

**Research Papers**  
**Issue RP0099**  
January 2011

*Scientific Computing and  
Operation (SCO)*

# NEMO-MED: EXTRA-HALO PERFORMANCE MODEL

**By Italo Epicoco**

University of Salento, Italy  
[italo.epicoco@unisalento.it](mailto:italo.epicoco@unisalento.it)

**Silvia Mocavero**

CMCC  
[silvia.mocavero@cmcc.it](mailto:silvia.mocavero@cmcc.it)

and **Giovanni Aloisio**

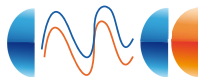
CMCC  
University of Salento, Italy  
[giovanni.aloisio@unisalento.it](mailto:giovanni.aloisio@unisalento.it)

**SUMMARY** The NEMO oceanic model, characterized by a resolution of  $1/16^\circ$  and tailored on the Mediterranean Basin used at CMCC, has been analyzed to discover possible bottlenecks to the parallel scalability. A detailed analysis of scalability on all of the routines called during a NEMO time step allowed to identify the SOR solver routine as the most expensive from the communication point of view. The function implements the red-black successive-over-relaxation method, an iterative search algorithm used for solving the elliptical equation for the barotropic stream function. The algorithm iterates until reach the convergence; a limit on the maximum number of iteration is also set up. The high frequency of data exchanging within this routine implies a high communication overhead. The NEMO code includes an enhanced version of the routine, that reduce the frequency of communication by adding an extra-halo region. The use of this optimization requires the selection of the optimal value of the extra-halo dimension to trade-off computation and communication. A performance model, allowing the choice of the optimal extra-halo value for a pre-defined decomposition, has been designed. The model has been tested on the MareNostrum cluster at the Barcelona Supercomputing Centre.

**Keywords:** NEMO, Extra-halo, Performance Model, Optimization

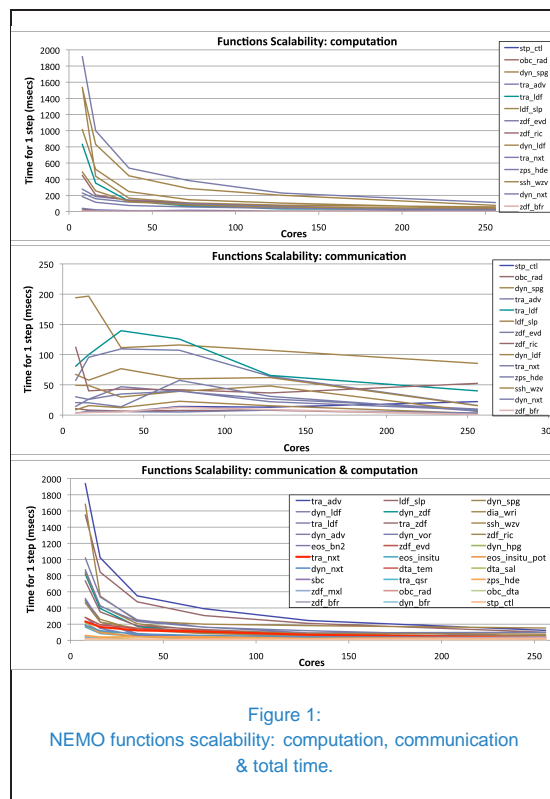
**JEL:** C63

*This work was carried out under the HPC-EUROPA2 project (project number: 228398) with the support of the European Commission Capacities Area - Research Infrastructures Initiative. The authors thankfully acknowledge the computer resources, technical expertise and assistance provided by the Barcelona Supercomputing Center, namely prof. Jose Maria Baldasano, prof. Jesus Labarta and their stuff members. The research leading to these results has received funding from the Italian Ministry of Education, University and Research and the Italian Ministry of Environment, Land and Sea under the GEMINA project.*



## INTRODUCTION

The NEMO model [5] has been evaluated on the MareNostrum platform. The profiling activity aimed at identifying those routines with poor parallel efficiency thus suitable to be optimized for improving the NEMO scalability. To this aim all of the functions at the first level, called by the step routine at each iteration, have been instrumented using the *paraver* tool [4], ignoring both the start-up and ending phases. Figure 1 shows computing, communication and total time for each of these functions, referring to a single time step. From the communication point of view, the *dyn\_spg* function is the most time consuming one. It spends almost all of its time within another function (the *sol\_sor* function).



