Efficient van der Waals functionals and other GPAW developments

Ask Hjorth Larsen[1,2]
Mikael Kuisma[3]
Paul Erhart[3]
Per Hyldgaard[3]

[1] Nano-bio Spectroscopy Group and ETSF Scientific development centre, Universidad del País Vasco UPV/EHU,
[2] Departament de Ciència de Materials i Química Física & Institut de Química Teòrica i Computacional (IQTCUB), Universitat de Barcelona, c /Martí i Franquès 1, 08028 Barcelona, Spain,
[3] Department of Applied Physics, Chalmers University of Technology

June 7, 2016

"Recent" developments

"New" features

- SG15 norm-conserving pseudopotentials
- `gpaw install-data`: Download and "autoinstall" PAW datasets, basis sets, pseudopotentials
- interface to libvdwxc for van der Waals functionals

Parallelization improvements

- load-balanced efficient atomic corrections to LCAO Hamiltonian
- load-balanced atomic corrections (of ΔH_{asp}) on all cores
- redistribute data among arbitrary grids (e.g. for libvdwxc)
- fine-grid quantities ($\tilde{n}$, $\tilde{\rho}$, $\tilde{v}$) parallel on all cores

van der Waals functionals

Form

- LDA correlation
- GGA exchange
- Plus non-local correlation:

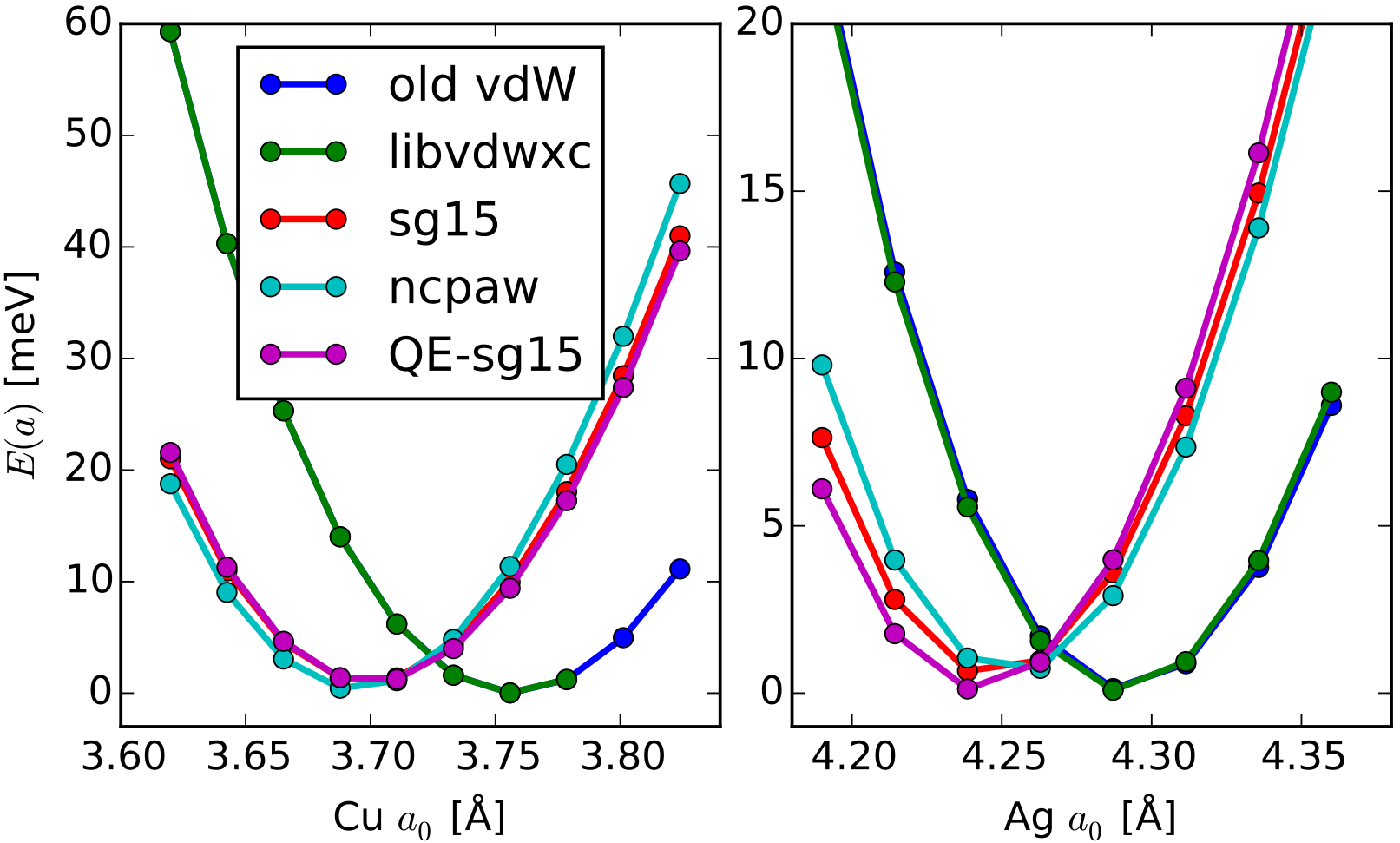
$$E_{\mathrm{c}}^{\mathrm{NL}}[n] = \frac{1}{2} \iint n(\mathbf{r}) \phi[n](\mathbf{r}, \mathbf{r}') n(\mathbf{r}) \, \mathrm{d}\mathbf{r} \, \mathrm{d}\mathbf{r}'$$

