QE, main strategies of parallelization and levels of parallelisms

Fabio AFFINITO

SCAI - Cineca

QUANTUM ESPRESSO FOUNDATION

«*What I cannot compute, I do not understand.*» (adapted from Richard P. Feynman)

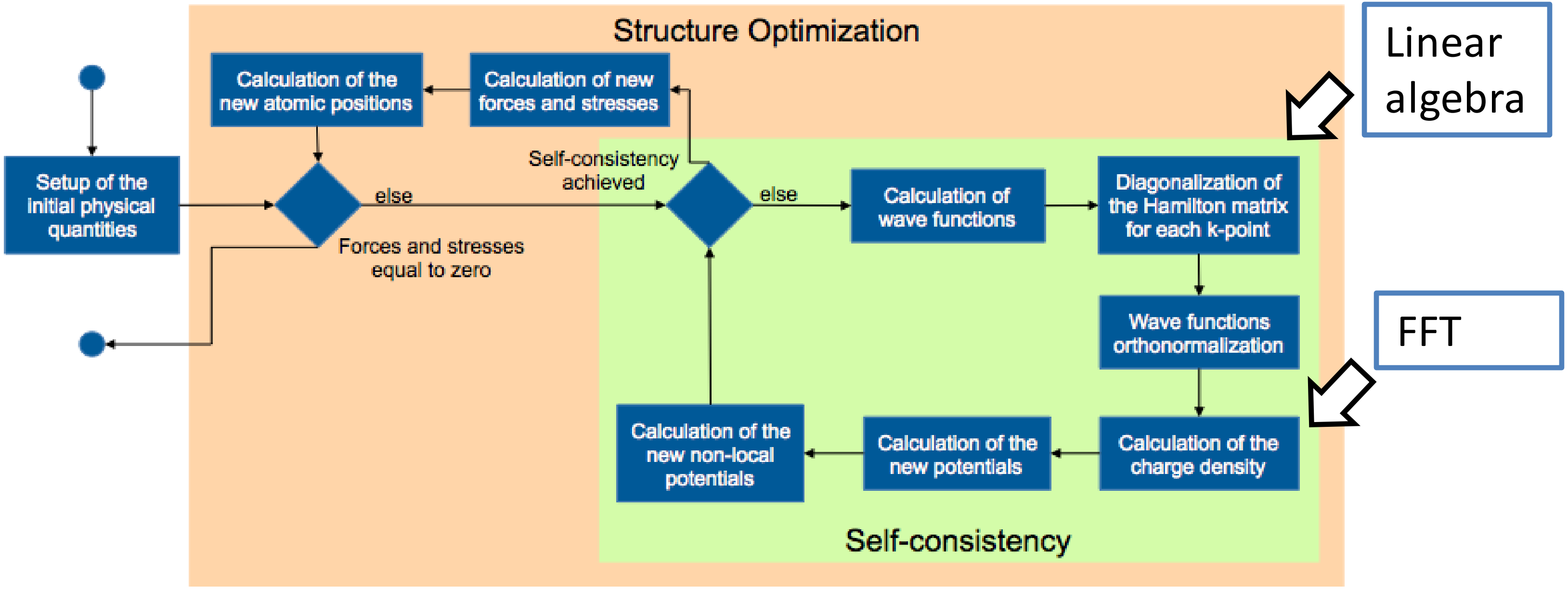
Quantum ESPRESSO: introduction

- Quantum ESPRESSO is an integrated software suite for atomistic simulations based on electronic structure, using density-functional theory(DFT), a plane waves (PW) basis set and pseudopotentials (PP)
- It is a collection of specific-purpose software, the largest being:
  - PWSCF
  - CP

  plus many other applications able to post-process the wavefunctions generated by PWscf (for example PHonon, GW, TDDFPT, etc)

PWscf

- As an example, let's watch at the structure of PWscf



Technical infos

- Quantum ESPRESSO is released under a GNU-GPL license and it is downloadable from www.quantum-espresso.org
- Mostly written in Fortran90
- Ongoing effort to increase the modularization (MaX CoE funded)
- It can use optimized libraries for LA and FFT (i.e. MKL, FFTW3, etc), but it can be also compiled without any external library
- MPI based parallelization: multiple communicators, hierarchical strategy
- OpenMP fine grained parallelization + usage of threaded libraries (OpenMP tasks will be soon implemented)

