The NEMO oceanic model performance analysis & optimization

Italo Epicoco, Silvia Mocavero, Giovanni Aloisio

CMCC, Italy

Introduction

NEMO (Nucleus for European Modeling of the Ocean) is a widely-used oceanic model written in Fortran 90 and parallelized using MPI with a domain decomposition in latitude/longitude. The computational domain is defined over two and three dimensional grids. In space the model uses the finite differencing method. The temporal integration is explicit, except for the vertical diffusion that uses the trapezoidal implicit scheme. Therefore, computations in the horizontal direction are non-recursive, while in the vertical direction it is necessary to recursively solve a tridiagonal system of equations. Due to the recursive solution in the vertical direction and the non-recursive solution in horizontal directions the MPI domain decomposition is made only over the horizontal plain. The vertical columns of the entire domain are divided for the number of vertical processes *jpnj,* while the horizontal lines are divided for the horizontal processes number *jpni*, with the number of subdomains given by *jpnij=jpni*jpnj*. Each subdomain is assigned to a single MPI process.

Subdomains have a single line of boundary data along the perimeter that are exchanged with the four surrounding processors above, right, below and left, following a cross communication pattern.

For the present analysis a high-resolution configuration of the model at 1/16°, named GLOB16 (http://www.cmcc.it/wp-content/uploads/2015/02/rp0247-ans-12-2014.pdf), has been taken into consideration.

GLOB16 analysis of scalability

The scalability analysis of GLOB16 high resolution configuration has been performed on two architectures: a BG/Q (named Vesta), located at the Argonne Leadership Computing Facilities (ALCF/ANL) and an iDataPlex equipped with Intel SandyBridge cores (named Athena), located at the CMCC Supercomputing Center. Details about the systems are reported in Table 1.

Table 1 - Architectures parameters

Design Parameters	BG/Q (ANL)	IBM iDataPlex SandyBridge (CMCC)
Processor	PowerPC A2	Intel Xeon Sandy Bridge
Cores/Node	16	16
Hardware Threads	4	2
Flop/clock/core	8	8
Flop / Node (GFlop)	204.8	332.8
Clock Speed (GHz)	1.6	2.6
RAM / core (GB)	1	4
Network	5D Torus	Infiniband 4x FDR

The analysis of scalability of NEMO 1/16° has been performed on both the architectures. Results are reported in Figure 1 and Table 2. The timing refers to those simulation steps that don't require I/O operations. The superlinearity on Athena near 1300 cores is due to a cache memory effect. The memory footprint for the reference configuration (320 cores) is about 3GB per process. On BG/Q the code scales up to 16K cores with a parallel efficiency of 46.25%, while on iDataPlex it scales up to 7K cores with a parallel efficiency of 48.0%. The SYPD reaches 0.14 on Vesta and 0.42 on Athena.

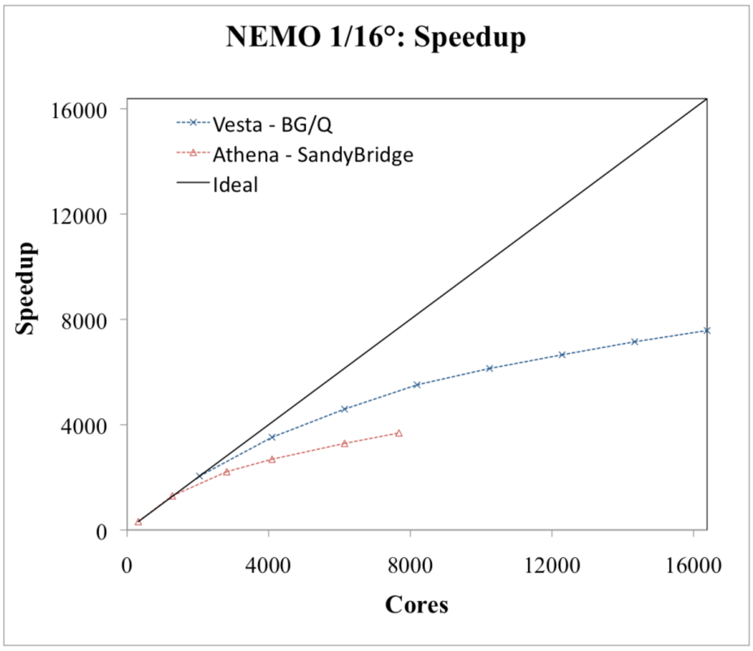

Table 2 - Elapsed time and speedup of NEMO 1/16° Athena (at CMCC) and Vesta (at ANL)

