HPC 4 DFT

or

Teaching new Tricks to an old Dog

Stefano de Gironcoli
Scuola Internazionale Superiore di Studi Avanzati
Trieste-Italy

The Dog I'm talking about

QUANTUMESPRESSO

HOME PROJECT DOWNLOAD RESOURCES PSEUDOPOTENTIALS CONTACTS NEWS & EVENTS

SEARCH

Search here...

Forum

NEWS

25.04.16

QUANTUM ESPRESSO V5.4.0

Version 5.4.0 of Quantum ESPRESSO is available for download. You can find all archives uploaded on QE-FORGE here.

31.01.16

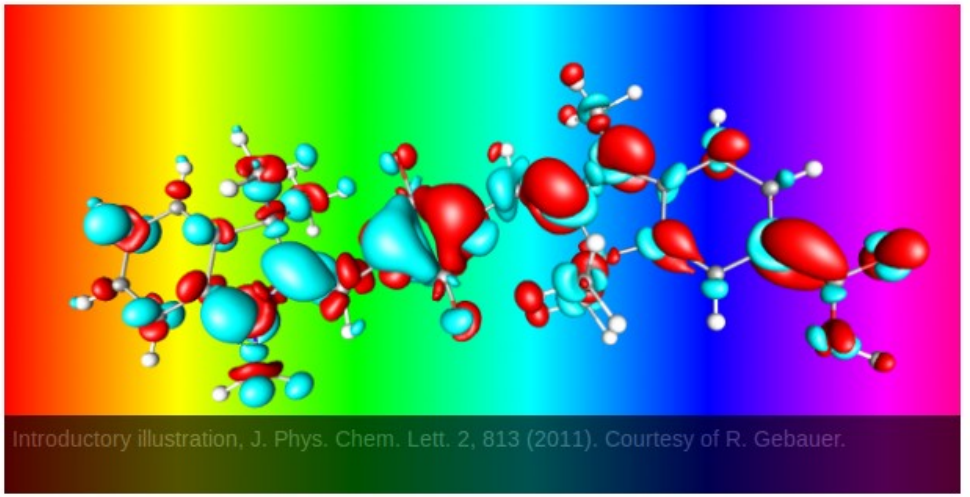
THE WALTER KOHN PRIZE

A prize for outstanding contributions in the field of quantum-mechanical materials and molecular modeling. More information here.

11.01.16

QUANTUM ESPRESSO V5.3.0

Version 5.3.0 of Quantum ESPRESSO is available for download.



Introductory illustration, J. Phys. Chem. Lett. 2, 813 (2011). Courtesy of R. Gebauer.

QUANTUM ESPRESSO

is an integrated suite of Open-Source computer codes for electronic-structure calculations and materials modeling at the nanoscale. It is based on density-functional theory, plane waves, and pseudopotentials.

READ MORE >

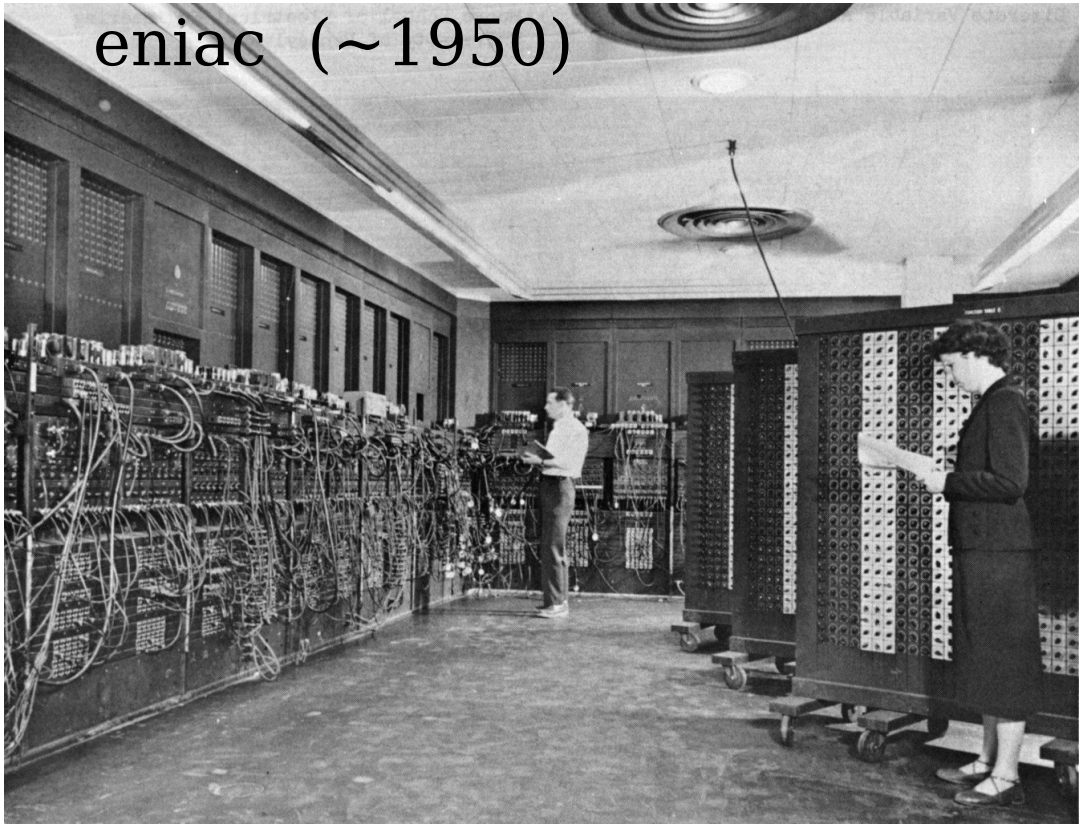
What I cannot compute, I do not understand (adapted from Richard P. Feynman)

Adapted by R. Sabatini on WordPress

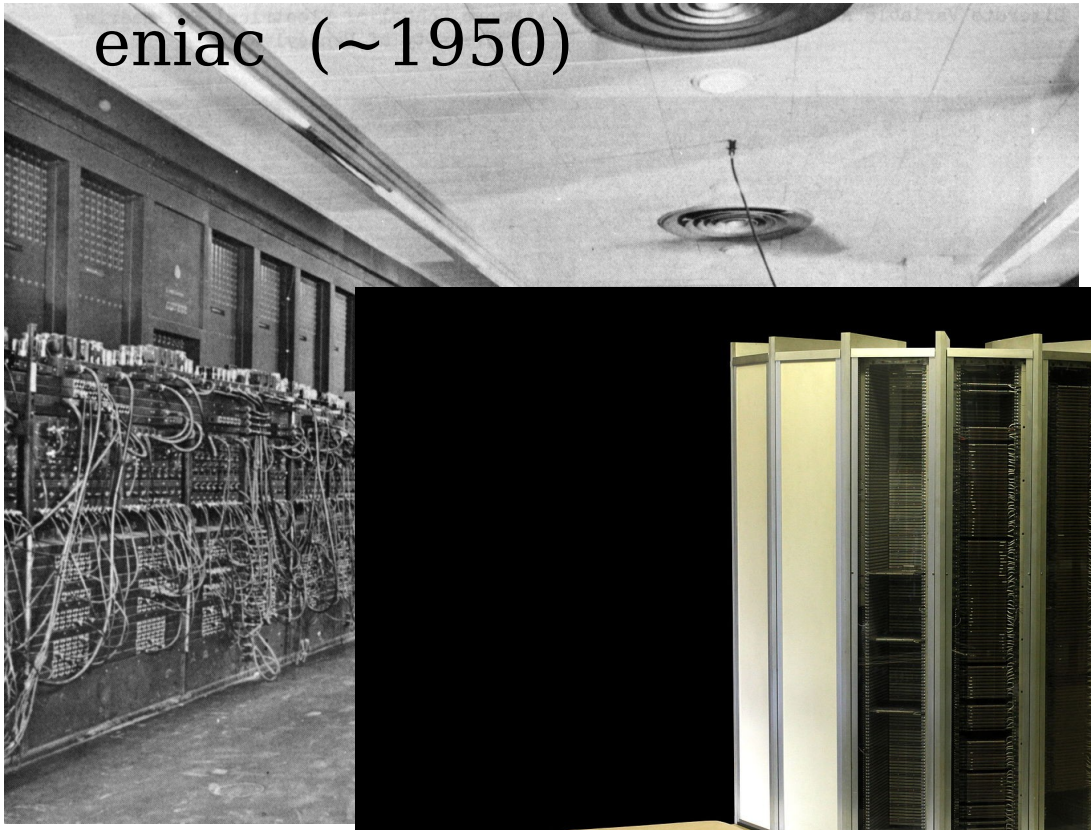
http://www.quantum-espresso.org/

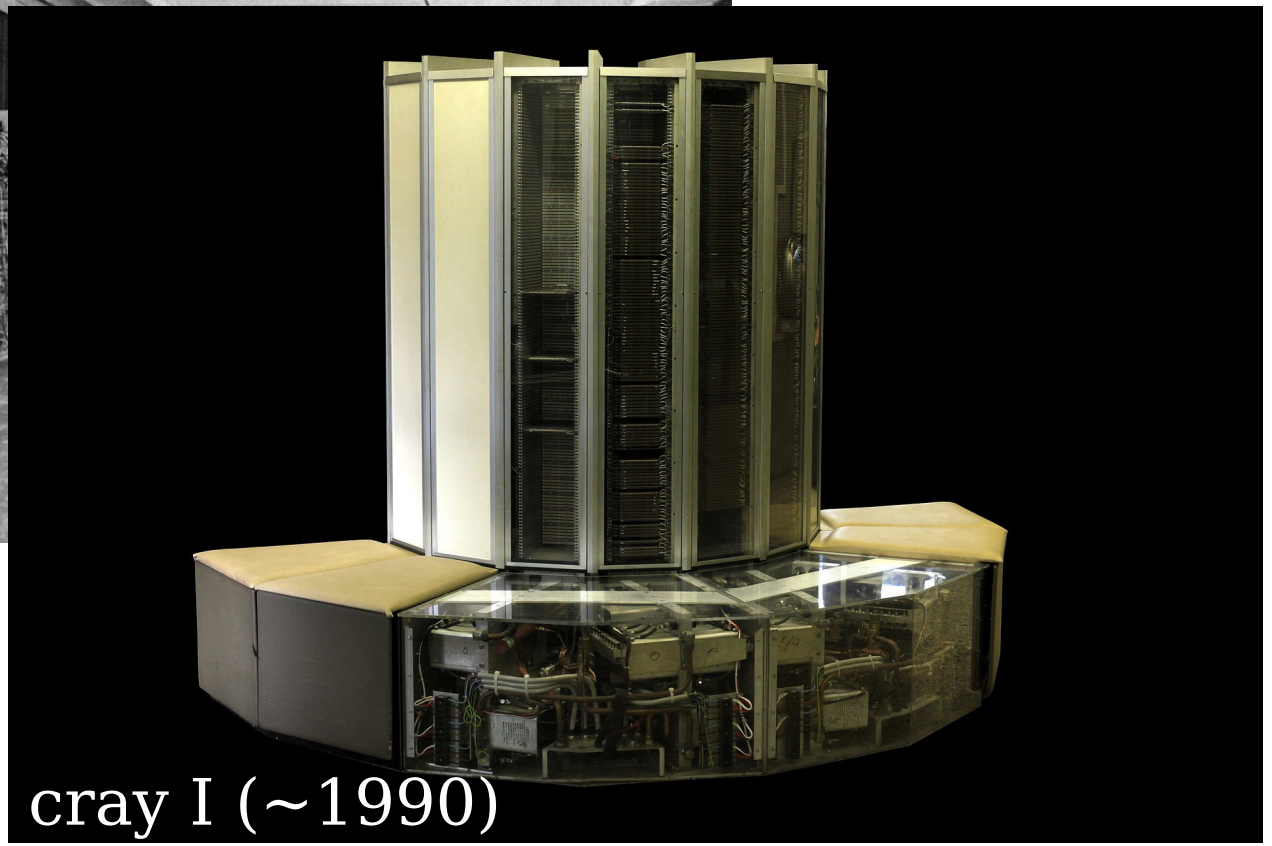
Why new Tricks are needed ?

