IL NUOVO CIMENTO Vol. 32 C, N. 2 Marzo-Aprile 2009
DOI 10.1393/ncc/i2009-10368-9

Colloquia: CSFI 2008

Large-scale computing with Quantum ESPRESSO

P. Giannozzi([1])([2]) and C. Cavazzoni([3])

([1]) *CNR-INFM DEMOCRITOS National Simulation Center - 34100 Trieste, Italy*
([2]) *Dipartimento di Fisica, Università di Udine - Via delle Scienze 208, I-33100 Udine, Italy*
([3]) *CINECA - via Magnanelli 6/3, 40033 Casalecchio di Reno, Bologna, Italy*

(ricevuto il 15 Maggio 2009; pubblicato online il 3 Agosto 2009)

Summary. — This paper gives a short introduction to Quantum ESPRESSO: a distribution of software for atomistic simulations in condensed-matter physics, chemical physics, materials science, and to its usage in large-scale parallel computing.

PACS 01.50.hv – Computer software and software reviews.
PACS 31.15.es – Applications of density-functional theory.

1. – Introduction

Quantum ESPRESSO (QE) starts in 2002 as a DEMOCRITOS initiative, in collaboration with SISSA, CINECA and with research groups in Princeton University, MIT, EPF Lausanne [1]. The name "ESPRESSO" stands for *opEn Source Package for Research in Electronic Structure, Simulation, and Optimization*, while "Quantum" stresses its scope: first-principle (*i.e.* based on the electronic structure) calculations within Density-Functional Theory (DFT) in a plane-wave (PW) pseudopotential (PP) approach. Building upon pre-existing codes, QE is evolving into a *distribution*, open to external contributions: QE is released under the terms of the General Public License (GPL).

Different categories of scientists may find QE useful for their research work:

– Those interested in methodological and algorithmic improvements who need a way to implement new developments and quickly reach the scientific community, without the hassle of maintaining "home-made" special-purpose codes.

– Those interested in applications rather than in development who need cutting-edge software tools, without the hassle of maintaining their own software.

– The increasing number of non-specialists (*e.g.*, experimentalists) who want to use numerical simulations to better understand their results; they need a set of robust and easy-to-use software tools that do not require a long training.

QE was conceived as a service to the scientific community at large, including non-specialists, and has the ambition to satisfy all of the above-mentioned categories of scientists. This imposes conflicting requirements: the software has to be powerful, fast and highly optimized, but at the same time easy to use and highly portable; it is thus necessary complex, but it has to be at the same time easy to understand and to modify, presenting a low barrier to new developers. QE offer no "ideological" ultimate answer, but a practical approach, inspired by experience:

- QE is a *distribution*, to which more packages can be added, without conforming to strict "coding rules" that represent a high barrier to potential contributors.
- Non-monolithic approach: QE is not a single code that does everything, but a series of smaller codes, each performing a well-specified task or group of tasks, lightly integrated and communicating between them with *data directories*. QE uses a hybrid format for data exchange in which a formatted XML-like file coexists with files containing large binary records in a data directory.
- Language. Fortran-95 was chosen because it provides an easy transition path from Fortran77, powerful and useful array constructs, a smooth path to modern programming paradigms (encapsulation, object-based programming).

The core components of QE are `PWscf` [2] and `CP` [3, 4]. The former performs self-consistent calculations (including structural optimization and molecular dynamics on the Born-Oppenheimer surface) in crystals, while the latter performs Car-Parrinello Molecular Dynamics (CP-MD) in cells with Periodic Boundary Conditions. Other important and well-established components are: `PHonon`, a set of codes for linear-response calculations; `PostProc`, utilities for visualization and data postprocessing; `atomic`, for pseudopotential generation and testing; `PWcond`, for ballistic conductance calculations.