This last function implements the red-black

Successive-Over-Relaxation (SOR) method [6], an iterative search method used for solving the elliptical equation for the barotropic stream function. The algorithm iterates until reaches the convergence. The *gcx* array represents the final solution and it is computed as shown in figure 2. *gcb* is the second member of the barotropic system and *gcp* the extra-diagonal elements of the barotropic matrix.

```

DO jj = .....
DO ji = .....
  ztmp = ..... gcb(ji, jj) &
  & - gcp(ji, jj, 1) * gcx(ji, jj-1) &
  & - gcp(ji, jj, 2) * gcx(ji-1, jj) &
  & - gcp(ji, jj, 3) * gcx(ji+1, jj) &
  & - gcp(ji, jj, 4) * gcx(ji, jj+1)

  ! Estimate of the residual
  zres = ztmp - gcx(ji, jj)

  gcx(ji, jj) = rn_sor * ztmp + (1-rn_sor) * gcx(ji, jj)
END DO
END DO

.....

! test of convergence
zres2 = MAXVAL( gcr(2:nlci-1, 2:nlcj-1) )
CALL mpp_max( zres2 ) ! max over the global domain
IF( zres2 < rn_resmax .OR. jn == nn_nmax ) THEN
  res = SQRT( zres2 )
ENDIF

```

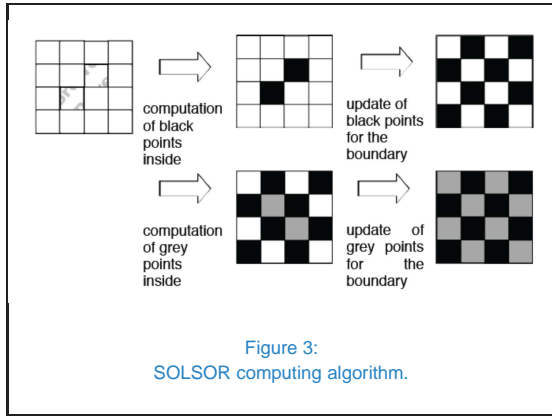
Figure 2: SOLSOR code fragment.

Converge is reached when the residual value is less than a threshold. At each iteration, the generic process computes the black points inside, updates the black points on the boundaries exchanging values with neighbors, computes grey points inside and finally updates grey point on the boundaries exchanging with neighbors (see figure 3).

Communications are very frequent (= # iterations \* 2 \* 4).

The NEMO code already implements an optimized algorithm for reducing the frequency of communication through the concept of extra-halo [1] region.

The report is organized as follows: the next section illustrates the extra-halo feature and how it works; in the further section the authors de-



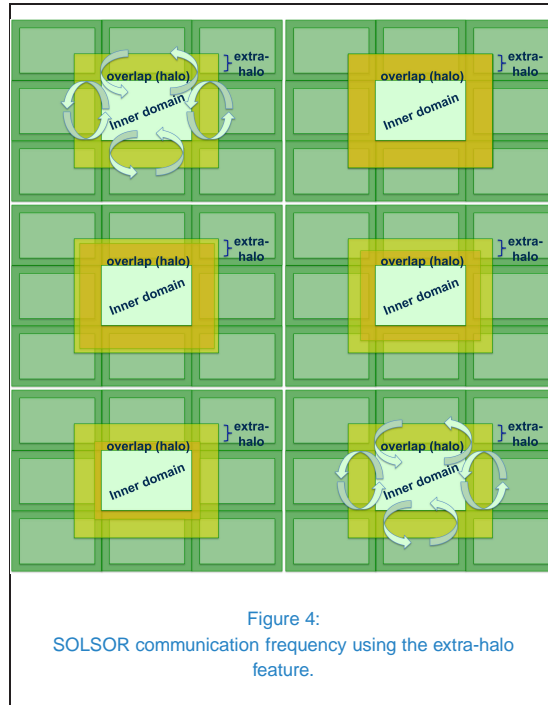
scribe the performance model that allows the selection of the optimal value for extra-halo region for a given number of processes; the model has been validated with a test configuration. Finally the model has been used to analyze the NEMO behavior exploiting extra-halo feature.

