Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 240 (2024) 52–60

Procedia Computer Science

www.elsevier.com/locate/procedia

Proceedings of the First EuroHPC user day

Quantum ESPRESSO towards performance portability: GPU offload with OpenMP ☆

Fabrizio Ferrari Ruffino[a], Laura Bellentani[b], Giacomo Rossi[c], Fabio Affinito[b], Stefano Baroni[d], Oscar Baseggio[d], Pietro Delugas[d], Paolo Giannozzi[e,a], Jakub Kurzak[f], Ye Luo[g], Ossian O'Reilly[f], Sergio Orlandini[b], Ivan Carnimeo[d,*]

[a]*CNR-IOM Istituto dell'Officina dei Materiali, area SISSA, via Bonomea 265, 34136 Trieste, Italy*
[b]*CINECA, Via Magnanelli 6/3, 40033 Casalecchio di Reno, BO, Italy*
[c]*Intel Corporation Italia S.p.A., via Santa Maria Valle 3, 20123 Milano, Italy*
[d]*SISSA, Scuola Internazionale Superiore di Studi Avanzati, via Bonomea 265, 34136 Trieste, Italy*
[e]*Dipartimento di Scienze Matematiche, Informatiche e Fisiche (DMIF), Università degli Studi di Udine, via delle Scienze 206, 33100 Udine, Italy*
[f]*Advanced Micro Devices, Inc., 2485 Augustine Drive, Santa Clara, CA 95054, USA*
[g]*Computational Science Division, Argonne National Laboratory, 9700 S Cass Ave, Lemont, IL 60559, USA*

Abstract

In recent years, significant strides have been made to enable Quantum ESPRESSO (QE) to operate on heterogeneous HPC architectures, effectively harnessing the computational capabilities of GPUs. Initially, emphasis has been placed on NVIDIA-based hardware and software infrastructure. Nonetheless, the HPC landscape is multifaceted and intricate, prompting the project to prioritize achieving peak performance across various GPU models. To this end, efforts are being directed towards empowering QE to leverage AMD and Intel GPUs through OpenMP offload directives. In this paper, coding approach and benchmark tests of the OpenMP porting of Plane-Wave Self-Consistent Field (`PWSCF`) code is presented and discussed.

Keywords: GPU; HPC; OpenMP offload; PWSCF; QE; Quantum ESPRESSO

☆ The submitted manuscript has been co-authored by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan. http://energy.gov/downloads/doe-public-access-plan.

* Corresponding author.
E-mail address: icarnimeo@sissa.it

1877-0509
10.1016/j.procs.2024.07.008

1. Introduction

Heterogeneous architectures based on GPU-accelerators have become a *de facto* standard in High Performance Computing (HPC), and the majority of the world's top exascale and pre-exascale machines are now equipped with GPUs [1]. In this context, molecular and material sciences are evolving in parallel with the advances in computer science and most electronic structure codes have either been adapted to accelerated architectures or are in the process of being so. Enabling quantum materials discovery and design towards exascale and extreme scaling performance on present and future HPC machines is one of the pillars of the MaX "MAterials design at the eXascale" Centre of Excellence for Supercomputing applications [2], of which Quantum ESPRESSO [3, 4, 5, 6, 7] stands as one of the lighthouse codes.

A significant challenge hindering this enabling is the varied landscape of hardware and software stacks provided by different vendors. Notably, each vendor offers specific technologies such as compilers, libraries, profilers, and languages, which must be incorporated into the codes to be optimized for a particular architecture. In this complex environment, a primary goal of scientific software development is to enhance the performance portability of the code across different architectures. An overview of performance comparison among different GPU porting approaches on different architectures can be found in the paper by Davis et al. [8].

The first porting of the Quantum ESPRESSO suite dates back to several years ago [4] and was fully based on CUDA Fortran in view of deployment on NVIDIA hardware and software stack. At a later stage [3], the suite was refactored using a mixed CUDA Fortran/OpenACC language model and all the most important codes of the suite (`PWSCF`, `CP` [9], `PHonon` [10, 11], `turbo_eels` [12, 13, 14, 15], `HP` [16, 17, 18]) were accelerated with a performance CPU-GPU speed-up[1] ranging between 4x and 6x for linear response codes, higher for ground-state codes. At the time of writing this paper, the porting of `turbo_davidson` [19], `turbo_lanczos` [20] and `turbo_magnons` [21, 22, 23] codes has been also finalized. Noticeably, the simplicity and minimal intrusiveness nature of OpenACC directives have simplified the code development process, lowered entry barriers for new developers, and facilitated software stack maintenance.

