3 PSMILe) linked by the component models in order to communicate with the coupler.
OASIS3 provides several transformations and 2D interpolations in order to adapt coupling fields from a source model grid to a target one. For each exchanged field, the user can define a set of required transformations and their order through the *namcouple* configuration file. Available transformations are grouped into five general classes and must be strictly applied following this logical order: time, pre-processing, interpolation, "cooking", and post-processing. This order is also supported by the OASIS3 software internal structure. It is worth noting here that transformations are usually independently performed on each field, thus transformations on a single field can be considered as a separate task. *BLASOLD* and *BLASNEW* transformations represent an exception to this rule, since they introduce functional dependence among fields. They perform, respectively before and after the interpolation phase, a linear combination of the current field with others or with itself.
The following steps characterized OASIS3 coupling activity:

- An initialization step, executed once for each

run. It includes some preliminary operations such as initialization of internal parameters, definition of logical I/O units descriptors, allocation of data structure for timing purpouse, definition of the communication environment among processes and grids, instantiation of variables and opening of required files.

- The execution of a loop over the time steps. For each time step, another loop over the sequencing index (*SEQ*) is performed. *SEQ* defines the order for fields to be transformed. Its aim is to allow overlapping of the coupling time with models computing time. A tag with small values of *SEQ* in the *namcouple* file must be associated to fields sent to the coupler from faster models; that allows overlap of OASIS3 coupling time spent over those fields with computing time of slower models. The loop over sequencing index includes: (i) evaluation of the number of fields to be exchanged among coupled models; (ii) retrieval of fields values from models; (iii) fields transformation through pre-processing, interpolation, cooking and post-processing operations; (iv) delivery of those fields to the target models.
- A finalization step, including arrays deallocation and file closing.

3 OASIS3 profiling

The OASIS3 coupler has been evaluated and profiled within the CMCC-MED model, developed at the CMCC. It is a 3-components coupled model consisting of the Echam5 (Roeckner et al. 2004) T159L31 atmospheric model, the OPA 8.2 (Madec et al. 1998) oceanic global model with a 2° resolution and the Nemo (Madec 1998) Mediterranean sea model with a 1/16° resolution. The atmospheric model provides the coupler with 26 fields defined on a 480x240 spatial grid; 17 are addressed to the ocean global model and the 9 remaining are addressed to the Mediterranean sea model. The ocean global model provides the coupler with 6 fields, defined on a 182x149 spatial grid, to be sent to the atmospheric model. Finally, the Mediterranean sea model provides the coupler with 3 fields, defined on an 871x253 spatial grid, to be addressed to the atmospheric model too. The total amount of managed fields, exchanged among the component models, is 35, with a coupling period of 2h 40′ and thus 279 coupling steps in a month. Table 1 lists performed transformation. Extrapolation (over 29 fields) and interpolation (over 35 fields) are the most frequent ones.

The coupled model has been profiled on a NEC SX9 cluster using FTRACE analysis tool in order to identify time-consuming functions. Optimization started from these functions. In particular, a FTRACE region has been defined in the OASIS3 code. The FTRACE output, shown in figure 1, highlights that *clim_import* takes about 1900 seconds followed by *scriprmp* and *extrap*. It is worth noting here that *clim_import* belongs to the CLIM library adopted for the communication among the coupler and the component models; it is devoted to receive fields exchanged among models. The elapsed time spent executing this function is actually an idle time, since the coupler has to wait for the component models to simulate the coupling period. We can thus safely ignore this function since it does not include coupling time.

As table 2 shows, *extrap* and *scriprmp* are the most time consuming transformations; they take about 96% of the total coupling time.

In the following sections we delve into details of the optimization performed on these functions.