## EXTRA-HALO FEATURE

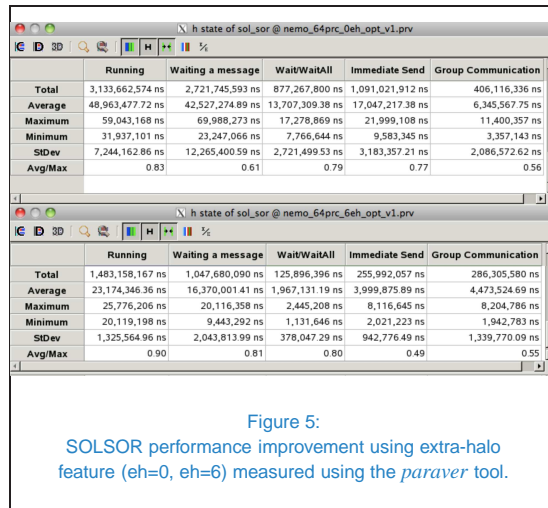
In order to reduce the frequency of communication within the *sol\_sor* routine, the size of the extra-halo region can be used. When the communication between two processes happens, they exchange not only the overlap region (also named halo), but a wider area formed by both the halo and an extra-halo region (see figure 4).

The dimension of this last region can be selected by the user at compile time and it influences the frequency of data exchanges: each process, after exchanging the data, performs computation on the *inner domain* + *halo* + *extra-halo - 1* region. At each next iteration only a line of extra-halo expires so that the process has no need to exchange for *extra-halo - 1* iterations.

The analysis using the *paraver* tool (see figure 5) shows the time spent by the application within the different states (running, waiting a message, group communication, etc.) executing the code on 64 processes (with a decomposition  $16 \times 4$  respectively on *i* and *j* directions),



using two different extra-halo values ( $eh=0$  and  $eh=6$ ).



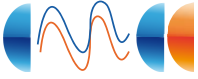
	Running	Waiting a message	Wait/WaitAll	Immediate Send	Group Communication
Total	3,133,662,574 ns	2,721,745,593 ns	877,267,800 ns	1,091,021,912 ns	406,116,336 ns
Average	48,963,477.72 ns	42,527,274.89 ns	13,707,309.38 ns	17,047,217.38 ns	6,345,567.75 ns
Maximum	59,043,168 ns	69,988,273 ns	17,278,869 ns	21,999,108 ns	11,400,357 ns
Minimum	31,937,101 ns	23,247,066 ns	7,766,644 ns	9,583,345 ns	3,357,143 ns
StdDev	7,244,162.86 ns	12,265,400.59 ns	2,721,499.53 ns	3,183,357.21 ns	2,086,572.62 ns
Avg/Max	0.83	0.61	0.79	0.77	0.56

	Running	Waiting a message	Wait/WaitAll	Immediate Send	Group Communication
Total	1,483,158,167 ns	1,047,680,090 ns	125,896,396 ns	255,992,057 ns	286,305,580 ns
Average	23,174,346.36 ns	16,370,001.41 ns	1,967,131.19 ns	3,999,875.89 ns	4,473,524.69 ns
Maximum	25,776,206 ns	20,116,358 ns	2,445,208 ns	8,116,645 ns	8,204,786 ns
Minimum	20,119,198 ns	9,443,292 ns	1,131,646 ns	2,021,223 ns	1,942,783 ns
StdDev	1,325,564.96 ns	2,043,813.99 ns	378,047.29 ns	942,776.49 ns	1,339,770.09 ns
Avg/Max	0.90	0.81	0.80	0.49	0.55

Figure 5:  
SOLSOR performance improvement using extra-halo feature ( $eh=0$ ,  $eh=6$ ) measured using the *paraver* tool.