Recently, in order to target a more diverse range of architectures, in particular based on Intel and AMD hardware and software stack, a new porting of the Quantum ESPRESSO suite based on OpenMP offload has been initiated. The OpenMP offload has been developed using directives based on the OpenMP API 5.1 standard [24], and from here onwards will be referred to as "OpenMP5" porting model. Currently, the development of `PWSCF` is about to be completed (branch `develop_omp5` in the official Quantum ESPRESSO repository [25]). The new porting model is designed to seamlessly accommodate both OpenACC and OpenMP5 directives in a single source code, enabling deployment on various hardware architectures (CPU, NVIDIA GPUs, and Intel/AMD GPUs) determined at compile time. This approach aims at achieving consistent performance across different GPU cards at runtime.

In this communication, we provide a brief overview of the porting model and discuss the computational performance and speed-up of `PWSCF`, based on calculation tests conducted on Leonardo (CINECA) and LUMI (CSC) EuroHPC superclusters.

2. Code development

The new OpenACC/OpenMP5 porting model for `PWSCF` is based on one unique source code, where two partially separated execution paths have been defined with directives. The first path is oriented to NVIDIA hardware and software stack and is based on an high level layer of OpenACC directives, an intermediate layer of CUDA Fortran libraries (e.g., FFTXlib, LAXlib and other libraries managing MPI and BLAS/LAPACK wrappers) and a low level layer of vendor's numerical libraries (e.g., cuSOLVER, cuFFT, cuBLAS, cuRAND). The second path is oriented to AMD and Intel hardware and software stack and is fully based on OpenMP5 directives, with the noticeable exception of a small HIP code portion in FFTXlib, that will be discussed in the following. The low level layer of vendor's

[1] The CPU-GPU speed-up was calculated by comparing executions on a CPU-based HPC machine (Galileo100 at CINECA) with GPU-based ones (Marconi100 at CINECA and Selene at Nvidia), utilizing an equal number of nodes for each comparison.

numerical libraries relies especially on oneAPI and ROCm utilities, that provide efficient Fast Fourier Transform (FFT) and linear algebra operations on GPU on AMD and Intel accelerators.

In order to simplify code development and deployment, the OpenACC and OpenMP5 execution paths are selected at compile time and are currently mutually exclusive, being not interfaced with each other (i.e., it is not possible to link oneAPI and ROCm backends when `PWSCF` is compiled with OpenACC, and, vice versa, it is not possible to link CUDA libraries when `PWSCF` is compiled with OpenMP5). Both `configure` and `cmake` compilation tools have been enabled.

A critical point in developing the mixed OpenACC/OpenMP5 porting model was how to tailor execution for CPU, CUF/OpenACC, and OpenMP5 paths, especially when each path necessitates routines specifically designed for a particular architecture. For instance, let us consider the case where distinct specialized algorithms are employed for CPU and GPU executions, and GPU execution is then further customized with a CUDA Fortran (or OpenACC) implementation, tailored for NVIDIA architectures, and an OpenMP5 implementation, for Intel/AMD architectures. Additionally, since variables can be stored in host or device memories, the CPU execution should be kept accessible also when the code is compiled with GPU flags. In the Quantum ESPRESSO suite, this scenario typically occurs, for example, in the FFTXlib library [26], for 3d FFTs, and in the `becmod` module, that contains many numerically intensive procedures with heavy matrix-matrix multiplications.

In Figure 1 very schematic pseudo-codes are represented to illustrate different strategies followed to cope with this issue. In the old purely CUDA Fortran approach [4], Fortran interfaces could recognize the `DEVICE` attribute of variables, invoking the right subroutine procedure accordingly. For instance, in the leftmost block of Figure 1, `v_d` is declared `DEVICE` in `abc_gpu`, and the type of the `arg`/`arg_d` variable in the parent code is sufficient to determine whether to invoke `abc_gpu` or `abc_cpu`. In order to resolve Fortran interfaces with OpenACC and OpenMP5 based models (rightmost block of Figure 1), we have defined three distinct derived types, one for each different execution path (CPU, CUF/OpenACC, OpenMP5). The `off` argument in `abc_cpu`, `abc_acc`, `abc_omp` is then declared with the specific type. Consequently, depending on the type of the `offload` flag in the parent code, a particular execution path is chosen.