- The vdW-DF family of functionals are true non-local density functionals, and not “corrections”.
- M. Dion, H. Rydberg, E. Schröder, D. C. Langreth, and B. I. Lundqvist, Phys. Rev. Lett. **92**, 246401 (2004)
- K. Berland and P. Hyldgaard, Phys. Rev. B **89**, 035412 (2014)

Motivation for libvdwxc

- Different codes have different implementations, numerics, van der Waals kernel parametrizations, making comparisons difficult
- We want to eventually resolve the PAW/vdW discrepancy
- Old GPAW vdW-DF implementation is not scalable beyond 20 cores
- We would also like vdW-DF in Octopus
- But we don't really like FORTRAN :(
- Conclusion: Write single reusable C library in the style of libxc

Motivation for libvdwxc



The PAW vdW discrepancy: Binding curves for Cu, Ag

libvdwxc

- Development: https://gitlab.com/libvdwxc/libvdwxc
- Documentation: http://libvdwxc.readthedocs.io
- Features: Non-local energy of vdW-DF family of functionals: (vdW-DF, vdW-DF2, vdW-DF-CX)
- Requirements: FFTW3 or FFTW3-MPI + MPI
- Uses kernel parametrization from Quantum Espresso (so far!)
- Compile GPAW with libvdwxc and use like this:

```
from ase.collections import g2
from gpaw import GPAW
from gpaw.xc.libvdwxc import vdw_df_cx

atoms = g2['CH3CH2OH']
atoms.center(vacuum=5.0)
atoms.calc = GPAW(xc=vdw_df_cx(),
                  txt='gpaw.txt')
atoms.get_potential_energy()
```

$\phi(\mathbf{r}, \mathbf{r}')$ and $q_0(\mathbf{r})$

- The kernel has the form $\phi(q_0(\mathbf{r}), q_0(\mathbf{r}'), \|\mathbf{r} - \mathbf{r}'\|)$
- q_0 is some GGA-like quantity: $q_0(\mathbf{r}) = q_0(n(\mathbf{r}), \|\boldsymbol{\nabla} n(\mathbf{r})\|)$
- The kernel can be pre-parametrized and stored in a file
- During a calculation, the double space integral must still be evaluated
- The double space integral is expensive but can be rewritten as a convolution using the method by Román-Pérez and Soler