Athena – SandyBridge		
#cores	Elapsed time per timestep (s)	Speedup
320	11.136	320
1280	2.747	1297
2816	1.609	2215
4096	1.325	2689
6144	1.083	3289
7680	0.967	3686
Vesta – BG/Q		
#cores	Elapsed time per timestep (s)	Speedup
2048	11.005	2048
4096	6.397	3523
6144	4.908	4592
8192	4.087	5514
10240	3.673	6136
12288	3.387	6655
14336	3.152	7150
16384	2.975	7577

Routine analysis

A more deep profiling at routine level could reveal the model bottlenecks. Figure 1 and 3 report the speedup for the main routines, respectively on Vesta and Athena. The speedup is evaluated as ratio between the execution time on 2048 and 4096 cores for Vesta and 320 and 2816 for Athena. The ideally value should be 2 for Vesta and 8.8 for Athena due to the considered number of cores.

The bottleneck to scalability on Vesta is related to the **north fold** boundary condition computation, introduced in order to handle the north boundary of a three-polar ORCA grid. On Athena, the scalability is limited by the **tra_ldf_bilap** routine, which computes the tracer lateral diffusion using the iso-level bilaplacian operator.

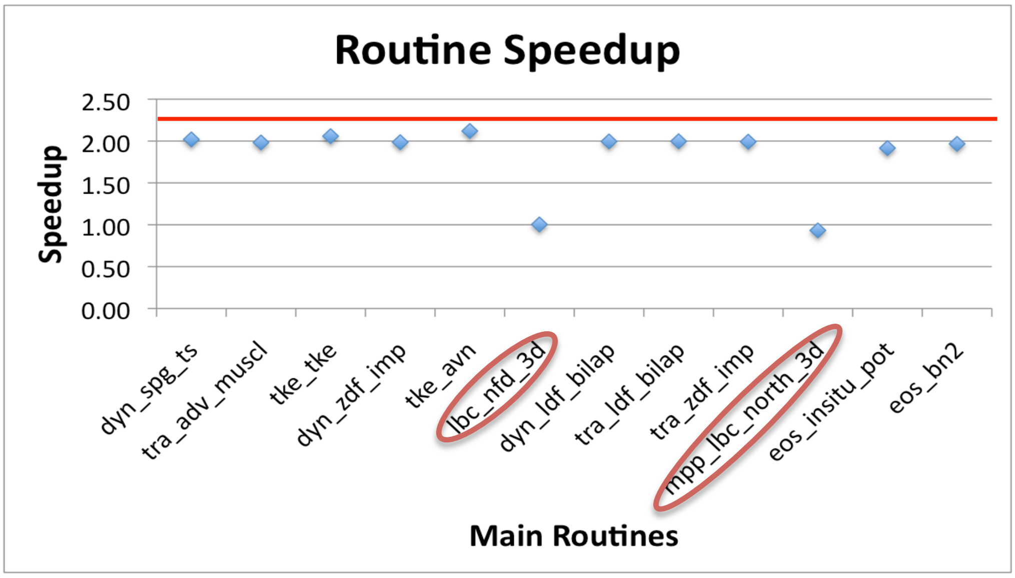

Figure 1 - Speedup of the main routines on Vesta (at ANL)