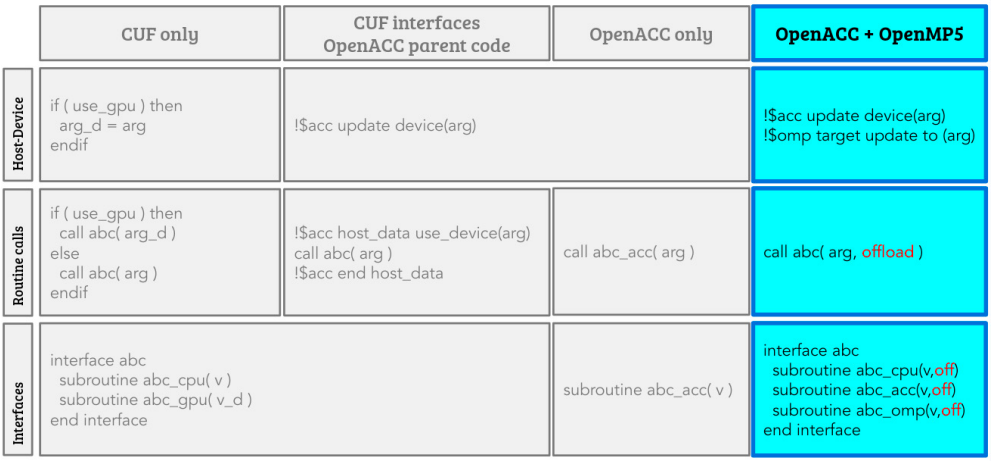

Fig. 1. Schematic representation of different GPU code implementation schemes.

Regarding performance, the majority of the computational cost of a typical `PWSCF` calculation is due to FFTs. Three-dimensional FFTs in the Quantum ESPRESSO suite are executed using FFTXlib [26], a specialized library tailored to accommodate the internal data distribution of the wavefunction (and charge density). It leverages the properties of DFT-related datasets, such as band structure, cutoff, and dual parameter and supports both slab and pencil decompositions. The former method involves a slab-based partition of the direct space, where the input function undergoes transformation across the entire x-y domain for a subset of values along the z-axis. Each subset is allocated to a specific processor, and the Fourier transform along the z-direction is carried out based on the 'z-stick' distribution of the reciprocal space: for each x-y point falling within the cutoff circle, the function is transformed across the entire

z-range. This algorithm entails one 2d FFT operation for x-y slab transforms, one stage of collective communication (typically MPI all to all), which involves distributing the z-sticks among processors, and one final 1d FFT operation on the z-sticks. The uniqueness of the QUANTUM ESPRESSO version of this algorithm lies in the z-stick decomposition of the reciprocal space and the utilization of the energy cutoff. This mapping of the square grid covering the direct space into a smaller one inscribed within the cutoff sphere allows for memory and data movement optimizations.

Pencil decomposition works similarly to slab decomposition, but transforms along the x and y directions separately, involving only 1d FFT operations. While it enables more efficient memory distribution, it requires an additional stage of communication compared to slab decomposition. Consequently, with the increasing memory availability of modern GPU devices, slab decomposition has become the preferred method in recent years.

Figure 2a provides a schematic overview of the main steps of an inverse FFT computation, performed with slab decomposition on 4 MPI processes: even on small systems, the time spent in communication and data movement significantly exceeds that related to actual 1d and 2d FFT computations.