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

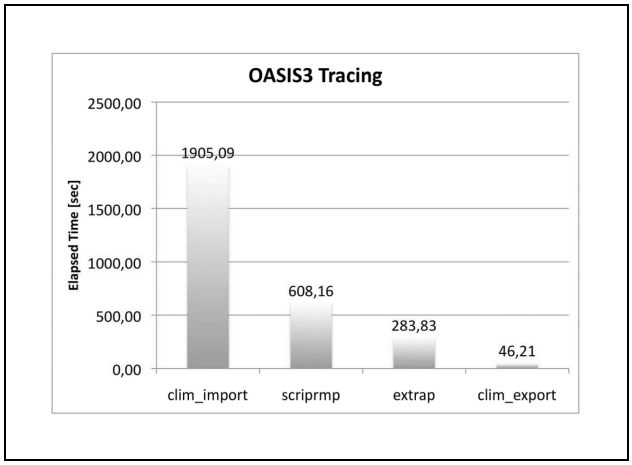

Figure 1: OASIS3 performance tracing.

3.1 Extrap analysis and optimization

The *extrap* function performs fields extrapolation over their masked points using the source grids. Since the adopted weights depend only on the source grid, it is reasonable to group fields into different datasets, each of these characterized by the same source grid and hence by the same weights. A field is also tagged with a NIO parameter, whose value is 1 if weights must be computed and written to file or 0 in case of reading from file. It is worth noting here that the NIO parameter is taken into account only for the first field of a given dataset and only during the first coupling step, it is ignored otherwise; in these cases weights are always read from memory. The flow chart in figure 2 gives an overview of the original OASIS3 algorithm. *wflag* is a boolean variable used to establish if

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	

weights and addresses values for dataset i are available in main memory or not. At the beginning, *wflag* is initialized to *FALSE* for all of the datasets; when the coupler processes the first field of the dataset i, the instruction control flow depends on NIO value. Following Branch A, both definition of weights and extrapolation of field, are jointly performed; weights are then stored in a file. Branch B is followed when NIO is 0 and extrapolation is performed by means of the stored weights. In both cases, weights are stored in main memory and *wflag* is asserted. That implies extrapolation to be performed reading weights from the main memory for every field of the same dataset and for each of further coupling steps. Considering the flow chart of figure 2, it is clear that:

1. weights and addresses values are written in a file only when OASIS3 transforms the first field of a given dataset, during the first coupling step, and its NIO value is equal to 1;
2. weights and addresses values are read from file only when OASIS3 transforms the first field of a given dataset, during the first coupling step and its NIO value is equal to 0;
3. weights and addresses values are read from main memory otherwise.

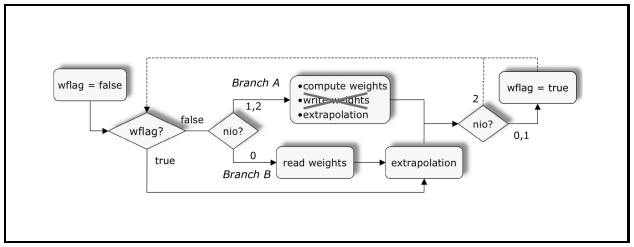

Figure 2: Flow chart of the *extrap* function.

These assertions reveal that even if weights are written, they are never read. We thus can optimize the *extrap* function skipping the writing procedure. Despite the introduction of this optimization, performance improvement, as shown in table 3, is very poor. This happens because weights writing is performed only during the first coupling step.

Performance analysis of the *extrap* function highlights also some numerical issues owing to the replication of the source code on two different branches (see figure 4). In particular, extrapolation of the first field of each dataset is performed during weights evaluation (Branch A); the others are extrapolated in a different branch (Branch B). Unfortunately, the compiler optimizes the two branches in different way introducing some optimizing transformation of floating point operations. Experiments show that if we change the order of a field in the *namcouple* configuration file, its value, after the extrapolation, is different. In particular, if we swap the position of two fields, the difference on the first field is about of $1.6 \cdot 10^{-14}$%. This displacement can absolutely be negligible. However, if we change the order of more then one field belonging to different datasets, this displacement produces a 0.25% difference on the netcdf output files generated by one simulated month. This discrepancy is relevant since it is due only to a different order of the fields in the *namcouple* file. To solve this problem, the code performing weights evaluation and extrapolation has been split. In this way, all the fields, including the first one of each dataset, is extrapolated using the same piece of code. The final solution is represented in figure 3.