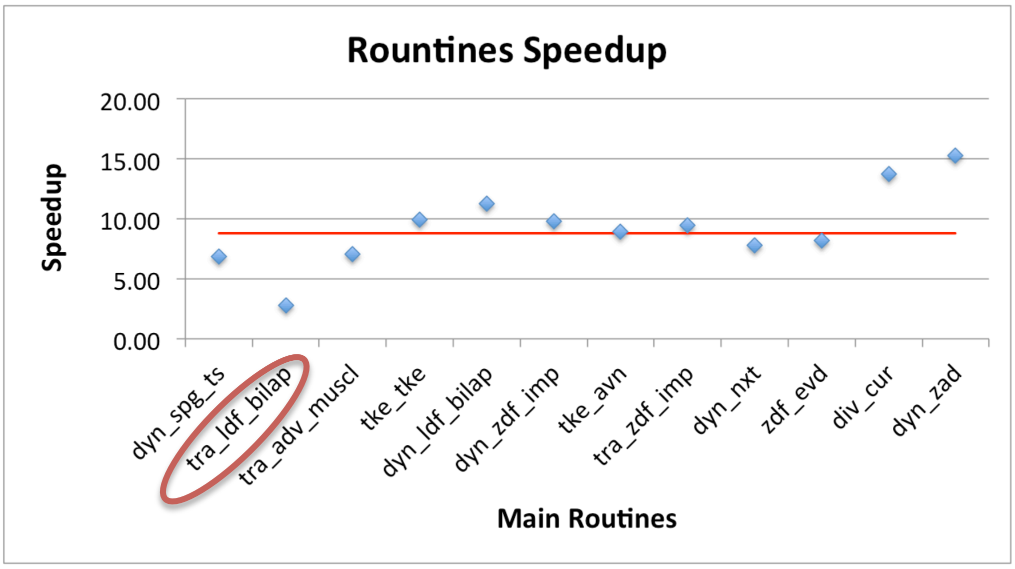

Figure 2 - Speedup of the main routines on Athena (at CMCC)

North Fold optimization

The folding of points at north of the domain handles the north boundary of a three-polar ORCA grid (see Figure 3). Such a grid has two poles in the northern hemisphere.

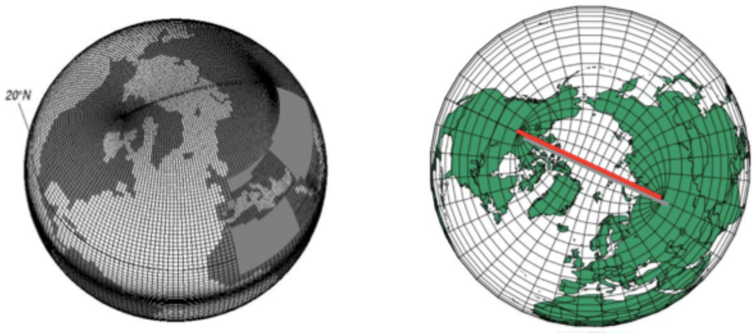

Figure 3 - tripolar ORCA grid

The folding slightly differs depending on the nature of the points (T-, U-, V-, F-point) to be folded and on the pivoting point (T- or F-point pivot). More in general the folding is achieved considering only the last 4 rows on the top of the global domain and by applying a rotation pivoting on the point in the middle. During the folding, the point on the top left is updated with the value of the point on bottom right and so on (see Figure 4).

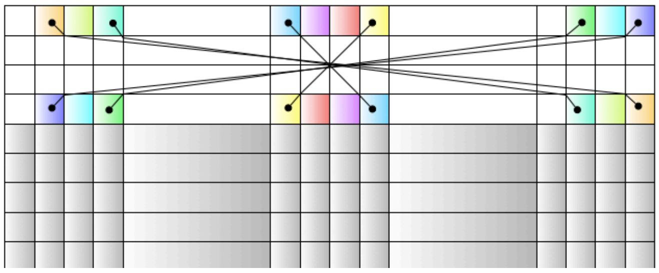

Figure 4 – north-fold algorithm

If we consider *jpiglo* the total number of column in the global domain, the sequential time is proportional to *jpiglo*. The original version of the parallel algorithm is based on the domain decomposition. Each MPI process takes care of a block of points and updates its points using values belonging to the symmetric process. An MPI_Sendrecv communication (as shown in Figure 5) is used for sending *jpi* point to the corresponding process. *jpi* is exactly the number of columns assigned to each process. If we consider the MPI processes disposed in a grid of *jpni x jpnj*, the following relation can be used to identify the pairs of processes that must exchange its north points for the north fold.

