PETSc in the NEMO stack software

Driving NEMO towards Exascale Computing

V. Boccia (INFN), L. Carracciuolo (CNR), L. D'Amore (University of Naples Federico II), A. Murli (SPACI, CMCC)

Some preliminar remarks

- On exascale systems there are some issues [1] related to *scalability*, *heterogeneity adaptivity* and, more in general, *resilience*

- Some issues, like algorithms scalability and software resilience are already well known to NEMO developers but only partially faced by means of:
  - some investments at software level to reduce communication number [2]
  - the use of a disk-based checkpointing to guarantee an embrional offline fault management

- However the problem of scalability still remains the main open issue.

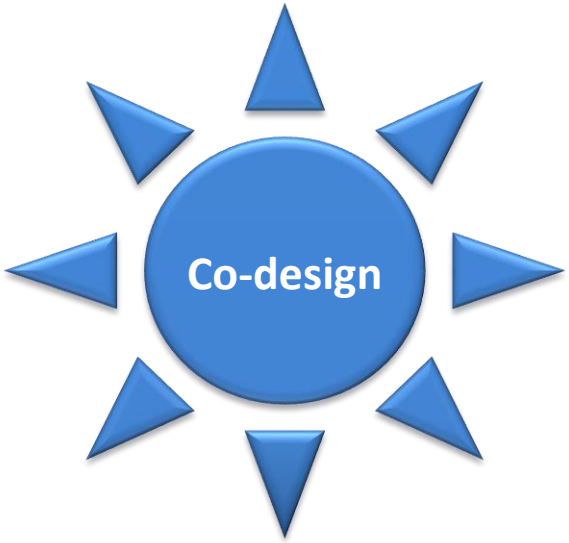
- Secondarily, **software ability to adapt its execution** to benefit from resources heterogeneity are not yet provided.

[1] F. Cappello, D. Wuebbles - «G8 ECS: Enabling Climate Simulation at Extreme Scale», G8 Exascale Projects Workshop - 12 November 2012 – Salt Lake City, USA
[2] I. Epicoco, S. Mocavero and G. Aloisio - «NEMO-MED: OPTIMIZATION AND IMPROVEMENT OF SCALABILITY», CMCC Research Paper Issue RP0096 January 2011 CMCC SCO Department.

The Exascale co-design model for climate simulations

Experts in Climate Science models

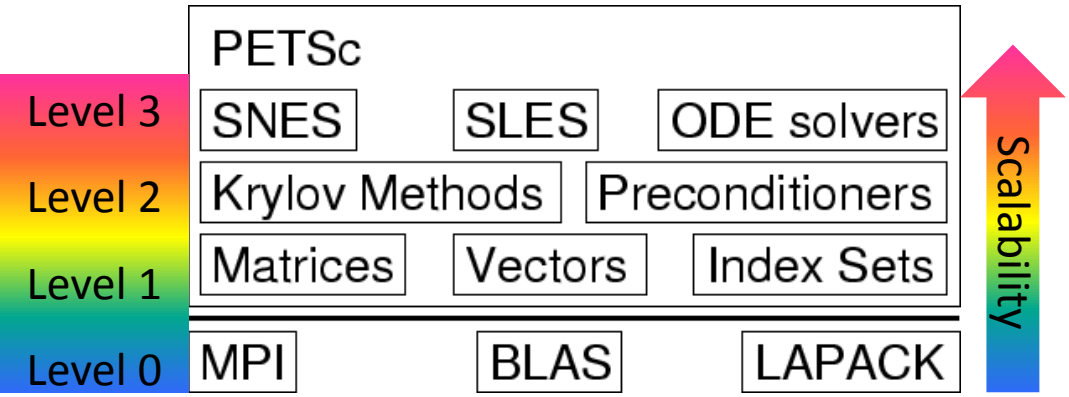
Experts in Numerical and Scientific Computing (new numerical methods, new algorithms, scientific libraries developers, …)



Experts in Computer Science (new emerging technologies, new programming environments, …)

In the context of IESP (International Exascale Software Project), groups of **experts in applications**, **computer and computational science**, are working together to produce new algorithms with features of super scalability, natural fault tolerance and the ability to adapt their execution on emerging hardware systems. Implementations of those algorithms will be included into already **consolidated scientific libraries** and packages able also to interact each other.

