Scalability Improvements for DFT Codes due to the Implementation of the 2D Domain Decomposition Algorithm

Massimiliano Guarrasi[a], Sandro Frigio[b], Andrew Emerson[a] and Giovanni Erbacci[a]

[a]*CINECA, Italy*
[b]*University of Camerino, Italy*

Abstract

In this paper we will present part of the work carried out by CINECA in the framework of the PRACE-2IP project aimed to study the effect on performance due to the implementation of a 2D Domain Decomposition algorithm in DFT codes that use standard 1D (or slab) Parallel Domain Decomposition. The performance of this new algorithm are tested on two example applications: Quantum Espresso, a popular code used in materials science, and , the CFD code BlowupNS.
In the first part of this paper we will present the codes that we use. In the last part of this paper we will show the increase of performance obtained using this new algorithm.

1. Introduction

Many challenging scientific problems require the use of Discrete Fourier Transform algorithms (DFT, [1]) with one of the most popular libraries used by the scientific community, the FFTW package [2] [3]. This library, which is free software, is a C subroutine library for computing DFTs in one or more dimensions with arbitrary input size consisting of both real and complex data. In order to compute a DFT on a distributed memory architecture FFTW uses a 1D block distribution of the data, called Slab Decomposition. However, codes that use standard Slab Decomposition algorithm have been shown to not scale well beyond a few hundred of cores, particularly when using small data arrays. This bottleneck is one of the major issues in order to scale on current PRACE Tier-0 systems that consists of several hundred thousands of cores. Thus, there is a clear need to improve the performance of the FFT solving algorithm on these massively parallel supercomputers.

One of the most promising methods available for solving this issue is a more scalable algorithm such as, for example, the 2D domain decomposition algorithm, which we have shown in a previous work can greatly increase the performance of the application when DFT forms the core of the calculation [4].

Thus, whereas the standard slab decomposition algorithm is faster on a limited number of cores because it only needs one global transpose, minimizing communication, the main disadvantage of this approach is that the maximum parallelization is limited by the number of data points on the largest axis of the 3D data array used (see also Figure 1). The performance can be further increased using a hybrid method, combining this decomposition with a thread-based parallelization of the individual FFTs, but the resulting increase is only small. In any case, for a cubic array with N^3 data points for instance, the maximum number of usable cores scales only as N.

On the other hand, the 2D decomposition algorithm requires another global transpose operation (see Figure 2). Nevertheless, it requires communication only between subgroups of all nodes. In other words, in the DFT computation, this algorithm pays a higher cost in terms of MPI communications, but it ensures higher scalability. Indeed, for a cubic data array with N^3 data points, this means that the maximum number of cores scale as N^2, significantly increasing the number of usable cores, especially when using small arrays.

