Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

Enabling of Quantum ESPRESSO to Petascale Scientific Challenges

Ivan Girotto[a]*, Nicola Varini[a]*, Filippo Spiga[a], Carlo Cavazzoni[b], Davide Ceresoli[c], Layla Martin-Samos[de], Tommaso Gorni[f]

[a]*Irish Centre for High End Computing, Tower Building, Trinity Technology and Enterprise Campus, Grand Canal Quay, Dublin 2, Ireland*
[b]*CINECA, via Magnanelli 6/3, 40033 Casalecchio di Reno, Italy*
[c]*CNR, Istituto di Scienze e Tecnologie Molecolari (CNR-STM), c/o Dept. Of Chemistry, University of Milan, via Golgi 19, 20133 Milan, Italy*
[d]*CNR, Istituto di Officina Molecolare (CNR-IOM), c/o SISSA, via Bonomea 235, 34136 Trieste, Italy*[d]
[e]*University of Nova Gorica, Materials Research Laboratory, 5270 Ajdovscina, Slovenia*
[f]*Visiting Scientist at the Irish Centre for High End Computing, c/o Università degli Studi di Modena e Reggio Emilia, Dipartimento di Fisica, Via Campi 213/A, 41100 Modena, Italy*

Abstract

In this paper we present development work carried out on Quantum ESPRESSO [1] software package within PRACE-1IP. We describe the different activities performed to enable the Quantum ESPRESSO user community to challenge frontiers of science running extreme computing simulation on European Tier-0 system of current and next generation. There main sections are described: 1) the improvement of parallelization efficiency on two DTF-based applications: Nuclear Magnetic Resonance (NMR) and EXact-eXchange (EXX) calculation; 2) introduction of innovative van der Waals interaction at the ab-initio level; 3) porting of PWscf code to hybrid system equipped with NVIDIA GPU technology.

Application Code: Quantum ESPRESSO

1. Introduction

The Quantum ESPRESSO package was selected in December 2010 as "community code" for Material Science during the first selection phase of PRACE-1IP. The process of selection is described in the PRACE-1IP public deliverable D7.2.1 'Interim Report on Collaboration with Communities' [15].

Quantum ESPRESSO is an integrated suite of computer codes for electronic-structure calculations and materials modeling based on density-functional theory (DFT, [2] [3]), plane waves basis sets (PW) and pseudopotentials (PP, [4]). It is freely available and distributed as open-source software under the terms of the GNU General Public License (GPL). The present applicability of Quantum ESPRESSO ranges from simple electronic structure calculations to the most sophisticated theoretical spectroscopies such as Nuclear Magnetic Resonance (NMR), Electron Paramagnetic Resonance (EPR), Raman and Scanning Tunneling Microscopy, etc. The simulation tools implemented in Quantum ESPRESSO are used across a wide range of R&D applications. The relevance of this code has been highlighted by its adoption in a number of key research groups, renowned institutions as well as in a number of commercial industries.

The work performed across different projects is presented in the following sections. It has been based on a strong partnership between PRACE partners and relevant members of the Quantum ESPRESSO user community. PRACE experts worked in close collaboration with developers to ensure that the Quantum ESPRESSO package can effectively exploit Tier-0 systems. In the next three sub-paragraphs we briefly introduce each of these three projects.

* Corresponding author. *E-mail address*: igirotto@ictp.ie, nicola.varini@gmail.com.

1.1. Scalability improvement of QE-GIPAW code and PWscf EXact eXchange (EXX) part

Simulating biological systems requires access to high-performance large-scale facilities. However, due to the expensive cost of world-class supercomputing resources this is usually awarded, as in the PRACE context, through highly competitive calls which final selection is based on both scientific and technical evaluation. Indeed, efficiency becomes extremely important to obtain high-level score within the technical evaluation.

QE-GIPAW and the PWscf EXact eXchange part included into the Quantum ESPRESSO suite were showing poor scalability beyond a certain scale so that they were considered not eligible to run on Tier-0 European infrastructures on previous calls. In the following we present the implementation of an additional level of parallelism over electronic bands in both of these two codes to overcome this limit and enable the user community to access at the petascale infrastructure. Actually, we introduced two new parallelization levels for the QE-GIPAW code. The validation of the development activity was based on a real data set proposed by Prof. Stefano De Gironcoli to study the cholesterol crystal equilibrium structures in human gallstones from first principles.

1.2. vdW interaction in Quantum ESPRESSO

Within traditional DFT schemes, exchange-correlation functionals (LDA and GGAs) do not account for non-local interactions such as van der Waals (vdW) [6]. Nevertheless, the explicit inclusion of non-local kernels through the direct implementation of existing vdW functionals results in a dramatic increase of computational needs. In 2004, Perez and Soler described a more efficient FFT-based way of implementing a particular class of such functionals [5]. Two of these have been included in the Quantum ESPRESSO package: vdW-DF [6] and vdw-DF2 [7]. However, in late 2010 it has been shown that the VV10 [8], that does not belongs to Perez-Soler functionals class, gives better results with respect to experiments. Successively, R. Sabatini and S. De Gironcoli, from SISSA institute in Tieste (IT), studied and released a variant of VV10, called rVV10. This new functional maintains the same level of accuracy than VV10 but it can be implemented with the Perez-Soler scheme. This allows DFT simulations to benefit of the most reliable vdW self-consistent description available up to date. The development work aims to implement rVV10 method in Quantum ESPRESSO. This will enable the user community to simulate large physical systems of particular interest for soft matter and molecular biology.