$$P(i, jpnj) \leftrightarrow P(jpni - i + 1, jpnj) \qquad \forall i, \;\; 1 \leq i \leq jpni$$

In the original implementation, each received message is placed in a buffer with a number of elements equal to the total dimension of the global domain (*jpiglo* elements).
Each process sweeps the entire buffer, but only a part of that computation is really useful for the process' sub-domain, resulting in a bunch of redundant operations that can be avoided. Since each process sweeps a buffer with *jpiglo* elements, the parallel time of the algorithm is still proportional to the dimension of the global domain.

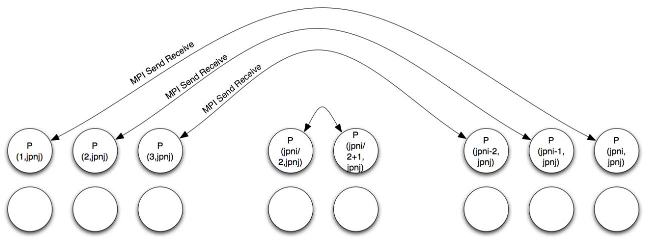

Figure 5 - communication pattern for the north-fold

Optimization

In our optimization, we reduced the buffer length only to the number of elements actually needed by the receiving process. The number of elements needed is exactly the dimension of the sub-domain. We avoid redundant operations and the parallel time is now proportional to the problem size and inverse proportional with the number of processes.

Iso-efficiency analysis

The iso-efficiency analysis (Figure 6) shows that the new parallel algorithm is cost optimal.

Iso-efficiency analysis

(before optimization)

$$T_1 = nt_c$$

$$T_p = nt_c + t_s + \frac{n}{p_i} t_w$$

$$T_O = p_i T_p - T_1 = (p_i - 1) nt_c + p_i t_s + nt_w$$

$$Eff = \frac{1}{1 + \frac{T_O}{T_1}} = \frac{1}{p_i + \frac{p_i t_s}{nt_c} + \frac{t_w}{t_c}}$$

It can not be kept constant

Is constant only if n grows up at the same rate of p

Constant

The efficiency can not be kept constant varying the problem size

(after optimization)

$$T_1 = nt_c$$

$$T_p = \frac{n}{p_i} t_c + t_s + \frac{n}{p_i} t_w$$

$$T_O = p_i T_p - T_1 = nt_w + p_i t_s$$

$$Eff = \frac{1}{1 + \frac{T_O}{T_1}} = \frac{1}{\frac{p_i t_s}{nt_c} + \frac{t_c + t_w}{t_c}}$$

Is constant only if n grows up at the same rate of p

Constant

The algorithm is cost optimal: the problem size must grow as O(p) to maintain efficiency constant

Figure 6- Iso-efficiency analysis comparing the algorithm before and after the optimization

Performance Evaluation

We evaluated the benefit coming from the optimization, using a high resolution configuration named GLOB16 on two architecture: the BG/Q named Vesta, located at the ALCF – Argonne National Laboratory (ANL) and the iDataPlex named MareNostrum (very similar to Athena), located at the Barcelona Supercomputing Centre.

As shown in Figure 7, on BG/Q, the optimization produced an improvement of 44.7% of in the execution time per step on 16K cores. The code scales well up to 16K cores with a parallel efficiency of 70.9%.

Figure 8 shows the performance results on MareNostrum. The optimization produced an improvement of 37.9% in the execution time per step on more than 8K cores. The code scales well up to 8K cores with a parallel efficiency of 81.9%.
A brief comparison between BG/Q and iDataPlex shows that on MareNostrum we can simulate more than half year per day using 8K cores while on Vesta we got a maximum of 0.25 years per day with 16K core. One of the main reasons is the different processor speed. However, a better SMT exploitation by the code could improve performance on the BG/Q system: a hybrid parallelization of NEMO will better suite on many core architectures.