The Román-Pérez–Soler method

- Discretize q_0 to 20-point grid (typically) and expand the kernel in $20 \times 20 = 400$ terms

$$\phi(q_0, q_0', d) = \sum_{\alpha,\beta=1}^{20} \phi_{\alpha\beta}(d) p_\alpha(q_0) p_\beta(q_0').$$

- p_α are fixed auxiliary functions to interpolate values on the coarse q_0 grid
- Let $\theta_\alpha(\mathbf{r}) = n(\mathbf{r}) p_\alpha(q_0(\mathbf{r}))$, and the energy becomes a convolution

$$E_{\mathrm{c}}^{\mathrm{nl}} = \frac{1}{2} \sum_{\alpha\beta} \iint \theta_\alpha(\mathbf{r}) \phi_{\alpha\beta}(\|\mathbf{r} - \mathbf{r}'\|) \theta_\beta(\mathbf{r}') \, \mathrm{d}\mathbf{r} \, \mathrm{d}\mathbf{r}',$$

$$= \frac{1}{2} \sum_{\alpha\beta} \int \theta_\alpha^*(\mathbf{k}) \theta_\beta(\mathbf{k}) \phi_{\alpha\beta}(k) \, \mathrm{d}\mathbf{k}.$$

Full algorithm

- Calculate $q_0(\mathbf{r})$ and $\theta_\alpha(\mathbf{r})$
- Transform $\theta_\alpha(\mathbf{r})$ to Fourier space $\rightarrow \theta_\alpha(\mathbf{k})$
- Calculate energy from convolution as well as derivative

$$F_\alpha(\mathbf{k}) = \sum_\beta \theta_\beta(\mathbf{k}) \phi_{\alpha\beta}(k) \tag{1}$$

- Transform $F_\alpha(\mathbf{k})$ back to $F_\alpha(\mathbf{r})$
- Evaluate potential using $F_\alpha(\mathbf{r})$

Main performance issue Fourier transforms

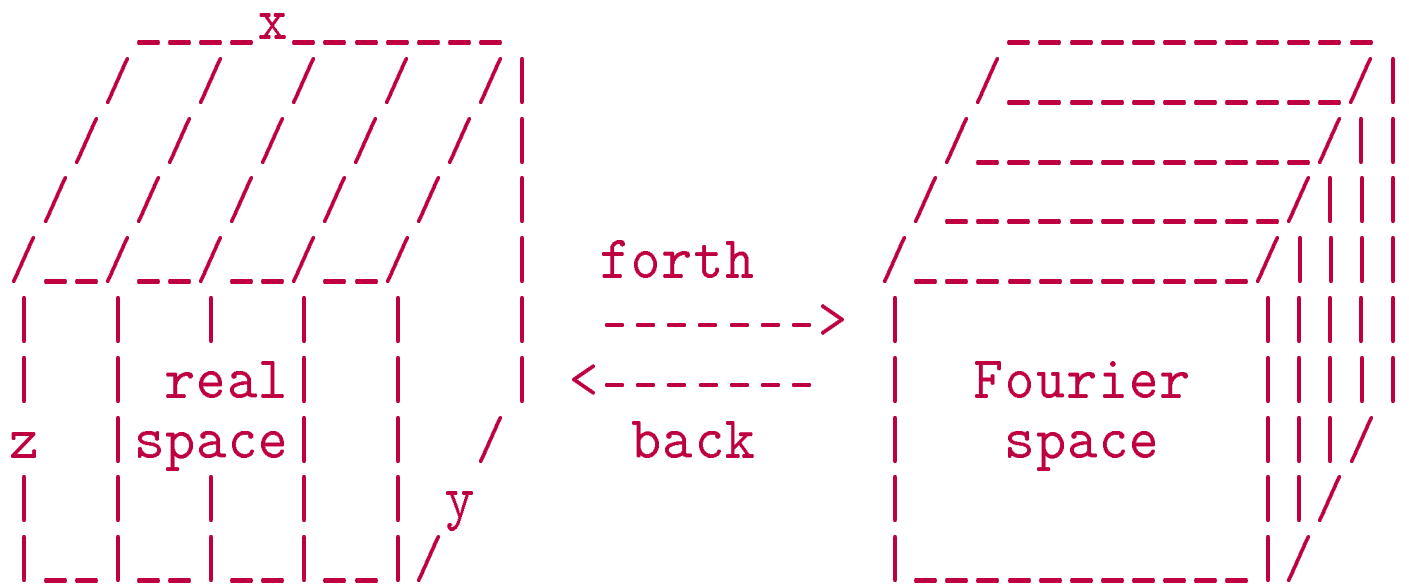
Parallel 3D Fourier transforms with FFTW3-MPI