We can notice that both running and communication time have been improved when the value of extra-halo increased due to respectively the reducing of the time spent performing useful instructions and the number of MPI calls. MPI



calls are reduced since the frequency of communications is reduced. However, we expected an increasing of computation due to the extension of the computing domain with the extra-halo region at each iteration. So, the question is "why running time is less with  $eh=6$ ?". To understand this behavior, it is important to analyze in detail the solver algorithm. The *sol\_sor* function calls the *lnk\_2d\_e* function for exchanging data between processes. Both the routines are characterized by two components: a running component buffering data before sending and after receiving and a communication one. So, if we reduce the number of calls to the *lnk\_2d\_e* routine, we consequently reduce both communication and computation of the same. Also the *sol\_sor* function has two components: the computation of the *gcx* matrix and the collective communication during the convergence test. If we increase the extra-halo value, computation increases, while collective communication remains the same. The total time is the sum of these four components, three of them depending on the value of extra-halo. How many can we increase extra-halo value to have a benefit? We can experimentally verify that after a threshold, computation increases more than communication decreases, so that the total time raises. This threshold depends on the decomposition, so we need of a performance model to select the optimal value of extra-halo when decomposition changes.

## EXTRA-HALO PERFORMANCE MODEL

A performance model for estimating the behavior of the SOR solver routine has been defined. It takes into consideration the four aspects above mentioned. The total time spent by the solver ( $T_{sol\_sor}$ ) is given by: (i) the time spent by the *sol\_sor* function performing communication without considering the *lnk\_2d\_e* ( $T_{com\_sol\_sor}$ ), (ii) the time spent by the *sol\_sor*

function performing computation without considering the *lnk\_2d\_e* ( $T_{useful\_sol\_sor}$ ), (iii) the sum of the time spent by the *lnk\_2d\_e* function performing communication ( $T_{com\_lnk\_2d\_e}$ ) and computation ( $T_{useful\_lnk\_2d\_e}$ ) by the number of calls of *lnk\_2d\_e*, which depends on both the extra-halo value and the number of iterations needed to reach the convergence.  $T_{com\_lnk\_2d\_e}$  and  $T_{useful\_lnk\_2d\_e}$  are related to the exchanged data dimension along the two domain directions ( $L_i$  and  $L_j$ ), which can be obtained from the decomposition.

The time spent by the collective communication depends only on the number of processes  $p$  (the convergence test is performed after the first 100 iterations and has a frequency of 10 iterations) and the useful time of the *sol\_sor* is related to the domain dimension (we have also a constant component for each iteration).

The performance model can be summarized by equation 1.

$$T_{sol\_sor} = T_{com\_sol\_sor} + T_{useful\_sol\_sor} + \left( \frac{2n}{eh+1} + 1 \right) * (T_{com\_lnk\_2d\_e} + T_{useful\_lnk\_2d\_e}) \quad (1)$$

where:

$n$  is the number of iterations needed to reach convergence,

$eh$  is the dimension of extra-halo region, in terms of number of rows/columns,

$T_{com\_sol\_sor}$  is given by equation 2,

$T_{useful\_sol\_sor}$  is given by equation 3,

$T_{com\_lnk\_2d\_e}$  is given by equation 4,

and  $T_{useful\_lnk\_2d\_e}$  is given by equation 5.