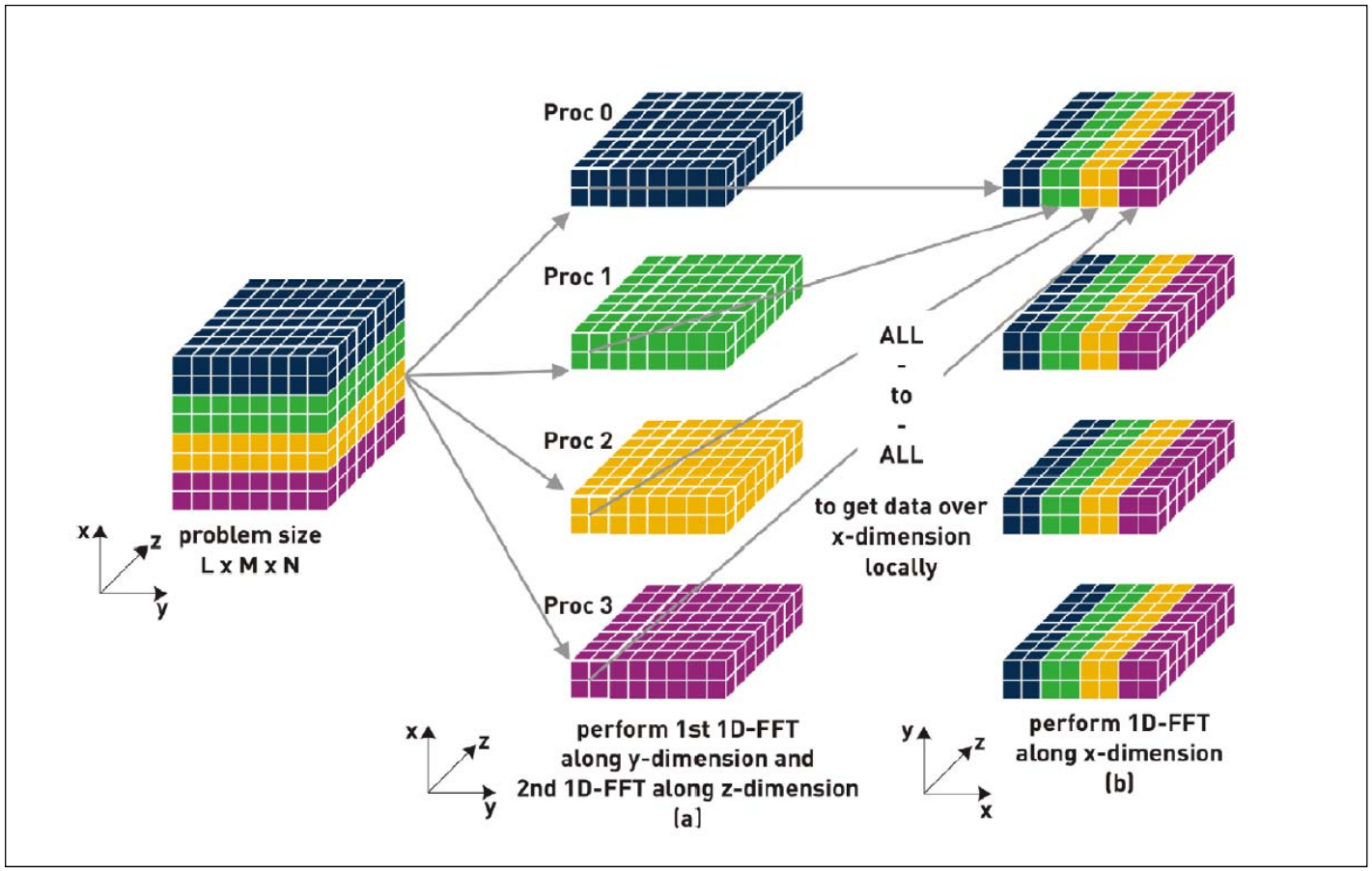

Figure 1: Slab decomposition of a 3D DFT. From [13]. In order to compute FFT we first split the computational domain between processes, then each process compute the FFT on X and Y. After this step the computational domain is transposed an finally the computation of the DFT on the third coordinate are executed.