hardware is changing and the code must adapt

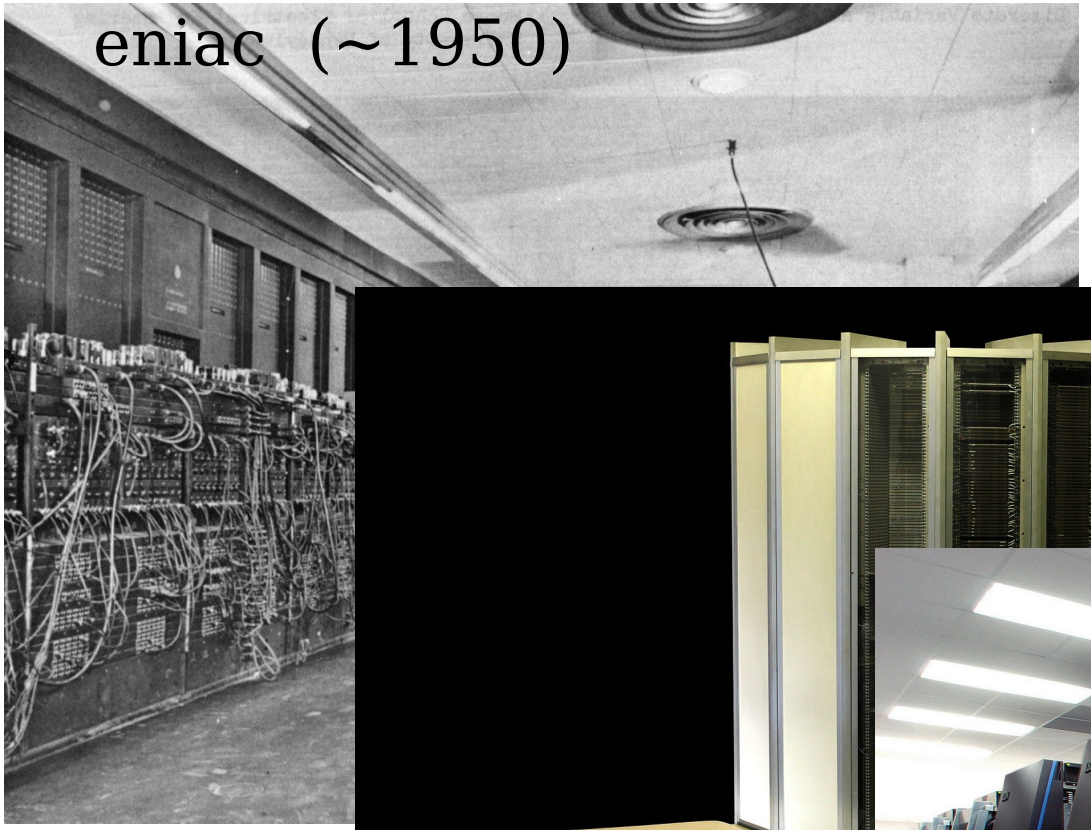


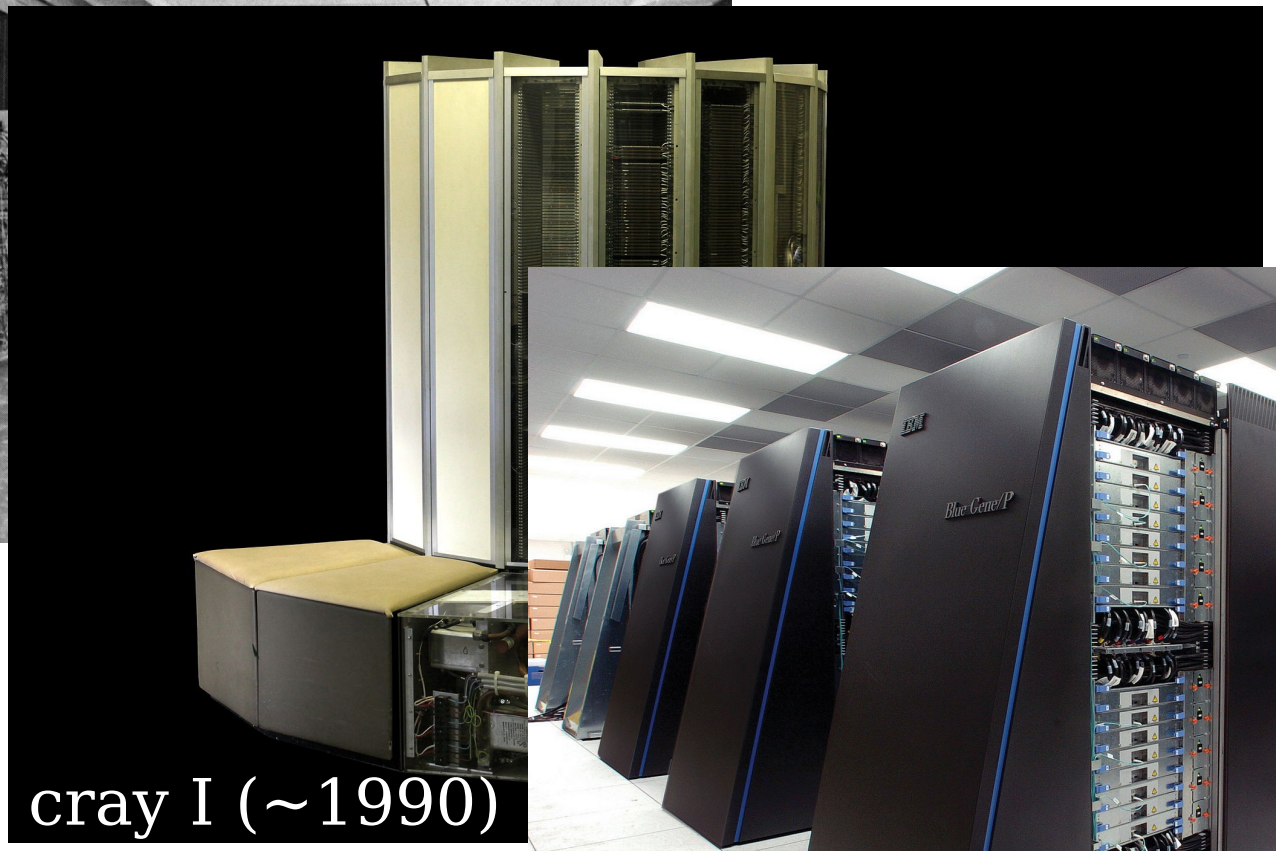
hardware is changing and the code must adapt



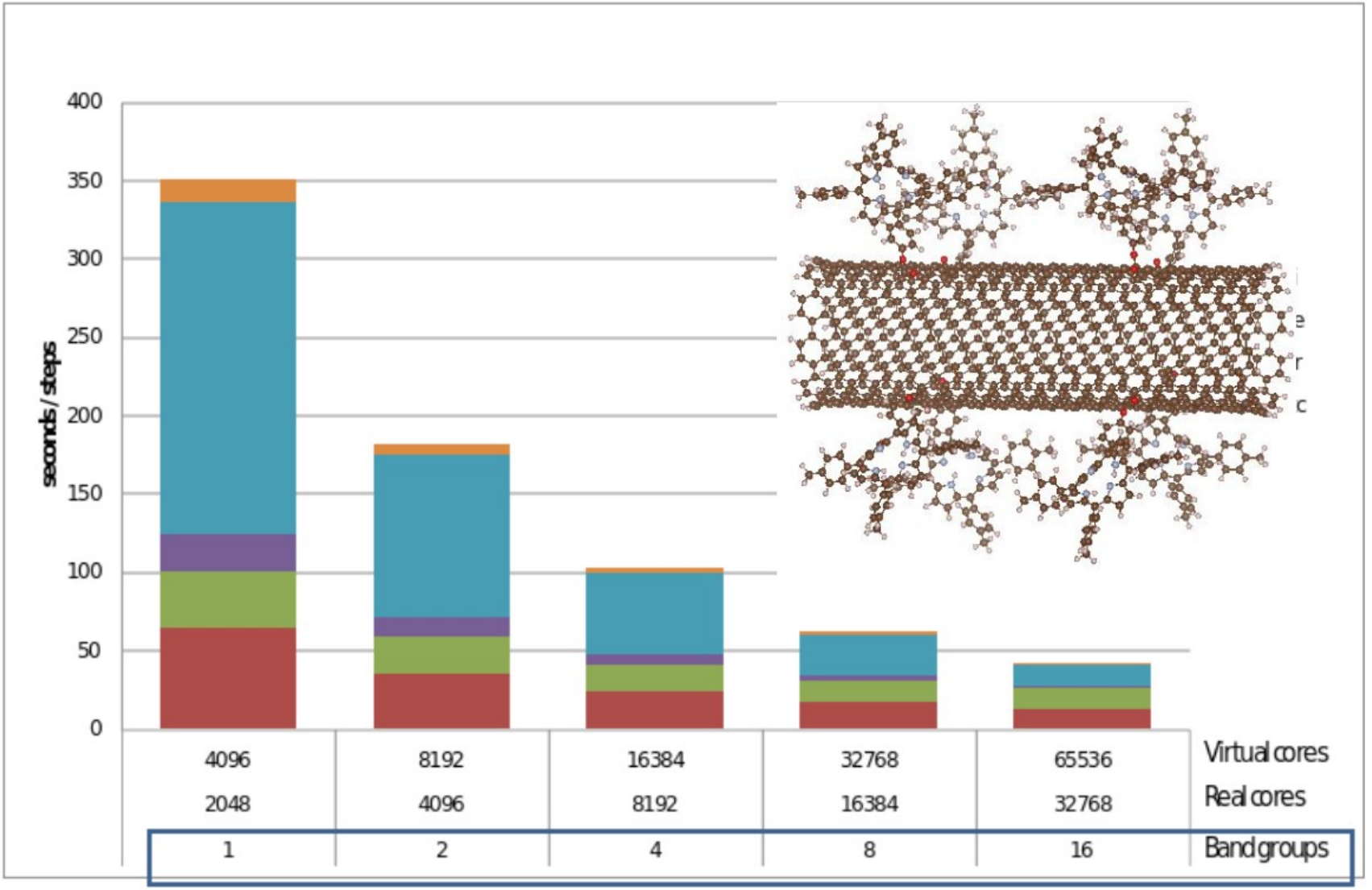


hardware is changing and the code must adapt





Parallel Computing with Quantum ESPRESSO



ABC of DFT

Hohenberg and Kohn theorem [1964]

The GS density can be uses as basic variable to describe the status of a quantum many-body system.

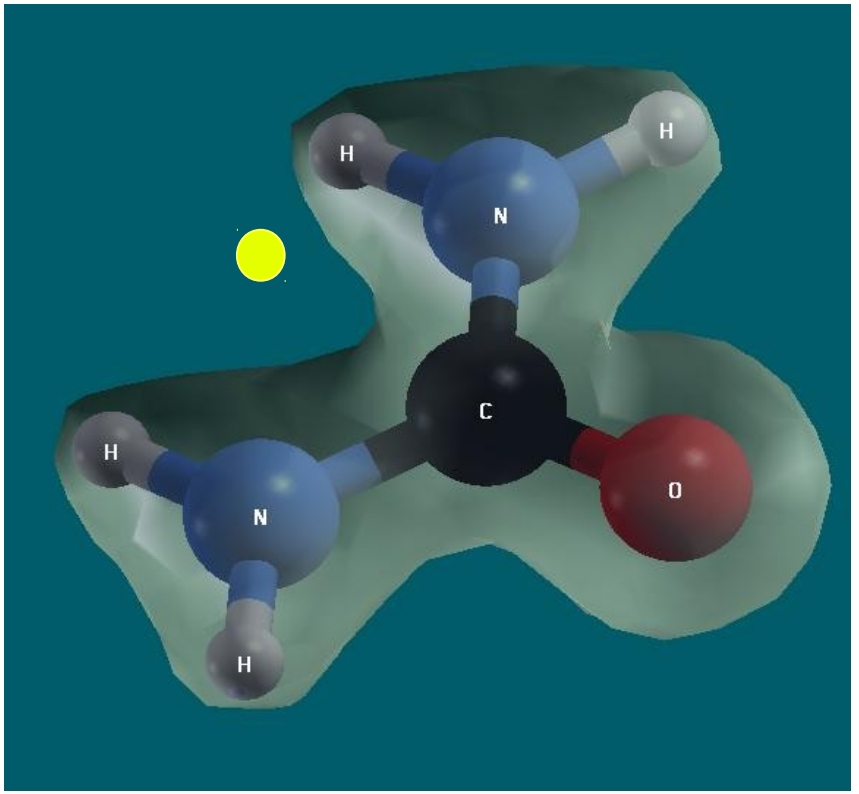
All properties of the system are therefore functionals of the GS density.

Independent electrons in an effective potential

The electrostatic potential felt by a *test* -e charge is therefore

$$V_{eff}(\mathbf{r}) = V_{eI}(\mathbf{r}, \mathbf{R}) + e^2 \int \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}'$$

Given a system of ions and electrons its GS electronic density n(r) is well defined (although not easy to compute)

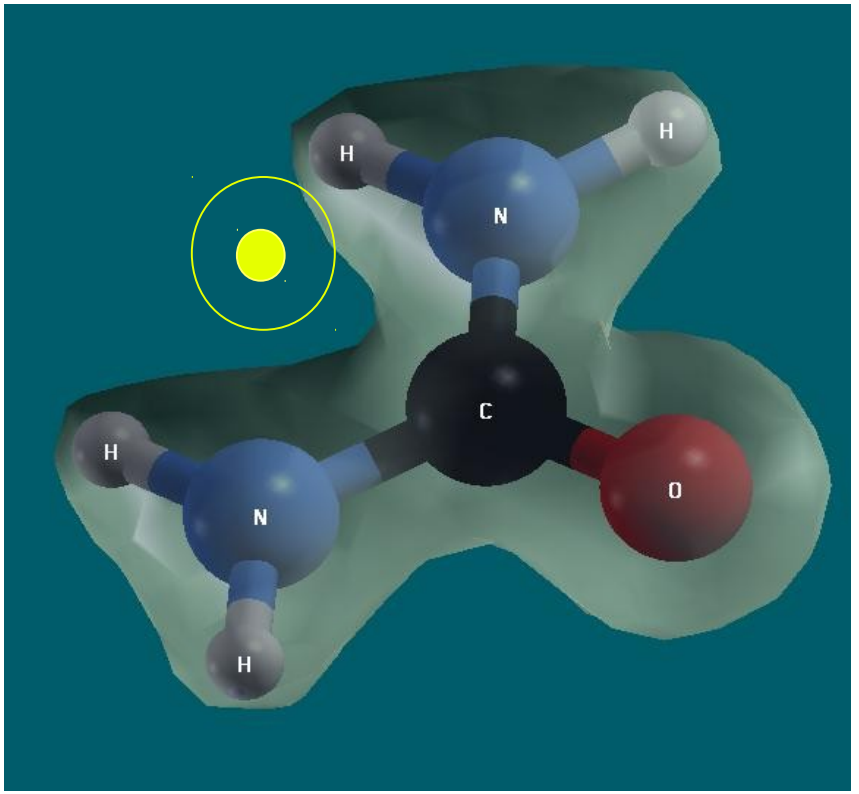


Independent electrons in an effective potential

The electrostatic potential felt by a *real* -e charge is instead

$$V_{eff}(\mathbf{r}) = V_{eI}(\mathbf{r}, \mathbf{R}) + e^2 \int \frac{n(\mathbf{r}') + \delta n(\mathbf{r}'|\mathbf{r})}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}'$$

because electrons *readjust* so as to avoid the extra charge and *screen* the perturbation.