Strategy

$$\widehat{f}(\mathbf{k}) = \iiint f(\mathbf{x}) \exp(-i\mathbf{k}\cdot\mathbf{x}) \, \mathrm{d}x_1 \, \mathrm{d}x_2 \, \mathrm{d}x_3$$

- A single FFT does not parallelize well
- What we really have is a 3D array of 1D transforms
- Different cores can do different (but whole) 1D transforms
- Luckily we can use FFTW-MPI for all this
- (Remaining operations are local!)

Parallel Fourier transforms



Algorithm

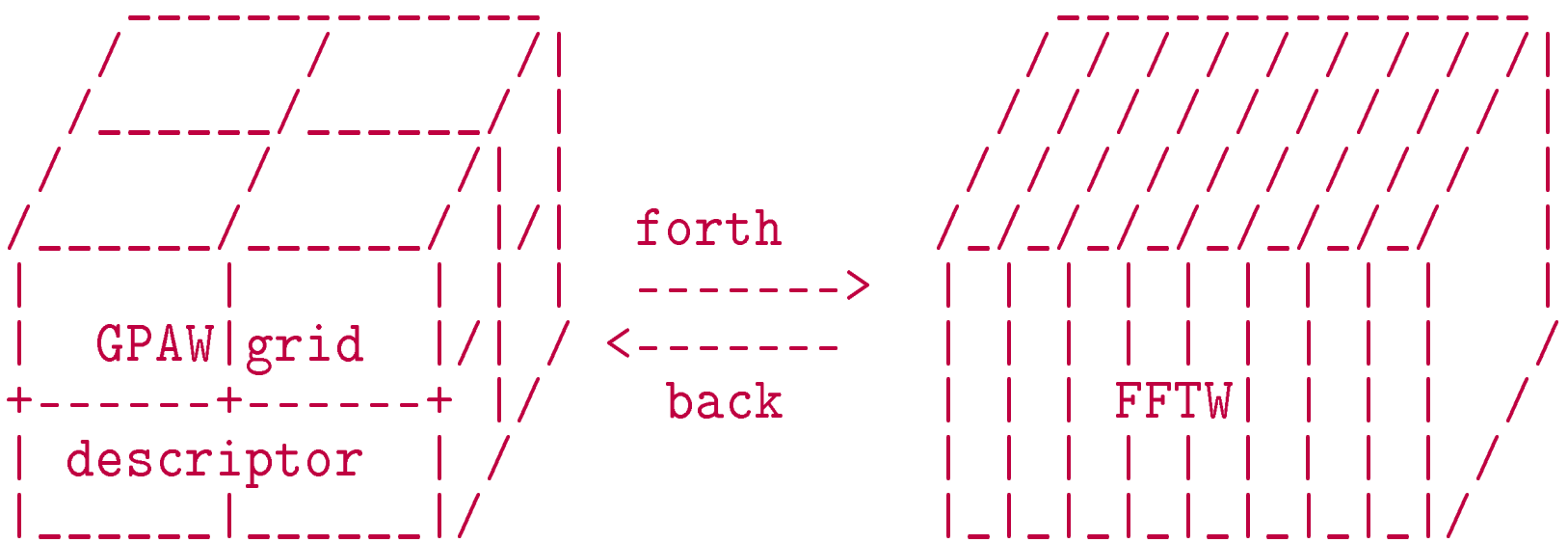
- Get data into block distribution over x only
- Take Fourier transform over y and z
- Transform to block distribution over y (parallel transpose)
- Fourier transform over x