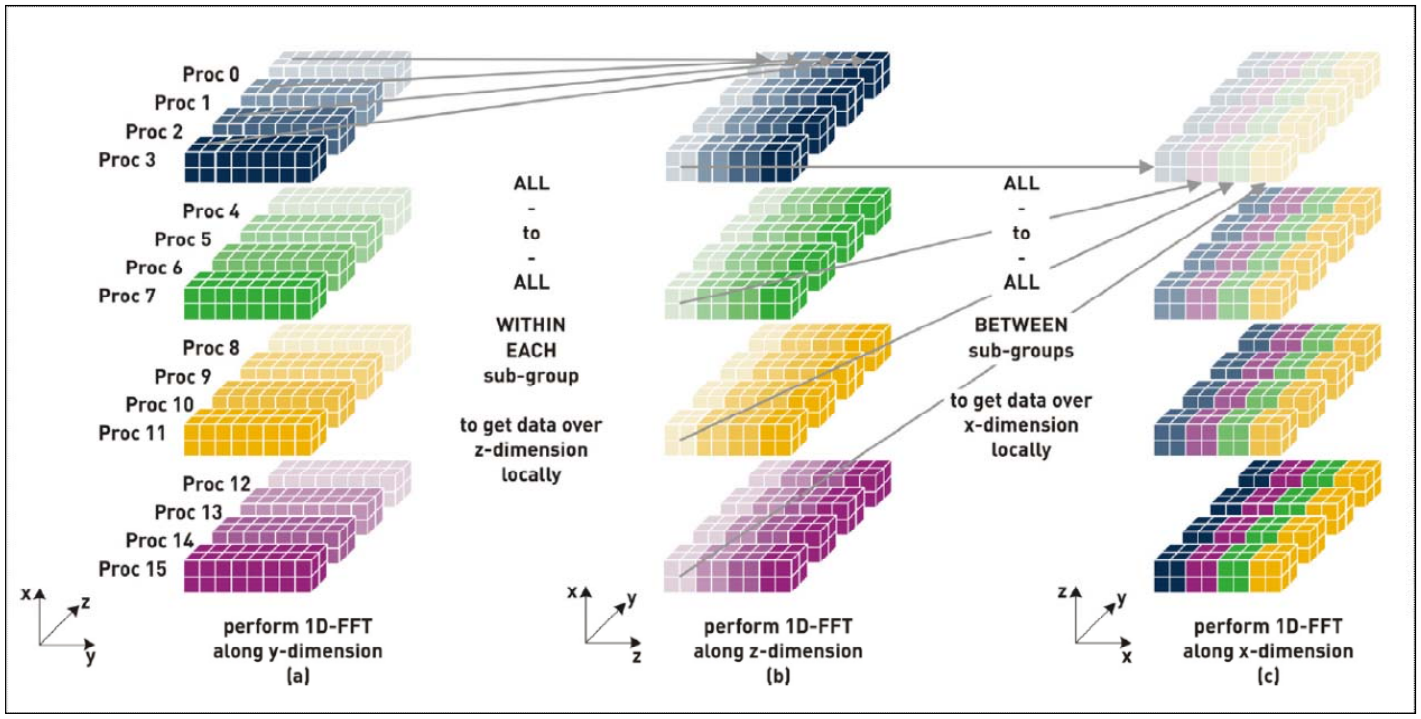

Figure 2: 2D Decomposition of a 3D DFT. From [13]. In order to compute FFT, the array was divided between 16 MPI processes, then each process compute the FFT on X. After this step the computational domain is transposed and the computation of the DFT on the second coordinate (Y) are executed. Finally, after another transposition operation the computation on the third coordinate (Z) is executed.

In this paper we will show how the use of the 2D domain decomposition can affect a CFD code (the BlowupNS code of University of Camerino, Italy) and a Material Science application (Quantum Espresso). Thus, a significant part of the next two sections will be devoted to the description of the benchmarks of BlowupNS that were performed using two parallel supercomputers present in the CINECA infrastructure:

- The PLX cluster [5] was chosen in order to test the performance on a typical Tier-1 system and to test the old version of the BlowupNS code available only on this cluster. The PLX system is an IBM iDataPlex DX60M3 Linux Infiniband cluster. It consists of 274 IBM X360M2 12-way compute nodes. Each node contains 2 Intel(R) Xeon(R) Westmere six-core E5645 processors [6], with a clock of 2.40GHz. All these compute nodes have 47GB of memory.

- The FERMI cluster [7], was chosen in order to test the performance of the codes on a typical Tier-0 system. FERMI is a Blue Gene/Q system [8] and has a massively parallel architecture. Each Compute Card (which we call a "compute node") features an IBM PowerA2 chip with 16 cores, working at a frequency of 1.6 GHz, with 16 GB of RAM and network connections. A total of 32 compute nodes are plugged into a so-called Node Card. Then 16 Node Cards are assembled in one midplane which is combined with another midplane and two I/O drawers to give a rack with a total of 32x32x16 = 16K cores.. In our BG/Q configuration there are 10 racks for a total of 160 K cores.