1.3. GPU version of PWscf code

GPU computing has revolutionized HPC by bringing the performance of the most powerful supercomputers of 10 years ago to today's' common workstations. Attractive price, performance, and interesting characteristics in term of power efficiency are the basic grounds for which currently, multiple GPUs are commonly plugged into both desktop machines as well as supercomputer nodes. Recent proliferation of reports in the scientific and technical literature show greater application performance on wide variety of computational problems using hybrid combinations of multiple GPUs and CPU computing resources. GPU-Computing experts from ICHEC and CINECA undertook the task of extending the PWscf [9] code included into the Quantum ESPRESSO suite to fully exploit computing platform equipped with NVIDIA GPUs.

The PWscf code is designed for highly scalable execution with the focus of maintaining a considerable level of portability. As described also in PRACE-2IP deliverable D8.1.2 'Performance Model of Community Codes' [16] this is mainly achieved through a modularized structure that implements several levels of parallelization and the compulsory use of standard scientific libraries such as BLAS, LAPACK and FFTW. The development activity mainly aimed to instrument the PWscf code to include NVIDIA CUDA libraries to exploit GPU capabilities on distributed computing resources, today accessible for European users through PRACE-2IP DECI [i] call. Only few computational kernels were ported on CUDA to improve the overall performance.

2. Method

In this section we describe the main development approaches adopted to reach the goals introduced in the previous section. The following paragraphs describe the activities related to each of three projects presented in this document. We also underline the constant interaction between PRACE partners and relevant scientists from the users community who have been at different levels involved across the various projects.

2.1. Increasing parallelism of QE-GIPAW code and PWscf EXact eXchange (EXX) part

Historically, the Quantum ESPRESSO package has been designed and used for modeling condensed-matter problems in which the physical system can be mapped into a periodically repeated primitive cell. In these cases, the number of electrons (and consequently the number of electronic bands) is a relatively small number while the number of k-points (Brillouin zone sampling) is usually high. As a result, the distribution of k-points across processes is a natural method of parallelization, especially considering that the DFT Hamiltonian is diagonal in k (the only source of communication arises from a sum in the calculation of the total energy). However, the boom of new organic and hybrid based electronics and optoelectronics (originated by the need of replacing silicon-based technologies), together with the fast evolution of supercomputers have opened new scientific challenges and have extended the boundaries among condensed-matter, chemistry and biology. Most of the new topics, involve systems with a huge number of atoms (subsequently, a huge number of electrons) and either low-symmetry (i.e., surfaces or wires) or no-symmetry at all (i.e., biological molecules). In order to address the present needs of the community, it is mandatory to define new parallelization levels and schemes that better exploit the system-of-interest characteristics (many electrons) on the modern supercomputers (scalability over thousands of cores).

By analyzing the implementation of the two codes QE-GIPAW and PWscf (EXact eXchange part only) we realized that the most computational intensive sections were nested into loops over electronic bands. Since these algorithms do not present data dependency along the band dimension, a new level of parallelization over the band index has been introduced. Once implemented, this simple concept becomes a breakthrough to simulate new scientific problems efficiently on large-scale supercomputers and to discover emerging physical effects that arise at a scale that was computationally unreachable before. The new implementation is showed schematically in figure 1.

```
do ibnd = 1, nbnd
    call invfft ( psic ( ibnd ) )
    operation over psic ( ibnd )
    call fft ( psic ( ibnd ) )
enddo
```

```
do ibnd = ibnd_start, ibnd_end,
    call invfft ( psic ( ibnd ) )
    operation over psic ( ibnd )
    call fft ( psic ( ibnd ) )
enddo
call MPI_Allreduction (psic , elctronic_bands_communicator)
```

Fig. 1. Code sections from QE-GIPAW code: (a) original implementation schema; (b) new implementation schema

GIPAW is a DFT-based method to compute Nuclear Magnetic Resonance properties (NMR), such as chemical shifts. GIPAW as implemented in QE-GIPAW uses pseudopotentials and plane wave basis set. Pseudopotential and plane waves provide the user with an excellent balance between easy-use, speed and accuracy. In order to run a NMR simulation with QE-GIPAW, it is first necessary to obtain the equilibrium geometry (together with its corresponding electronic ground-state density and potential) via SCF calculation with PWscf.

One of the most computational intensive algorithms of the QE-GIPAW code is to evaluate the linear response of the wave functions to an external magnetic field. This is done by solving the Sternheimer equation which in practice amount to solving $n = 1..N$ independent linear systems of the form:

$$(H^{(0)} - E_n^{(0)})\, \Psi_n^{(1)} = H^{(1)}\, \Psi_n^{(0)}$$

where $(H^{(0)} - E_n^{(0)}) = G^{-1}$ is the inverse of the Green's function, $H^{(1)}$ is the perturbing magnetic field, and $\Psi_n^{(0)}$ are the unperturbed wave-functions, obtained by a previous SCF calculation with PWscf. Here $\Psi_n^{(1)}$ are the unknowns and the index n runs over all occupied electronic states. Previous to the present work, the Sternheimer equations are solved band-by-band by the Conjugate Gradient method, with a clever re-grouping of unconverged bands, in order to exploit level-3 BLAS operations. In the our new implementation, we further distribute the occupied bands over CPUs, and the CG algorithm has been modified to work with groups of bands.