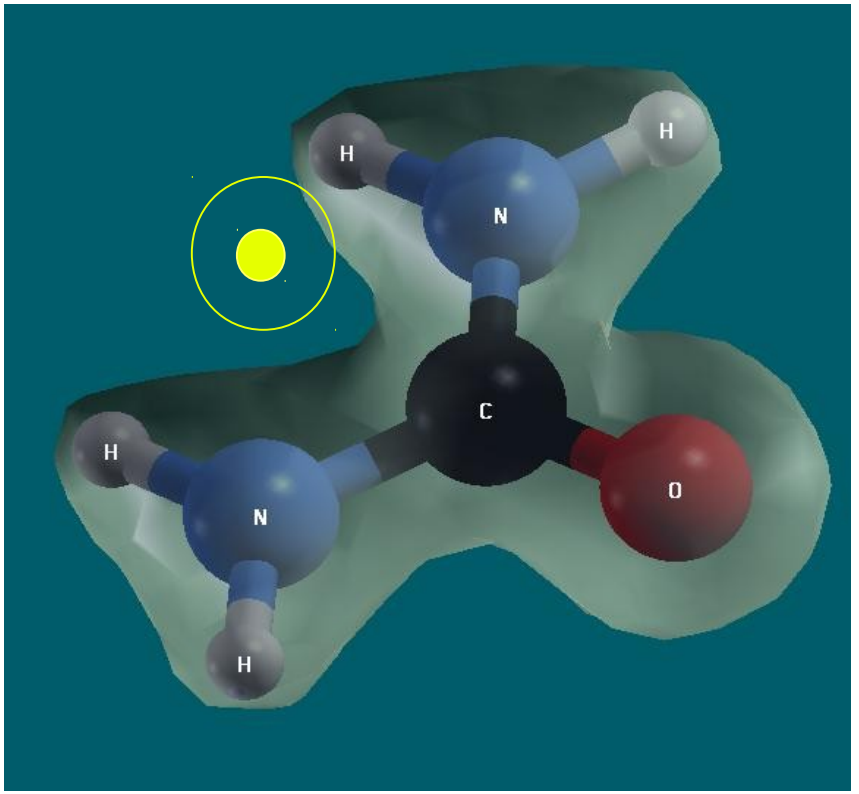


Independent electrons in an effective potential

The electrostatic potential felt by a *real* -e charge is instead

$$V_{eff}(\mathbf{r}) = V_{eI}(\mathbf{r}, \mathbf{R}) + e^2 \int \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}' + v_{scr}(\mathbf{r})$$

because electrons *readjust* so as to avoid the extra charge and *screen* the perturbation.



Independent electrons in an effective potential

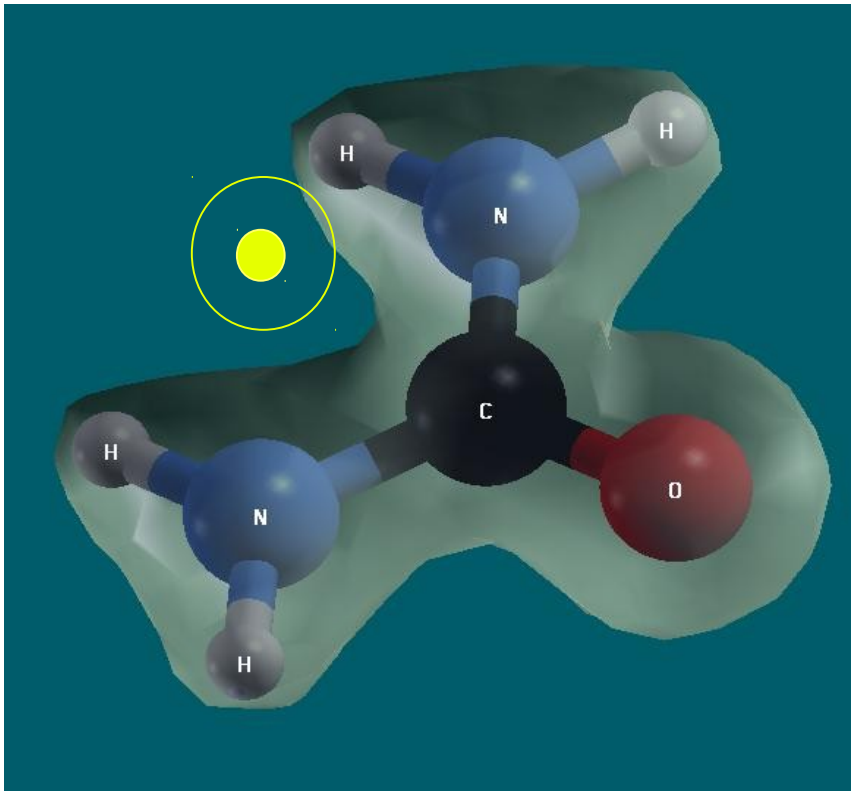
The electrostatic potential felt by a *real* -e charge is instead

$$V_{eff}(\mathbf{r}) = V_{eI}(\mathbf{r}, \mathbf{R}) + e^2 \int \frac{n(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}' + v_{scr}(\mathbf{r})$$

We can imagine electrons would move in this potential

$$\left[T_e + V_{eff}(\mathbf{r})\right] \phi_i(\mathbf{r}) = \varepsilon_i \, \phi_i(\mathbf{r})$$

including *dynamical screening* and Pauli principle (*exchange*) effects.



Independent electrons in an effective potential

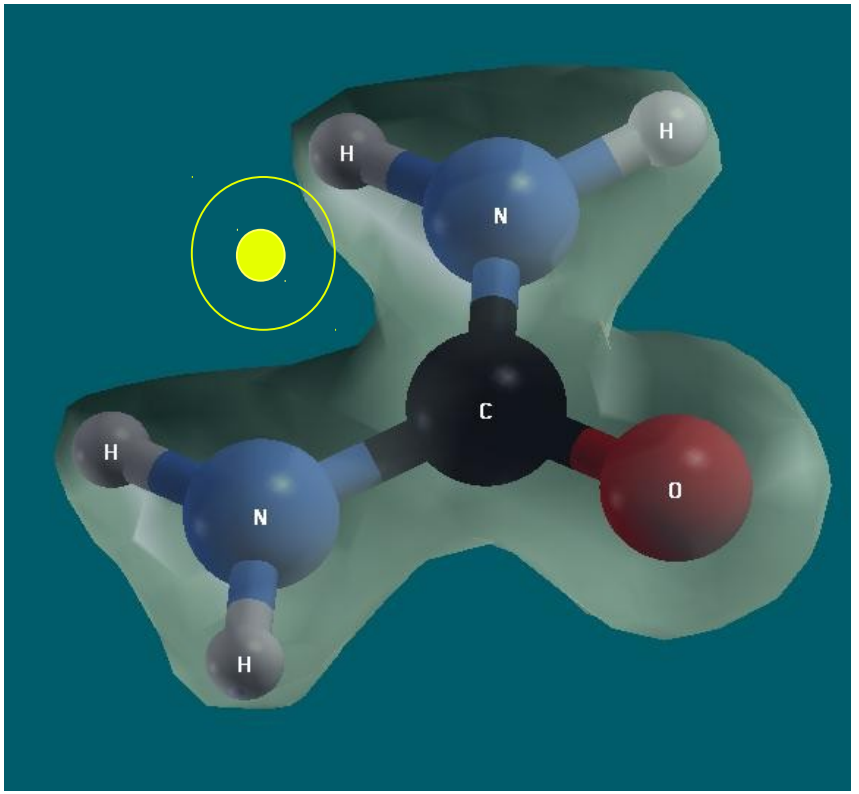
$$V_{eff}(\mathbf{r}) = V_{eI}(\mathbf{r},\mathbf{R}) + e^2 \int \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r}' + v_{scr}(\mathbf{r})$$

$$\left[T_e + V_{eff}(\mathbf{r})\right] \phi_i(\mathbf{r}) = \varepsilon_i \, \phi_i(\mathbf{r})$$

$$n(\mathbf{r}) = 2 \sum_i |\phi_(\mathbf{r})|^2$$

self-consistent
single-particle
set of equations

ex: HF, DFT, GW, ...



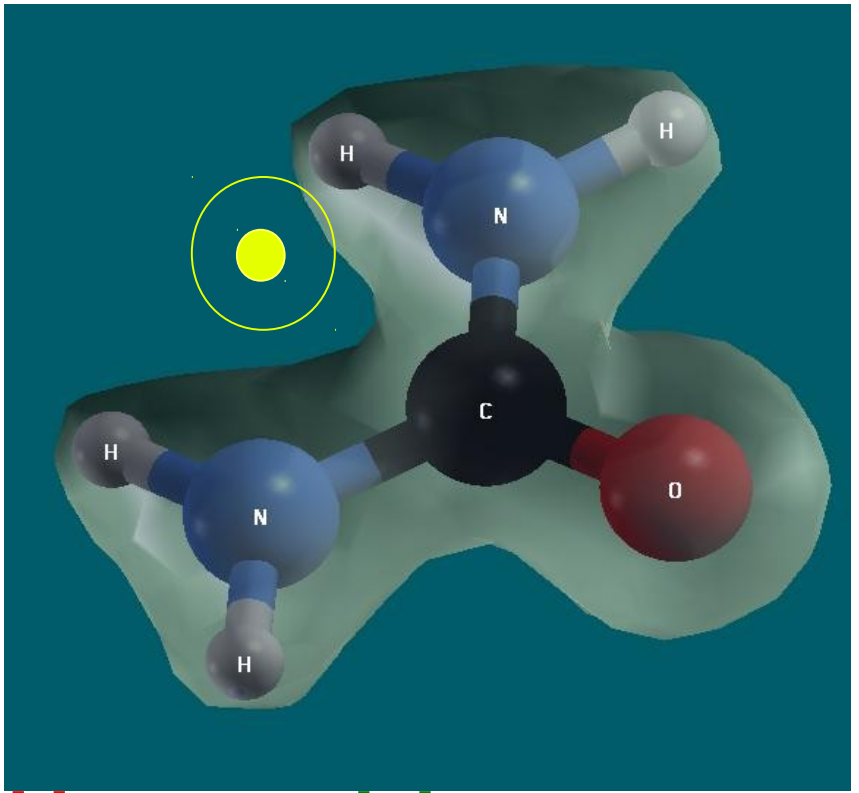
Independent electrons in an effective potential

$$V_{eff}(\mathbf{r}) = V_{eI}(\mathbf{r},\mathbf{R}) + e^2 \int \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r}' + \frac{\delta E_{xc}[n]}{\delta n(\mathbf{r})}$$

$$\left[T_e + V_{eff}(\mathbf{r})\right] \phi_i(\mathbf{r}) = \varepsilon_i \, \phi_i(\mathbf{r})$$

$$n(\mathbf{r}) = 2 \sum_i |\phi_(\mathbf{r})|^2$$

DFT: Kohn-Sham [1965]
self-consistent
single-particle
set of equations



as simple as a Mean-field approach but exact !