Performance tricks

- FFT output goes into input buffer
- $\theta_\alpha(\mathbf{r})$ and friends are *strided* (α axis is contiguous, not $\mathbf{r}$)
- Work on transposed output
- In general, write non-silly loops (memory order, buffers)

(None of this is advanced in any way!)

Distribution from GPAW to FFT format



Performance and old vdW bottleneck

XC 3D grid:	10312.154	268.410	2.4%
VdW-DF integral:	10043.744	84.202	0.8%
Convolution:	61.796	61.796	0.6%
FFT:	120.897	120.897	1.1%
gather:	8655.951	8655.951	77.5%
iFFT:	178.561	178.561	1.6%

- Performance on molecule from S22 dataset
- Old implementation: 3 hours, 320 cores
- libvdwxc: less than half an hour, 32 cores

Parallelization in GPAW

3D main CPU mesh

- k-points/spins $\psi_{\underline{k}n}(\mathbf{r})$
- bands/orbitals $\psi_{k\underline{n}}(\mathbf{r})$, $H_{\underline{\mu}\nu}$, $c_{\mu\underline{n}}$
- Domains $\psi_{kn}(\underline{\mathbf{r}})$

Temporary redistributions to "world"

- ScaLAPACK $H_{\underline{\mu\nu}}$, $c_{\underline{\mu n}}$ (old)
- Atomic quantities ΔH_{asp} (new)
- Fine-grid (Poisson, XC) (new)

Parallel timings

Write timings:

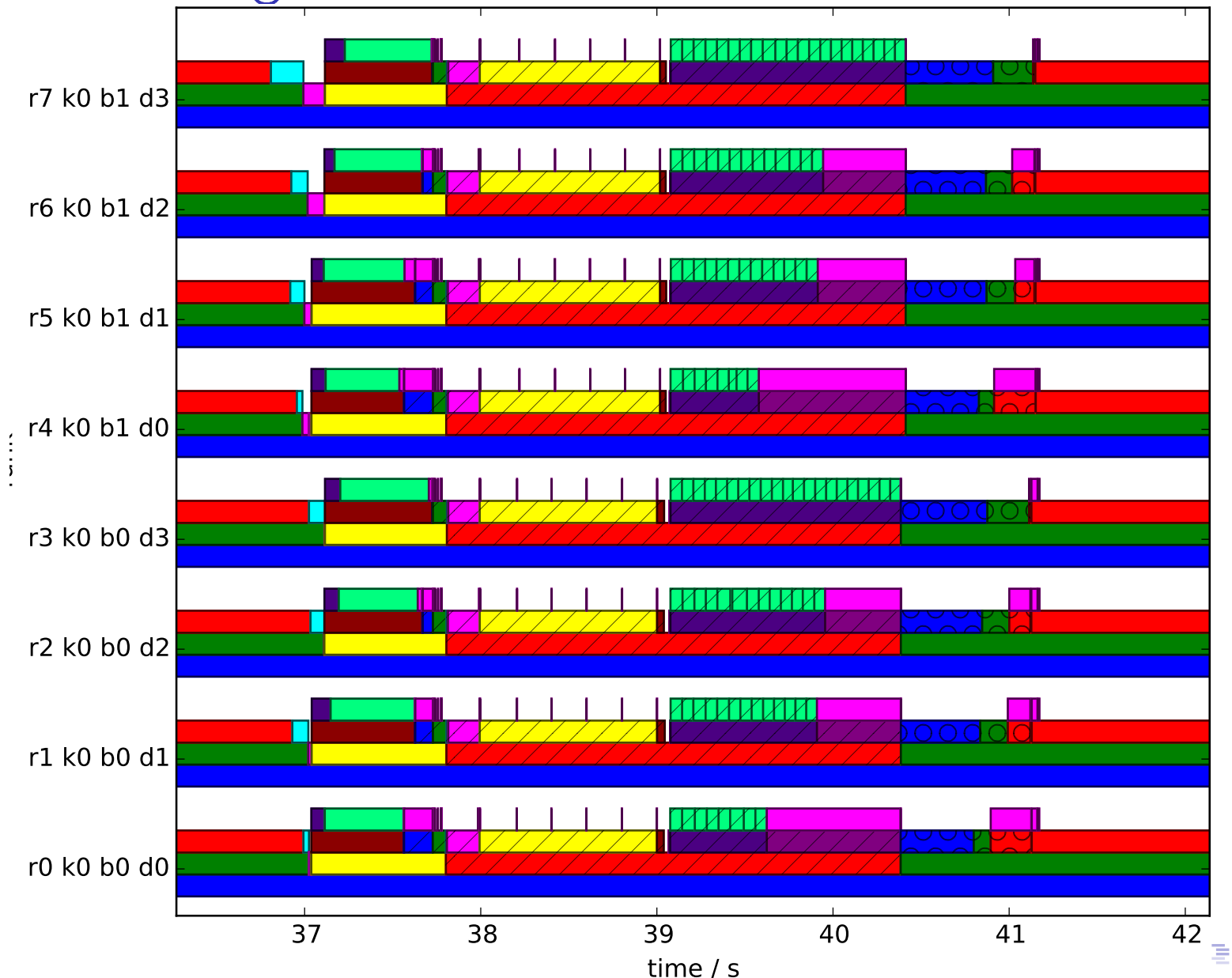
```
from ase.io import read
from gpaw import GPAW
from gpaw.utilities.timing import ParallelTimer

system = read('055.cube')
system.calc = GPAW(mode='lcao',
                   basis='dzp',
                   timer=ParallelTimer())
system.get_potential_energy()
```