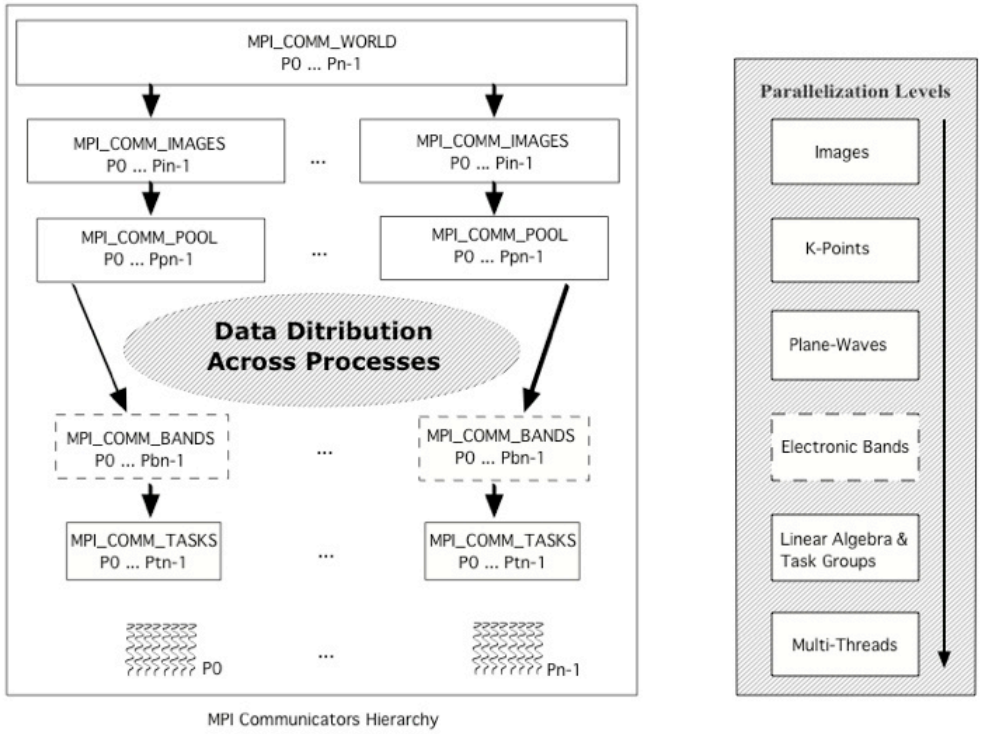

Fig. 2. The picture stands for the levels of parallelism implemented within the Quantum ESPRESSO software packages. On the right hand side the hierarchy is logically represented from the higher to the deeper parallelism, expressed using multithread on top of MPI distribution. The left hand side shows how this hierarchy is mapped on MPI groups of processes (or communicators). From the top to the bottom four of these levels are divided in smaller groups from previous level (each black arrow stands for a splitting stage), starting from MPI_COMM_WORLD, which represents all the MPI processes. Readers must consider that: *in = n / #images, pn = in / #pools, bn = pn / #bands and tn = bn / #tasks*. The dashed squares represents the new level of parallelism described in this paper.

To implement this additional level of parallelism a new MPI communicator has been introduced named as *elctronic_bands_communicator*. The new schema of process decomposition allows re-creating the original version of the replicated data structure (*psic* in this case) across all the processes as soon as they have updated the local section of data (from *ibnd_start* to *ibnd_end*). The new hierarchy of MPI communicators is shown in figure 2.

```
do i = 1, m
…
   do iq = 1, nqi
      …
      do ibnd = 1, nbnd
         call fft ( rho ( ibnd ) )
         operation over rho ( ibnd )
         call fft ( rho ( ibnd ) )
      enddo
      …
   endo
…
endo
```

```
do i = 1, m
…
   do iq = 1, nqi
      …
      do ibnd = ibnd_start, ibnd_end
         call fft ( rho ( ibnd ) )
         operation over rho ( ibnd )
         call fft ( rho ( ibnd ) )
      enddo
      call MPI_Allreduction(psic, inter_elctronic_bands_communicator)
      …
   endo
…
endo
```

Fig. 3. Code sections from EXX code: (a) original implementation schema; (b) new implementation schema

The code analysis brought out that the GIPAW calculation is based on a simultaneous run over a 7-fold loop at one of the outermost routines. The image_communicator, already implemented in the modular structure of the package, is designed for such purpose but it was not yet used for the QE-GIPAW code. We introduce the new

communicator to take advantage also of this level of parallelism. As presented in the next section even higher scalability is reachable whether this is exploited. Although the model presented here is quite simple the introduction of new levels of parallelism required a considerable re-factoring effort due to the complexity of the software structure. As second goal, we aimed to improve the scalability of the Exact-Exchange (EXX) code. Since our target was to enhance the parallelisation for simulating big systems we worked at the bands level as described for the GIPAW code. As presented in the figure 3 once again it was possible to parallelize the inner loop over the electronic bands by introducing the electronic_bands MPI communicator (i.e., by splitting the ibnd across processors to perform on a different local section of the array rho). However, in this case the parallelisation was much more effective because it was implemented on a high level loop that performs the whole computational workload of the EXX applications. In other words, the sum over occupied states that appear in the Green function calculation of the EXX self-energy has been distributed across processors. In the next section we will show the improvements obtained with these new parallelisation schemes.

2.2. Implementation of rVV10 method

To carry out this project ICHEC opened a scholarship position as outcome of the co-operation between PRACE staff members and the Quantum ESPRESSO user community, represented by Prof. Stefano De Gironcoli. Tommaso Gorni (from University of Modena & Reggio Emilia) visited the Irish centre for a period of 10 weeks during which he worked in close collaboration with ICHEC staff. During the scholarship he had been in constant connection with main authors of the rVV10 method who are mainly based in Trieste (IT). The development can be briefly described in two main phases: 1) checking the reliability of the rVV10 approximation and 2) implementing rVV10 with the Perez-Soler method and testing it.

During the first phase, the new functional rVV10 was tested. As the new method is an approximation of the VV10, a first step of validation was due to compare results with the original method. It was developed a testing subroutine that receives the charge density as input and calculate both the energy and the potential contribution of rVV10 at the end of each self-consistent iteration. At this stage we wanted only to validate the correctness of the method. In fact, this was trivially implemented with a double integral to compute energy on local data and then performing an *all_to_all* distribution across processes. While this process required almost half of the project time, the good results obtained on the S22 set provided a reference for the further Perez-Soler implementation, and encouraged to proceed (see section below).