$$T_{com\_sol\_sor} = \frac{n-110}{10} * (k_x \log(p)) = k_5 \log(p) \quad (2)$$

$$T_{useful\_sol\_sor} = n * (k_v N + k_p) \quad (3)$$

where:

$N = (jpi + 2eh - 2)(jpi + 2eh - 2)$ ,  
 $jpi$  is the subdomain maximum i-dimension,  
 and  $jpi$  is the subdomain maximum j-dimension.

$$T_{com\_lnk\_2d\_e} = k_1 L + k_2 \quad (4)$$

where:

$L = L_i + L_j$  for corner processes,  
 $L = 2L_i + 2L_j$  for internal processes,  
 $L = 2L_i + L_j$  for processes on vertical boundary,  
 $L = L_i + 2L_j$  for processes on horizontal boundary,  
 $L_i = (jpi + 2eh)(ol + eh)$ ,  
 $L_j = (jpi + 2eh)(ol + eh)$ ,  
 and  $ol$  is the overlap (number of halo lines).

$$T_{useful\_lnk\_2d\_e} = k_3 L + k_4 \quad (5)$$

where:

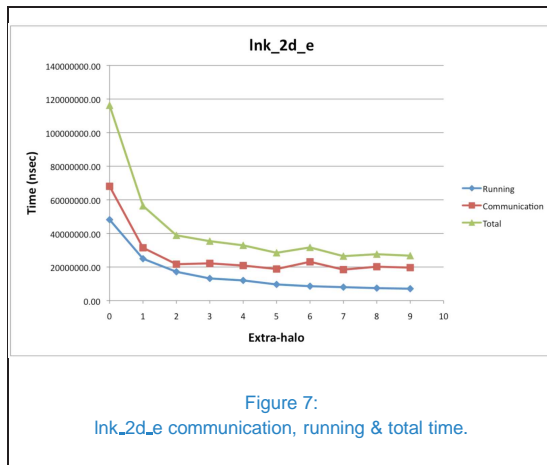
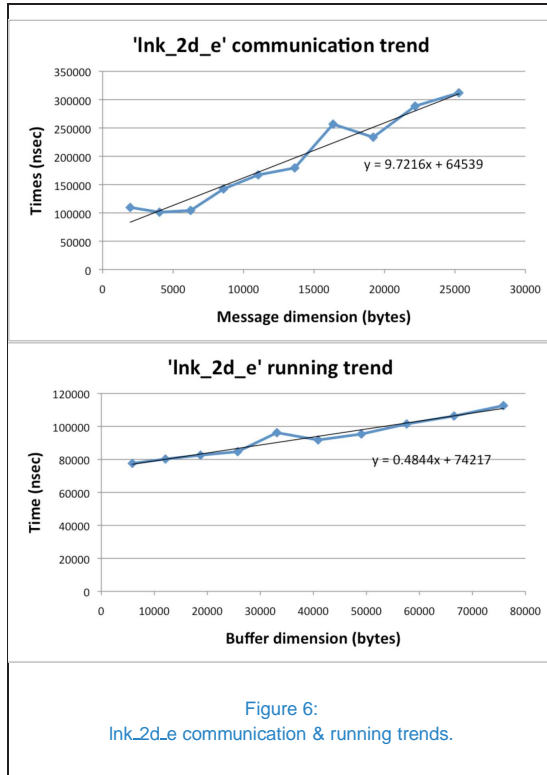
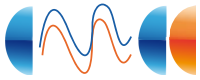
$L = 5L_i + 5L_j$  for corner processes,  
 $L = 6L_i + 6L_j$  for internal processes,  
 $L = 6L_i + 5L_j$  for processes on vertical boundary,  
 and  $L = 5L_i + 6L_j$  for processes on horizontal boundary.