More recent additions and extensions include: `GIPAW`, for chemical shifts and EPR factor calculations; `Wannier90`, Wannier-function package; `XSPECTRA`, calculating X-ray spectra. For further and more updated information on the capabilities of QE, we refer to the main page of the QE wiki: `http://www.quantum-espresso.org/wiki/index.php/Main_Page`.

2. – Parallel computing with Quantum ESPRESSO

QE is parallelized using the Message-Passing paradigm, via calls to standard MPI (Message Passing Interface) library routines. Five parallelization levels are present, allowing some form of effective execution on all kinds of parallel machines. The different levels are organized as a *hierarchy* of processor groups, identified by different MPI communicators. In this hierarchy, groups implementing coarser-grained parallelizations are split into groups implementing finer-grained parallelizations. Table I contains a summary of the five levels. In the following we give a short description of each of them.

Image parallelization. Implemented by dividing processors into n_{image} groups, each taking care of one or more images, *i.e.* a point in the configuration space, used by the NEB (Nudged Elastic Band) method.

Pool parallelization. Implemented by further dividing each group of processors into n_{pool} pools of processors, each taking care of one or more **k**-points.

In both cases, good scalability of CPU time (but no scalability for RAM) can be achieved with a modest amount of communication among processors. These two approaches are well suited for cheap hardware (*e.g.*, PC clusters with Gigabit Ethernet).

TABLE I. – *Summary of parallelization levels in QE.*

Group	Distributed quantities	Communications	Performances
image	NEB images	very low	linear CPU scaling, fair to good load balancing; does not distribute RAM
pool	**k**-points	low	almost linear CPU scaling, fair to good load balancing; does not distribute RAM
plane-wave	PW, **G**-vector coefficients, **R**-space FFT arrays	high	good CPU scaling, good load balancing, distributes most RAM
task	FFT on electron states	high	improves load balancing
linear algebra	subspace Hamiltonians and constraints matrices	very high	improves scaling, distributes more RAM

They are useful however only for calculations using more than one **k**-point, or for NEB or similar calculations, thus leaving out important cases like typical CP-MD simulations.

Plane-wave parallelization. Performed over **r**- and **G**-space grids. The data: columns of PW coefficients in **G**-space, planes of real-space values in **r**-space, is distributed across the n_{PW} processors of each pool. A parallel 3-dimensional Fast Fourier Transform (FFT) is performed over such distributed data. Our implementation [3,4] covers the case of multiple **G**-space and **r**-space grids, required by modern (ultrasoft) PP. Very good scalability for both CPU and RAM can be achieved, but communication is heavy: fast communication hardware is needed for good performances.

The three well-established parallelization levels above described allow good CPU and memory scalability up to several tens of processors (the total number of MPI processes is $N = n_{\mathrm{image}} \times n_{\mathrm{pool}} \times n_{\mathrm{PW}}$) for systems including several tens to hundreds atoms. In order to effectively solve larger systems (order of 500 and more atoms) on massively parallel machines, two further parallelization levels have been recently added.

Task-group parallelization. When the number of processors in a pool, n_{PW}, exceeds the number N_z of planes in **r**-space for electron states, plane-wave parallelization does not scale well any longer. The solution is to subdivide each pool into n_{task} task groups [5]. Each task group, composed of $n_{\mathrm{FFT}} = n_{\mathrm{PW}}/n_{\mathrm{task}}$ processors, takes care of different groups of electron states to be Fourier-transformed, while each FFT is parallelized inside a task group. Ideally, $n_{\mathrm{FFT}} = N_z/k$ where k is an integer; typically, an optimal choice for n_{task} yields $k = 2$–8.