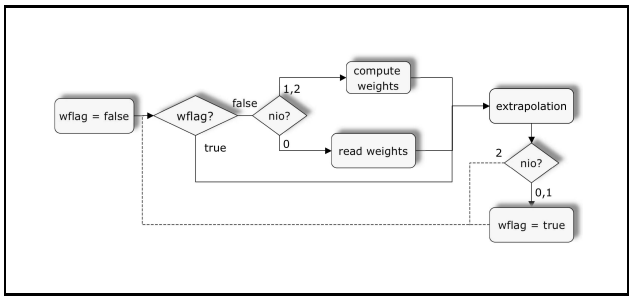

Figure 3: Optimized *extrap* transformation.

3.2 Scripr analysis and optimization.

The *scriprmp* routine implements the interpolation techniques offered by the Los Alamos National Laboratory SCRIP1.4 library. In particular, it performs a remapping of the fields using weights and addresses 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 whether the file containing remapping weights exists. If not, they are first evaluated and then written to a file for the further coupling steps.
At each coupling step, an access to the file is performed. The main optimization concerns the management of remapping weights into the main memory, in order to reduce the time spent for I/O operations. As detailed in table 4, this optimization reduces elapsed time for the *scriprmp* function of 40%. The overall optimizations of the sequential version of OASIS3, performed on both the *scriprmp* and the *extrap* functions, is shown in table 5. As previously described, the optimizations were mainly focused on reducing the I/O time. The main contribution to the optimization has been gained in the *scriprmp* function. The overall performance improvements is 27% of the whole coupling time.

4 Per field parallelization

In order to further reduce the elapsed time of coupling transformations, a parallel version of the algorithm has been developed. Adopted parallel approach is the master/slaves model. Since computation of each field is independent from the others, slaves do not communicate with each other. The master distributes

Table 3: *extrap* performance evaluation

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

Table 4: *scripr* performance evaluation

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

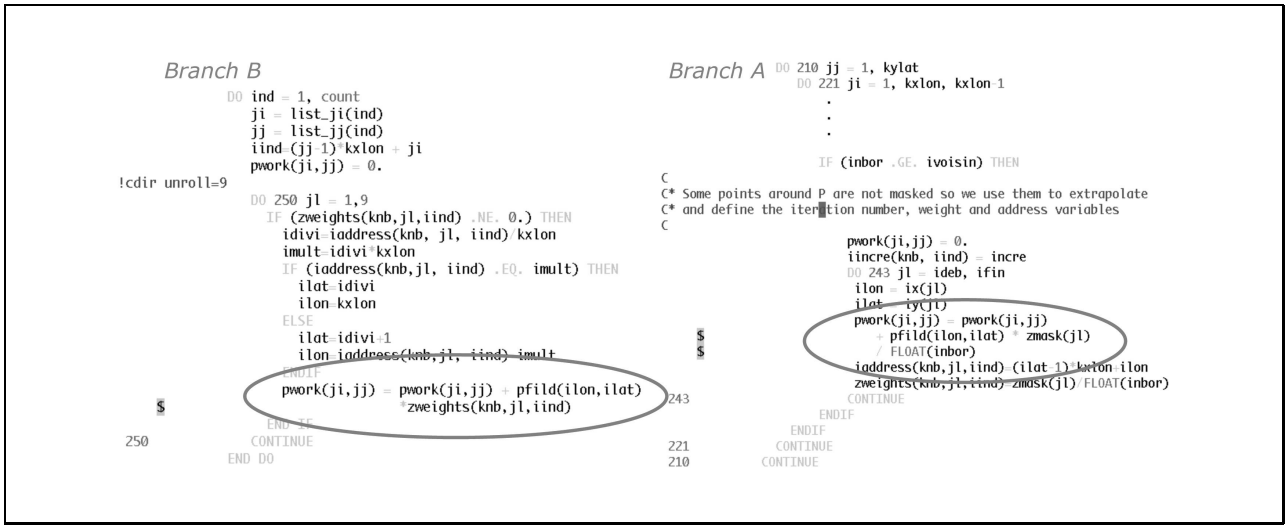

Figure 4: *extrap* numerical displacement.

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**

fields to slaves and collects them after the execution of coupling operations, using the MPI library, as shown in figure 5. Each OASIS3 process is then in charge of computing all of the foreseen transformations for each assigned fields. The design of the parallel algorithm is driven by two main factors:

1. load balancing among OASIS3 processes;
2. maximum communication reduction.