In the second phase the rVV10 was implemented following the much more efficient FFT-based Perez-Soler model which computes the energy as a single integral over the charge, instead of double, allowing to use local data only. This is available in two separate files: Modules/xc vv10.f90 and PW/generate vv10 kernel table.f90, following the same strategy of the previous vdW functional implemented in Quantum ESPRESSO.

All the testing was done on the S22 Test Set [10], a set of 22 dimers widely used to check the vdW functionals.

2.3. GPU of PWscf code

This development project was carried out at ICHEC with the supervision of Carlo Cavazzoni and the fundamental contribution of scientist from the Quantum ESPRESSO developers community such as Paolo Gianozzi and Layla Martin-Samos.

The GPU code was developed starting from the serial version of the code. Filippo Spiga and Ivan Girotto published more detailed information on this first phase of the project in [11]. The most computationally expensive sections of the Self-Consistency Field (SCF) were gradually enabled to run on NVIDIA GPUs, see also figure 13 of the PRACE-2IP public deliverable D8.1.3 'Prototype Codes Exploring Performance Improvements' [17]. The performance of each section was assessed step by step. Here are listed the main milestones performed during the phase of development:

- GEMM operations were replaced transparently with phiGEMM operations
- 3D-FFT was accelerated using CUFFT wherever possible
- LAPACK functions were replaced with equivalent MAGMA functions
- Computationally expensive routines were replaced by CUDA kernels

GEMM operations (mainly DGEMM or ZGEMM) may represent up to 35-45% of all the wall-time of a typical representative benchmark. phiGEMM extends the basic mapping presented by Fatica [12] implementing a heterogeneous SGEMM, DGEMM, CGEMM, and ZGEMM matrix multiplication that can use multiple devices (including the host processor). It has been explicitly designed to treat large matrices (bigger than any memory space currently available on GPU technology) as well as very rectangular matrices. At chapter 8 of his CUDA book “Application design and development” [18], Rob Farber shows how phiGEMM delivers more than 1 TFlop/s performing a 15GBytes, double precision matrix-matrix multiplication on a hybrid GPU system equipped with 2 x 6cores Intel Xeon X5670 @ 2.93GHz and 4 x NVIDIA C2050. Once installed, the library is linkable to scientific code as well as other commonly used math libraries (i.e., Intel-MKL, NetLib, ESSEL, etc...). The phiGEMM can be also leveraged into a high-exposure, commonly accepted HPC benchmark for conventional and co-processor accelerated computers. PWscf code was instrumented to use phiGEMM library to perform matrix-matrix multiplication when running on GPU hybrid system.

CUFFT is a free NVIDIA library to provide a simple interface for computing FFT on NVIDIA GPU technology. As claimed by NVIDIA the application of this library within a single node strongly outperforms common FFT library such as FFTW or Intel-MKL. One of the main bottlenecks of the PWscf code is the series of 3D-FFT that are performed to transform wave-functions, charge density and potentials from reciprocal to real space and vice versa. C wrappers for memory management and calls to CUFFT have been implemented, as the FORTRAN nature of the code does not allow to directly interface CUFFT efficiently.

The MAGMA project aims to develop a dense linear algebra library for heterogeneous Multi-core+GPU architectures to replace the LAPACK version for CPU. MAGMA has been used to accelerate the solution of eigenvalues and eigenvectors to solve the commonly applied Devidson iterative diagonalization. The replacement of LAPACK to the original FORTRAN code is quite simple as presented in figure 4.

Finally two CUDA kernels were explicitly developed: the *newd* routine to update the nonlocal part of the electron-ion interaction potential (in the ultra soft pseudopotential framework [13]) and the *addusdens* routines which adds the charge density related to the hard component of the pseudo-potential and the electron charge density to obtain the total charge density.

For the parallel version the phiGEMM library and the CUDA kernels are re-used since they act on local data, but the main problems performed on distributed data need a deeper analysis. A parallel algorithm for 3D-FFTs is implemented as a series of local 1D-FFTs, combined with data transposition that is implemented *ad hoc* to accommodate different grids. Then, 3D-FFTs are performed independently over all the electronic bands of a given physical system. The GPU implementation aims to collect the distributed grids on shared memory nodes to execute in parallel bunch of independent 3D-FFTs by using CUDA libraries on re-aggregated data. This approach is still under implementation. We are still looking for the best implementation to minimize the inter-process communication. Moreover, an on-going analysis aims to better understand whether this model speed-up the traditional CPU-based approach.

The introduction of MAGMA library as described above can be though for both serial and parallel versions of PWscf code. In fact, also the CPU-based implementation let users define at compile time whether they want to perform Davidson iterative diagonalization using LAPACK on local data or ScaLAPACK on distributed data. Actually, LAPACK is the default configuration as this implementation is more efficient for small and medium

size physical systems. However, for large parallel execution it is recommended to use ScaLAPACK to optimize both memory distribution and process communication. Unfortunately there is no GPU implementation of ScaLAPACK yet.