Linear-algebra parallelization. Subspace diagonalization (`PWscf`) or iterative orthonormalization (`CP`) require linear algebra operations (diagonalizations and multiplications) on square $N_b \times N_b$ matrices, where N_b is the number of electron states (or a small multiple of it). For large systems ($N_b \sim$ several thousands) this becomes a serious bottleneck, so both CPU and RAM have to be distributed. The *Linear algebra* group is a subset of each pool, forming a square grid of $n_{\mathrm{diag}}^2 \leq n_{\mathrm{PW}}$ processors. All involved matrices are distributed across this grid; all matrix operations are performed in parallel, using Cannon's parallel matrix-matrix multiplication and SCALAPACK (or custom) parallel diagonalization routines. The choice of a square grid is natural since all involved matrices are square and this grid yields optimal performances. If $n_{\mathrm{diag}}^2 \neq n_{\mathrm{PW}}$, some processors in the pool will be idle. This is not really a waste of resources because the

TABLE II. – *CPU time* (s) *per electronic time step (*CP *code) on an IBM Blue Gene/P (BG/P) and a SGI Altix for a fragment of an Aβ-peptide in water containing 838 atoms and 2311 electrons in a* $22.1 \times 22.9 \times 19.9\,\text{Å}^3$ *cell, as a function of* $n_{\text{FFT}} \times n_{\text{task}}$.

Machine	32×1	64×1	128×1	32×4	256×1	32×8	64×4	64×8
BG/P	321.7	162.6	92.1	65.9	57.7	42.3	41.2	28.4
Altix	118.7	63.0	38.9	31.0	31.4	21.3	23.1	

linear-algebra group operates only on linear-algebra operations, whereas the computation of the matrix elements—a much more demanding task—is performed in parallel on all processors. Moreover, the optimal number of processors to be used for the linear algebra group depends upon the size of the matrices, and it can be smaller (even much smaller) than the number of processors in each pool.

The different processor groups described above are identified by a hierarchy of MPI communicators. The *world* communicator include all processors and is split into n_{image} image communicators, which in turn are split into into n_{pool} pool communicators. Each pool communicator is finally split into n_{task} task communicators, while at the same time, a suitable subset of the processors of the pool is included in the linear algebra communicator. Task groups and linear algebra group are both obtained from the same pool but they are independent: one is not obtained from the other.

With a judicious usage of the available parallelization levels, simulations on systems containing several hundreds of atoms have become quite standard (see table II for an example). Excellent scalability on up to 4800 processors has been demonstrated in a system of 1500 atoms, even in a case where coarse-grained parallelization does not help.

Parallelism in QE is evolving to keep the pace with current technological trends in the high-performance and technical computing world. The push towards the increase in the number of cores inside a single CPU is likely to continue and CPU with up to 12-24 cores will be common in the near future. It will be extremely important to distribute the work inside a CPU in an optimal way in order to take advantage of the multicore architecture. MPI does not seem able to efficiently manage the load both among CPUs and inside a single CPU. For this reason QE developers are now working to mix MPI with a multi-threads paradigm (OpenMP) to manage computation inside a CPU and leave MPI to manage communications between different CPUs. A version of QE that supports a mixed MPI-OpenMP parallelism will be available during 2009.

* * *

We acknowledge contributions to Quantum ESPRESSO from more people than we can mention here. We thank V. MINICOZZI for providing the Aβ-peptide test.

REFERENCES

[1] SCANDOLO S., GIANNOZZI P., CAVAZZONI C., DE GIRONCOLI S., PASQUARELLO A. and BARONI S., *Z. Kristallogr.*, **220** (2005) 574.
[2] DAL CORSO A., *A Pseudopotential Plane Waves Program (PWscf) and some case studies*, in *Quantum-Mechanical Ab-initio Calculation of the Properties of Crystalline Materials*, *Lect. Notes Chem.*, edited by PISANI C., Vol. **67** (Springer, Berlin) 1996.
[3] CAVAZZONI C. and CHIAROTTI G., *Comput. Phys. Commun.*, **123** (1999) 56.
[4] GIANNOZZI P., DE ANGELIS F. and CAR R., *J. Chem. Phys.*, **120** (2005) 5903.
[5] HUTTER J. and CURIONI A., *Chem. Phys. Chem.*, **6** (2005) 1788.