It is necessary to consider that different fields could require different number and type of transformations; moreover they are also defined on different grids at different resolutions. This implies that coupling time cannot be considered constant for every field. Since computing time of each parallel task is not uniform and not known at compile time, a dynamic scheduling approach should be preferable (Quinn 2004). However, in this case, this choice should introduce an overhead of the same order of magnitude of computing time. For this reason, a static scheduling algorithm to distribute fields to available processes, has been implemented. Fields are allocated to processes taking into account the sequencing index (*SEQ*) and the presence of a correlation among fields. That happens when a field is a linear combination of other fields, using the *BLASNEW* and/or *BLASOLD* transformations. The sequencing index defines an order for fields to be transformed. It has been introduced to allow overlapping coupling time with models computing time. Indeed, fields sent to the coupler from faster models must be tagged, in the *namcouple* file, with smaller values of *SEQ*. This way, the OASIS3 coupling time spent over these fields is overlapped with computing time of slower models. This constraint introduces a temporal dependence among processes: the process performing transformation of a field with a high *SEQ* value should wait for those processes responsible for fields with a smaller sequencing index. To avoid some processes idle time, the scheduling policy must take into account the *SEQ* value of each field: fields with the same *SEQ* must be uniformly distributed to the available processes. In this case, the maximum number of fields with the same sequencing index gives the maximum level of parallelism. 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 to know results form those with smaller *SEQ* value; this implies that the sequencing order does not introduce communication among processes.

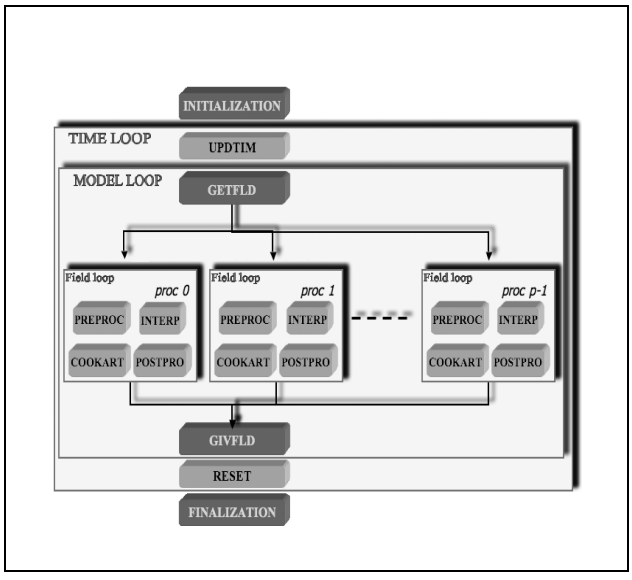

Figure 5: OASIS3 parallel approach.

If a given field A is a linear combination of one or more fields B, C (*BLASNEW* and *BLASOLD* transformations), the process performing A must wait for the transformations of B and C to be completed and

must communicate with the respective processes. In order to avoid communications, the scheduling algorithm aggregates fields with functional dependence and assigns them to the same process. Since coupling time taken by each field is not known at scheduling time, the algorithm aims at balancing the number of fields assigned to each process. The scheduling algorithm is structured as follow:

1. fields are aggregated in different groups G_{SEQ} according to the *SEQ* value;
2. within each G_{SEQ}, fields are further grouped in bundles $B_{SEQ,i}$ according to functional dependencies established by *BLASNEW* and *BLASOLD* transformations. Each bundle is ranked with an integer given by the cardinality of the set: $r_i = |B_{SEQ,i}|$;
3. for each G_{SEQ}, the bundles $B_{SEQ,i}$ are sorted by rank in descending order;
4. each MPI process is labeled with an integer l_j representing the number of fields currently assigned to process j. For each G_{SEQ}:
   (a) l_j is initialized to zero;
   (b) bundle $B_{SEQ,i}$, with maximum rank and not assigned yet, is associated with the process j having the minimum l_j;
   (c) the rank of bundle r_i is added to l_j;
   (d) the algorithm iterates from step b for each bundle belonging to current G_{SEQ}.