The  $k$ s parameters for communication terms

should be evaluated taken into consideration both measured latency and bandwidth (inter- and intra-node) of MareNostrum, while the  $k$ s parameters for useful terms should be evaluated measuring the time spent for computation. Two kinds of problems happened: on one hand, the strange behavior of communications on MareNostrum, which is very different from the expected one (we have theoretically evaluated the behavior on an ideal architecture with the nominal values declared for MareNostrum using the *dimemas* tool [3]); on the other hand, the granularity of the computational parts that limits the reliability of the measured times. For these reasons,  $k$ s parameters have been experimentally evaluated using the minimum square methods on a set of several runs for which we fixed a predefined domain decomposition and changed the extra-halo value. For both the  $lnk\_2d\_e$  terms, when extra-halo changes also the data dimension exchanged is different: the two graphs in figure 6 show the trend of communication and running time (in microseconds) related to data dimension while the graph in figure 7 shows the running, communication and total time (the sum of the previous components) of the  $lnk\_2d\_e$  when extra-halo increases.

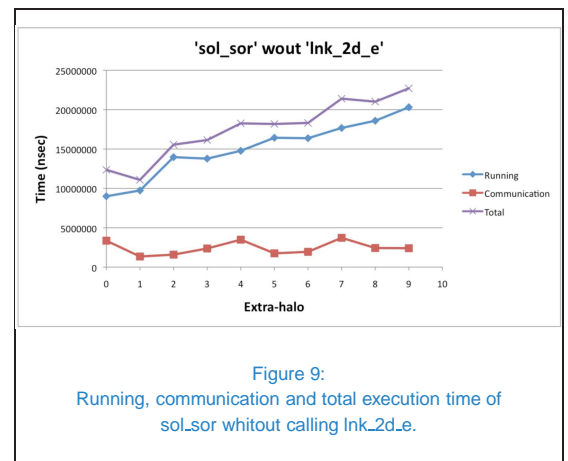
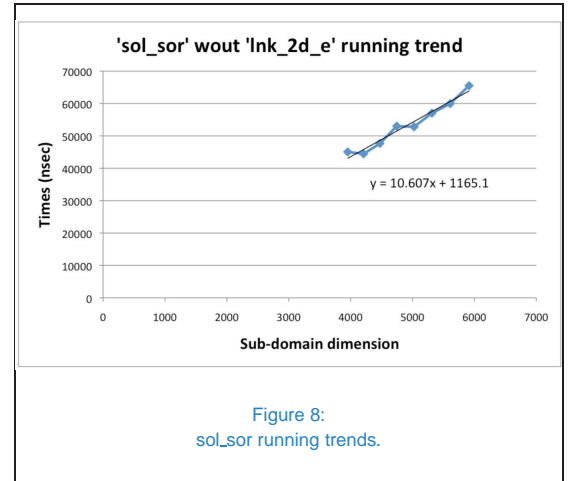
Regarding the analysis of the  $sol\_sor$  terms, we have taken into consideration only the trend of the computing time. Indeed, the communication time is only related to the number of processes and then it is constant when extra-halo changes for a fixed decomposition. Figure 8 shows the running trend of  $sol\_sor$  routine without considering the time spent calling the  $lnk\_2d\_e$ , while figure 9 shows running, communication and total time (in microseconds) of the function related to data dimension.

Figure 10 represents the running, communication and total time of the entire  $sol\_sor$  (including the  $lnk\_2d\_e$  function). As we can notice, af-