2. The codes

BlowupNS is a non public code for studying the behaviour of the “blowups” for complex-valued solutions of the hydrodynamic equations, which are predicted in Li *et al*, 2008 [9] and as seen in a two dimensional simplified version Boldrighini et al [17]. It considers the 3D viscous Navier-Stokes equations in the whole space R^3 with no external force, i.e.,

$$\begin{cases} \dfrac{\partial \vec{u}(x,t)}{\partial t} + (\vec{u}(x,t)\cdot\nabla)\vec{u}(x,t) = \Delta\vec{u}(x,t) - \nabla p(x,t) \\ \nabla\cdot\vec{u}(x,t) = 0 \end{cases}$$

In Fourier space the equations have the following form:

$$\vec{v}(\vec{k},t) = e^{t\vec{k}^2}\vec{u}(\vec{k},0) + i\int_0^t e^{-(t-s)\vec{k}^2} ds \int_{R^3} \left(\vec{v}(\vec{k}-\vec{k}',s)\cdot\vec{k}'\right)\left(\vec{v}(\vec{k}',t) - \frac{\vec{k}\cdot\vec{v}(\vec{k}',t)}{\vec{k}\cdot\vec{k}}\vec{k}\right) d\vec{k}'$$

The integral equation is provided with suitable initial conditions. The code studies real solutions of the equation, which however always correspond to complex solutions in x-space.

The code was implemented by discretizing the time integral with the rectangular algorithm, whereas the space convolution was computed by a Discrete Fast Fourier Transform algorithm on a large region.

The old version of the code was parallelized using MPI by dividing the 3D computational domain only on the first coordinate (slab decomposition) with the FFTW library being used to pass from the spatial domain to its reciprocal domain. The new version of BlowupNS has been modified in order to use the 2D parallel domain decomposition paradigm when computing the DFTs. This work was performed using the 2Decomp&FFT library [14], but it requires not only the rewriting of the routines involved in the FFT computation, but also others parts of the program. In fact, three 3D FFT need to be executed in sequence, and they were all implemented using the *decomp_2d_fft_3d* subroutine from 2Decomp&FFT library, while the initialization of the environment variables and the array and plan creation were done using the *decomp_2d_init*, the *decomp_2d_fft_init* and the *decomp_2d_fft_get_size* subroutines from the same library. In the next section we show how the change of domain decomposition can affect the scalability of the BlowupNS code.

The Quantum Espresso code [10][11] (Q.E., open Source Package for Research in Electronic Structure, Simulation, and Optimization) is a freely available (GPL) suite of software for performing materials science using Density Functional Theory (DFT). The package includes ground state calculations, structural optimisations, *ab- initio* molecular dynamics, linear response (phonons) and spectroscopy. It uses a plane wave basis set and implements a range of pseudopotentials and exchange correlation functionals, including Hartree-Fock and hybrid functionals. It is developed by the collaboration of the DEMOCRITOS National Simulation Center (Trieste) with CINECA and several other materials science centers in Europe and the USA.

Quantum Espresso is written in Fortran 90, consisting of over 300,000 lines of code in nearly 1000 source files. A number of external libraries are required including an FFT library (e.g. FFTW3) as well as BLAS, LAPACK and ScaLAPACK for linear algebra operations.

The 3D FFT is a key component of Quantum Espresso, and is implemented as a set of modules which are shared between several of the executables. At a high level, the FFT grids are distributed as slices (planes) in real space, and as columns (rays) in Fourier space. The reason for the column distribution in Fourier space is that not every column will necessarily have the same number of non-zero Fourier coefficients. The columns can be distributed among the processes to load balance the number of non-zero Fourier coefficients per process (see Giannozzi *et al*, 2004[12] for a full description of this technique).