Unfortunately such an approach cannot guarantee a good load balancing for each configuration, since it assumes that each field takes the same coupling time. Moreover, the balancing is also influenced by the order of fields in the *namcouple* configuration file. More accurate algorithm should take into account the different coupling time requested by each field balancing the load according to it.
The resulting 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 *SEQ* values, *BLASNEW* and *BLASOLD* transformations;
2. at each coupling step, the master process of OASIS3 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 coupling transformations on the assigned fields and sends them to the master;
4. master process exports them to the models.

4.1 Parallel model

In this section, we define the analytic model of the execution time of our parallel algorithm. The coupled model elapsed time depends on many factors: number of processes assigned to a single component model; overhead introduced by communications among processes of a model (intra-model communication overhead); coupling transformations and so on. In this paper we focus only on the aspects affecting the coupler behavior. Elapsed time of the CMCC-MED model can be devised as the sum of the following components:

1. initialization of the computing environment;
2. time spent by component models, also including intra-model communications;
3. computing time to perform coupling transformations: it is important to properly evaluate this time, since it could partially overlap with computing time of component models;
4. communication overhead within the coupler: also this time could be partially overlapped with models computing time;
5. finalization of the simulation.

Since we are interested in the coupler parallel behavior, we establish the number of processes assigned to component models and considered time for executing models as intrinsically sequential, constant and independent from the number of processes assigned to the coupler. This choice is also supported by the consideration that the parallelization effort concerned only the coupler and not the whole coupled model. Moreover, a careful analysis leads us to infer that initialization and finalization operations cannot be parallelized. The intrinsically sequential time, T_{seq}, can be expressed as:

$$T_{seq} = T_{init} + T_{models} + T_{end} \tag{1}$$

hence the parallel time is given by:

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

where num_{couple} represents the total number of coupling steps occurring during the simulation; T_{couple} is the elapsed time required by the slowest process to transform its assigned fields; and T_{com} represents the communication overhead occurring in a coupling step to transfer fields from the OASIS3 master process to slaves and back.
The aforementioned *SEQ* values can be used to partially overlap coupling time with computing time spent by the component models. Let us define $\mathcal{F}_i$ as the set of fields assigned to the process i. This set may contain fields with different *SEQ* values; it can also be thought as the union of disjoined subsets $\mathcal{G}_{i,j}$ containing fields assigned to process i with *SEQ* j . Defining s^* as the maximum *SEQ* value, we have:

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

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

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

where index i represents the process and T_{tr_k} is the elapsed time for coupling transformations applied on field k. Communication overhead has been modeled according to the standard linear communication model (Foster 1995),(Nupairoj et al. 1994). At each coupling step, OASIS3 takes T_{pp} time for point-to-point communications and T_{broad} time for broadcast communications. Thus,

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

where T_s and T_b are machine dependent parameters and represent respectively the communication latency and inverse of the effective throughput of the communication channel. L_{im_j} and L_{ex_j} are the lengths (in bytes) of field j, used respectively during import and export operations; p is the number of involved processes. n^* is the highest cardinality (over the processes i) of set containing fields with value of SEQ equal to s^*, given by:

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

5 Parallel performance analysis

The model previously described has been validated performing several tests on the NEC SX9 cluster available at the CMCC Supercomputing Center. Some preliminary tests have been executed in order to experimentally evaluate the latency and throughput of the communication channel. Since the architecture consists of 7 nodes and 16 processors for each node, both intra-node and inter-node communications may take place. However, we can safely generalize considering only inter-node communication. Indeed, the best configuration for the CMCC-MED would map the OASIS3 master process on the same node of the slowest component model master process (in order to minimize the time for communication among models and coupler). OASIS3 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	$3.40 \cdot 10^{-06}$
T_b	$2.30 \cdot 10^{-11}$

Table 7: Sequential time

Init Time	$2.08 \cdot 10^{+01}$
Models Time	$3.67 \cdot 10^{+03}$
End Time	$3.73 \cdot 10^{-05}$

It is worth reminding that performance analysis is mainly focused on the evaluation of the coupler parallelization; then the number of processes assigned to the component models has been pre-defined, 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.