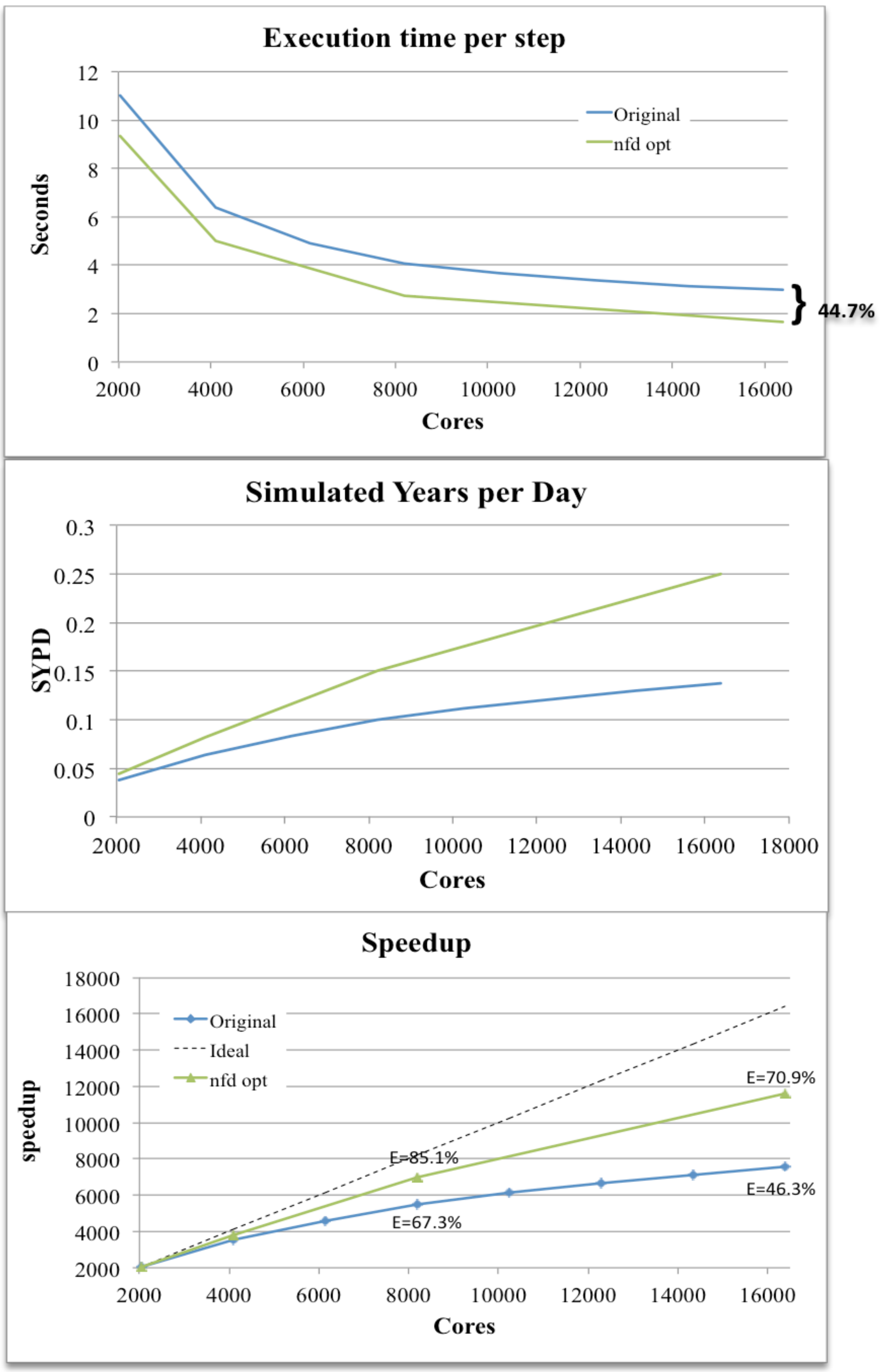

Figure 7 - Performance improvement on Vesta BG/Q