A preliminary implementation of the 2D domain decomposition algorithm has been carried out, using as before the 2Decomp&FFT library. This was done by modifying the *fft_base* module of Q.E. v.5.03 which includes the main features

needed to execute a DFT operation. For this implementation we simply map the planar FFT grid on an arbitrary 2D domain, using as before *decomp_2d_init*, the *decomp_2d_fft_init* and the *decomp_2d_fft_get_size* subroutines. The size of the 2D domain and the number of strips per process were chosen in order to have the best load balancing possible. This first step clearly could require moderate communications between processes, if the number of stripes per process in the 2D decomposition is different from those used in the standard 1D decomposition. After this step the computation of the DFT (or the reverse-DFT if required) was carried out using the subroutine *decomp_2d_fft_3d* from 2Decomp&FFT library. Using this subroutine an array distributed along the Z coordinate was obtained (whereas the input array is distributed along the X coordinate). In order to use this array in the current version of the code another transposition is required; this transposition was made using the *transpose_z_to_x* subroutine. Clearly, both the DFT computation and the transposition require more MPI communications than the standard DFT computation of Q.E. Furthermore, in the current version of our code the columns will not have the same number (or any) of non-zero Fourier coefficients and this could create some issues due to poor load balancing. We are working to solve these issues in order to significantly increase the performance of the Q.E. code. Nevertheless we expect that the performance of the codes will not change significantly, at least for our current version, due to the imperfect load balancing of non-zero Fourier coefficients.

Since the work on this implementation is still in progress, in the next section only the results from the BlowupNS code will be presented.

3. Performance improvements

This section will be devoted to the performance tests of the new version of the BlowupNS code. In order to highlight the increase of performance due to the inclusion of the new domain decomposition, we will first report a test performed on the PLX cluster using both the old and the new versions of the program. In the second part of this section we will report the scalability test performed on FERMI using only the new version (the only one able to scale up to several thousands of cores) using small 3D arrays.

For the benchmarks on PLX we use the same input array of 1024x64x64 grid points with both versions of the program. Clearly in the old version the parallel domain decomposition is implemented only in the X direction (according to the domain decomposition method used by FFTW library), whereas in the new version the domain decomposition is implemented both in the X and Y coordinates.

In Figure 3 we plot the results of this first scalability test on PLX. In the left panel we plot the normalized[1] execution time obtained using both the old version of the code and the new one. In the right panel we plot the efficiency[2] up to 256 cores. We can easily see that whereas the old code scales only up to 64 cores as expected, the new one scales well at least up to 256 cores. This improvement is due to the implementation of the 2D domain decomposition algorithm.

In order to test the real efficiency of the code using moderate size arrays on current PRACE Tier-0 systems, we repeat the same scalability tests on the FERMI cluster. Since the old version of the code cannot run on FERMI due to memory requirements, we report the results obtained using only the new version. We test the performance of the code up to 32768 cores, using two different sized arrays with 4092x256x256 and 4096x512x512 grid points respectively.