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

With this configuration, T_{seq} time components have been evaluated, as shown in table 7. Table 8 lists coupling time of each field.

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

The performance model demonstrated that scalability is heavily limited by the coarse grained parallelization based on both the distribution of the fields among the processors and the different kind and number of transformations performed on the fields. The scalability analysis shows that the algorithm reaches a 50% efficiency with 13 processors, corresponding to a computational load of about 3 fields per process. The developed parallel approach heavily influences the load balancing among processors. The communication overhead takes just almost 2% of the coupling time and it cannot be considered the limitation factor.

Figures 6-8 depict coupling time (on one simulated month $num_{couple} = 279$), speed-up and efficiency of the parallel algorithm with a number of processors ranging from 1 to 35. The analytic performance model approximates the real behavior of the algorithm with a standard deviation of 2.4%, hence

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

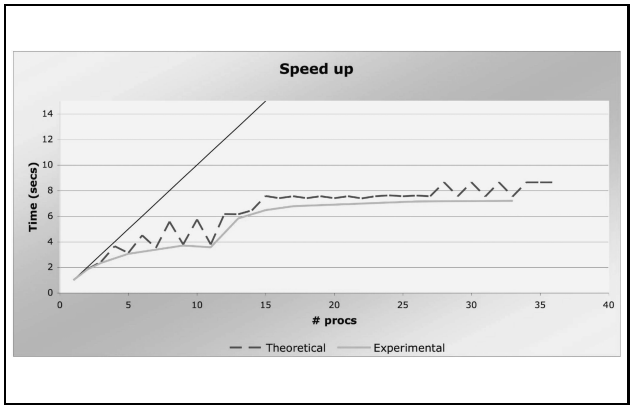

Figure 6: Parallel OASIS3 speedup.

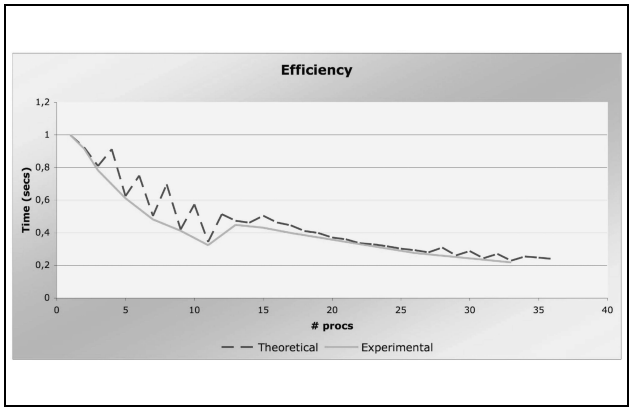

Figure 7: Parallel OASIS3 efficiency.

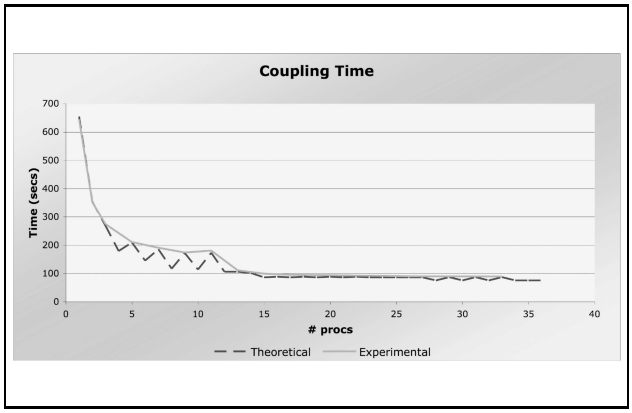

Figure 8: Parallel OASIS3 execution time.