Why PETSc



The gerarchic organization of PETSc lets software developers to use PETSc objects at the most suitable level to guarantee high levels of scalability

- **PETSc** developers are working, in exascale context, to produce, i.e., iterative methods superscalable implementations [3,4];
- Similar investments are in progress on other well known scientific libraries as i.e. **TRILINOS** (that is already able to «interact» with PETSc);
- Other softwares (i.e. **TAO** used to solve optimization problems) are already based on PETSc.

[3] L. Curfman McInnes, «Recent Advances in PETSc Scalable Solvers», Lectures at the Department of Computer Science of Technische Universität Darmstadt, 2 July 2012
[4] B. Smith, , «The Portable Extensible Toolkit for Scientific computing», Lectures at the The Argonne Training Program on Extreme-Scale Computing, August 2013

Why NEMO

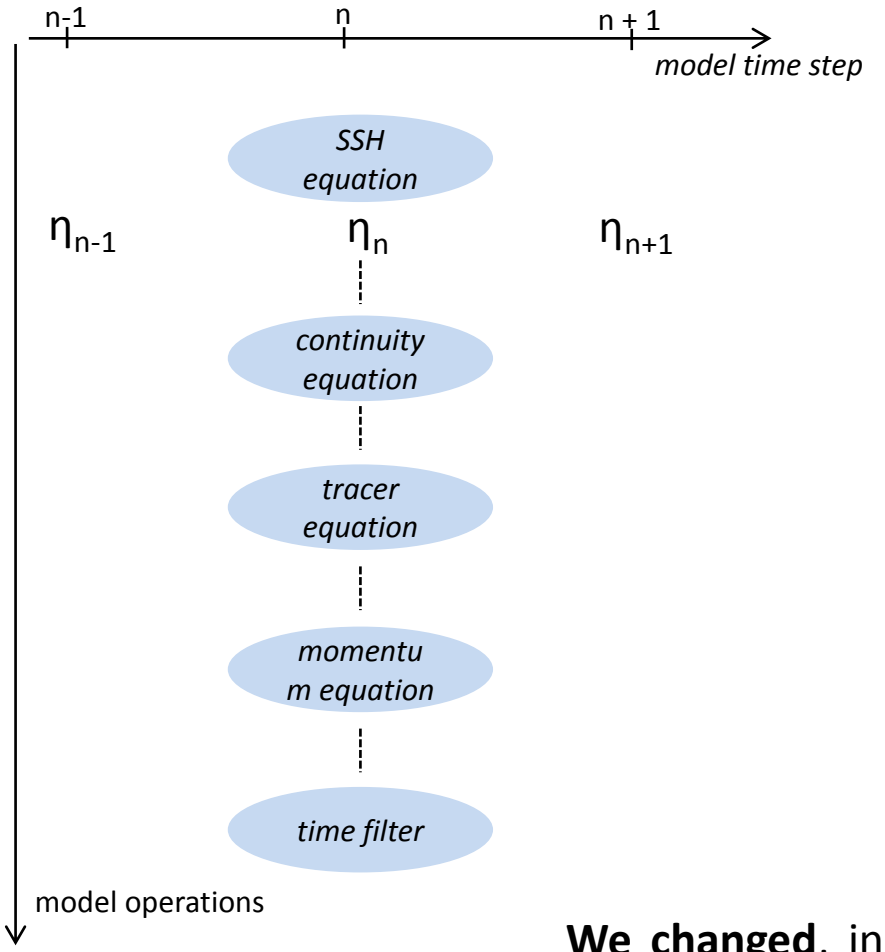
- NEMO* **(Nucleus for European Modelling of the Ocean)** is a modelling framework for oceanographic research
- **NEMO** is used by a large community: 240 projects in 27 countries
- The framework evolution and reliability are organised and controlled by a European Consortium
- **NEMO has to be redesigned to meet exascale requirements**

We investigated on the benefits coming from the use of PETSc (and other consolidated scientific libraries) in NEMO code in terms of robustness, portability and scalability.

*http://www.nemo-ocean.eu/

What we did inside NEMO

NEMO working schema (ORCA2-LIM configuration)



We focused on See Surface Height equation, modeled by PDEs

η_n calculation (by means of Leapfrog method)

needs…

Linear system solution (NEMO uses ad-hoc implementations of PCG & SOR methods)

We changed, in NEMO, **the code** used to solve linear system by means of some PETSc Krylov Subspace Methods (KSP) solvers.