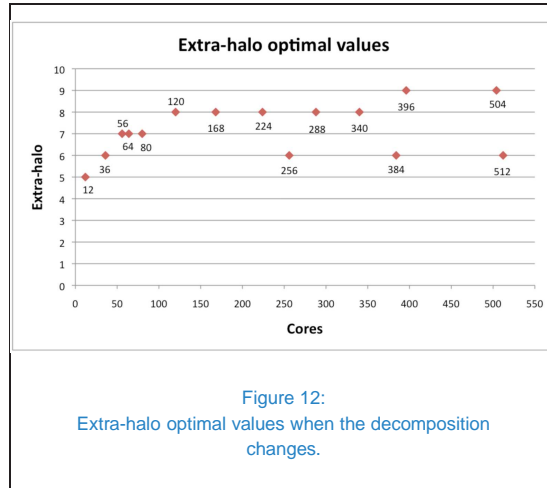
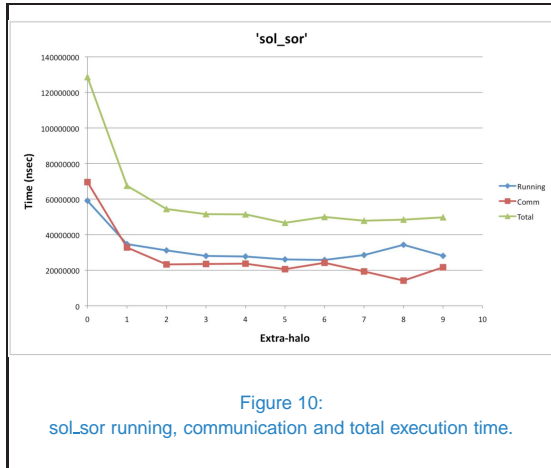
ter a threshold, it is not convenient to increase extra-halo and the model helps us to define the value of this threshold when the decomposition changes.

The model has been validated using the decomposition 16x4 (see figure 11) and it has been used to evaluate optimal extra-halo val-

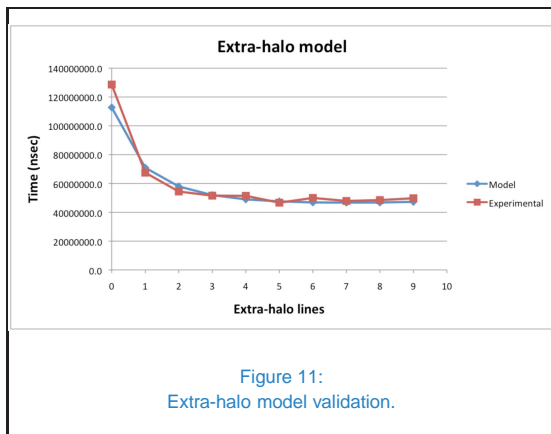


ues (see figure 12) for all of the decompositions taken into consideration to evaluate the scalability of the NEMO code. As we can notice in table 1, the optimal value of extra-halo increases with the number of processes: the number of communications increases with the number of processes and then the use of an higher value of extra-halo is much more useful to reduce the communication frequency. There are three exceptions to this rule when the number of processes along  $i$  direction is equal to 128. In these cases, the minimum number of rows assigned to each process is 6, so data that each process can give to their neighbors is equal to 6: extra-halo can not exceeds this value (even if





the performance model suggests another thing) because extra-halo algorithm implements only the exchange between neighbors.



**Table 1**  
Extra-halo optimal values when the decomposition changes.

#proc along i	#proc along j	#proc	eh optimal value
6	2	12	5
12	3	36	6
14	4	56	7
16	4	54	7
16	5	80	7
20	6	120	8
24	7	168	8
28	8	224	8
128	2	256	6
32	9	288	8
34	10	340	8
128	3	384	6
36	11	396	9
42	12	504	9
128	4	512	6

## NEMO ANALYSIS OF SCALABILITY

Using the extra-halo performance model, we have performed an analysis of scalability comparing the original NEMO code with a new version developed by the SCO Division of CMCC characterized by the optimization of the *obc\_rad* routine, with the version adding to the *obc\_rad* optimization the use of extra-halo values suggested by the model. Table 2 and figure 13 show the results of the analysis respectively highlighting the total execution time and related

efficiency and the speed-up of the previous mentioned parallel versions.