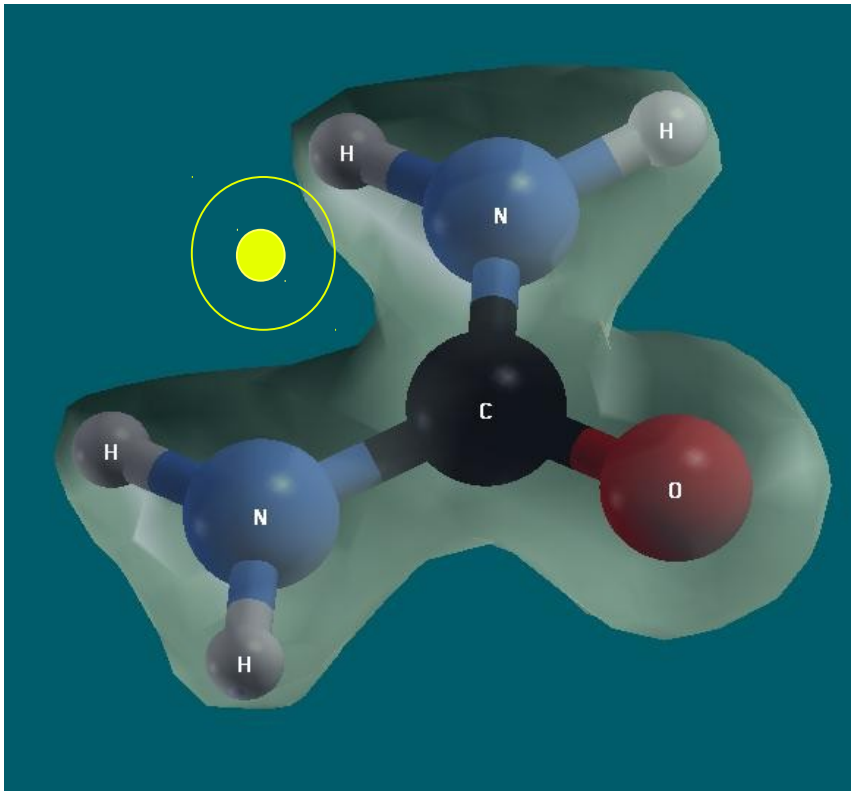
Independent electrons in an effective potential

$$V_{eff}(\mathbf{r}) = V_{eI}(\mathbf{r},\mathbf{R}) + e^2 \int \frac{n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r}' + \frac{\delta E_{xc}[n]}{\delta n(\mathbf{r})}$$

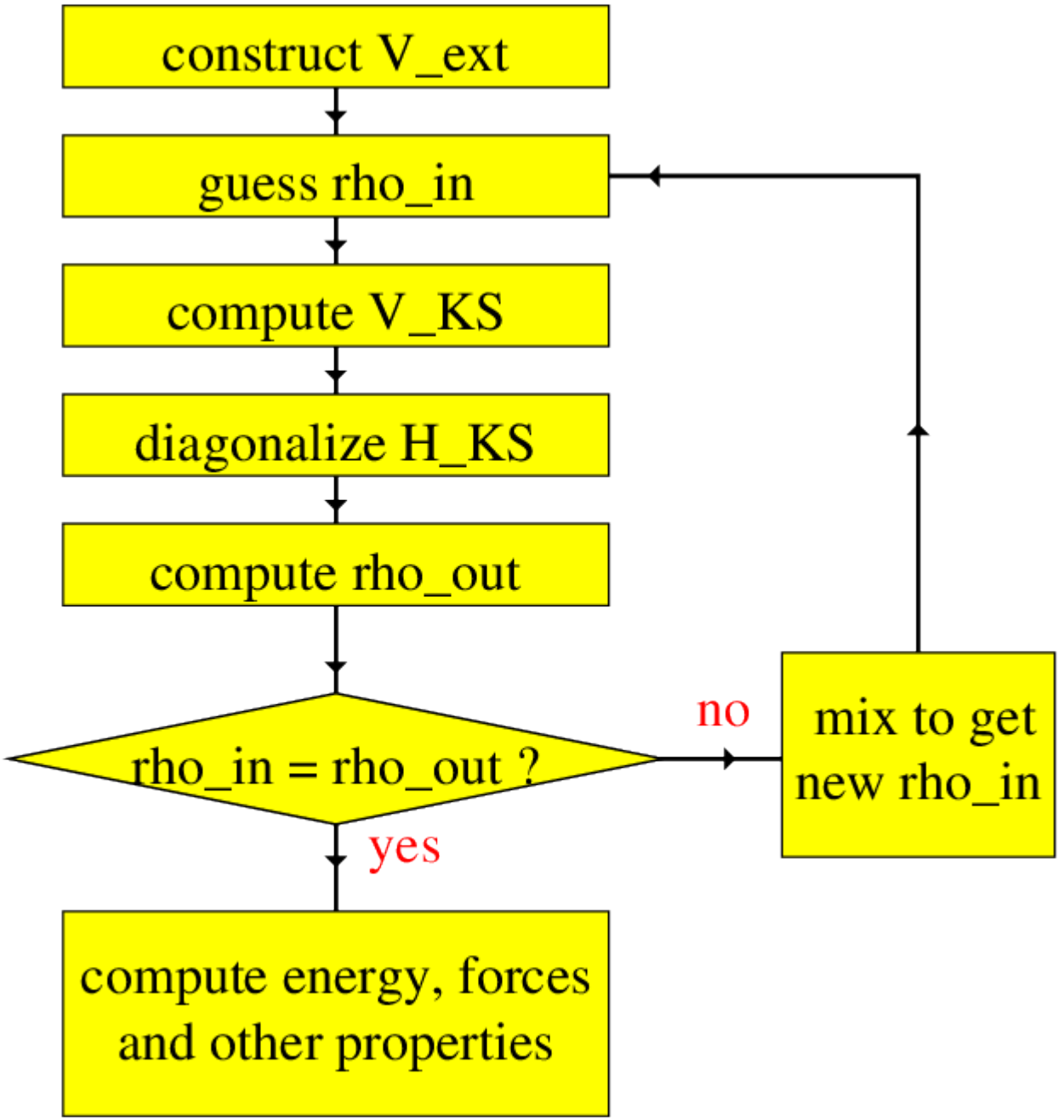
$$\left[T_e + V_{eff}(\mathbf{r})\right] \phi_i(\mathbf{r}) = \varepsilon_i \, \phi_i(\mathbf{r})$$

$$n(\mathbf{r}) = 2 \sum_i |\phi_(\mathbf{r})|^2$$

DFT: Kohn-Sham [1965] self-consistent single-particle set of equations



$E_{xc}[n]$ is not known exactly → approximations


construct V_ext
guess rho_in
compute V_KS
diagonalize H_KS
compute rho_out
rho_in = rho_out ?
no
mix to get new rho_in
yes
compute energy, forces and other properties

The Building Blocks

Diagonalize the hamiltonian

Build the density

Calculate the KS potential

The Building Blocks

Diagonalize the hamiltonian

*needs an efficient computation of H*psi*

Build the density

Calculate the KS potential

The Building Blocks

Diagonalize the hamiltonian

*needs an efficient computation of H*psi*

Build the density

n*eeds an efficient BZ sampling and fast psi(r)*

Calculate the KS potential

The Building Blocks

Diagonalize the hamiltonian

*needs an efficient computation of H*psi*

Build the density

needs an efficient BZ sampling and fast psi(r)

Calculate the KS potential

needs Poisson's solver and xc functionals

The Building Blocks

Diagonalize the hamiltonian

*needs an efficient computation of H*psi*

Build the density

needs an efficient BZ sampling and fast psi(r)

Calculate the KS potential

needs Poisson's solver and xc functionals

FFT are a fundamental tool for all these operations

The KS hamiltonian and the wfc in a PW basis set

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + V_{KS}(r)\right]\varphi_i(r) = \varepsilon_i\varphi_i(r)$$

thanks to Bloch theorem $\quad i \rightarrow kv$

$$|\varphi_{kv}\rangle = \sum_{k+G} c^v_{k+G}\,|k+G\rangle \qquad |k+G| < G_{max}$$

$$\varphi_{kv}(r) = \langle r|\varphi_{kv}\rangle = \sum_{k+G} c^v_{k+G}\,\langle r|k+G\rangle$$

$$= \sum_{k+G} c^v_{k+G}\,\frac{e^{i(k+G)r}}{\sqrt{\mathcal{V}}}$$

the KS eq. becomes a matrix eigenvalue problem

$$\sum_{k+G'} \langle k+G|H_{KS}|k+G'\rangle\; c^v_{k+G'} = \varepsilon_{kv}\; c^v_{k+G}$$

Exact diagonalization is expensive

find eigenvalues & eigenfunctions of H (k+G,k+G')

Expensive to store H matrix: NPW^2 elements to be stored

Expensive (CPU time) to diagonalize H matrix exactly, ~ NPW^3 operations required.

NPW >> Nb = number of bands required = Ne/2 or a little more (for metals).

So it's ok to determine just the lowest few eigenvalues

=> Iterative Diagonalization (Davidson,CG,ParO,...)

an efficient computation of H_psi is required.

The KS hamiltonian and the wfc in a PW basis set

$$\sum_{k+G'} \langle k+G|H_{KS}|k+G'\rangle \; c^v_{k+G'} = \varepsilon_{kv} \; c^v_{k+G}$$

$$\langle k+G| - \frac{\hbar^2}{2m}\nabla^2|k+G'\rangle = \frac{\hbar^2}{2m}(k+G)^2 \; \delta_{GG'}$$

diagonal in reciprocal space

The KS hamiltonian and the wfc in a PW basis set

$$\sum_{k+G'} \langle k+G|H_{KS}|k+G'\rangle \; c^{v}_{k+G'} = \varepsilon_{kv} \; c^{v}_{k+G}$$

$$\langle k+G| -\frac{\hbar^2}{2m}\nabla^2|k+G'\rangle = \frac{\hbar^2}{2m}(k+G)^2 \; \delta_{GG'}$$

diagonal in reciprocal space

$$\begin{aligned}\langle k+G|V_{KS}(r)|k+G'\rangle &= \frac{1}{\mathcal{V}}\int_{\mathcal{V}} V_{KS}(r) \;\; e^{-i(G-G')r} \; dr \\ &= \frac{1}{\Omega}\int_{\Omega} V_{KS}(r) \;\; e^{-i(G-G')r} \; dr \\ &= V_{KS}(G-G')\end{aligned}$$

a local potential becomes a convolution

*as such its application to a vector would require N**2 ops*

The dual space formalism

Discrete Fast Fourier Transform allows to go back and forth...

$$\tilde{f}(G_k) = \frac{1}{\Omega}\int_{\Omega} f(r)\exp(-iG_k r)dr = \frac{1}{N}\sum_j f(r_j)e^{-i2\pi\frac{jk}{N}}$$ fwfft

$$f(x_j) = \sum_k \tilde{f}(G_k)\exp(iG_k x_j) = \sum_k \tilde{f}(G_k)e^{i2\pi\frac{jk}{N}}$$ invfft

... in N log N operations

The dual space formalism

*H * psi can be computed very efficiently*

psi(r) = invfft[psi(k+G)]

vpsi(r) = v(r) * psi(r)

vpsi(k+G) = fwfft[vpsi(r)]

hpsi(k+G) = h2/2m (k+G)**2 * psi(k+G) + vpsi(k+G)

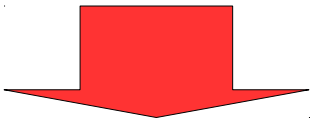
The result is exact if the FFT grid can describe Fourier components up to $2G_{max}$ *where psi is limited to* G_{max}

NB: *this is also the required grid to describe correctly the charge density (i.e. the square of the wavefunctions) and the Hartree potential.*

Problems for Plane-Wave basis

Core wavefunctions:
Sharply peaked close to nuclei due to deep Coulomb potential.

Valence wavefunctions:
Lots of wiggles near nuclei due to orthogonality to core wavefunctions



High Fourier component are present
i.e. large kinetic energy cutoff needed

$$r_{1s} \approx 1/Z \qquad Ecut \approx \left(\frac{2\pi}{r_{1s}}\right)^2 \approx 40Z^2$$

Use

PseudoPotentials

An analogy!

- "Dummy cops" used by many law-enforcement agencies!
- Don't care about internal structure as long as it works right!
- But cheaper!!
- Obviously it can't reproduce all the functions of a real cop, but should be convincing enough to produce desired results….

Borrowed from Shobhana Narasimhan, JNCASR

Ultra Soft PseudoPotentials

$$\hat{V}^{USPP} = V_{loc}(r) + \sum_{lm} |\beta_l\rangle D^0_{lm} \langle\beta_m|$$

Orthogonality with USPP:

$$\langle\psi_i|S|\psi_j\rangle = \langle\psi_i|\psi_j\rangle + \sum_{lm} \langle\psi_i|\beta_l\rangle q_{lm} \langle\beta_m|\psi_j\rangle = \delta_{ij}$$

where $$q_{lm} = \int Q_{lm}(r) dr$$

leading to a generalized eigenvalue problem

$$[H_{KS} - \varepsilon_i S] \ |\psi_i\rangle = 0$$

How things scale with system size ?

N_{at}	number of atoms	Ω system volume $\propto N_{at}$
N_{elec}	number of electrons	$\propto N_{at}$
N_{band}	number of bands	$\approx N_{elec}$
N_{PW}	number of plane waves	$\propto \Omega\, G_{max}^3, \quad \Omega E_{cut}^{3/2}$
N_r	number of FFT grid points	$\approx 10 N_{PW}$
N_K	number of BZ k-points	$\propto 1/\Omega$

computational cost

1 Hpsi	$\propto N_{PW} + N_r \log N_r + N_r + N_{band} N_{PW}$
Iter. Diag.	$\propto N_{band}$ `Hpsi` $+ N_{band}^2 N_{PW} + N_{band}^3$
new rho	$\propto N_K\, (N_r \log N_r + N_r)$
new pot	$\propto N_r \log N_r + N_r$

strongly dependent on N_{elec} *and* N_{PW} (E_{cut})

Parallel Programming Paradigms

Most modern electronic-structure codes use one of the following approaches, or a combination of them:

MPI – Message Passing Interface

Many processes are executed in parallel, one per processor, accessing their own set of variables. Access to variables on another processor is achieved via *explicit* calls to MPI libraries.

Open MP – open Multi Processing

A single process spawns sub-processes (*threads*) on other processors that can access and process the variables of the code. Achieved with compiler directives and/or via call to “multi-threading” libraries like Intel MKL or IBM ESSL.

Quantum ESPRESSO exploits both MPI and OpenMP parallelization: the former is well-established, the latter is quickly evolving and stabilizing.

There is also some work for GPUs and accelerators not discussed here.

MPI vs Open MP

MPI – Message Passing Interface

distributed data, explicit communications

multicore processors (workstations,laptops), cluster of multicore processors interconnected by a fast communication network.