The code has ben tested and ported on number of architectures (both Intel and AMD platforms equipped with NVIDIA GPUs) to finally obtain a robust and reliable product for the worldwide scientific community. For this purpose a huge effort was spent to make the GPU version available within the official package on which C wrappers and CUDA kernels are self-contained (PW/cuda) and modularized. The CUDA section becomes active part of the package only if users specify --enable-cuda while running the ./configure command. More information on how to build the GPU version of PWscf are available within the file README.GPU included in the source code [14]. All the main functionalities commonly utilized for Self-Consistency Field (SCF) simulation are covered, indeed both k-point and gamma-point calculations are supported. Only non-collinear calculation [19] is still not implemented.

```
#if defined(__CUDA) && defined(__MAGMA)
        CALL start_clock( 'MAGMA_DSYGVD' )
        CALL magmaf_dsygvd( 1,'V','U',n,v,ldh,s,ldh,e,work,lwork,iwork,liwork,info )
        CALL stop_clock( 'MAGMA_DSYGVD' )
#else
        CALL start_clock( 'DSYGV' )
        CALL DSYGV( 1,'V','U',n,v,ldh,s,ldh,e,work,lwork,info )
        CALL stop clock( 'DSYGV' )
#endif
```

Fig. 4. Section of PWscf code to show how MAGMA library can easily replace standard calls to LAPACK.

3. Results

In this section we describe the result obtained. Performance analysis were performed mostly by ICHEC staff on both PRACE Tier-0 systems and PRACE Tier-1 GPU systems. Details are reported in table 1.

Table 1: PRACE supercomputing architectures

Name	Computer	Site	Country	Year
JUGENE	IBM Blue Gene/P Solution	Forschungszentrum Juelich (FZJ)	Germany	2009
PLX	IBM iDataPlex DX360M3, Xeon E5645 6C 2.40 GHz, Infiniband QDR, 528 x NVIDIA Tesla M2070	CINECA	Italy	2011
CURIE	Bullx S6010 Cluster, Xeon 2.26 Ghz 8-core, QDR Infiniband	Commissariat a l'Energie Atomique (CEA)	France	2010
STONEY	Bull Novascale R422-E2, Xeon X5560 2.8 GHz, QDR Infiniband, 48 x NVIDIA Telsa M2090	Irish Centre for High End Computing (ICHEC)	Ireland	2009

3.1. Scalability and performance improvement of QE-GIPAW code and PWscf EXact eXchange (EXX) part

Both the validation and the performance analysis of the new codes were performed using the scientific data-set proposed by Prof. Stefano De Gironcoli and reported in the PRACE-1IP public deliverable D7.2.1 'Interim Report on Collaboration with Communities' [15]. In particular, a cholesterol system with 592 atoms and 600 electronic bands was used. For completeness it must be underlined that restarts are not implemented and it is necessary to reach the convergence to end the simulation. Table 2 shows the results obtained running QE-GIPAW calculation on the CURIE Tier-0 system.

The column “Number of bands” indicates how the processes are gathered respect the *bgrp* MPI communicator level (see previous section). For instance, both the first and the second rows of the table show the elapsed time obtained with the same number of processors (64). Considering the first two rows of the table, the results tell that the explicit band parallelisation does not give any improvement compared to the old version (Number of bands = 1). However, just doubling the number of cores (rows 3 and 4) the effect of new development becomes relevant. In this case the old version of the code, number of bands = 1, delivers a low value of efficiency (58%) already at 128 cores. On the other hand users can now push up to 256 cores to reach the same level of efficiency. At this level of scalability we obtain 52% if comparing with the old version of the code (see row 5) while the value become 57% if comparing the same version of the code using a number of bands equal one. It must be

underlined that while the efficiency of the overall code falls down the Green function keeps scaling quite well, as reported in the last column. Indeed, at this level 1/3 of the overall time is spent to solve the diagonalization problem that does not take advantage from the parallelization over electronic bands. This is the limit point where users are suppose to utilize the other level of parallelism introduced. As described in the previous section the code implements seven independent calculations at the outermost routine. For our experiments we took the limit point at 128 cores (rows 3 and 4). By using this new parallelism and running over 896 cores the code scales with 73% of efficiency respect to 128 cores (93% if compared with the same number of cores while using the old version of the code). In table 2 shows the efficiency in regards to the run at 64 cores, so a lower value is reported.

Although this is the best result reachable in term of efficiency, further scaling with the introduction of hybrid MPI+OpenMP approach shows that is possible to reduce up to 36 minutes the time of response while running at 3584 cores. Thanks to the new development performed in PRACE1IP-T7.2 users can now easily scale at least 14 times the number of cores in regards to the old version of the same code.

Table 2: QE-GIPAW benchmark data

# Cores	# Threads	Number of bands	Elapsed Time (min)	Efficiency %	#Green function (min)
64	1	1	621.15	1.00	425.94
64	1	2	694.66	0.89	441.67
128	1	1	533.33	0.58	358.32
128	1	2	416.69	0.73	242.11
256	1	4	300.79	0.52	137.87
896	1	2	81.59	0.54	58.71
1792	2	2	51.82	0.42	37.60
3584	2	2	36.00	0.31	24.81

The result analysis continues now with the EXX calculation. As described in the previous section the routine Vexx is carrying out almost all the computational workload. In order to test the impact of the new parallelization we simulate a system of 108 atom with 800 bands on the Tier-0 JUGENE machine. Although the low number of atoms, full convergence requires a huge amount of core hours as shown in the table 3, almost 300000 hours were necessary to complete this computation. For the sake of completeness it must be underlined that the energy cutoff used for the fock operator (EXX operator) is its by-definition cutoff, i.e. four times the wave function cutoff.

Table 3: EXX benchmark data

# Cores	Number of bands	Vexx Time(s)	Efficiency %	Elapsed Time(s)	Efficiency %
32768	64	417.18	1.00	523.45	1.00
65536	128	215.96	0.97	311.59	0.84

The routine Vexx, that calculate the Exact-Exchange potential, scales perfectly by doubling the number of cores and the number of band groups. We measure an efficiency of around ~97% moving from 32768 to 65536 cores if we compare the time of execution of the single Vexx routine. However, as presented in table 2 also the global time scales almost perfectly at large number of cores. Indeed, since Vexx is extremely computational intensive in regards to the overall time of execution the code scales efficiently up to 65536 cores. The level of parallelism introduced by the new *elctronic_bands_communicator* definitely impact much more the EXX code than the QE-GIPAW code as in this case most of the computational workload can be parallelized over the electronics bands.