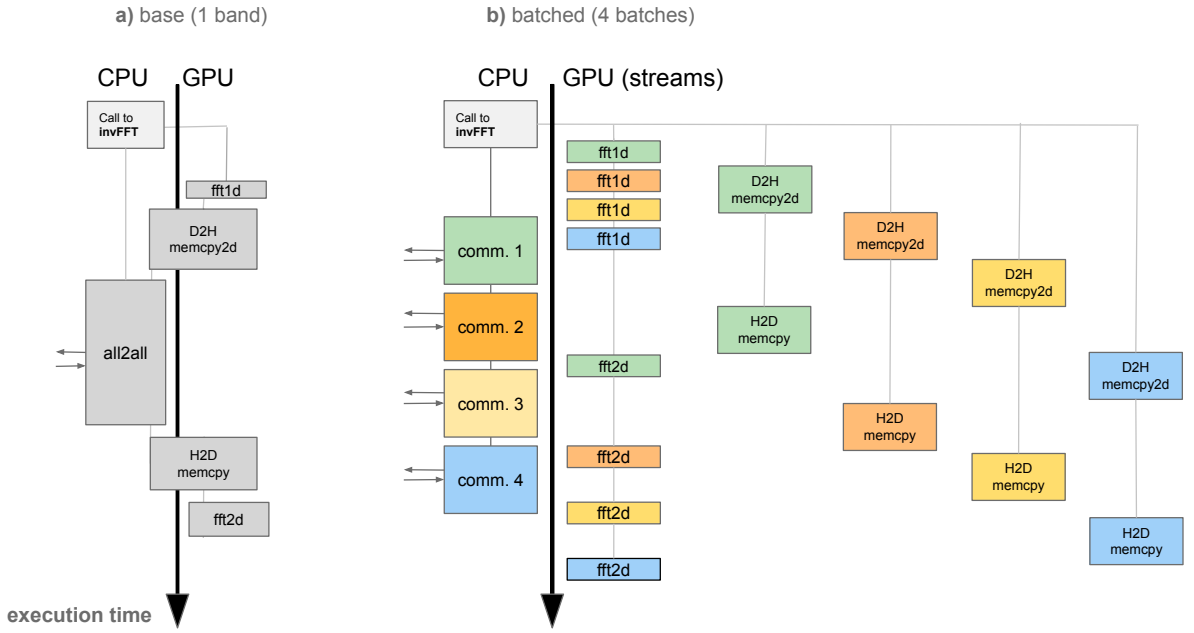

Fig. 2. Illustrative scheme of FFT with slab decomposition: a) base accelerated inverse FFT algorithm; b) asynchronous batched inverse FFT algorithm with 5 streams and 4 sub-batches (one for each color). The main data movements are done with memcpy2d and memcpy routines. MPI communications without GPU-aware MPI are performed among CPUs and require host-device synchronizations, whereas when GPU-aware MPI is enabled they involve GPU-direct device-device communications. The time scale of the two diagrams is not the same.

The GPU porting of FFTs for NVIDIA-based architectures [4, 3] leveraged the extensive internal parallelism of GPU cards to concurrently process many bands at a time within each FFT operation. This was achieved by segmenting the wavefunction data structures into batches and sub-batches, each containing a fixed number of bands, and using streams to process sub-batches asynchronously. This algorithm, initially developed using CUDA Fortran [4], has not undergone rewriting in OpenACC during the recent refactoring [3]. Instead, it has been expanded to support AMD-based architectures using the HIP language, since OpenMP5 does not allow to work on multiple GPU streams within a single CPU task. For Intel architectures, FFTXlib has been currently ported using the base algorithm shown in Figure 2a, and work is currently underway to include the asynchronous streams.

In the batched algorithm, schematically summarized in Figure 2b, all the 1d and 2d FFTs, needed to fully Fourier transform a batch, are performed on one dedicated stream (usually stream 0), while data movement and copies are performed on different streams, one for each sub-batch. Other small operations are performed on stream 0 too, by

means of explicit HIP kernels. MPI communications are called for each sub-batch asynchronously with respect to the computation and the data movements related to all the other sub-batches. In this way, the overlap in time between communication and computation is brought out at its maximum, taking into account the constraints coming from the finite bandwidth of CPU and GPU communications.

Work is currently still in progress to finalize the OpenMP5 porting, in particular to optimize eigensolver operations.

3. Numerical results

The developments described in the previous section allowed to significantly accelerate FFTs and basic matrix operations in `PWSCF` on Intel and AMD GPU-based machines. Such operations are especially relevant to compute the application of the Kohn-Sham Hamiltonian to a generic function (`h_psi` subroutine in `PWSCF`), that is the core of the iterative methods used in plane-wave based electronic structure codes. For example, in the CPU execution of `PWSCF` shown in Figure 3, `h_psi` took around 75% of one SCF interation. In this section we will discuss numerical performance of `PWSCF`, comparing CPU and GPU executions on Galileo100, Leonardo and LUMI machines, whose main technical features are briefly summarized in Table 1.

HPC cluster	Galileo100	Leonardo	LUMI
Processors	2 x Intel Xeon E5-2697	Intel Ice Lake	AMD EPYC™ 7A53
Cores	36	32	64
RAM	128 GB	512 GB DDR4	512 GiB DDR4
Accelerators	-	4 x NVIDIA A100 GPUs with 64 GB HBM2	4 x AMD Instinct™ MI250X GPU modules with 2 x 64 GB of HBM memory
Intra-node links	-	NVLink 3.0 (200 GB/s)	In-package Infinity Fabric™ (400 GB/s), and single/double Infinity Fabric (100/200 GB/s)
Network	Intel OmniPath, 100 Gb/s	DragonFly+ 200 Gbps of 2x dual port NVIDIA Mellanox Infiniband HDR100	PCIe links to the Slingshot-11 (200 GB/s)

Table 1. Systems specification for the CPU (Galileo100) and GPU (Leonardo and LUMI) partitions

Calculations on CPU have been performed on Galileo100 with the official release `qe-v7.2` [27], compiled with Intel Ifort 2021.5.0, Intel MPI 2021.5 and Intel MKL version 2022.0.0; GPU runs have been performed on Leonardo and LUMI machines. Simulations on Leonardo have been done with the official release `qe-v7.3.1` [28], compiled with NVHPC 23.1 and CUDA 11.8. On LUMI we employed the development version of `PWSCF` in the `develop_omp5` branch [25], based on the `qe-v7.2` release, and available on the official Quantum ESPRESSO repository on GitLab. The code has been compiled with CRAY HPE 15.0.1 and ROCm 5.2.5.