The OPA-NEMO SSH equation solver in PETSc

```
….
/* Creation and definition of the elements of A /*
ierr = MatCreate(PETSC_COMM_WORLD,&A); CHKERRQ(ierr);
ierr = MatSetFromOptions(A); CHKERRQ(ierr);
….
/* Creation and definition of the elements of b and x */
ierr = VecCreate(PETSC_COMM_WORLD,&b); CHKERRQ(ierr);
ierr = VecSetFromOptions(b); CHKERRQ(ierr);
….
ierr = VecCreate(PETSC_COMM_WORLD,&x); CHKERRQ(ierr);
ierr = VecSetFromOptions(x); CHKERRQ(ierr);
….
/* Creation and definition of KSP object   */
ierr = KSPCreate(PETSC_COMM_WORLD,&ksp);CHKERRQ(ierr);
ierr = KSPSetOperators(ksp,A,A,SAME_NONZERO_PATTERN);CHKERRQ(ierr);
ierr = KSPSetInitialGuessNonzero(ksp,PETSC_TRUE);
ierr = KSPSetFromOptions(ksp);CHKERRQ(ierr);
….
/* Solution of the system by KSP object   */
ierr = KSPSolve(ksp,b,x);
….
/* KSP object status   */
ierr = KSPGetIterationNumber(ksp,&its);CHKERRQ(ierr);
ierr = KSPGetConvergedReason(ksp,&reason); CHKERRQ(ierr);
….
```

We executed PETSc implementation of the NEMO SSH module:

- selecting, at runtime, the algorithm to solve the linear system (LEAST SQUARE)
- selecting, at runtime, the most suitable solver implementation (MPI-based, OpenMP, Cuda, …)

```
$> ./Solve -ksp_type lsqr -ksp_rtol 1e-7 -threadcomm_nthreads 8  -threadcomm_type openmp -log_summary
$> ./Solve -ksp_type lsqr -ksp_rtol 1e-7 -threadcomm_nthreads 8 -threadcomm_type pthread -log_summary
$> ./Solve -ksp_type lsqr  -ksp_rtol 1e-7 -threadcomm_type nothread  -log_summary  -mat_type seqaijcusparse -vec_type seqcusp
```

MULTITHREAD

GPGPU

Results vs. constraints

- We obtained:
  - Accuracy level preservation (and software robustness) ✓
  - but a more portable version on ssh-equation solver ✓

- The NEMO SSH module **is not suitable to verify software scalability** because of the small and fixed problem dimension size (actually NEMO model configurations work at maximum 1/16°)

We have to identify other suitable case studies

The other case studies

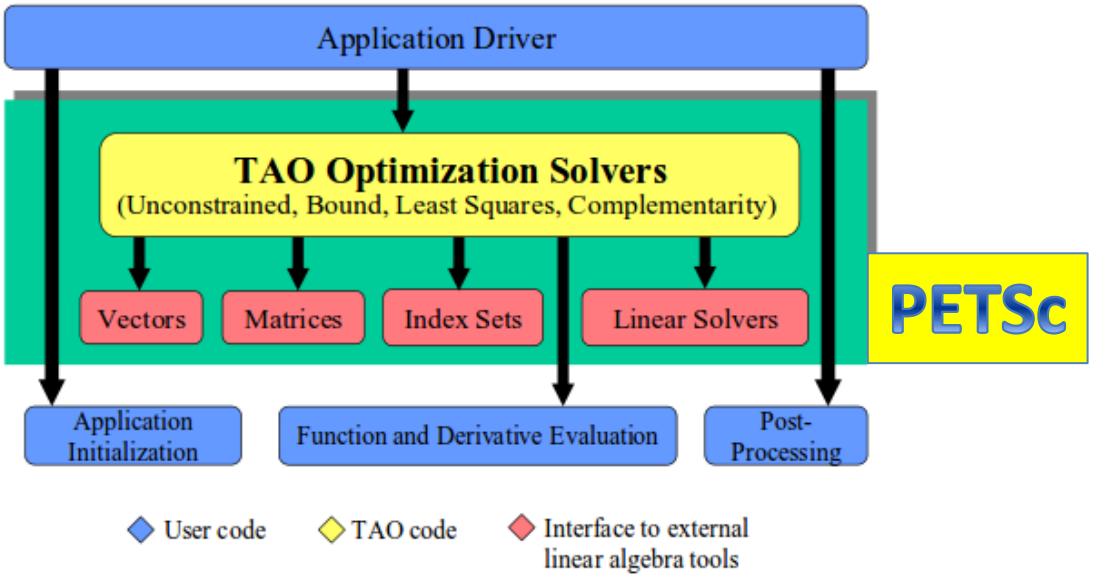
1. The L-BFGS optimization algorithm used in OceanVar [6] and NEMOVAR [5] data assimilation systems