3.2. Result analysis of rVV10 approach

The best result obtained in this section of the project, it is the highly improved accuracy of the rVV10 approach functional with respect to older implementation of vdW functionals. The new approach also keeps the same computational efficiency. Figure 5 shows the deviation from the reference values of the S22 binding energies.

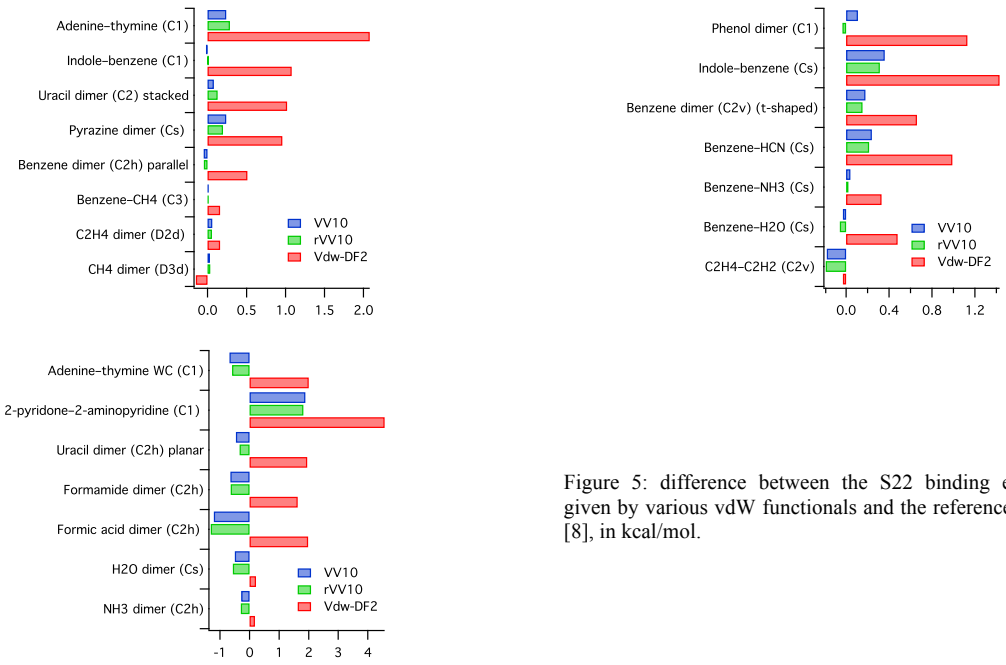

Figure 5: difference between the S22 binding energies given by various vdW functionals and the reference values [8], in kcal/mol.

rVV10 reproduces the VV10 results very closely, which, in turn, show a better accuracy than vdW-DF2 ones (in 20 cases over 22). The argon interaction energy curve (figure 6) is also reported to check the behavior of rVV10 far from equilibrium.

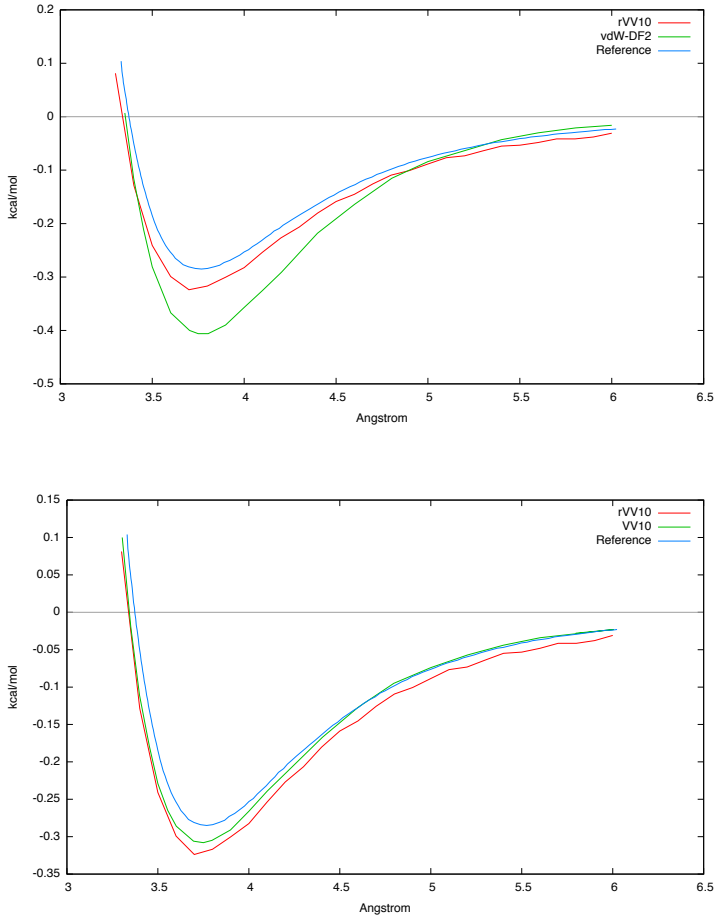

Figure 6: a) agreement of rVV10 with VV10 (above); b) Improvement from vdWDF2

3.3. Acceleration of PWscf code

Performance analysis is a complex task if considering the introduction of accelerators. By looking at profiling data, a typical PWscf execution can be FFT-bounded or diagonalization-bounded. The profiling may change also if considering few SCF iterations rather than the entire SCF loop until convergence or the SCF loop plus the relax step (where also new atomic displacements, forces and stresses are calculated). As it is not possible to select benchmarking data-set to balance equally all the computing footprint factors, we decided to select inputs provided by users community for this purpose. These cases are likely to represent the physical systems currently researched by several research groups worldwide. In the following, we present data underlining the shortest time-to-solution achievable by using heterogeneous system equipped with GPU. It is important to remark that adding GPU capabilities does not directly increase the scalability of the calculation. Scalability improvement by using GPU for large physical simulated systems might come as a side-effect of the increase in work-load of each MPI process. An efficient data distribution (so a good MPI implementation) is still the main responsible of scalability gains (or losses).