The parallel efficiency and speed-up have been evaluated taking as reference time the wall clock time of the application with 12 processes. Due to the amount (8 GB) of main memory per node available on MareNostrum [2], the execution of the sequential version of the model is prohibitive requiring at least 20 GB with used configuration. The tests have been executed on 1-day simulation: the minimum wall clock time

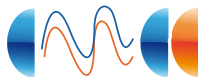
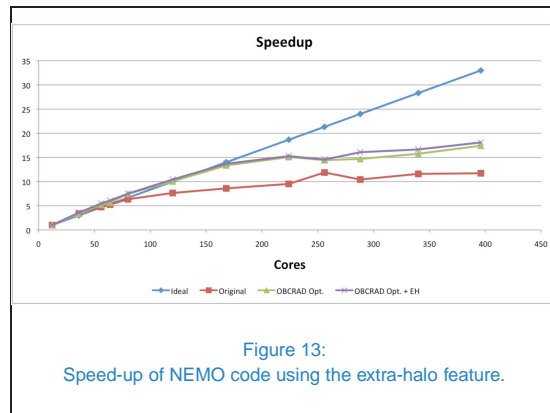


Table 2

Performance analysis of NEMO code using the extra-halo feature.

Decomposition	Cores	OBCRAD exec. time (sec)	opt. efficiency (%)	OBCRAD efficiency (%)	opt. efficiency (%)	OBCRAD opt. and Extra- halo exec. time (sec)	OBCRAD opt. and Extra-halo efficiency (%)
6x2	12	1281.28		100.00		1265.75	100.00
12x3	36	382.47		111.67		352.55	119.67
14x4	56	244.73		112.19		230.69	117.57
16x4	64	226.17		106.22		207.92	114.14
16x5	80	171.54		112.04		168.46	112.71
20x6	120	127.54		100.46		121.19	104.44
24x7	168	95.98		95.35		92.38	97.87
28x8	224	84.95		80.80		82.83	68.26
128x2	256	88.70		67.71		86.92	68.26
32x9	288	87.24		61.20		78.70	67.01
34x10	340	81.26		55.65		75.99	58.79
36x11	396	73.53		52.80		69.93	54.85

happens on 396 cores. Efficiency increases compared with both the original version and the optimized *obc\_rad* one, such as speed up. The execution time is reduced of about 4.9% for this decomposition.

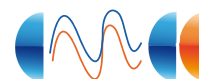


experimental evaluation. The use of a performance model simulating the behavior of the extra-halo feature allows choosing the value for extra-halo reducing at minimum the execution time of the *sol\_sor* routine. For the future, we plan to (i) evaluate the behavior of the code using the extra-halo optimal values given by the suggested performance model over 396 cores (ii) modify the extra-halo feature implementation in order to extend the exchange of data not only to the neighbors of each process.

## CONCLUSIONS AND FUTURE WORK

In this work, we presented the definition of a performance model for the extra-halo feature of the NEMO oceanic model. This feature, developed by the NEMO team, allows the user to reduce the frequency of communication within the *sol\_sor* routine implementing the SOR algorithm for solving elliptical equation. However, once the configuration is fixed, the choice of the extra-halo value is usually the result of an





## Bibliography

- [1] R. Benshila. Optimization of the sor solver for parallel run. Technical Report Technical Report v1, LOCEAN-IPSL, ESOPA Team, 2005.
  - [2] Barcelona Supercomputing Centre. Marenostrum user's guide. Technical report, BSC-CNS, 2009.
  - [3] Barcelona Supercomputing Centre. Dimemas overview. *BSC Performance Tools*, 2010.
  - [4] Barcelona Supercomputing Centre. Paraver overview. *BSC Performance Tools*, 2010.
  - [5] G. Madec. Nemo ocean engine. Technical Report Technical Report 27 ISSN No 1288-1619, Institut Pierre-Simon Laplace (IPSL), 2008.
  - [6] D. M. Young. Iterative methods for solving partial difference equations of elliptic type. *Trans. Amer. Math. Soc.*, 76:92–111, 1954.
- 