2. The Shallow Water equations used to model waves, where the wavelength is significantly larger than the depth of the medium. They can be considered a simplified version (a "*toy model*") of the more complex and worldwide diffused ocean models.

[5] K. Mogensen, M. Balmaseda, A. Weaver, et al., "The NEMOVAR Ocean Data Assimilation System as Implemented in the ECMWF Ocean Analysis for System 4", ser. ECMWF technical memorandum. European Centre for Medium-Range Weather Forecasts, 2012.
[6] S. Dobricic and N. Pinardi, "An oceanographic three-dimensional variational data assimilation scheme," Ocean Modelling, vol. 22, no. 3-4, pp. 89 – 105, 2008.

The L-BFGS algorithm

- Among consolidated scientific libraries, TAO includes a variety of solvers based on optimization algorithms (among them the L-BFGS algorithm) for several classes of problems.
- As TAO is built on PETSc, it inherits from PETSc all the features and the approach in declaring, defining and using objects.



The TAO implementation of the L-BFGS algorithm

```
/* Creation and definition of the vector x of the initial guess values */
ierr = VecCreate(PETSC_COMM_WORLD, &x);
ierr = VecSetFromOptions(x);
…
/* Begin of the instructions to define the elements of x vector */
…
/* End of the instructions to define the elements of x vector */
…
/* Creation and definition of TaoSolver object */
ierr = TaoCreate(PETSC_COMM_SELF, &tao);
ierr = TaoSetInitialVector(tao,x);
ierr = TaoSetFromOptions(tao);
…
/* Solve the application */
ierr = TaoSolve(tao);
/* TaoSolver status */
ierr = TaoGetTerminationReason(tao, &reason);
…
```

We wrote the needed code to execute the TAO L-BFGS solver:

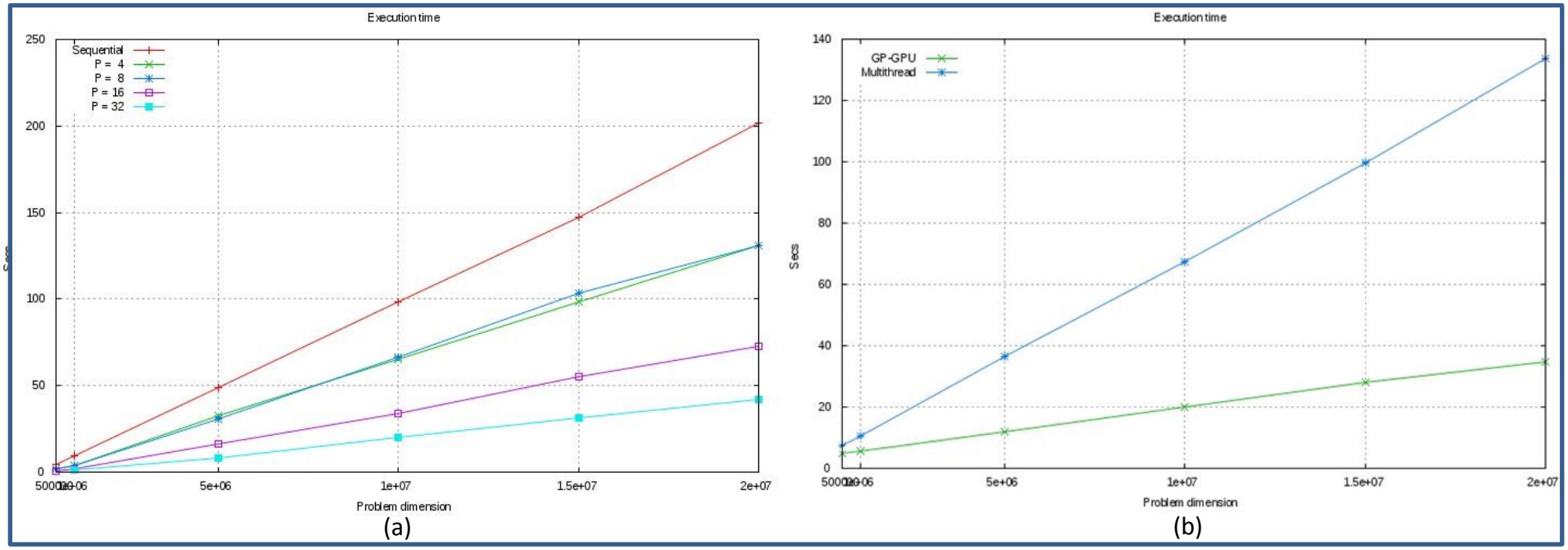
- selecting, at runtime, the parameters affecting TaoSolver behaviour
- selecting, at runtime, the most suitable solver implementation (MPI-based, OpenMP, Cuda, …)

```
$> ./TaoOptimize -tao_method tao_lmvm -tao_max_funcs 10000 -threadcomm_nthreads 8 -threadcomm_type openmp -log_summary
$> ./TaoOptimize -tao_method tao_lmvm -tao_max_funcs 10000 -threadcomm_nthreads 8 -threadcomm_type pthread -log_summary
$> ./TaoOptimize -tao_method tao_lmvm -tao_max_funcs 10000 -threadcomm_type nothread -vec_type seqcusp -log_summary
```