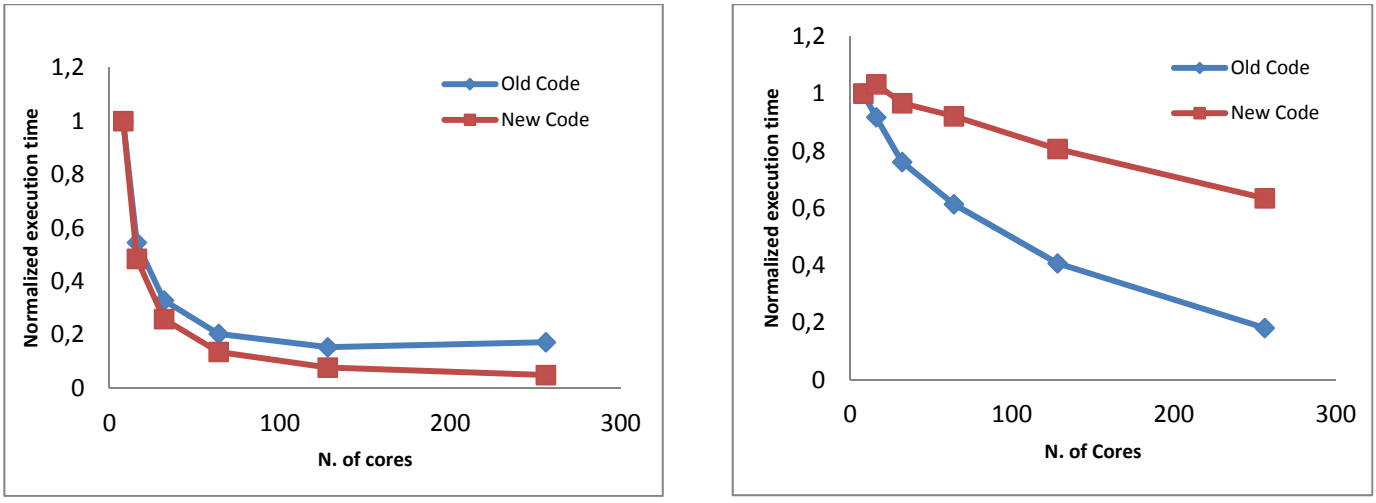

Figure 3:Left panel: Normalized execution time for both the two version of the BlowupNS code on PLX cluster. Right panel: Efficiency obtained from the same data.

[1] Normalized execution time = (execution time using N cores) / (execution time using 8 cores). We normalize the execution time using the execution time for 8 core, because this is the minimum number of cores useable due to memory requirements.

[2] With the term *Efficency* we intend to use the following formula: Efficency = [8 * (execution time using 8 cores)] / [(N * (execution time using 8 cores)].

In Figure 4 we report the normalized[3] execution time of the new version of BlowupNS code on FERMI. The left panel in Figure 4 reports the time needed to compute only the DFT, whereas the right panel reports the total execution time. All values of the execution time are normalized to the execution time of the result with the lowest core count. We analyze the scalability of the code using arrays with two different sizes but, due to memory requirements, for the larger size array we use more cores than for the smaller array. From the left panel we can easily see that the FFT module scales well at least up to 32768 cores.

In Figure 5 we plot the efficiency[4] obtained from the data of Figure 4. We see that the FFT module has a very good efficiency at least up to 10000 cores. On the other hand, as we can see from right panel in Figure 5, for small size arrays we obtain a reasonable efficiency only up to a few thousand cores. Clearly, if we increase the array size we expect that the efficiency of the code will be higher.

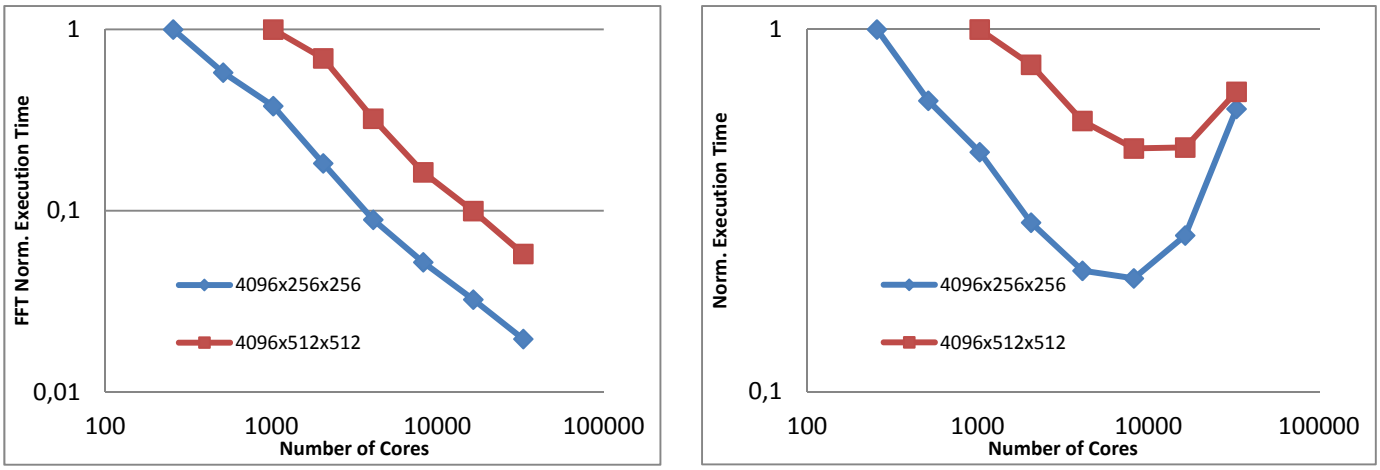

Figure 4:Left Panel: Normalized execution time of the routine that compute only the FFT. Right Panel: Normalized execution time of the whole code. The red lines refers to the execution of the code using an array with size 4096x512x4512 and blue lines to an array with size 4096x256x256.