Open MP – open Multi Processing

shared data multiprocessing

parallelization inside a multicore processor, standalone or part of a larger network.

Network important factors

Bandwidth and Latency

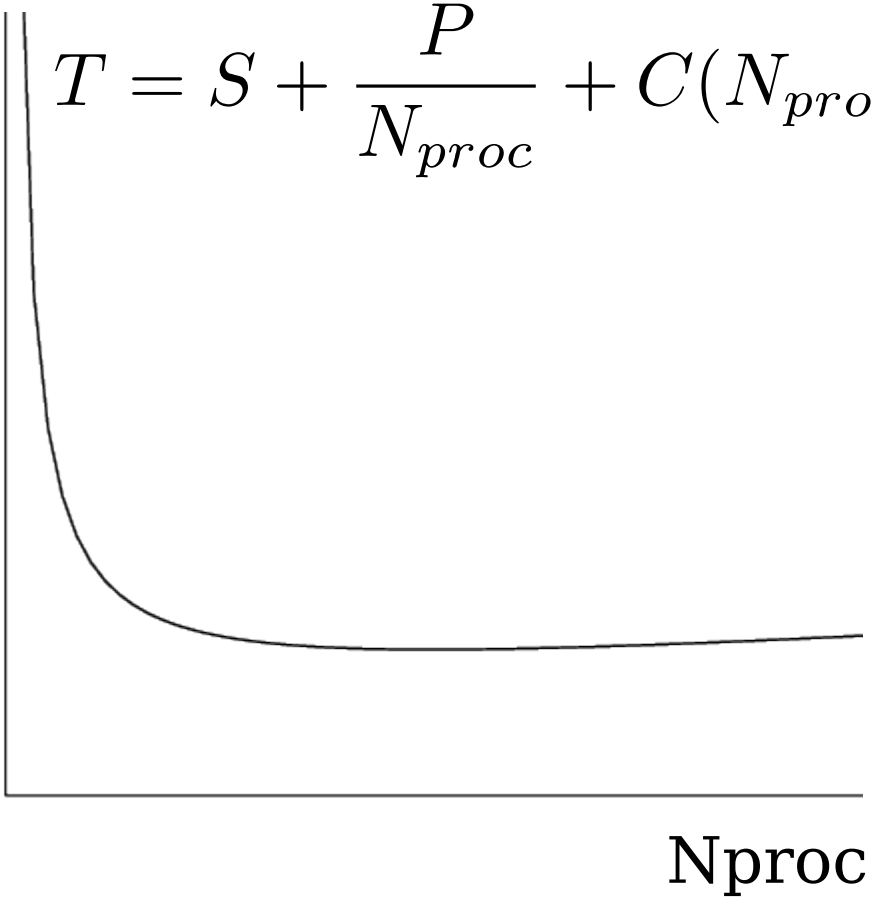
A fast interconnection is important but how often and how much one communicates is also very important.

Blocking vs non blocking communications may also play a role.

Disk I/O is typically bad and should be avoided.
On parallel machines even more so.
If RAM allows for it just keep things in memory.

Amdahl's law

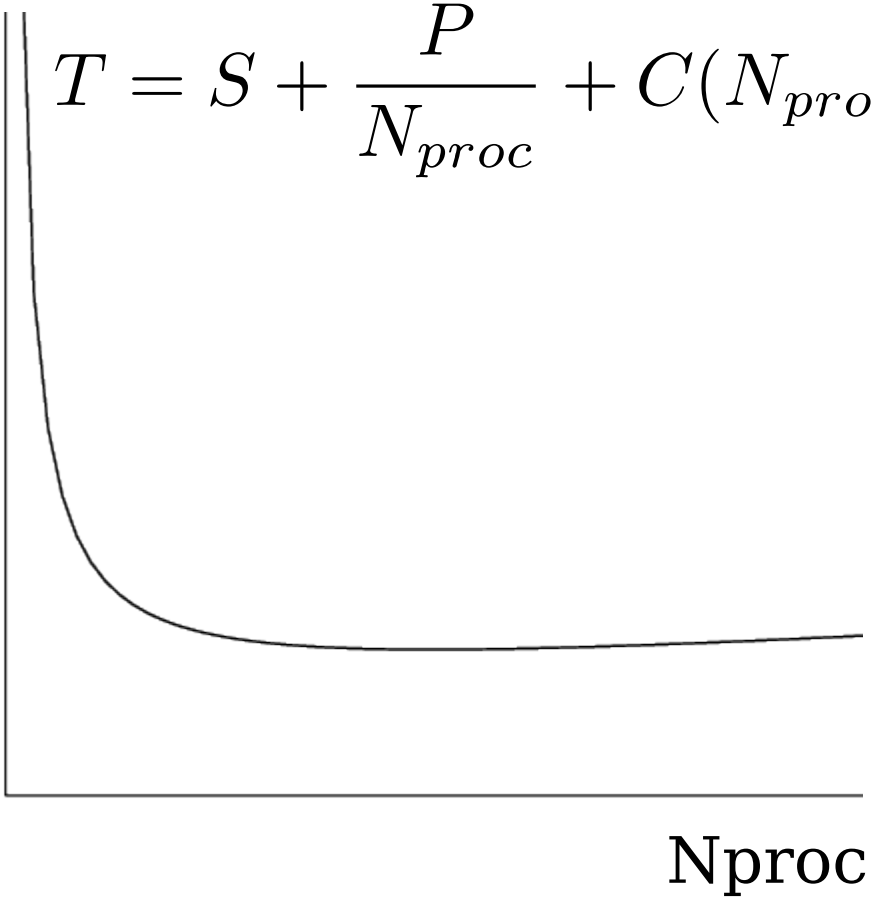
$$T = S + \frac{P}{N_{proc}} + C(N_{proc})$$



No matter how well you parallelize your code on the long run the scalar fraction dominates.

Amdahl's law

$$T = S + \frac{P}{N_{proc}} + C(N_{proc})$$



No matter how well you parallelize your code on the long run the scalar fraction dominates. Even before communication becomes an issue.

Strong Scaling vs Weak Scaling

Strong Scaling: scaling when system size remains fixed
Weak Scaling: scaling when system size also grows

Strong Scaling is much more difficult to achieve than Weak Scaling.

Many times one does not need to perform huge calculations but rather (many) medium/large calculations.

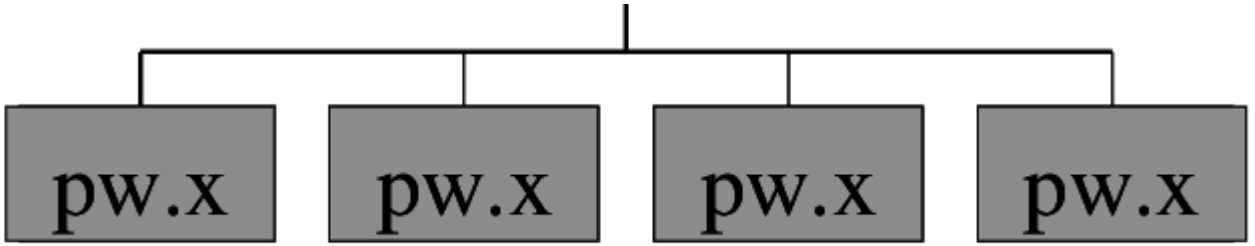
Extreme parallelization would not be needed but queue scheduler constraints enforce the use of many cores so strong scaling is desirable.

When many independent runs are needed, automated scheduling and job management is becoming important.

MPI – Message Passing Interface

multiple processes, multiple data, single program

```
prompt> mpirun -np 4 pw.x < pw.in > pw.out
```



Multiple copies of the code start,input is broadcast, data are distributed, work is performed on the local data,communication of partial results is frequent.

High communication needs, reduction of memory/proc footprint.

MPI – Message Passing Interface

a simple example

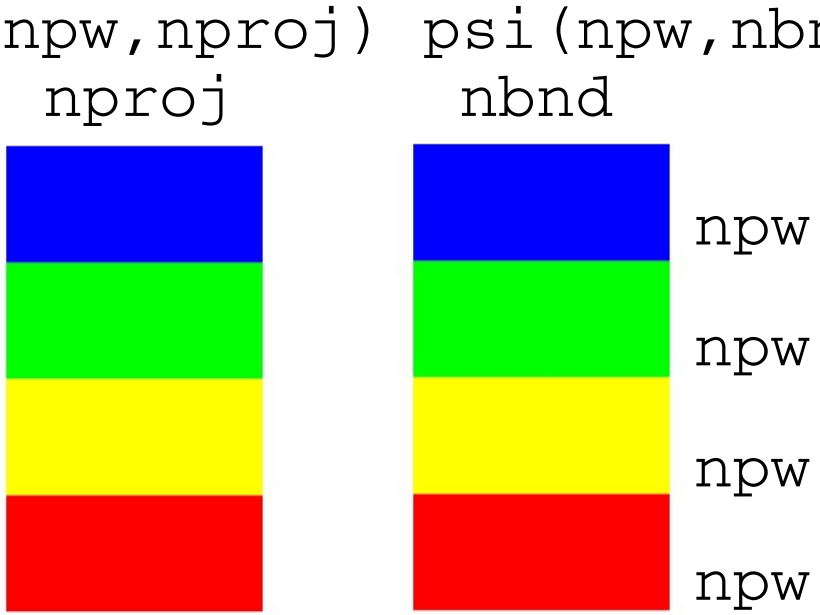
$$\langle \beta_i | \psi_j \rangle = \sum_{k+G} \beta_i^*(k+G)\psi_j(k+G)$$

MPI – Message Passing Interface

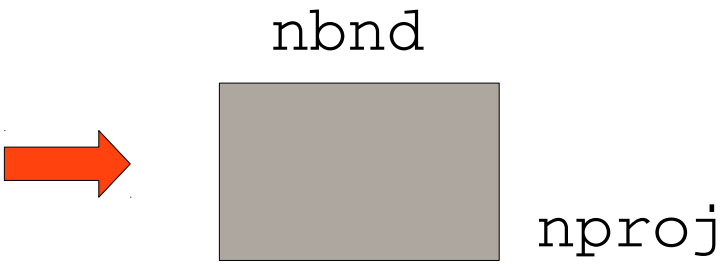
a simple example

$$\langle \beta_i | \psi_j \rangle = \sum_{k+G} \beta_i^*(k+G)\psi_j(k+G)$$

beta(npw,nproj) psi(npw,nbnd)



betapsi(nproj,nbnd)



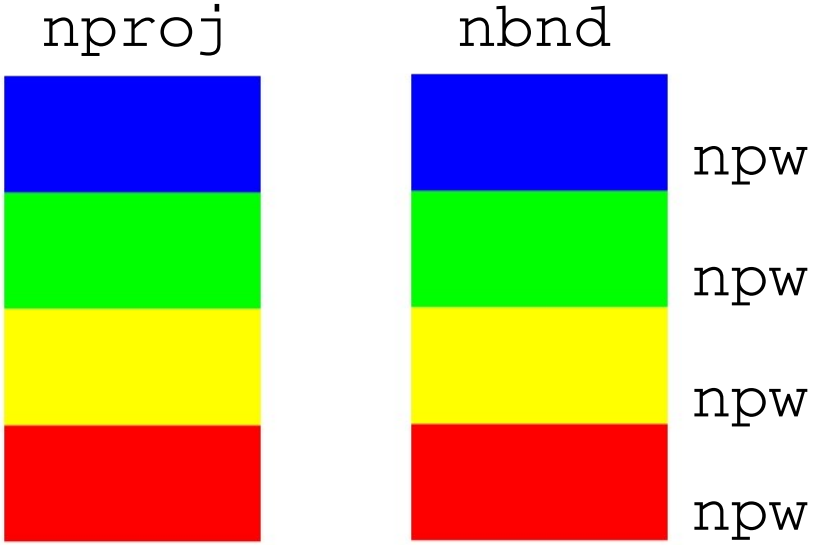
how one gets betapsi?