Figure 3 shows that the performance of the OpenMP5 version of `h_psi` on LUMI is equivalent to that of the corresponding OpenACC version on Leonardo. The test case is a reduced version of the CrI_3 system previously benchmarked in Ref.[3], with an orthorhombic cell of 480 atoms, for a total of 3240 electrons, with 1944 bands and a per-band FFT grid of dimension `(120,192,640)`, which make it a good candidate to assess the effectiveness of the FFT parallelization via *R&G* distribution and band-batching. The executions on LUMI are performed by progressively enabling improvements of the FFT algorithm with respect to the "base" one (cf. Figure 2a). Calculations labeled as "many" use the batched FFT algorithm (cf. Figure 2b); GPU-MPI awareness is enabled for the "many aware" results. The CPU-GPU speed-up of the last two bars is also in line with other calculations published in another work [3], using the same machine (Galileo100) as CPU reference.

In Figure 4 the scaling over plane waves (*R&G*) on LUMI, at fixed number of pools, is shown for the same system, and it is noteworthy that using the batched FFT algorithm and GPU-aware MPI it is possible to decrease communication bottlenecks and scale the calculations beyond 2 nodes.

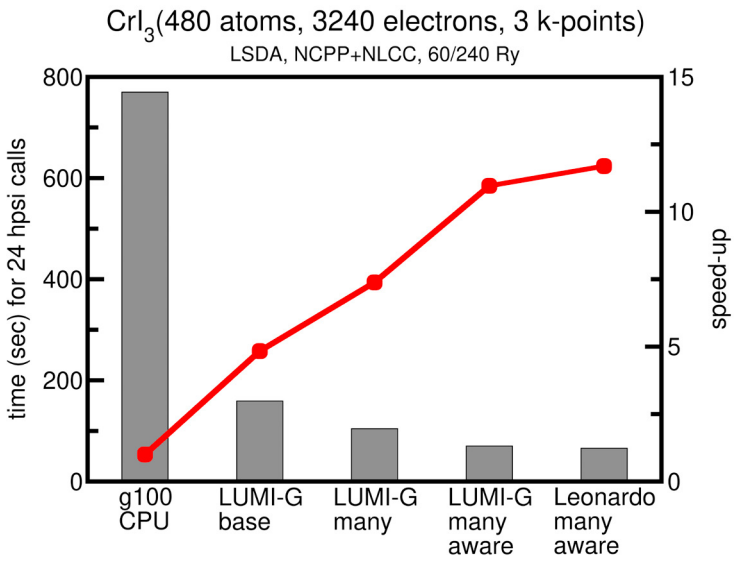

Fig. 3. Wall time of `h_psi` calls in one SCF iteration of CrI_3 system, performed on different HPC clusters using 2 nodes. Speed-up on the right axis is computed with respect to CPU time. "many" refers to the batched FFT algorithm, "aware" refers to GPU-aware MPI, "base" is without neither "many" nor "aware".

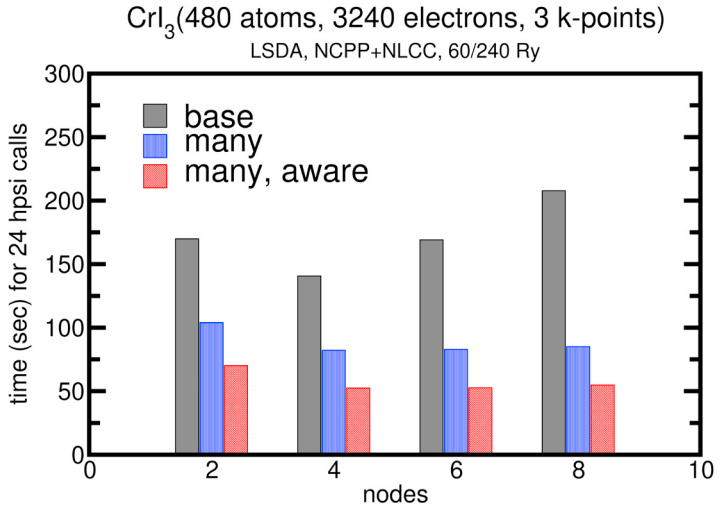

Fig. 4. *R&G* scaling of `h_psi` over the number of nodes in LUMI cluster. "many" refers to the batched FFT algorithm, "aware" refers to GPU-aware MPI, "base" is without neither "many" nor "aware".