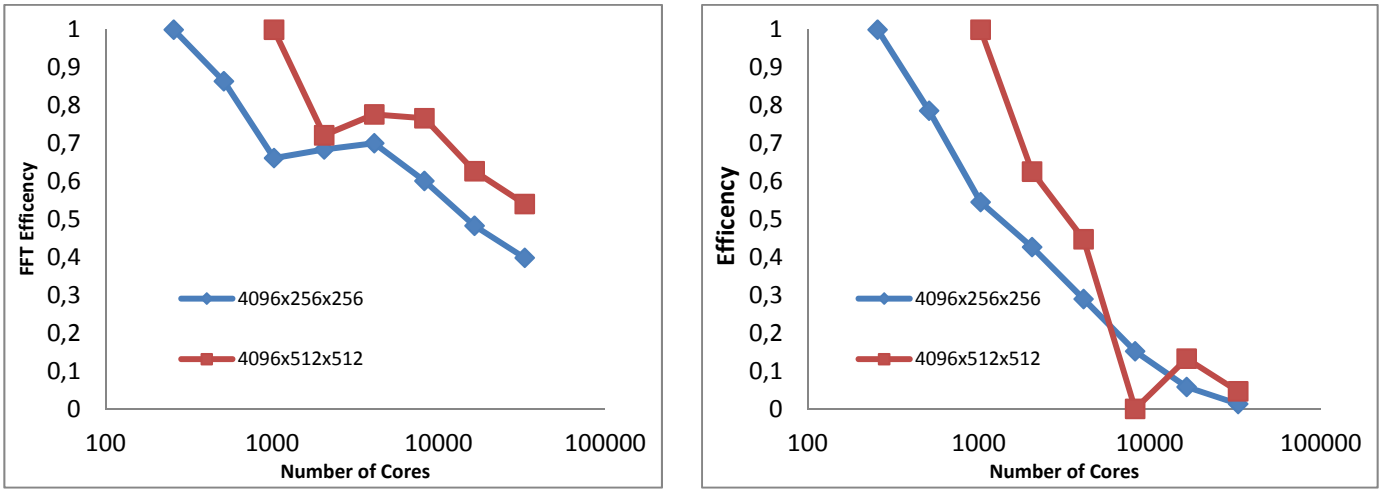

Figure 5: Left Panel: Efficiency of the routine that compute the FFT. Right Panel: Efficiency of the whole code. The red lines refer to the execution of the code using an array with size 4096x512x4512 and blue lines to an array with size 4096x256x256.

[3] Normalized execution time = (execution time using N cores) / (execution time using N_0 cores). We normalize the execution time using the execution time for N_0 core. For the blue lines $N_0 = 256$, whereas for the red lines $N_0 = 1024$. These choices are due to memory requirements of FERMI cluster.

[4] With the term *Efficency* we intend to use the following formula: Efficency = [N* (execution time using N cores)] / [(N_0 * (execution time using N_0 cores)]. For the blue lines $N_0 = 256$, whereas for the red lines $N_0 = 1024$. See also footnote 3.

4. Conclusions

As predicted in our previous work [4], we demonstrated that, at least for those codes in which the computation of a 3D FFT forms the main part of the calculation, the use of a 2D domain decomposition algorithm can significantly improve the performance. In particular, for the BlowupNS CFD code we demonstrate that the new version of the code (i.e. using the 2Decomp&FFT Library) can scale efficiently on the FERMI cluster ensuring a good scalability at least up to 2048 cores also using small arrays. More generally we can say that the use of a 2D decomposition algorithms in this application allows it to scale efficiently on current PRACE Tier-0 systems. Furthermore the implementation of the 2D decomposition algorithm reduces significantly the memory requirements of the code, allowing us to use more accurate grids and also a more symmetric domain configuration.