MPI – Message Passing Interface

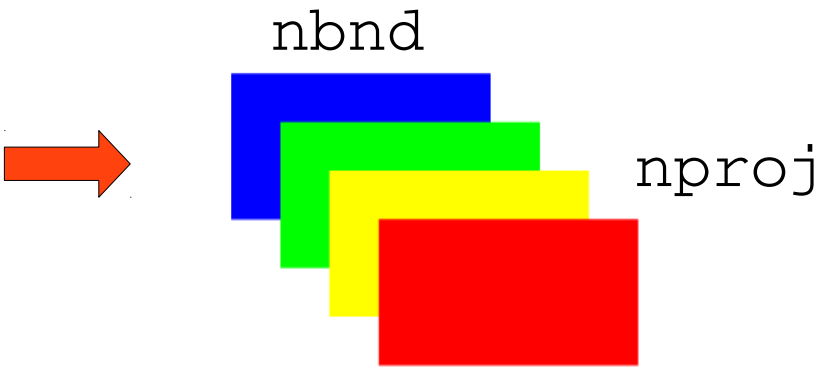
a simple example

$$\langle \beta_i | \psi_j \rangle = \sum_{k+G} \beta_i^*(k+G)\psi_j(k+G)$$

beta(npw,nproj) psi(npw,nbnd)



betapsi(nproj,nbnd)




```
CALL ZGEMM( 'C', 'N', nproj, nbnd, npw, (1.0_DP,0.0_DP), &
                  beta, npwx, psi, npwx, (0.0_DP,0.0_DP), &
                  betapsi, nprojx )
```

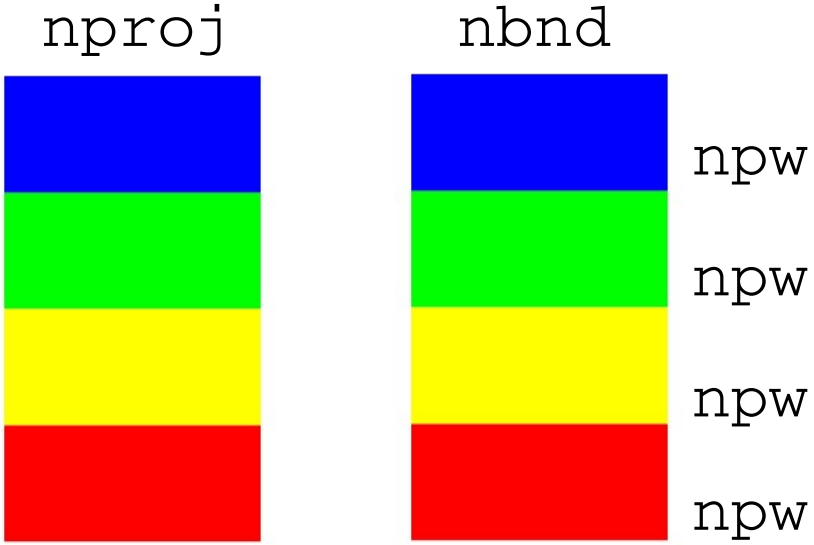
each processor has a partially summed betapsi

MPI – Message Passing Interface

a simple example

$$\langle \beta_i | \psi_j \rangle = \sum_{k+G} \beta_i^*(k+G)\psi_j(k+G)$$

beta(npw,nproj) psi(npw,nbnd)

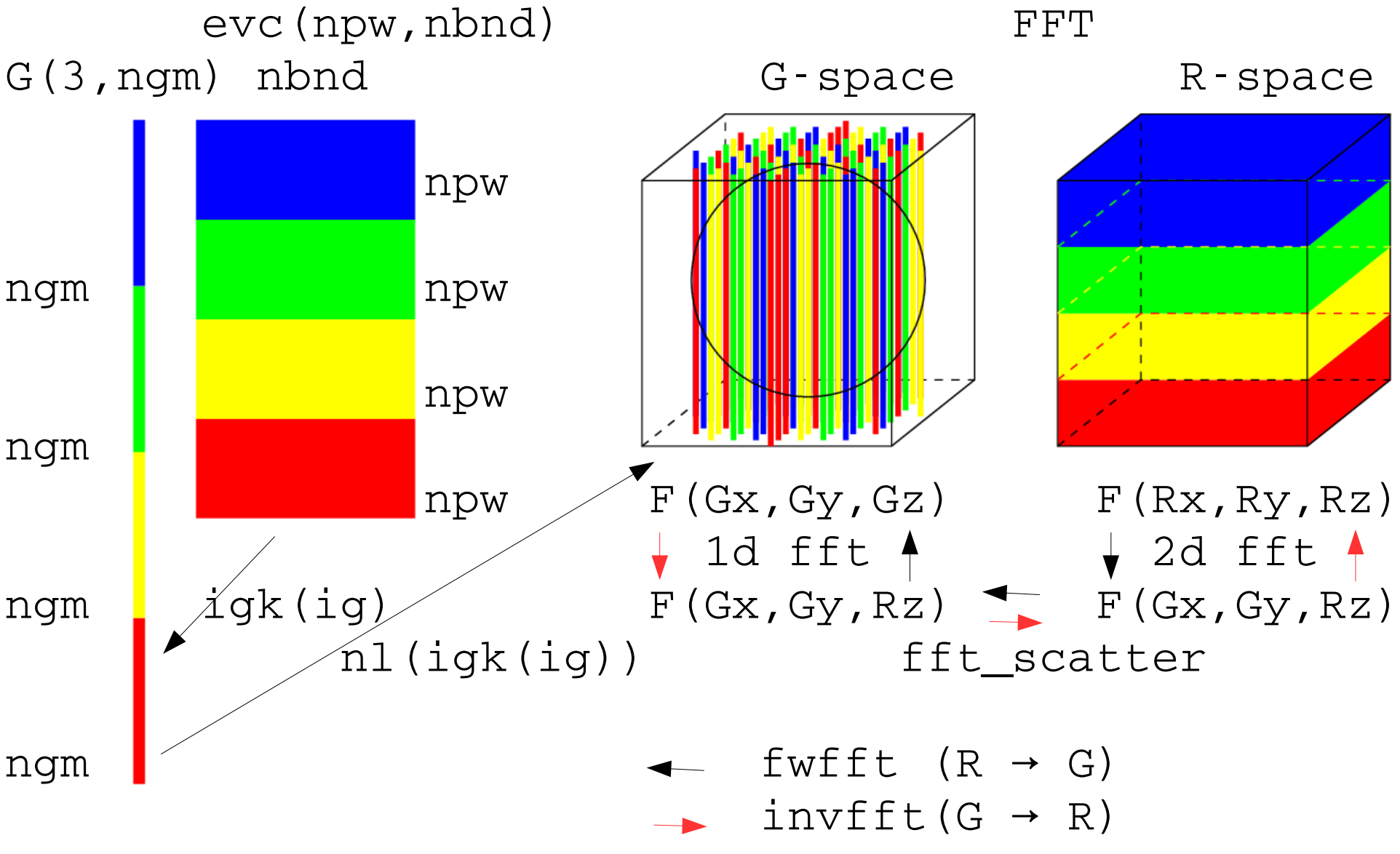


betapsi(nproj,nbnd)


```
CALL ZGEMM( 'C', 'N', nproj, nbnd, npw, (1.0_DP,0.0_DP), &
                  beta, npwx, psi, npwx, (0.0_DP,0.0_DP), &
                  betapsi, nprojx )
CALL mp_sum( betapsi, intra_bgrp_comm )
```

at the end each processor has the complete betapsi !

R & G parallelization



FFT scalability

- computation time

$$T_{\mathtt{comp}} \propto T_{\mathtt{scalar}}/N_{\mathtt{proc}}$$

- communication time

$$T_{\mathtt{comm}} \propto (N_{\mathtt{proc}} - 1)\,(\alpha + \beta N_{\mathtt{data}})$$

$$N_{\mathtt{data}} = N_{\mathtt{col/proc}} \times N_{\mathtt{plane/proc}}$$

$$N_{\mathtt{col/proc}} \propto \frac{nr1 \times nr2}{N_{\mathtt{proc}}} \qquad N_{\mathtt{plane/proc}} = \frac{nr3}{N_{\mathtt{proc}}}$$

- for large number of processors communication time becomes important and latency becomes dominant

- the number of processors that can be used efficiently is limited by nr3 grid dimension !

MPI – Message Passing Interface

hierarchy of parallelization in PW

```
mpirun -np $N pw.x -nk $NK -nb $NB -nt $NT -nd $ND
                                        < pw.in > pw.out
```

```
-nk (-npool, -npools)    # of k-point pools

-nb (-nband, -nbgrp, -nband_group) # of band groups

-nt (-ntg, -ntask_groups) # of FFT task groups
-nd (-ndiag, -northo,-nproc_diag, -nproc_ortho)
                          # of linear algebra groups
```

$N = $NK x $NB x $NT x $Nproc_R&G

MPI – Message Passing Interface

hierarchy of parallelization in PW

- **R & G space parallelization**

  data are distributed, communication of results is frequent. High communication needs, reduction of processor memory footprint.

MPI – Message Passing Interface

hierarchy of parallelization in PW

- **R & G space parallelization**

data are distributed, communication of results is frequent. High communication needs, reduction of processor memory footprint.

- **K-point parallelization**

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + V_{eff}(\mathbf{r},\mathbf{R})\right]\phi_{\mathbf{k}v}(\mathbf{r}) = \varepsilon_{\mathbf{k}v}\phi_{\mathbf{k}v}(\mathbf{r})$$

$$n(\mathbf{r}) = 2\sum_{\mathbf{k}v}|\phi_{\mathbf{k}v}(\mathbf{r})|^2$$

different k-points are completely independent during most operations. Needs to collect contributions from all k-points from time to time. Mild communication needs, no lowering of the processor memory footprint, unless all k-point are kept in memory...

MPI – Message Passing Interface

hierarchy of parallelization in PW

- **R & G space parallelization**
  data are distributed, communication of results is frequent. High communication needs, reduction of processor memory footprint.

- **K-point parallelization**
  different k-points are completely independent during most operations. Needs to collect contributions from all k-points from time to time. Mild communication needs, no lowering of the processor memory footprint, unless all k-point are kept in memory...

- **Band parallelization**
  different processors deal with different subset of the bands. Computational load distributed, no memory footprint reduction for now.

Some recent work on an alternative iterative method

A PARALLEL ORBITAL-UPDATING APPROACH FOR ELECTRONIC STRUCTURE CALCULATIONS *

XIAOYING DAI†, XINGAO GONG‡, AIHUI ZHOU† , AND JINWEI ZHU†

Abstract. In this paper, we propose an orbital iteration based parallel approach for electronic structure calculations. This approach is based on our understanding of the single-particle equations of independent particles that move in an effective potential. With this new approach, the solution of the single-particle equation is reduced to some solutions of independent linear algebraic systems and a small scale algebraic problem. It is demonstrated by our numerical experiments that this new approach is quite efficient for full-potential calculations for a class of molecular systems.

arXiv:1405.0260v2 [math.NA] 20/11/2014

A PARALLEL ORBITAL-UPDATING BASED OPTIMIZATION METHOD FOR ELECTRONIC STRUCTURE CALCULATIONS *

XIAOYING DAI†, ZHUANG LIU‡, XIN ZHANG§, AND AIHUI ZHOU¶

Abstract. In this paper, we propose a parallel optimization method for electronic structure calculations based on a single orbital-updating approximation. It is shown by our numerical experiments that the method is efficient and reliable for atomic and molecular systems of large scale over supercomputers.

arXiv:1510.07230v1 [math.NA] 25/10/2015

ParO method in a nutshell

•Given trial eigenpairs: $\{|\phi_i^{(n)}\rangle, \varepsilon_i^{(n)}\}$

•Solve <u>in parallel</u> the *nbnd* linear systems

$$(H_{KS} + \lambda S)|\tilde{\phi}_i^{(n)}\rangle = (\varepsilon_i^{(n)} + \lambda) S|\phi_i^{(n)}\rangle$$

•Build the reduced Hamiltonian

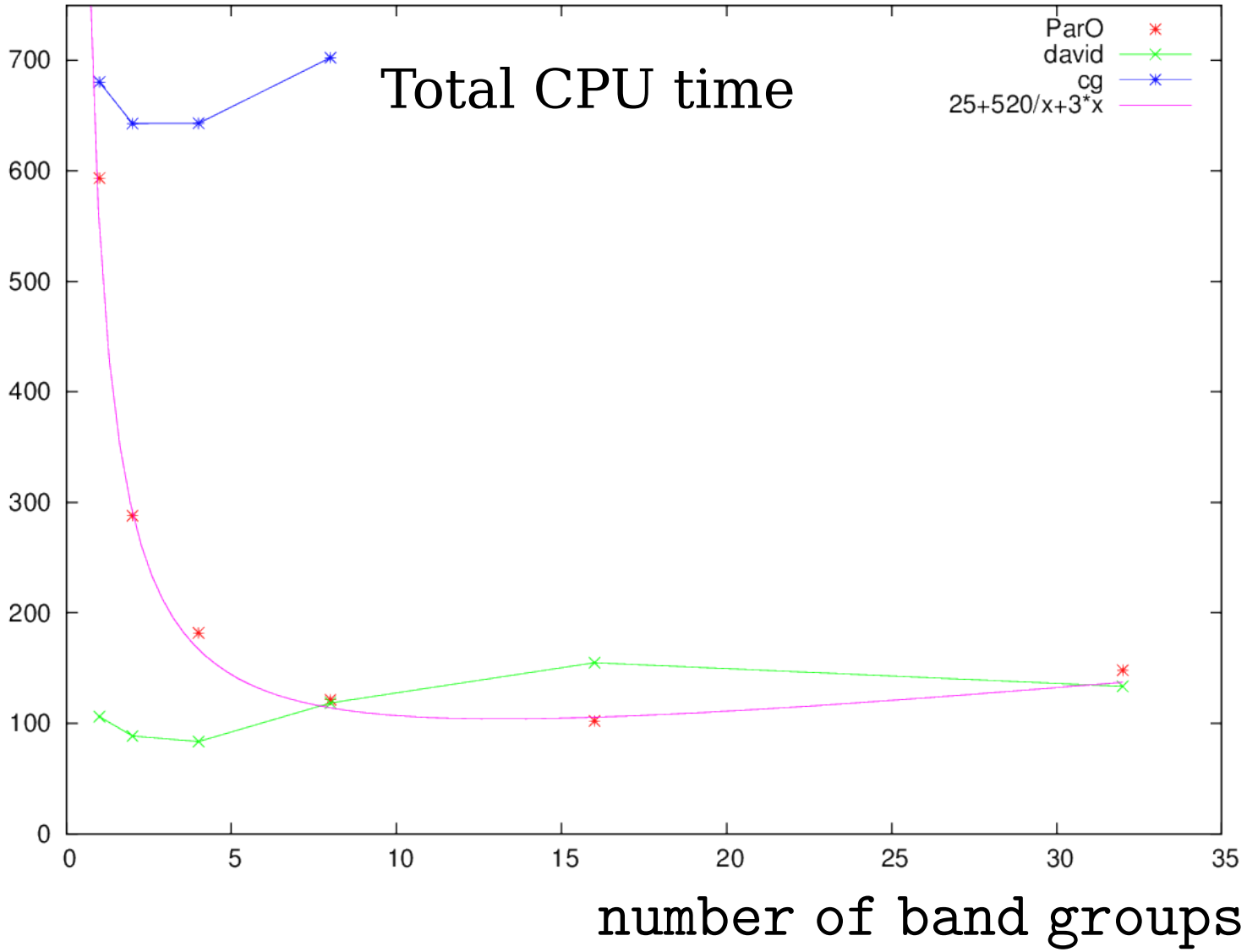
$$\tilde{H}_{ij} = \langle\tilde{\phi}_i^{(n)}|H_{KS}|\tilde{\phi}_j^{(n)}\rangle, \quad \tilde{S}_{ij} = \langle\tilde{\phi}_i^{(n)}|S|\tilde{\phi}_j^{(n)}\rangle$$