In Figure 5 the performance of `h_psi` with respect to the number of pools (independent groups of k-points distributed among MPI processes) is shown, using one pool per node, for the CsI test case[29]. This system has a cubic CsCl crystal structure, with 768 electrons, 461 bands and a per-band FFT grid of dimension `(180,180,135)`. Its limited memory footprint allows distributing the workload with R&G parallelization intra-node only, while, thanks to the large number of available k-points, the system can scale with pools up to 89 nodes.

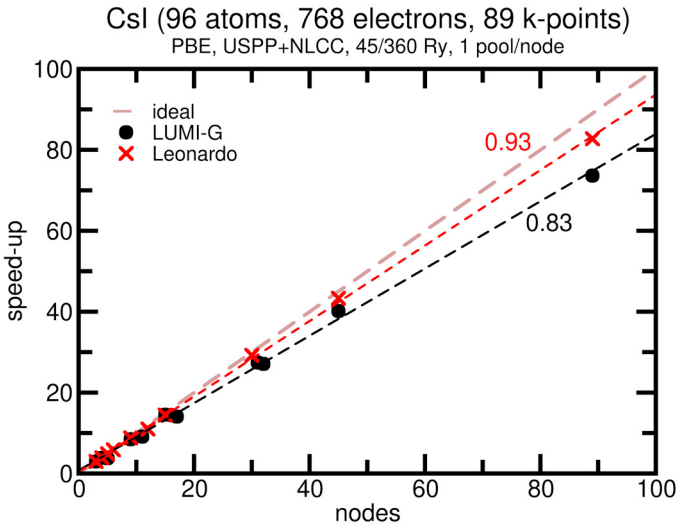

Fig. 5. Speed-up of pool scaling of `h_psi` in LUMI and Leonardo cluster (using one pool per node). Slope of the linear regression is reported for each dataset.

The scaling shown here is in line with the speed-up of pool parallelism shown in another previous work [3], and the slope of the regression is reasonably similar between Leonardo and LUMI executions.

4. Conclusions

The main numerically intensive parts of `PWSCF`, in particular FFT and basic linear algebra operations such as matrix multiplications, have been ported to GPU using OpenMP5, in order to address Intel and AMD cards and software stack. OpenMP5 directives have been seamlessly integrated in the previous CUF/OpenACC code, resulting in a unique source code that can be tailored at compile time for execution on CPU, NVIDIA and Intel/AMD hardware and software stack. Performance portability of `h_psi`, one of the most computationally heavy steps of the SCF calculation (i.e., the application of the Kohn-Sham Hamiltonian to a generic function), has been tested with calculations on Galileo100, LUMI and Leonardo clusters. In such tests, very similar computational times and speed-ups have been found between the two GPU execution paths of `PWSCF` (CUF/OpenACC and OpenMP5), both for *R&G* and pools parallelism.

The porting of other parts of `PWSCF` with OpenMP5, particularly with regard to the eigensolver, is still underway. When using the accelerated version of `PWSCF`, the main bottleneck of OpenMP5 executions on LUMI remains the eigensolver, and overcoming this step is crucial to approaching performance portability between the OpenACC and OpenMP5 versions of the code, although absolute parity may be limited by intrinsic differences in hardware architectures.

Concerning Intel GPUs, an asynchronous algorithm for batched FFTs fully based on OpenMP offload is currently under development. Since explicit streams are not supported in OpenMP, it is based on `nowait` and `depend` clauses, and the batch related workload is distributed among OpenMP tasks. GPU streams associated to each sub-batch are implicitly defined according to each task.

Acknowledgements

This work was funded by the European Union through the MaX ”MAterials design at the eXascale” Centre of Excellence for Supercomputing applications (Grant agreement No. 101093374, co-funded by the European High Performance Computing joint Undertaking (JU) and participating countries) and by the Italian MUR, through the PRIN

project ARES (grant number 2022W2BPCK) and the Italian National Centre for HPC, Big Data, and Quantum Computing (grant No. CN00000013), founded within the Next Generation EU initiative. The authors acknowledge EuroHPC JU Development Access Call for granting computational resources on LUMI cluster (EHPC-DEV-2023D06-013 and EHPC-DEV-2021D04-129 projects). The authors IC, FFR, LB, OB acknowledge CSC for technical support received at the LUMI GPU / Nomad CoE hackathon Sept 4-6, 2023. YL was supported by the Office of Science, U.S. Department of Energy, under contract DE-AC02-06CH11357.

References

[1] Top 500 list of supercomputers. https://www.top500.org (last access Apr, 16th 2024).

[2] MaX: Materials at the eXascale. An EU Centre of Excellence for Supercomputing Applications. https://www.max-centre.eu (last access Apr, 16th 2024).