MULTITHREAD

GPGPU

Some remarks



- The TAO L-BFGS algorithm is portable on different parallel paradigms and it mantains a good level of performance.
- To obtain execution times, on a multinode system, which are comparable with that obtained using a GP-GPU based system, we need to use at least P = 32 tasks.
- The use of a larger number P of tasks cannot be useful because of the fixed problem size which cannot be increased due to the limited amount of memory on GP-GPU device.

SYSTEMS CHARACTERISTICS

(a) cluster of 8 nodes, connected by Infiniband technology, each of them with two processors quad core Intel Xeon 5410@2.33GHz.
(b) node with two processors quad core Intel I7 950@3.07GHz and a Tesla C1060.

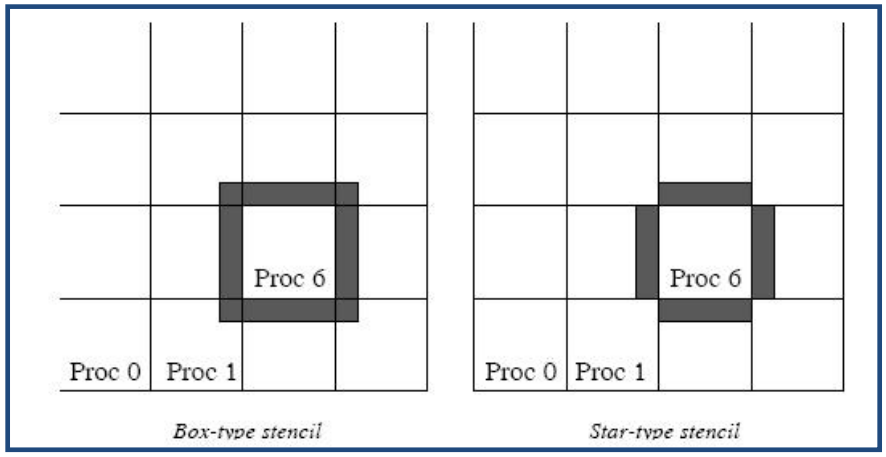
The «toy model» to test scalability

- We used the **shallow water algorithm** to evaluate software scalability (both “weak” and “strong”) when the problem size increases (out of the NEMO software constraints)
- With the terms strong and weak scalability, measured respectively by the SpeedUp S and the Scaled SpeedUp S_S metrics we study how:
  - to solve a fixed size problem in less time as more and more processors are used in parallel (*strong scaling*)
  - to use additional computational resources effectively to solve increasingly larger problems (*weak scaling*)

$$\boldsymbol{S = \frac{T_1(N)}{T_p(N)}} \qquad \boldsymbol{S_S = \frac{pT_1(N)}{T_p(pN)}}$$