•Diagonalize the <u>small</u> *nbnd* x *nbnd* reduced Hamiltonian to get the new estimate for the eigenpairs

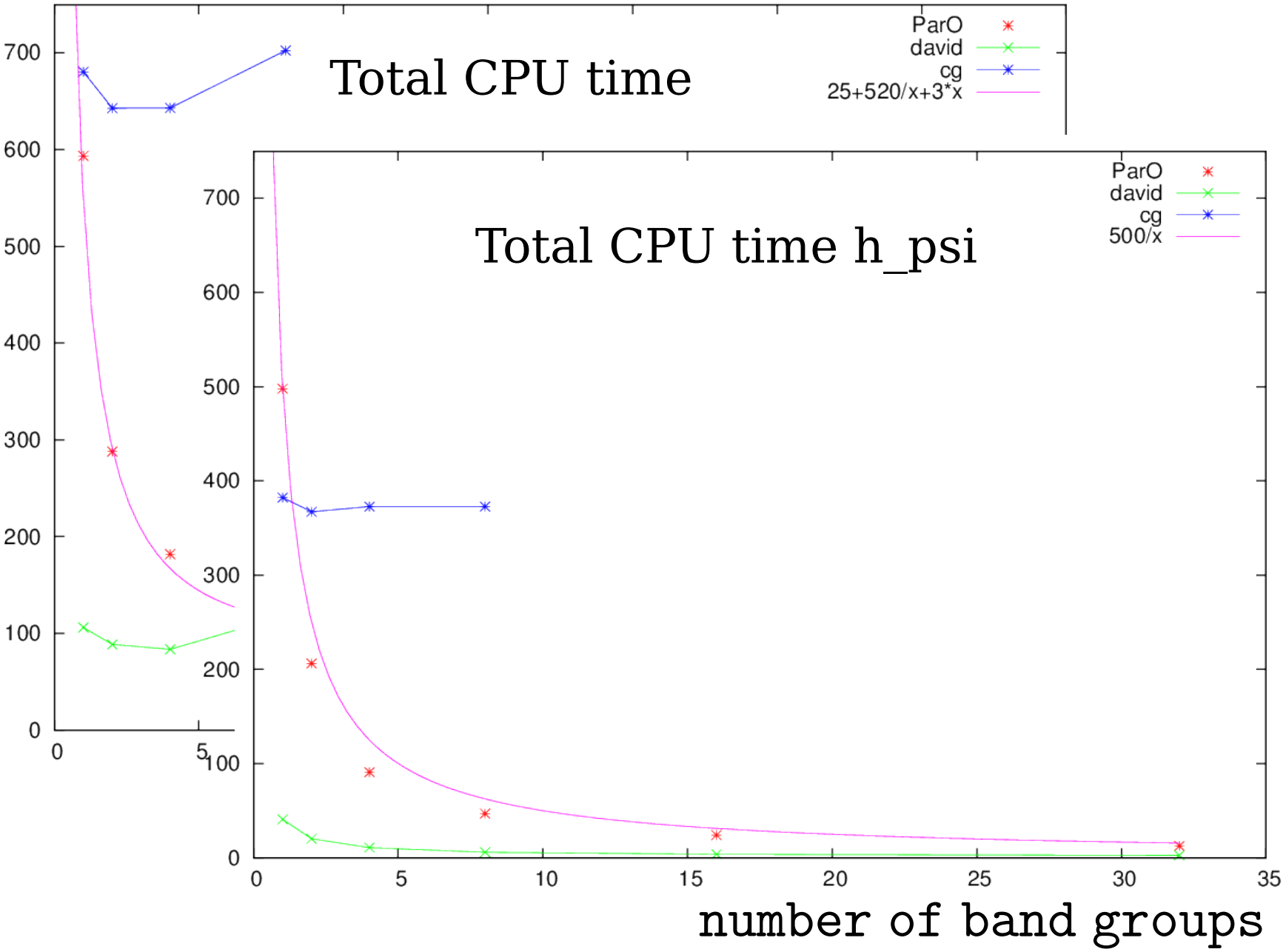
$$(\tilde{H} - \varepsilon\tilde{S})v = 0 \quad \longrightarrow \quad \{|\phi_i^{(n+1)}\rangle, \varepsilon_i^{(n+1)}\}$$