Relevant quantities

- N_w: number of plane waves (used in wavefunction expansion)
- N_g: number of G-vectors (used in charge density expansion)
- N_1, N_2, N_3: dimensions of the FFT grid for charge density (for Ultrasoft PPs there are two distinct grids)
- N_a: number of atoms in the unit cell or supercell
- N_e: number of electron (Kohn-Sham) states (bands)
- N_p: number of projectors in nonlocal PPs (sum over cell)
- N_k: number of k-points in the irreducible Brillouin Zone

Parallelization strategy

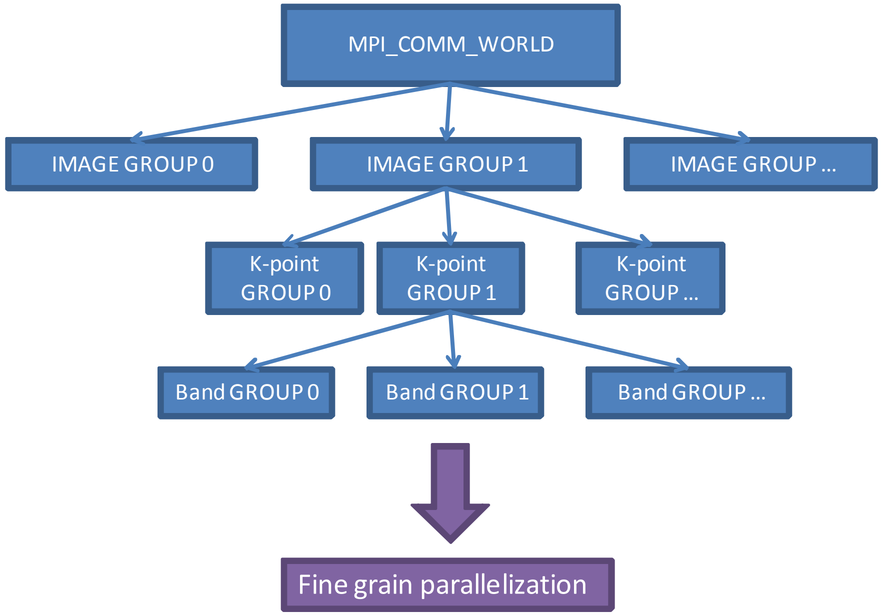
- Goals:
  - Load balancing
  - Reduce communication
  - Fit the architecture (intranode/internode)
  - Exploit asynchronism and pipelining

Coarse grain parallelization levels

1. Plane-waves (MPI_Comm_World)
2. Images
3. K-points
4. Bands

+ a finer grain data distribution

Coarse grain, high level QE data distribution



QUANTUM ESPRESSO FOUNDATION

Fine grain parallelization levels

Data can be furtherly redistributed in order to accomplish specific tasks, such as FFT or linear algebra (LA) routines