The shallow water in PETSc

- The **shallow water problem** is modeled by PDEs which can be discretized in space by a finite differences scheme
- PETSc offers the Distributed Arrays (DMDA) objects that simplify the distribution and the management of the domain data (all the physical quantities on the domain region) in a distributed memory system
- In the global representation of the DMDA associated vectors, each process stores a non-overlapping rectangular portion of the grid points. In the local representation these rectangular regions are extended in all directions by a stencil width which is useful to perform finite difference operations

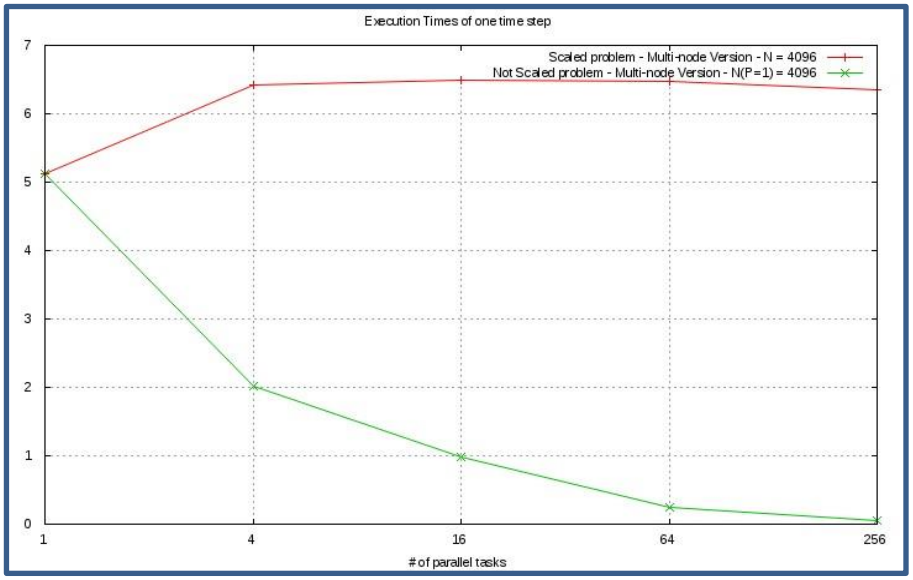


Our implementation of the shallow water problem in PETSc

```
.....
/ * Creation and definition of a DMDA object */
ierr = DMDACreate2d(PETSC_COMM_WORLD, DMDA_BOUNDARY_GHOSTED,
       DMDA_BOUNDARY_GHOSTED,  DMDA_STENCIL_STAR,n,n,PETSC_DECIDE,
       PETSC_DECIDE,ndof,1,NULL,NULL,&da);
ierr = DMDASetFieldName(da,0,"S_height");
ierr = DMDASetFieldName(da,1,"x_velocity");
ierr = DMDASetFieldName(da,2,"y_velocity");
/* Creation of both a global and local arrays associated to DMDA */
ierr = DMCreateGlobalVector(da,&H);
ierr = DMCreateLocalVector(da,&Hloc);
.....
for(itimestep=1;itimestep<=nstep;itimestep++) {
/* begin code related to the work made on local array */
ierr = DMGlobalToLocalBegin(da,H,INSERT_VALUES,Hloc);
ierr = DMGlobalToLocalEnd(da,H,INSERT_VALUES,Hloc);
.....
/*code related to the work made on local array using PETSc interfaces to BLAS1
operations implementation */
.....
/*end code related to the work made on local array*/
ierr = DMLocalToGlobalBegin(da,Hloc,INSERT_VALUES,H);
ierr = DMLocalToGlobalEnd(da,Hloc,INSERT_VALUES,H);
}
.....
```

- DMDA objects are defined and created
- The global and local vectors can be used to update values related to the physical quantities.
- The update phases are performed by BLAS1 type operations
- The use of optimized BLAS implementations gives to the PETSc based applications the chance to exploit all the potentiality of the underling computing system (from GP-GPU to many cores based systems)