[3] Ivan Carnimeo, Fabio Affinito, Stefano Baroni, Oscar Baseggio, Laura Bellentani, Riccardo Bertossa, Pietro Davide Delugas, Fabrizio Ferrari Ruffino, Sergio Orlandini, Filippo Spiga, and Paolo Giannozzi. Quantum ESPRESSO: One further step toward the exascale. *Journal of Chemical Theory and Computation*, 19(20):6992–7006, 2023. PMID: 37523670.

[4] Paolo Giannozzi, Oscar Baseggio, Pietro Bonfà, Davide Brunato, Roberto Car, Ivan Carnimeo, Carlo Cavazzoni, Stefano de Gironcoli, Pietro Delugas, Fabrizio Ferrari Ruffino, Andrea Ferretti, Nicola Marzari, Iurii Timrov, Andrea Urru, and Stefano Baroni. Quantum ESPRESSO toward the exascale. *J. Chem. Phys.*, 152(15):1–11, 04 2020. 154105.

[5] P. Giannozzi, O. Andreussi, T. Brumme, O. Bunau, M. Buongiorno Nardelli, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, M. Cococcioni, N. Colonna, I. Carnimeo, A. Dal Corso, S. de Gironcoli, P. Delugas, R. A. DiStasio, Jr., A. Ferretti, A. Floris, G. Fratesi, G. Fugallo, R. Gebauer, U. Gerstmann, F. Giustino, T. Gorni, J. Jia, M. Kawamura, H-Y Ko, A. Kokalj, E. Kucukbenli, M. Lazzeri, M. Marsili, N. Marzari, F. Mauri, N. L. Nguyen, H-V Nguyen, A. Otero-de-la Roza, L. Paulatto, S. Ponce, D. Rocca, R. Sabatini, B. Santra, M. Schlipf, A. P. Seitsonen, A. Smogunov, I. Timrov, T. Thonhauser, P. Umari, N. Vast, X. Wu, and S. Baroni. Advanced capabilities for materials modelling with Quantum ESPRESSO. *J. Phys. Condens. Matter*, 29(46):1–30, 2017. 465901.

[6] Paolo Giannozzi, Stefano Baroni, Nicola Bonini, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Gui do L Chiarotti, Matteo Cococcioni, Ismaila Dabo, Andrea Dal Corso, Stefano de Gironcoli, Stefano Fabris, Guido Fratesi, Ralph Gebauer, Uwe Gerstmann, Christos Gougoussis, Anton Kokalj, Michele Lazzeri, Layla Martin-Samos, Nicola Marzari, Francesco Mauri, Riccardo Mazzarello, Stefano Paolini, Alfredo Pasquarello, Lorenzo Paulatto, Carlo Sbraccia, Sandro Scandolo, Gabriele Sclauzero, Ari P Seitsonen, Alexander Smogunov, Paolo Umari, and Renata M Wentzcovitch. Quantum ESPRESSO: a modular and open-source software project for quantum simulations of materials. *J. Phys. Condens. Matter*, 21(39):1–19, sep 2009. 395502.

[7] Sandro Scandolo, Paolo Giannozzi, Carlo Cavazzoni, Stefano de Gironcoli, Alfredo Pasquarello, and Stefano Baroni. First-principles codes for computational crystallography in the Quantum ESPRESSO package. *Zeitschrift für Kristallographie*, 220(5-6):574–579, 2005.

[8] Joshua H. Davis, Pranav Sivaraman, Joy Kitson, Konstantinos Parasyris, Harshitha Menon, Isaac Minn, Giorgis Georgakoudis, and Abhinav Bhatele. Taking GPU programming models to task for performance portability, 2024.

[9] P. Giannozzi, F. De Angelis, and R. Car. First-principle molecular dynamics with ultrasoft pseudopotentials: Parallel implementation and application to extended bioinorganic systems. *The Journal of Chemical Physics*, 120(13):5903–5915, 04 2004.

[10] S. Baroni, S. de Gironcoli, A. Dal Corso, and P. Giannozzi. Phonons and related crystal properties from density-functional perturbation theory. *Rev. Mod. Phys.*, 73(2):515–562, 2001.

[11] P. Giannozzi, S. De Gironcoli, P. Pavone, and S. Baroni. Ab-initio calculation of phonon dispersions in semiconductors. *Phys. Rev. B*, 43(9):7231–7242, 1991.

[12] Iurii Timrov, Nathalie Vast, Ralph Gebauer, and Stefano Baroni. Electron energy loss and inelastic x-ray scattering cross sections from time-dependent density-functional perturbation theory. *Phys. Rev. B*, 88:064301, Aug 2013.