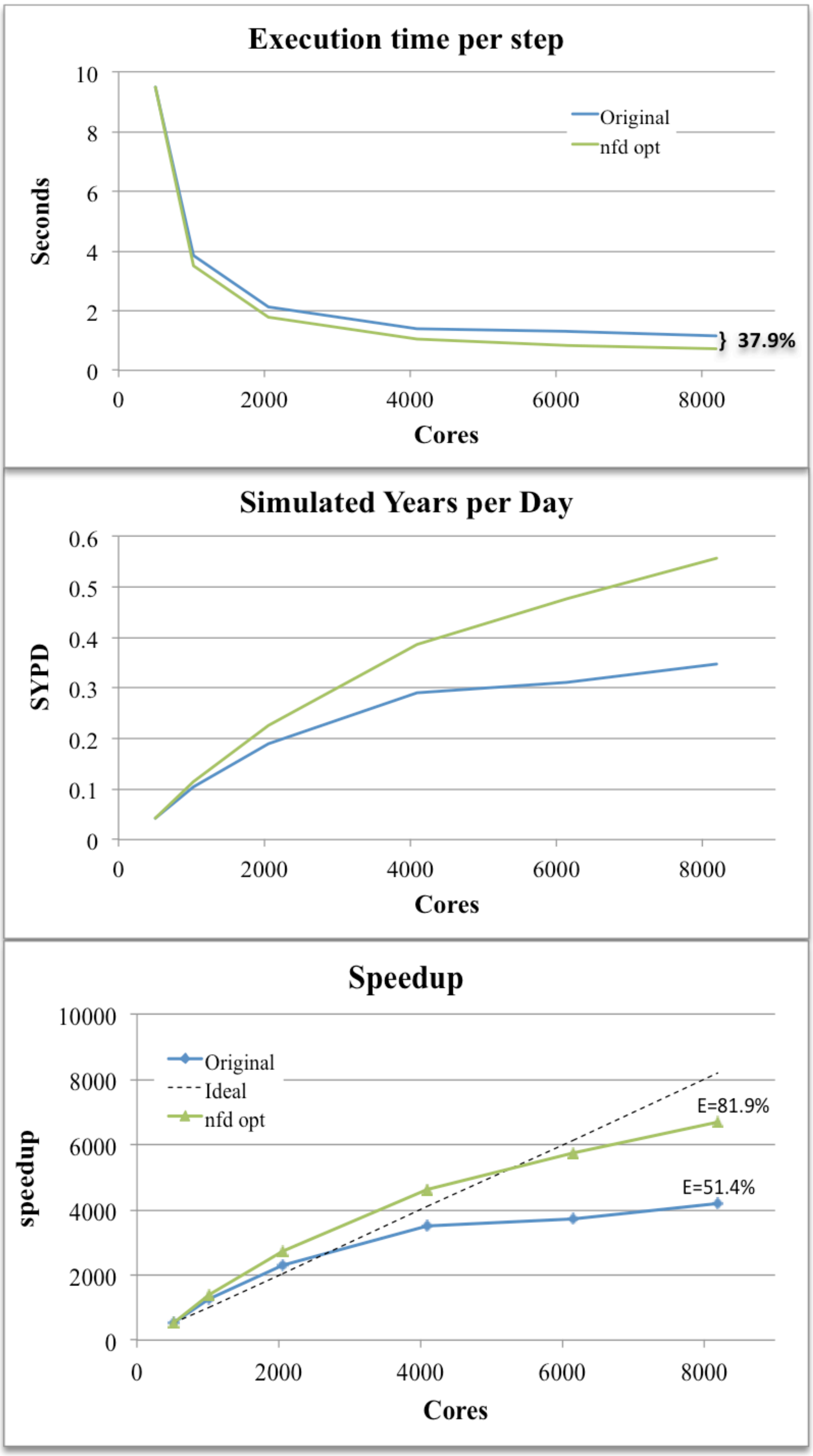

Figure 8 - Performance improvement on MareNostrum iDataPlex