•Repeat if needed in order to improve solution at fixed Hamiltonian

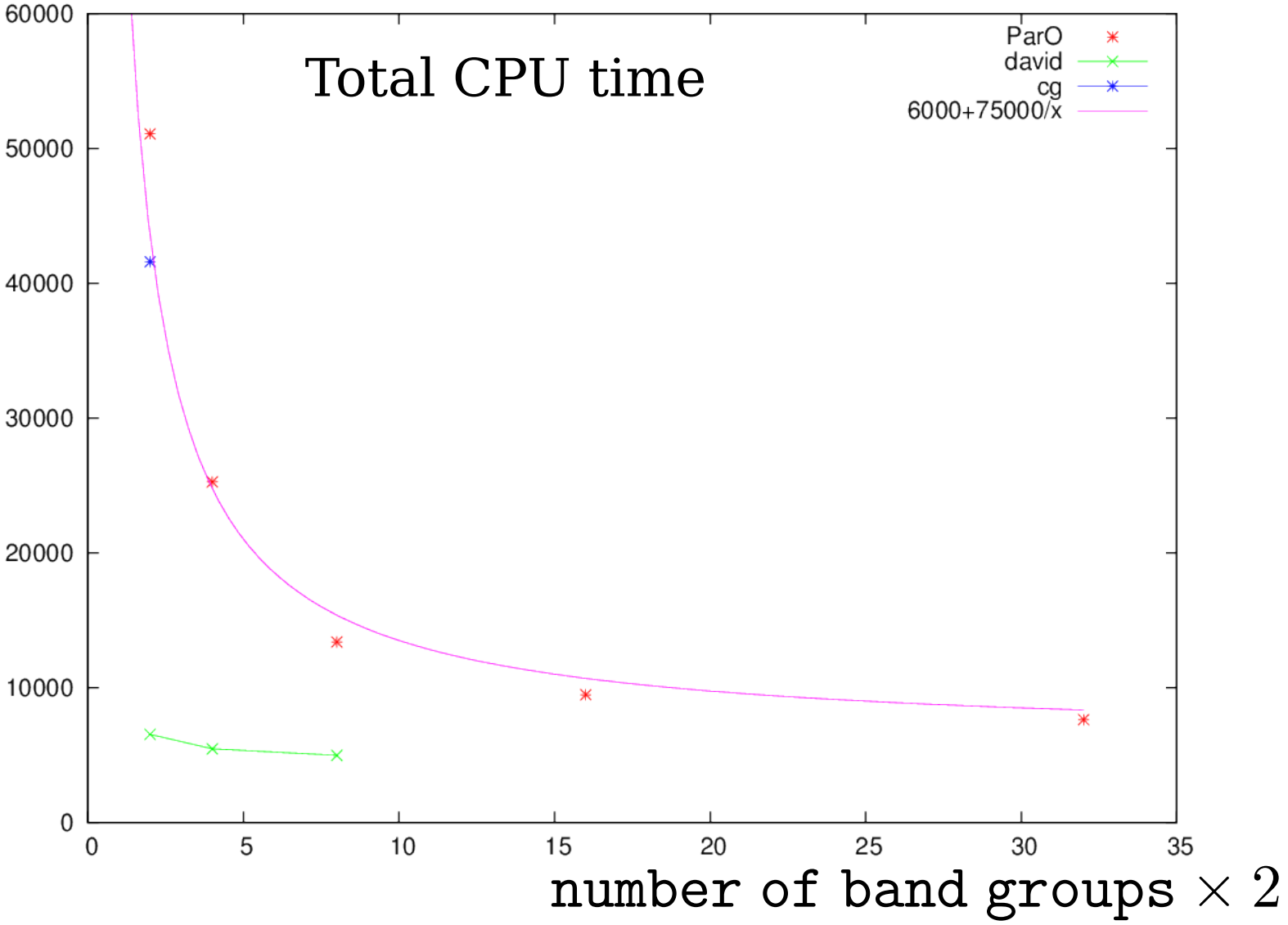
216 Si atoms in a SC cell : Timing



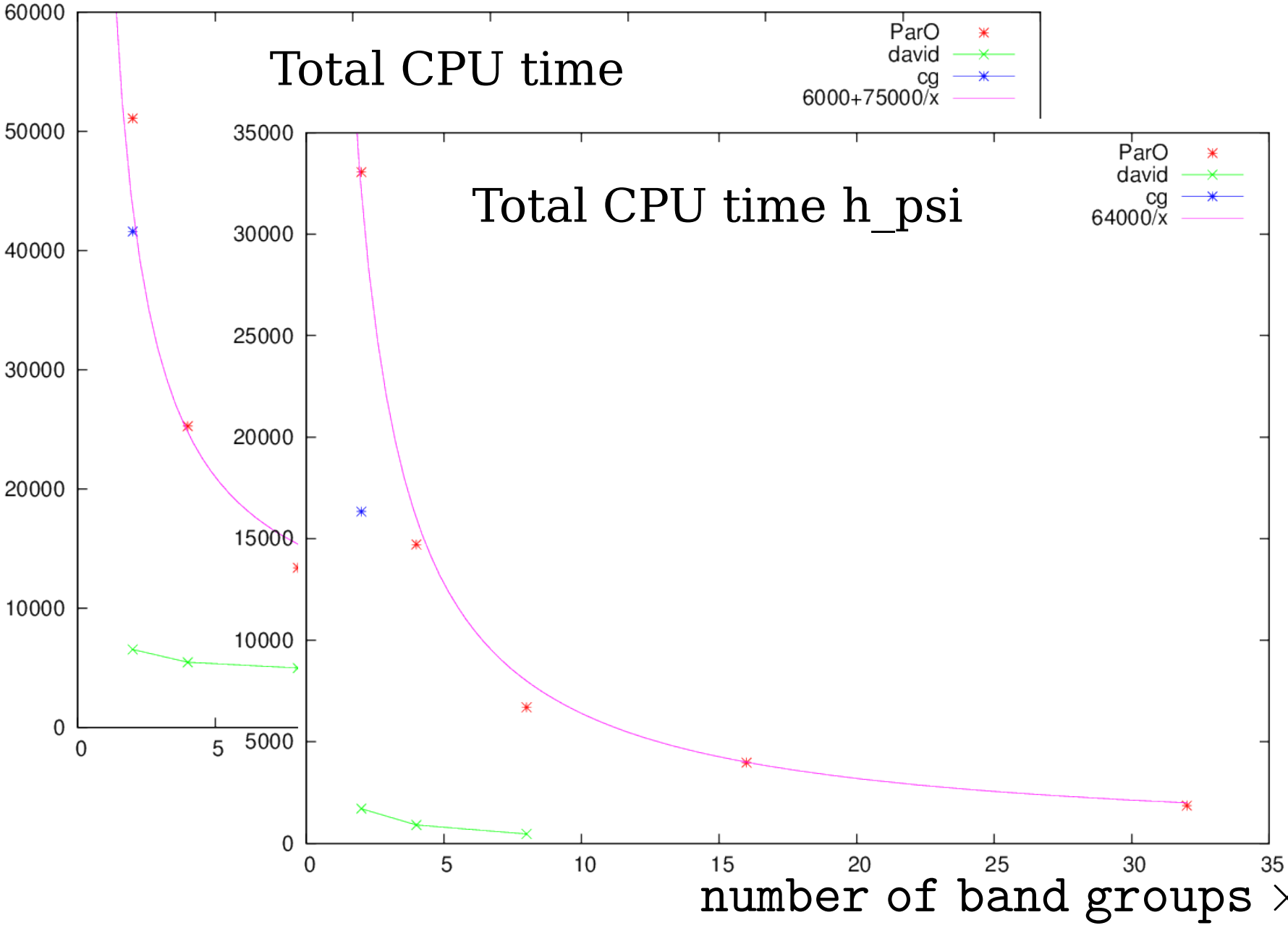
216 Si atoms in a SC cell : Timing



Not only Silicon: BaTiO3 320 atms, 2560 el

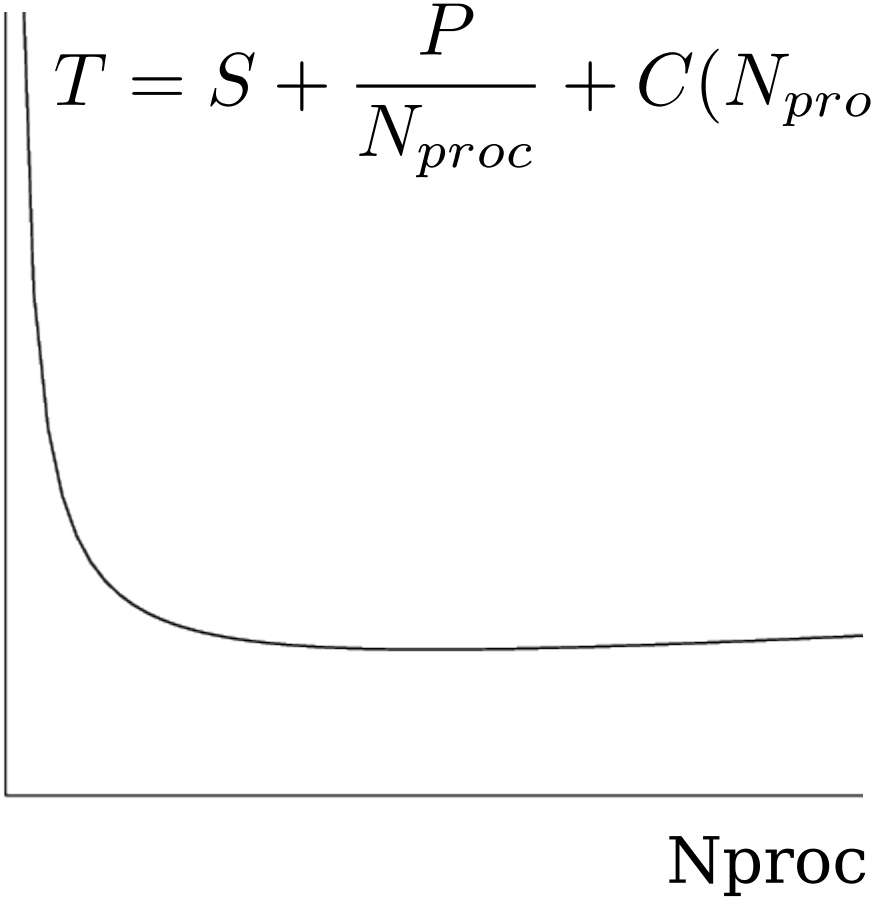


Not only Silicon: BaTiO3 320 atms, 2560 el



Amdahl's law

$$T = S + \frac{P}{N_{proc}} + C(N_{proc})$$



No matter how well you parallelize your code on the long run the scalar fraction dominates.

MPI – Message Passing Interface

additional levels of parallelization

- **linear algebra parallelization**
  diagonalization routines are parallelized.
  The paralelization is done by a dedicated parallel communicator. It only affects the diagonalization of the small dense matrices build by Davidson or ParO methods.

- **task group parallelization**
  FFT data are redistributed to perform multiple FFTs at the same time. Needed when number of processors is large compared with FFT dimension (nr3).
  It complicates the code significantly and interferes with band paralellization. Only applies to wfc FFTs, not to rho & potential FFTs.

MPI – Message Passing Interface

additional levels of parallelization

- **linear algebra parallelization**

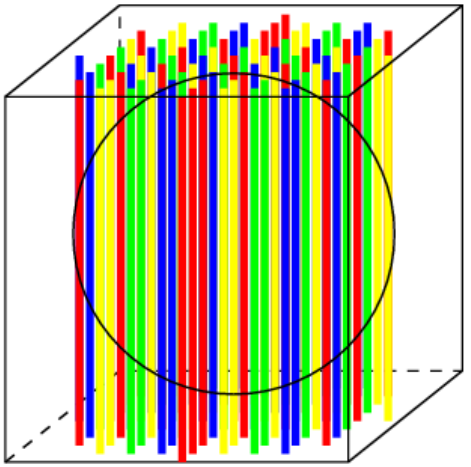
  diagonalization routines are parallelized.
  The paralelization is done by a dedicated parallel communicator. It only affects the diagonalization of the small dense matrices build by Davidson or ParO methods.

- **task group parallelization**

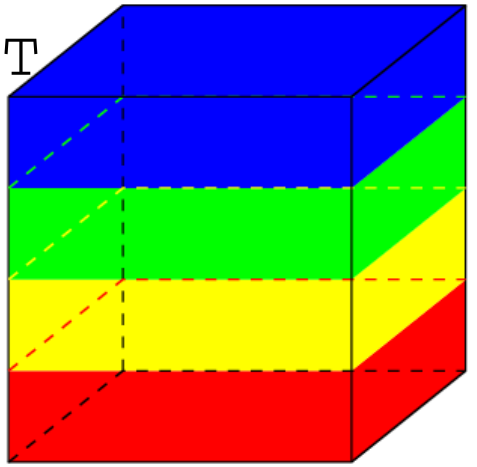
  FFT data are redistributed to perform multiple FFTs at the same time. Needed when number of processors is large compared with FFT dimension (nr3).
  It complicates the code significantly and interferes with band paralellization. Only applies to wfc FFTs, not to rho & potential FFTs.

A different R&G distribution might be more useful

Old FFT distribution



FFT



F(Gx,Gy,Gz)

1d fft

F(Gx,Gy,Rz)

F(Rx,Ry,Rz)

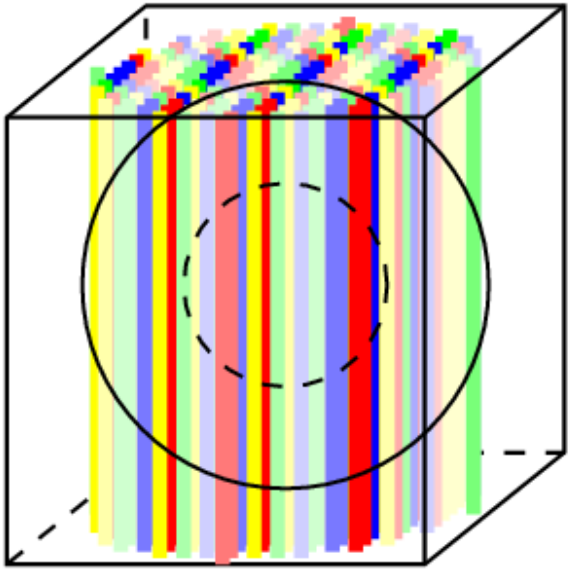
2d fft

F(Gx,Gy,Rz)

fft_scatter

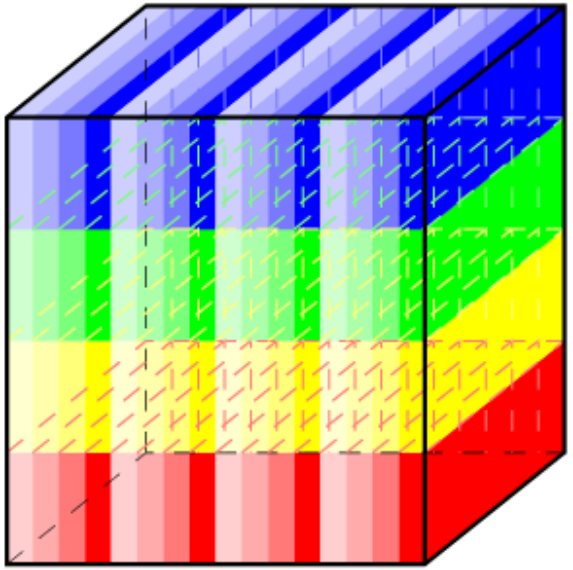
New FFT distribution

fft_scatter



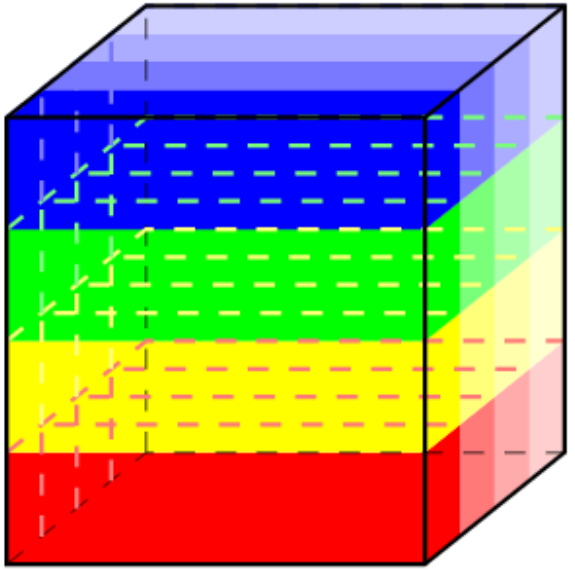
F(Gx,Gy,Gz)

F(Gx,Gy,Rz)



F(Gx,Gy,Rz)

F(Gx,Ry,Rz)



F(Gx,Ry,Rz)

F(Rx,Ry,Rz)

Extended FFT scalability $N_{\mathtt{proc}} = N_{\mathtt{P}_1} \times N_{P_2}$

- computation time

$$T_{\mathtt{comp}} \propto T_{\mathtt{scalar}}/N_{\mathtt{proc}}$$

- communication time

$$T_{\mathtt{comm}} \propto (N_{\mathtt{P}_1} - 1)\left(\alpha + \beta N^{(1)}_{\mathtt{data}}\right) + (N_{\mathtt{P}_2} - 1)\left(\alpha + \beta N^{(2)}_{\mathtt{data}}\right)$$

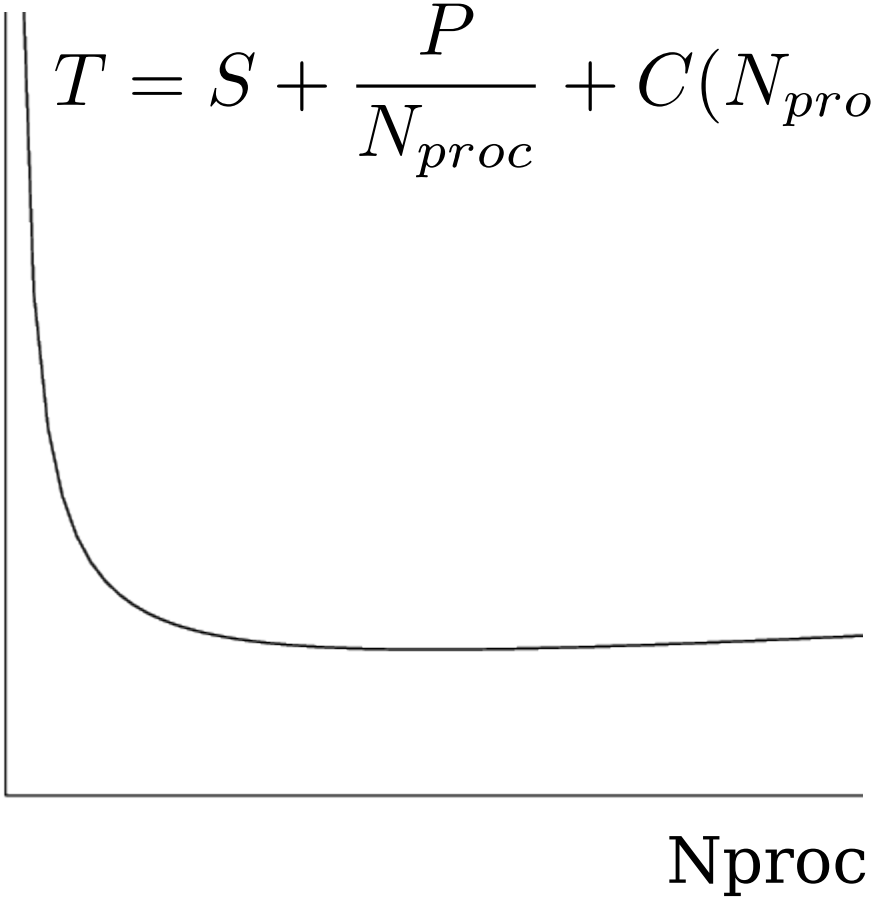
$$N^{(1)}_{\mathtt{data}} \propto \frac{nr1 \times nr2 \times nr3}{N_{\mathtt{proc}} N_{\mathtt{P}_1}}, \quad N^{(2)}_{\mathtt{data}} \propto \frac{nr1 \times nr2 \times nr3}{N_{\mathtt{proc}} N_{\mathtt{P}_2}}$$

- for large number of processors communication time grows more slowly, ideally as $\sqrt{N_{\mathtt{proc}}}$

- the number of processors that can be used efficiently could be extended well beyond nr3 grid dimension !

Amdahl's law

$$T = S + \frac{P}{N_{proc}} + C(N_{proc})$$



No matter how well you parallelize your code on the long run the scalar fraction dominates.

Conclusions

- Several levels of parallelization implemented
- R&G data ditribution, parallel FFT .
- k-point parallelization
- bgrp parallelization / ParO method
- Careful profiling of the code.
- extend bgrp parallelization to other parts of te code by distributing different loops in different routines (nats, nkb, ngm, nrxx, …). Only local effects: incremental!
- diag parallelization
- Except for some memory reduction there is not much gain in parallelizing dense matrix diagonalization.
- New algorithms with better communication are needed.
- task parallelization/ new FFT distribution
- Test the new FFT distribution and verify the improved scalability.
- Verify the compatibility with task-groups.