Tests for the single node version (serial + OpenMP) have been performed on ICHEC's workstation equipped with 2 six-core Intel Xeon CPU X5650 @ 2.67GHz, 2 NVIDIA GPU C2050 and 24 GByte of RAM. Scaling tests on multiple GPU nodes have been performed using the parallel version of the code on CINECA PLX and ICHEC STONEY systems. Figure 7 compares wall-time and speed-up between single core, single CPU (6 cores) and also using one GPU. Adding one GPU to a single core PWscf achieves 5.54x acceleration for gamma-point and 7.81x for k-point. The acceleration becomes 3.54x and 3.49x respectively if we consider the use of one GPU on top of a single CPU system. For this test we used 'AUSURF112', a medium size input benchmark for PWscf, (a gold surface of 112 atoms).

The distributed version still represents an on-going challenge. Two real scientific data-sets from the user community were analyzed:

- MGST.hex.rx: system composed by 216 atoms of 4 different species (Ge,Mn, Te,Sb), provided by Dr. Wei Zhang (AACHEN-RTWH, Germany) and executed on CINECA PLX
- CdSe-159: system composed by 159 atoms of 2 different species (Cd, Se), provided by Dr. Arrigo Calzolari (CNR-NANO, Italy) and executed on ICHEC STONEY

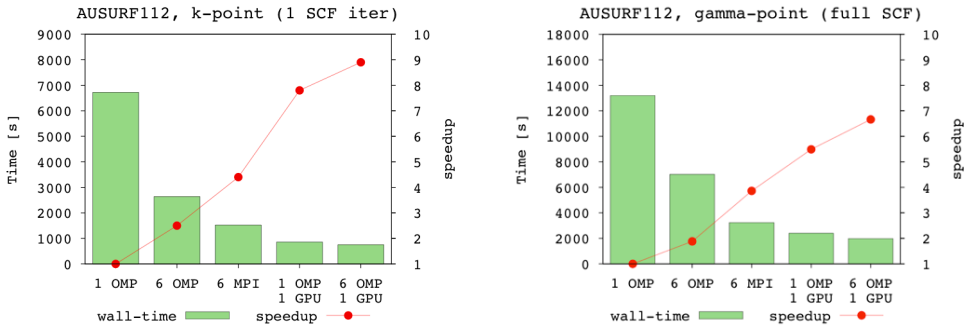

Figure 7: Best results obtained running QE-PWscf on a six-core Intel Xeon X5650 and one Tesla C2050: a) single SCF iteration using K-points; b) full SCF cycle using gamma-point

Early experiments (with parallel 3D-FFT executed on the CPU) based on the 216 atoms data-set were performed scaling up to 32 nodes of CINECA PLX (12 Intel cores and two Tesla M2070 x node), running one SCF cycle. Results show a speed-up of between 2.5x and 3.5x if comparing CPU versus GPU versions of the code running on the same number of CPU nodes, figure 8a. Obviously, in the case of the GPU version GPUs are used on top of the CPU system. Figure 8b compares the time of simulation on a given number of nodes either using GPUs+CPUs or only CPUs. Again, each node include 12 Intel CPU cores and 2 NVIDIA GPU as the elapsed time was registered performing on CINECA GPU Cluster. The use of 144 cores and 24 GPUs together outperforms the 144-CPU-cores-only computation by a factor of 3.3 (8510 Vs. 2510, elapsed time seconds). The same time-to-solution is achievable using 384 cores or more on the same platform.

In case of "CdSe-159", the average acceleration is 2-times by using 1:1 as MPI:GPU ratio and 4 OpenMP per socket (the maximum is 2.52-times with 2 nodes up to 1.9-times with the 14 nodes). By over-subscribing the each GPU card by placing 2 MPI per socket and increasing the MPI:GPU ratio to 2:1 the average acceleration is

even better for low number of nodes (2.96-times using 2 nodes or 2.64-times using 4 nodes) but not for large number of nodes (drop to 1.67-times using 12 nodes or 1.63-times using 14 nodes). Both the test cases presented not scale more than 8/10 nodes. They represent medium size systems with relatively small the memory.

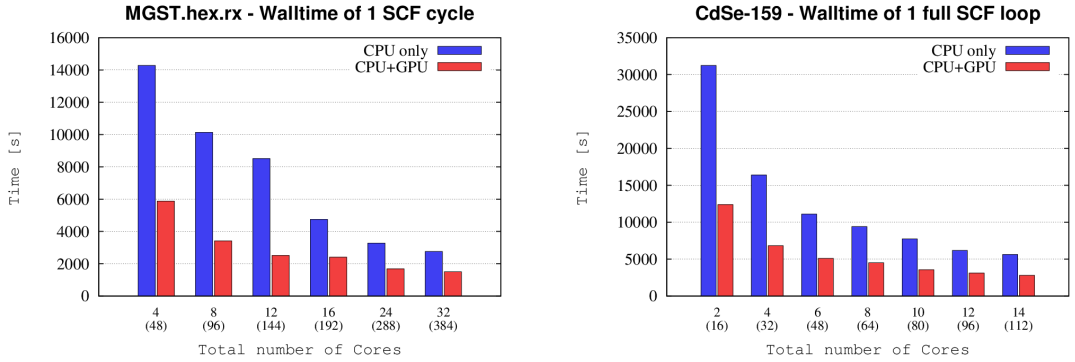

Figure 8: The two graphs show wall-time of parallel MPI+OMP versus MPI+OMP+CUDA. The Y-Axis reports elapsed time of execution while the number of compute node is reported on X-Axis. Experiments were performed on CINECA PLX Cluster where each compute node is equipped with 12 Intel cores and two Tesla M2070. Two different data-sets have been benchmarked: a) data-set provided by Dr. Wei Zhang; b) data-set provided by Dr. Arrigo Calzolari