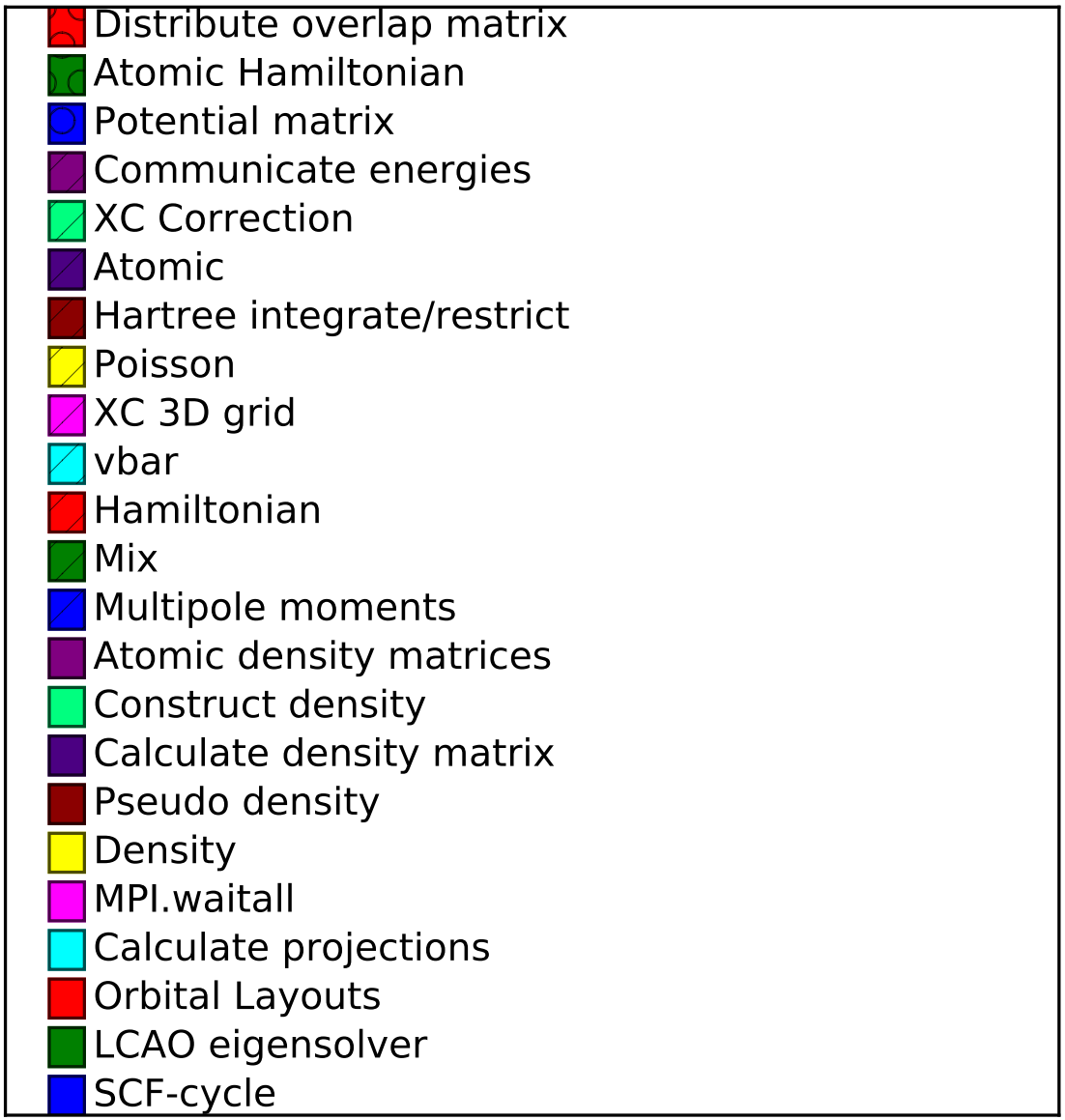
Plot timings:

```
gpaw-plot-parallel-timings timings.*.txt
    --interval=20:50
```