it can be considered reliable. As confirmed by the swing trend of the speed-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 different number and kind of transformations deteriorate performance even if the number of fields is divisible by the number of processes (i.e. $p = 5, 7$). Experimental data obtained analyzing parallel performance is also reported in table 9. As we previously highlighted, a limit of our proposed approach is that the scheduling policy considers the time taken for coupling transformations constant for each field. Better performance could be achieved taking into account the different computational load required by applying transformations on different fields and trying to better balance the load among processors. But a per-field parallelization is 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 a spatial domain decomposition. The timing model can be used for further considerations concerning the suitability of a distributed approach for this problem. In our case, two main reasons restrict the adoption of a distributed approach: (i) the parallelism level of the proposed algorithm is strongly limited by the number of fields to be transformed (it is rare that the number of exchanged fields is greater than 100); (ii) communication overhead in a distributed environment has a stronger impact on parallel performance. Several distributed approaches and frameworks exploit the architectures heterogeneity to improve the parallelization level. In this context different frameworks exist: MapReduce, GRIDSs, Condor are characterized by efficient mechanisms for managing resources, enhancing the fault tolerance and handling node heterogeneity. Generally, these frameworks use I/O operations for communication and hence they are not suitable for coupler parallelization since the communication overhead would exceed the computing time.

5.1 Implementation details

The implementation of the parallel algorithm has been fully integrated in the official version of the OASIS3 coupler, distributed by CERFACS. Code modification has been made minimizing the impact on the structure of the original code. Taking into account that the CLIM libraries, used by the coupler to communicate with the component models, supports both MPI1 and MPI2, the parallel model has been accordingly implemented. More in detail, with MPI1 (Gropp et al. 1996) implementation, a MPMD (Foster et al. 1997) approach is adopted; component models and coupler are executed launching different executables. Only the process itself then knows its "specialization"; an initialization step where the colour of models is exchanged allows each process to know masters and slaves of each model. A communicator for each model, including the coupler, is created using the *MPI_Comm_split* function.

The MPI2 (Gropp et al. 1998) implementation follows a different approach: with the *mpirun* command, only the coupler processes are instantiated. The executables names and the number of processes to be spawn for each component model are also passed through the command line to the OASIS3 executable. In this case, the OASIS3 communicator is duplicated from the *MPI_COMM_WORLD* at the beginning; other communicators are then 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, communications are executed within it.

6 Comparison with the pseudo parallel version

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

7 Evaluation of different scheduling policies

Our proposed approach for scheduling and for mapping the fields to the processors, suffers mainly because coupling time, for each field, is not known at compile time. Thus, the algorithm assigns each field the same weight. More efficient algorithms can be taken into consideration in order to reduce the parallel time. A dynamic scheduling algorithm would distribute fields to processes according to a request/response approach. At the beginning, one field for each process is assigned. The generic process i requests a new field to be transformed as soon as it ends the transformation of the current field. This approach generally behaves better with respect to the round-robin algorithm, but it is still influenced by the order of the fields in the *namcouple* configuration file. The best case for this dynamic approach is when fields are ordered from the most time consuming to the less one. In this case, the dynamic allocation of fields behaves exactly as the MaxMin approach (Maheswaran et al. 1999). The worst case happens when fields are sorted in descending order. Figure 9 depicts performance obtained with different scheduling approaches.

The MaxMin algorithm is a static approach, but it assumes that coupling time for each field is already known. Fields are sorted in descending order with respect to the coupling time and each field is assigned to the process with the current minimum computing load. This approach is the best one, but it requires a profiling phase in order to establish the coupling time for each field.

8 Conclusions

In this work, we presented optimization and parallelization of one of the most deployed coupler. Before dealing with parallelization of a code, it is necessary to deeply understand why it badly performs on the target architecture; that involves optimizations. The profiling phase is mandatory to identify hot-spot functions and to drive optimization. Further level of improvement can be reached with parallelization, after a deep analysis of the algorithm and identification of both data and functional dependencies. In the case here discussed, with just the optimization and elimination of useless I/O operations, the coupling time has been reduced of 27%. Even if the parallelization strategy is coarse grained, it allowed a coupling time reduction up to 80% of the original sequential version, with 13 processors (see table 10).

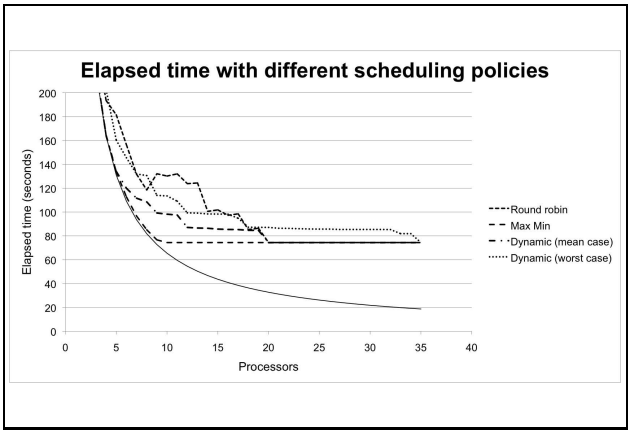

Figure 9: Elapsed time of the OASIS3 using different scheduling policies.