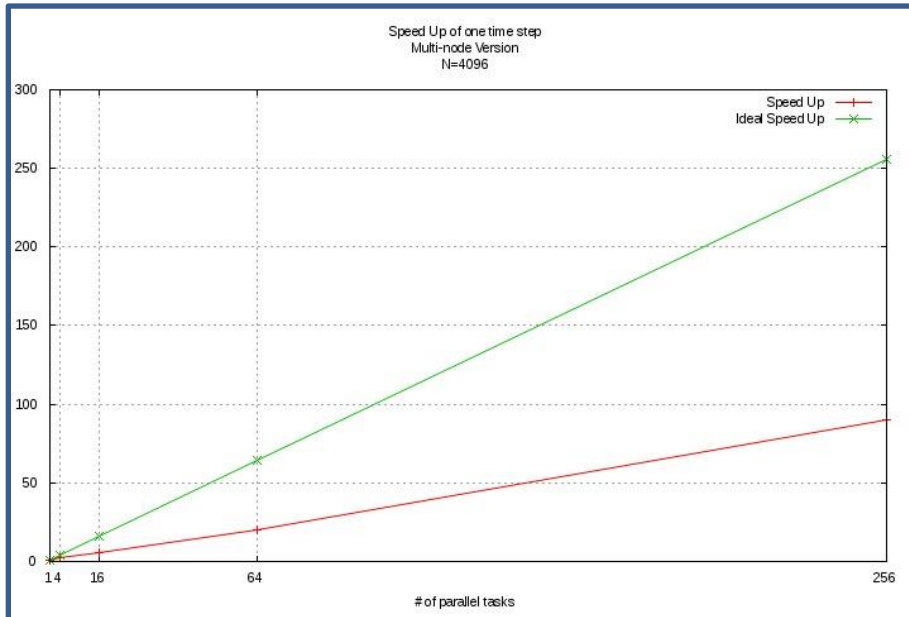
Some results: Weak vs Strong scaling

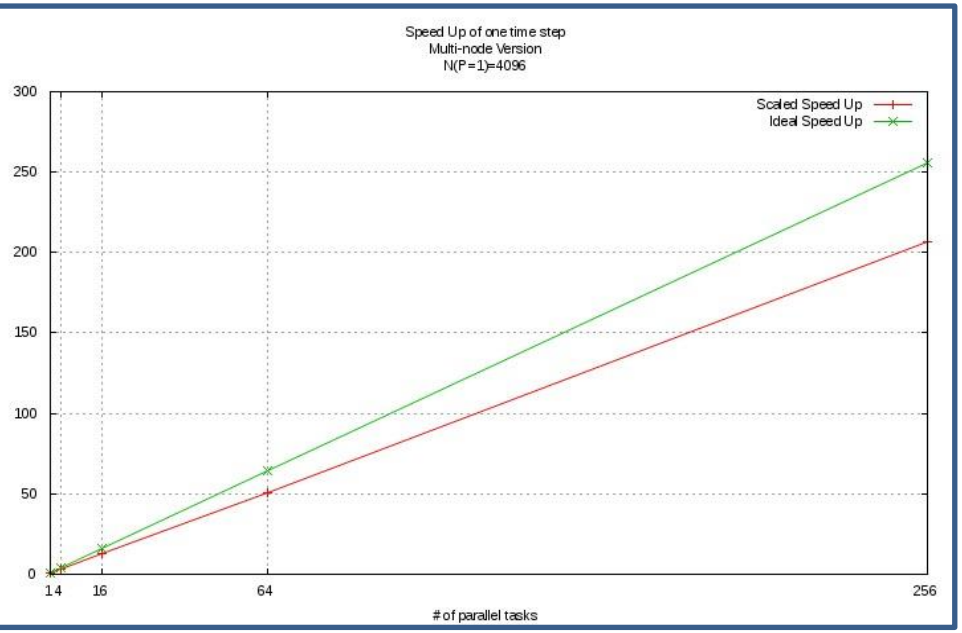


SYSTEM CHARACTERISTICS

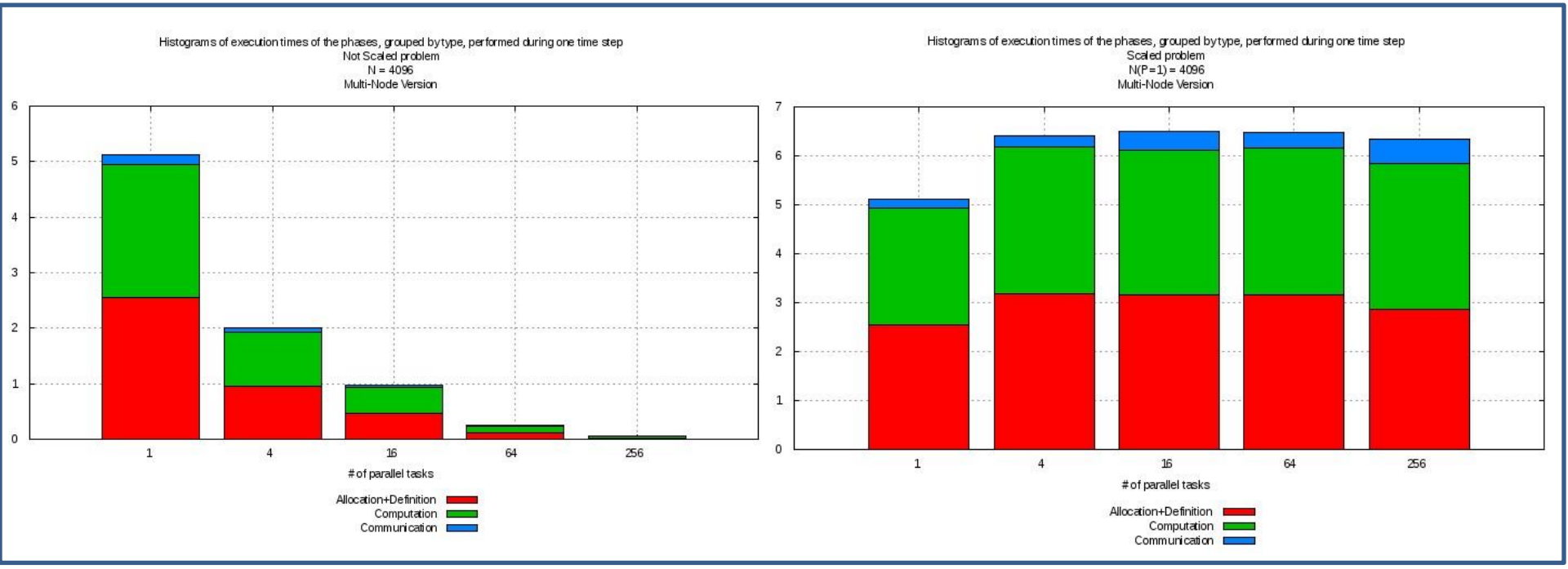
CMCC Athena system: IBM iDataplex cluster with 482 nodes the following features:

- IBM iDataplex DX360M4 dual processor Intel Xeon E5-2670 (8 cores, 2,6GHz);
- RAM per core: 4GB
- Infiniband 4x FDR interconnection





Weak vs Strong scaling results: algorithm phases analysis



We observe that all the algorithm phases preserve scalability properties both in scaled problem and in not scaled one (the percentage of each phase respect to the TTE remains quite constant, by varying the number of tasks)

Summarizing

- Some results, obtained by three case studies, have been presented to show how the use of consolidated scientific libraries in well known Ocean Model code improves software quality in terms of robustness, portability and scalability.
  - Our PETSc version of NEMO SSH has been used to verify aspects related to software **robustness** and **portability**
  - The TAO L-BFGS algorithm implementation gave us the opportunity to verify software **portability** and to face some preliminary issues related to **software performance and scalability on different hardware architectures**,
  - Our implementation of the Shallow Water algorithm in PETSc gave us the chance to discuss some aspects related to **weak and strong scalability**
- Software performances study (related to all the algorithms phases) on different architectures, gives useful information to realize software able to exploit **efficiently hybrid parallel paradigms**.

Conclusions

- Robust and consolidated scientific libraries, such as PETSc, are versatile and capable of adapting to different execution environments (preventing the user to take into account the details related to different computing environments)

- The use of PETSc in NEMO future redesign will involve advantages both in terms of saving time in software development and in terms of software quality

- If the initial effort in using scientific libraries may seem heavy, on the contrary, this approach ensures a longer life for software (amortizing the initial time investments)

Acknowledgment

This work has been realized also thanks to the use of the S.Co.P.E computing infrastructure at the University of Naples in the framework of P.O.N. ReCaS Project