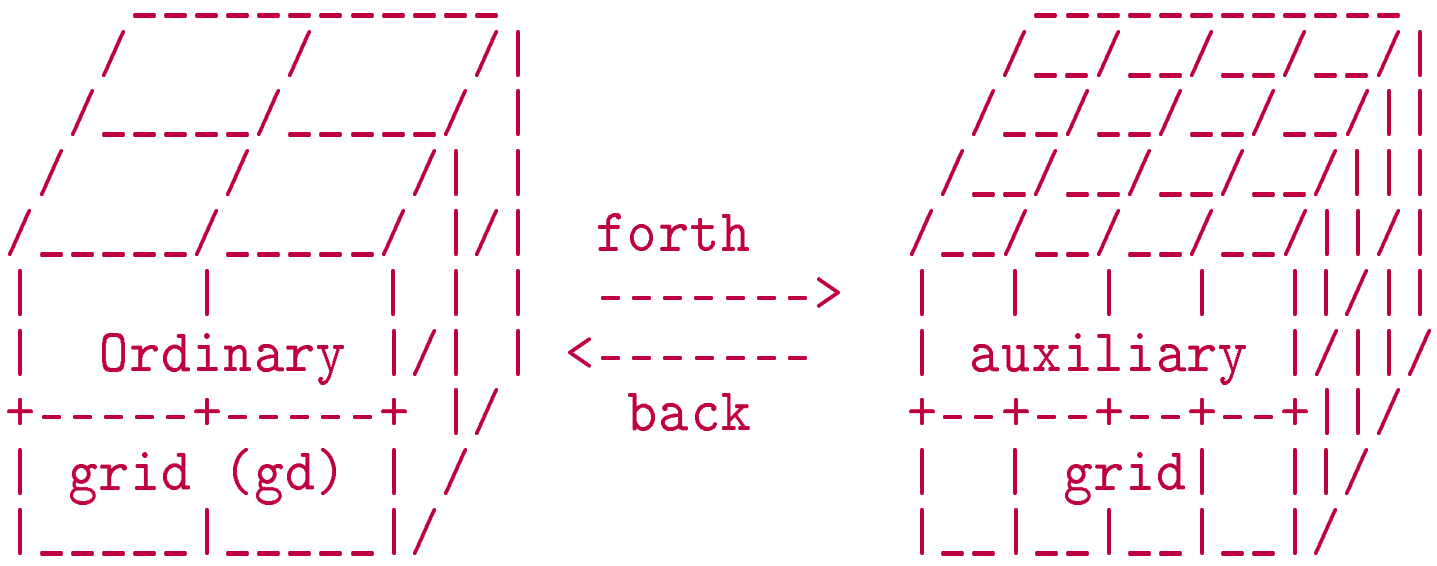
Parallel timings



Parallel timings



Finegrid operations on `world`



- Use “spare” cores from k-point/band parallelization for grid-only operations (XC, Poisson)
- Redistribute from band/k-point comm masters to world
- Then interpolate to fine grid
- Fine grid `finegd` now supports this distribution!

Example

```
from ase.collections import g2
from gpaw import GPAW

system = g2['H2O']
system.center(vacuum=3.0)
system.calc = GPAW(mode='lcao',
                   basis='dzp',
                   parallel=dict(band=2,
                                 augment_grids=True))
system.get_potential_energy()
```

In GPAW output:

```
Total number of cores used: 8
Domain Decomposition: 1 x 2 x 2 (coarse grid)
                      2 x 2 x 2 (fine grid)
```

```
askhl@loki:~$ gpaw install-data
Available setups and pseudopotentials
  [*] https://wiki.fysik.dtu.dk/gpaw-files/gpaw-setups-0.9.20000.tar.gz
      https://wiki.fysik.dtu.dk/gpaw-files/gpaw-setups-0.9.11271.tar.gz
      https://wiki.fysik.dtu.dk/gpaw-files/gpaw-setups-0.9.9672.tar.gz
      https://wiki.fysik.dtu.dk/gpaw-files/gpaw-setups-0.8.7929.tar.gz
      https://wiki.fysik.dtu.dk/gpaw-files/gpaw-setups-0.6.6300.tar.gz
      https://wiki.fysik.dtu.dk/gpaw-files/gpaw-setups-0.5.3574.tar.gz

Current GPAW setup paths in order of search priority:
   1. /home/askhl/install/gpaw-basis-NAO-sz+coopt-NGTO-0.9.11271
   2. /home/askhl/install/gpaw-basis-pvalence-0.9.11271
   3. /home/askhl/install/sg15_oncv_upf_2015-05-20
   4. /home/askhl/install/gpaw-setups-0.9.11271
   5. /usr/local/share/gpaw-setups
   6. /usr/share/gpaw-setups

Run gpaw DIR to install newest setups into DIR.
Run gpaw DIR --version=VERSION to install VERSION (from above).
See gpaw --help for more info.
```

More on performance

Obvious performance improvements mostly done now!

PW bottlenecks

- Timings from high-precision S22 test ($29 \times 21 \times 21$ Å)
- XC; no longer a problem with finegrid on world
- FFT mixer 30%
- Unbenchmarked part of Hamiltonian 25%
- Poisson — extremely fast but serial

Davidson

- When will it have band parallelization?

Thank you for listening!

https://gitlab.com/libvdwxc/libvdwxc