For the Quantum Espresso code, the results of this first implementation of the 2D domain decomposition algorithm will be published in a paper at the next ParCo conference 2013 [16].

5. Acknowledgements

This work was financially supported by the PRACE-2IP project [15] funded in part by the EUs 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-283493.
The work is achieved using the PRACE Research Infrastructure resources FERMI and PLX at CINECA in Italy.
BlowupNS is a non public code developed by Dr. Sandro Frigio and collaborators at University of Camerino, Italy.
Quantum ESPRESSO is an Open Source distribution initiative of the DEMOCRITOS National Simulation Center(Trieste, Italy) and SISSA (Trieste, Italy), in collaboration with CINECA National Supercomputing Center (Italy), the Ecole Polytechnique Fédérale de Lausanne (Swiss), Université Pierre et Marie Curie (France), Princeton University (USA), and Oxford University (UK).
The authors acknowledge Dr. Carlo Cavazzoni for his technical support on the implementation of 2D Domain Decomposition algorithm on the Quantum Espresso code.
The authors are very grateful to D. Erwin and Prof. Aykanat for their useful comments and suggestions.

References

[1] Cooley, J. W. & Tukey, J., 1965. An algorithm for the machine calculation of complex Fourier Series. *Mathematics of Computation*, Issue 19, pp. 297-301
[2] M. Frigo & S. G. Johnson, 2013, Available at: http://www.fftw.org/
[3] M. Frigo & S. G. Johnson, "The Design and Implementation of FFTW3", Proceedings of the IEEE 93 (2005) 216
[4] M. Guarrasi, G. Erbacci & A. Emerson, “Auto-tuning of the FFTW Library for Massively Parallel Supercomputers”, PRACE white paper. Available at http://www.prace-ri.eu/IMG/pdf/wp55_auto-tuning_of_fftw_library_for_massively_parallel_supercomputers-cineca-prace2ip-wp12.1.pdf
[5] CINECA, s.d. *PLX cluster.* Available at: http://www.hpc.cineca.it/content/ibm-plx-gpu-user-guide
[6] Intel, s.d. *Xeon Westmere.* Available at http://ark.intel.com/products/48768/Intel-Xeon-Processor-E5645-%2812M-Cache-2_40-GHz-5_86-GTs-Intel-QPI%29
[7] CINECA, s.d. *FERMI cluster.* Available at: http://www.hpc.cineca.it/content/fermi-reference-guide
[8] IBM BLUE GENE/Q. Available at http://www-03.ibm.com/systems/technicalcomputing/solutions/bluegene/
[9] D. Li, Ya. G.Sinai,” Blowups of complex solutions of the 3D Navier-Stokes system and renormalization group method”, J. Eur. Math. Soc. 10, 267-313 (2008)
[10] Quantum Espresso. Available at http://www.quantum-espresso.org/
[11] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. Fabris, G. Fratesi, S. de Gironcoli, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, R. M. Wentzcovitch, J.Phys.:Condens.Matter 21, 395502 (2009), http://arxiv.org/abs/0906.2569
[12] P. Giannozzi, F. De Angelis and R. Car, “First-principle molecular dynamics with ultrasoft pseudopotentials: Parallel implementation and application to extended bioinorganic systems” , J. Chem. Phys. 120 (2004)
[13] R. Schultz, 2008, 3D FFT with 2D decomposition, CS project report, Center for molecular Biophysics, Available at: https://cmb.ornl.gov/members/z8g/csproject-report.pdf
[14] N. Li, & S. Laizet, 2010. “*2DECOMP&FFT – A highly scalable 2D decomposition library and FFT interface.*”, Cray User Group 2010 Conferences, Edinburgh.
[15] PRACE: Partnership for advanced Computing in Europe. Available at http://www.prace-ri.eu.
[16] International Conference on Parallel Computing, https://www.conftool.net/parco2013/.
[17] C. Boldrighini, S. Frigio, and P. Maponi: “Exploding solutions of the complex two-dimensional Burgers equations: Computer simulations”, Journal of Mathematical Physics, 53 (2012)