[13] Iurii Timrov, Nathalie Vast, Ralph Gebauer, and Stefano Baroni. turboeels-a code for the simulation of the electron energy loss and inelastic x-ray scattering spectra using the Liouville-Lanczos approach to time-dependent density-functional perturbation theory. *Comp. Phys. Comm.*, 196:460–469, 2015.

[14] Oleksandr Motornyi, Michele Raynaud, Andrea Dal Corso, and Nathalie Vast. Simulation of electron energy loss spectra with the turboeels and thermo_pw codes. In *XXIXTH IUPAP Conference on Computational Physics CCP2017*, volume 1136 of *J. Phys. Conf. Ser.* IUPAP, 2018. 39th IUPAP Conference on Computational Physics (CCP), Univ Pierre Marie Curie Sorbonne, Paris, France, JUL 09-13, 2017.

[15] Oleksandr Motornyi, Nathalie Vast, Iurii Timrov, Oscar Baseggio, Stefano Baroni, and Andrea Dal Corso. Electron energy loss spectroscopy of bulk gold with ultrasoft pseudopotentials and the Liouville-Lanczos method. *Phys. Rev. B*, 102:1–12, 2020. 035156.

[16] Iurii Timrov, Nicola Marzari, and Matteo Cococcioni. HP – A code for the calculation of Hubbard parameters using density-functional perturbation theory. *Comp. Phys. Comm.*, 279:1–17, 2022. 108455.

[17] Iurii Timrov, Nicola Marzari, and Matteo Cococcioni. Hubbard parameters from density-functional perturbation theory. *Phys. Rev. B*, 98:1–15, Aug 2018. 085127.

[18] Iurii Timrov, Nicola Marzari, and Matteo Cococcioni. Self-consistent Hubbard parameters from density-functional perturbation theory in the ultrasoft and projector-augmented wave formulations. *Phys. Rev. B*, 103:1–14, Jan 2021. 045141.

[19] Brent Walker, A. Marco Saitta, Ralph Gebauer, and Stefano Baroni. Efficient approach to time-dependent density-functional perturbation theory for optical spectroscopy. *Phys. Rev. Lett.*, 96:1–4, Mar 2006. 113001.

[20] Dario Rocca, Ralph Gebauer, Yousef Saad, and Stefano Baroni. Turbo charging time-dependent density-functional theory with Lanczos chains. *J. Chem. Phys.*, 128(15):1–14, 2008. 154105.

[21] Tommaso Gorni, Oscar Baseggio, Pietro Delugas, Stefano Baroni, and Iurii Timrov. Turbomagnon – A code for the simulation of spin-wave spectra using the Liouville-Lanczos approach to time-dependent density-functional perturbation theory. *Comp. Phys. Comm.*, 280:1–15, 2022. 108500.
[22] Tommaso Gorni, Oscar Baseggio, Pietro Delugas, Iurii Timrov, and Stefano Baroni. First-principles study of the gap in the spin excitation spectrum of the CrI_3 honeycomb ferromagnet, 2022.
[23] Pietro Delugas, Oscar Baseggio, Iurii Timrov, Stefano Baroni, and Tommaso Gorni. Magnon-phonon interactions enhance the gap at the Dirac point in the spin-wave spectra of CrI_3 2D magnets, 2021.
[24] OpenMP Architecture Review Board. OpenMP application program interface version 5.1, 2020.
[25] Quantum ESPRESSO development branch `develop_omp5`. https://gitlab.com/QEF/q-e/-/tags/qe-7.2-omp5-1.0 (last access Apr, 16th 2024).
[26] Michael Wagner, Victor Lopez, Julian Morillo, Carlo Cavazzoni, Fabio Affinito, Judit Gimenez, and Jesus Labarta. Performance analysis and optimization of the fftxlib on the intel knights landing architecture. In *2017 46TH International Conference on Parallel Processing Workshops (ICPPW)*, International Conference on Parallel Processing Workshops, pages 243–250. CRAY Supercomp Company; Intel; ARM; OCF; SGI; Univ Bristol; VirginiaTech; Int Assoc Comp & Communications, 2017. 46th International Conference on Parallel Processing Workshops (ICPPW), Bristol, England, AUG 14-17, 2017.
[27] Quantum ESPRESSO release 7.2. https://gitlab.com/QEF/q-e/-/tags/qe-7.2 (last access Apr, 16th 2024).
[28] Quantum ESPRESSO release 7.3.1. https://gitlab.com/QEF/q-e/-/tags/qe-7.3.1 (last access Apr, 16th 2024).
[29] Procurement repository of CINECA. https://gitlab.hpc.cineca.it/procurement/tier1-tecnopolo/benchmarks/-/blob/main/QuantumESPRESSO/LEONARDO-datacentric/inputfiles/csi.in (last access Apr, 16th 2024).