4. Conclusion

The high complexity of supercoputing infrastructure expose scientific community to largely improve legacy code to efficiently perform large-scale simulation. This concept becomes even more strength if considering the introduction of acceletaros on top of genaral purpose CPU systems. On the other hand the costant upgrade of complex sceintific code requires to both developers and scientists a huge effort, large amount of computational resourses and remarkable experitises in high performance computing.

The results presented in this paper proof that the model of strong collaboration between scientific communities and PRACE partners can overcome these problems by delivering a considerable contribution to tackle such challanges. Concetrating on selected communities a given mount of PRACE resources, we demostrated that it is possible to enable complex scientific code to petascale Tier-0 system.

Other than highly scalable code, as a follow-up of this project, Prof. Stefano De Gironcoli submitted a proposal for PRACE production calls on Tier-0 system that has been accepted with high referee score at the first evaluation step. Final decision is till an on-going process at the time of writing. The GPU enabled version of PWscf has been the subject of many presentation, on-line articles and pulished on an IEEE conference proceeding. At least two manuscripts are on process of writing and they will be shortly submitted.

All the source code developed is available in [14]. Althought the GPU version is still downloadble as a separated *SVN branch* all the work presented was recently released with the new Quantum ESPRESSO 5.0 version.

Acknowledgement

This work was financially supported by the PRACE project funded in part by the EU's 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-261557. We would also like to acknowledge the support from Science Foundation Ireland (grant 08/HEC/I1450). The authors would thank Prof. Paolo Giannozzi for his support to integrate our improvements in the Quantum ESPRESSO suite, as well as Giovanni Erbacci and Maria Francesca Iozzi who have overviewed the overall project.

Reference

[1] P. Giannozzi and et al. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. Journal of Physics: Condensed Matter, 21(39), 2009.

[2] R. G. Parrand and W. Yang. Density Functional Theory of Atoms and Molecules. Oxford University Press, 1989.

[3] R. M. Dreizler and E. K. U. Gross. Density Functional Theory. Springer-Verlag, 1990.

[4] W. E. Pickett. Pseudopotential methods in condensed matter applications. Comput. Phys. Rep., 9, 1989.

[5] Guillermo Roman-Perez and Jose M. Soler. Efficient Implementation of a van der Waals Density Functional: Application to Double-Wall Carbon Nanotubes. Physical Review Let- ters, 103,096102, 2009.

[6] M. Dion, H. Rydberg, E. Schorder, D. C. Langreth and B. I. Lundqvist. Van der Waals Density Functional for General Geometries. Physical Review Letters, Volume 92, Number 24, 2004.

[7] Kyuho Lee, Eamonn D. Murray, Lingzhu Kong, Bengt I. Lundqvist and David C. Langreth. Higher-accuracy van der Waals density functional. Physical Review B, 82, 081101(R), 2010.

[8] Oleg A. Vydrov and Troy Van Voorhis. Non local van der Waals density functional: The simpler the better. The Journal of Chemical Physics, 133,244103, 2010.

[9] A. Dal Corso. A pseudopotential plane waves program (pwscf) and some case studies. Lecture Notes in Chemistry, Vol. 67, C. Pisani editor, Springer Verlag, Berlin, 1996.

[10] S22 test set downloadable from http://www.begdb.com

[11] F. Spiga and I. Girotto, phiGEMM: a CPU-GPU library for porting Quantum ESPRESSO on hybrid systems, Proceeding of 20th Euromicro International Conference on Parallel, Distributed and Network-Based Computing -- Special Session on GPU Computing and Hybrid Computing, IEEE Computer Society, ISBN 978-0-7695-4633-9, pp. 368-375 (2012)

[12] M. Fatica. Accelerating linpack with cuda on heterogenous clusters. In Proceedings of 2nd Workshop on General Pur- pose Processing on Graphics Processing Units. ACM, 2009.

[13] D. Vanderbilt. Soft self-consistent pseudopotentials in a generalized eigenvalue formalism. Phys. Rev. B, 41(11), 1990.

[14] Quantum ESPRESSO project official repository: http://qe-forge.org.

[15] D7.2.1 PRACE-1IP, Interim report on collaboration with communities: http://www.prace-ri.eu/IMG/pdf/d7.2.1_1ip.pdf

[16] D8.1.2 PRACE-2IP, Performance Model of Community Codes: http://www.prace-ri.eu/IMG/pdf/D8-1-2_2IP.pdf

[17] D8.1.3 PRACE-2IP, Prototype Codes Exploring Performance Improvement: http://www.prace-ri.eu/IMG/pdf/d8.1.3_2ip.pdf

[18] Rob Farber, CUDA Application Design and Development, Morgan Kaufmann; 1 edition (November 14, 2011), ISBN-13: 978-0123884268

[19] J Kubler, K -H Hock, J Sticht and A R Williams. Density functional theory of non-collinear magnetism. J. Phys. F: Met. Phys. 18 469, 1998

[i] Distributed European Computing Initiative