Table 10: Parallel OASIS3 improvements

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

As we expected, the coarse grained parallel approach cannot guarantee a good load balancing and it limits the level of parallelism. The counterpart is that communication overhead is 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 computing requirements and to take into account coupling time of each field, allowing a better load balance. If a scheduling algorithm could know coupling time for each field, it should be able to better distribute load among processes. The scheduler can obtain this information by means of a profiling phase of the coupled model; otherwise, the scheduler could self adapt, keeping track of the time taken by each field to simulate a month and using this information for the scheduling policy of the next month;
- memory bank conflicts (about 40%) (NEC 2006) during OASIS3 execution on the vector machine could be resolved by means of a further optimization step. Bank conflicts occur when two or more processes try to simultaneously access to the same memory bank. The code can be suitably modified avoiding bank conflicts;
- OASIS4 (Valcke et al. 2007) is the new parallel version of the coupler, developed by CERFACS and based on a geographical domain decomposition of fields among processes. Performance evaluation of this new coupler can be performed using the CMCC-MED couple model. These two

parallel approaches can be integrated in a unique solution;

- the CMCC Supercomputing Center has also 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 behavior of the code on a many core system compared with a vector one;
- the parallel coupler has been validated on a set of available transformations. A complete test of available transformations is needed;
- climate change studies involve several coupled models. They are obtained using different climate models, but also different couplers. Performance comparison of parallel OASIS3 with other couplers such as the NCAR CPL coupler (Bryan et al. 1996) represents a further step to evaluate pros and cons of our approach.

References

Euro-Mediterranean Centre for Climate Change http://www.cmcc.it

Bryan F.O., Kauman B.G., Large W.G., Gent P.R. (1996), The NCAR CSM FluxCoupler, NCAR Technical Note NCAR/TN-424+STR, National Center for AtmosphericResearch.

NEC Corporation (2006), SUPER-UX performance tuning guide.

Foster I., Geisler J., Tuecke S., Kesselman C. (1997), 'Multimethod Communication for High-Performance Metacomputing', *Proceedings of ACM/IEEE Supercomputing* , 1–10.

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

Snir M., Huss-Lederman S., Lumsdaine A., Lusk E., Nitzberg B., Saphir W., Snir M. (1998), *MPI The complete reference: Volume 2, the MPI-2 extensions*, The MIT Press.

Madec G. (2008), NEMO ocean engine, Institut Pierre-Simon Laplace (IPSL).

Madec G., Delecluse P., Imbard M. & Levy C. (1998), OPA 8.1 Ocean General Circulation Model Reference Manual, Institut Pierre-Simon Laplace (IPSL).

Maheswaran M., Ali S., Siegel H. J., Hensgen D., Freud R. (1999), 'Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneus Computing Systems', *8th Heterogeneus Computing Workshop (HCW99)*.

Nupairoj N., Ni L.M. (1994), 'Performance Evaluation of Some MPI Implementations on Workstation Clusters', *Proceedings of the 1994 Scalable Parallel Libraries Conference (SPLC94)* , 98–105.

Quinn M.(2004), *Parallel programming in c with mpi and openmp*, McGraw Hill.

Roeckner E., Brokopf R., Esch M., Giorgetta M., Hagemann S., Kornblueh L., Manzini E., Schlese U. & Schulzweida U. (2004), The atmospheric general circulation model ECHAM5, Max Planck Institute (MPI).

Snir M., Otto S., Huss-Lederman S., Walker D., Dongarra J. (1996), *MPI: The Complete Reference*, The MIT Press.

Valcke S. (2006), OASIS3 User Guide, CERFACS.

Valcke S., Redler R. (2007), OASIS4 User Guide, CERFACS.