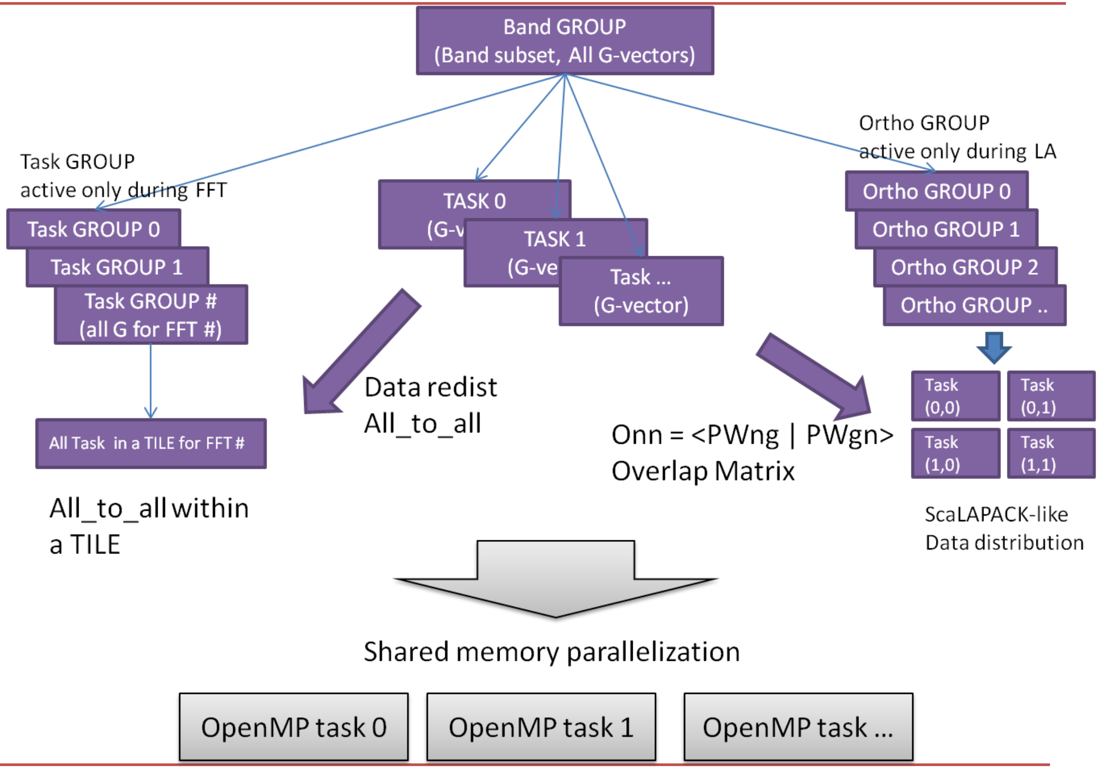


Image parallelization

- A trivial parallelization can be made on images. Images are loosely coupled replica of the system and they are useful for
  – Nudged Elastic Band calculations
  – Atomic Displacement patterns for linear response calculation

  and in general for all the cases in which you want to replicate N times your system and perform identical simulations (ensemble techniques).

```
mpirun -np 64 neb.x -nimage 4 -input inputfile.inp
```

k-point parallelization

- If the simulation consists in different k-points, those can be distributed among n_{pools} pools of CPUs
- K-points are tipically independents: the amount of communications is small
- When there is a large number of k-points this layer can strongly enhance the scalability
- By definition, n_{pools} must be a divisor of the total number of k-points

```
mpirun -np 64 pw.x -npool 4 -input inputfile.inp
```

Band parallelization

- Kohn-Sham states are split across the processors of the band group. Some calculations can be independently performed for different band indexes.
- In combination with other levels of parallelism can improve performances and scalability
- For example, in combination with k-points parallelization:

```
mpirun -np 64 pw.x -npool 4 -bgrp 4 -input inputfile.inp
```

Linear algebra parallelization

- Distribute and parallelize matrix diagonalization and matrix-matrix multiplications needed in iterative diagonalization (SCF) or orthonormalization (CP). Introduces a linear-algebra group of n_{diag} processors as a subset of the plane-wave group. $n_{diag} = m^2$, where m is an integer such that $m^2 \leq n_{PW}$.
- Should be set using the – ndiag or -n_{ortho} command line option, e.g.:

```
mpirun -np 64 pw.x -ndiag 25 -input inputfile.inp
```

Task-group parallelization

- Each plane-wave group of processors is split into n_{task} task groups of n_{FFT} processors, with $n_{task} \times n_{FFT} = n_{PW}$;
- each task group takes care of the FFT over N_e/n_t states.
- Used to extend scalability of FFT parallelization.
- Example for 1024 processors
  - divided into $n_{pool} = 4$ pools of $n_{PW} = 256$ processors,
  - divided into $n_{task} = 8$ tasks of $n_{FFT} = 32$ processors each;
  - Subspace diagonalization performed on a subgroup of $n_{diag} = 144$ processors :

```
mpirun -np 1024 pw.x -npool 4 -ntg 8 -ndiag 144
-input inputfile.inp
```

OpenMP parallelization

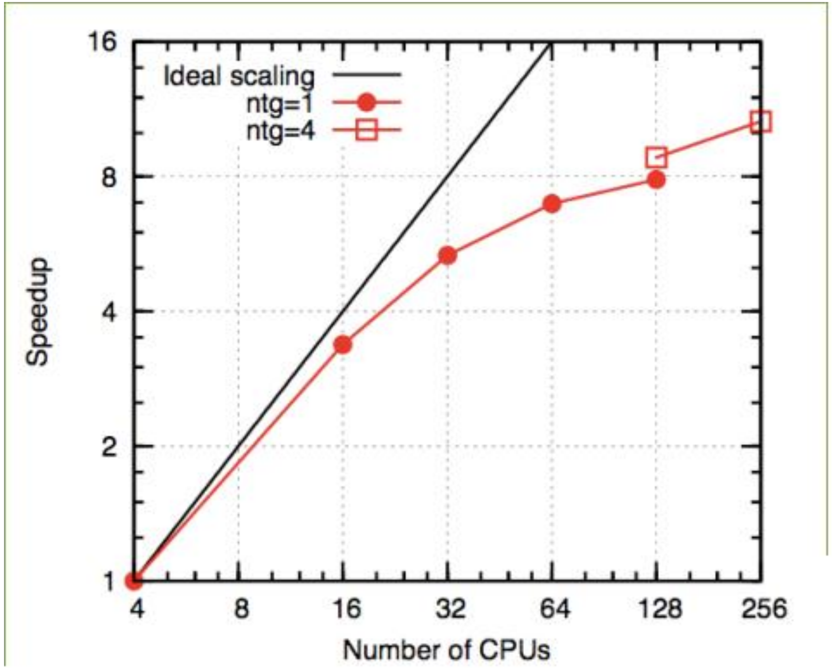
- Explicit with workshare directives on computationally intensive for-loops
- Implicit, when using external thread-safe libraries, e.g.
  - MKL for linear algebra and fft (DFTI interface)
  - FFTW/FFTW3

- Usually scalability on threads is quite poor (no more than 8 threads).
- Ongoing effort to enhance OpenMP scalability using tasking techniques
  - Necessary when working on many-cores architectures

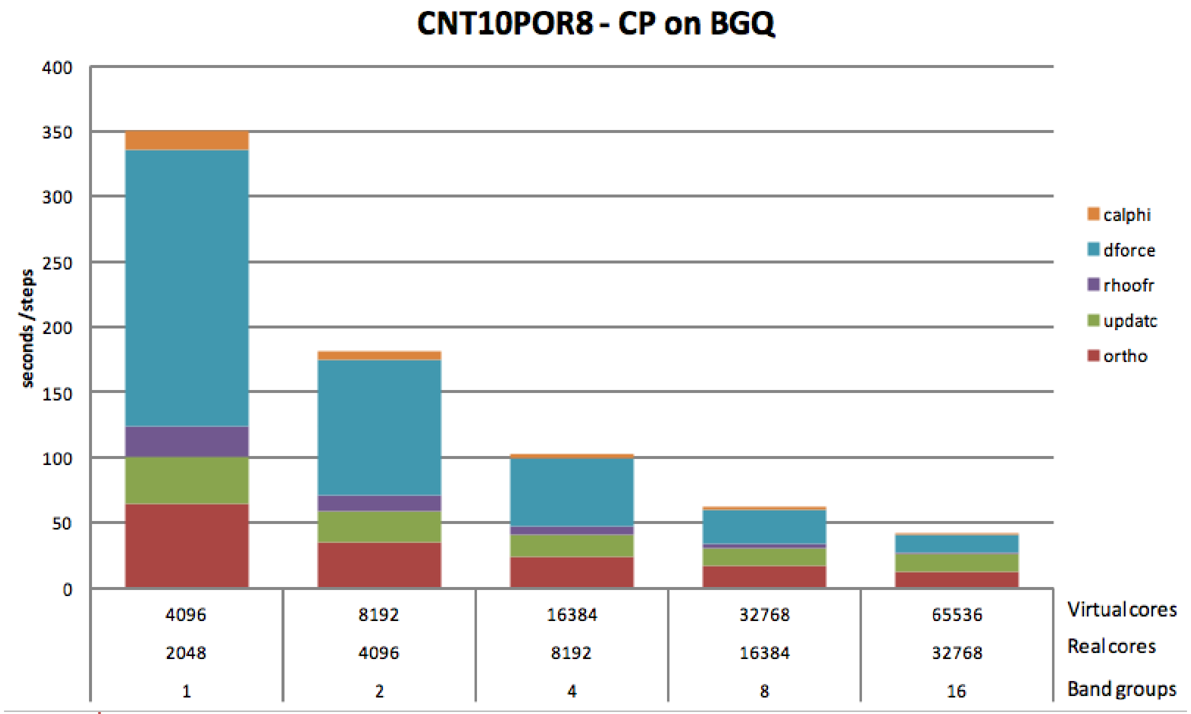
Some examples

- 128 water molecules, PW calculation (IBM Power6), MPI-only

- When scalability saturates, using task-groups permitted to push further..



Some examples



QUANTUM ESPRESSO FOUNDATION