Numerical Methods for DFT and Beyond

Ville Havu

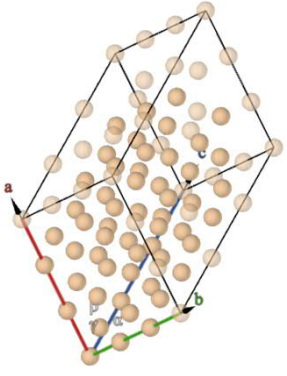
Wednesday, 27 October 2021

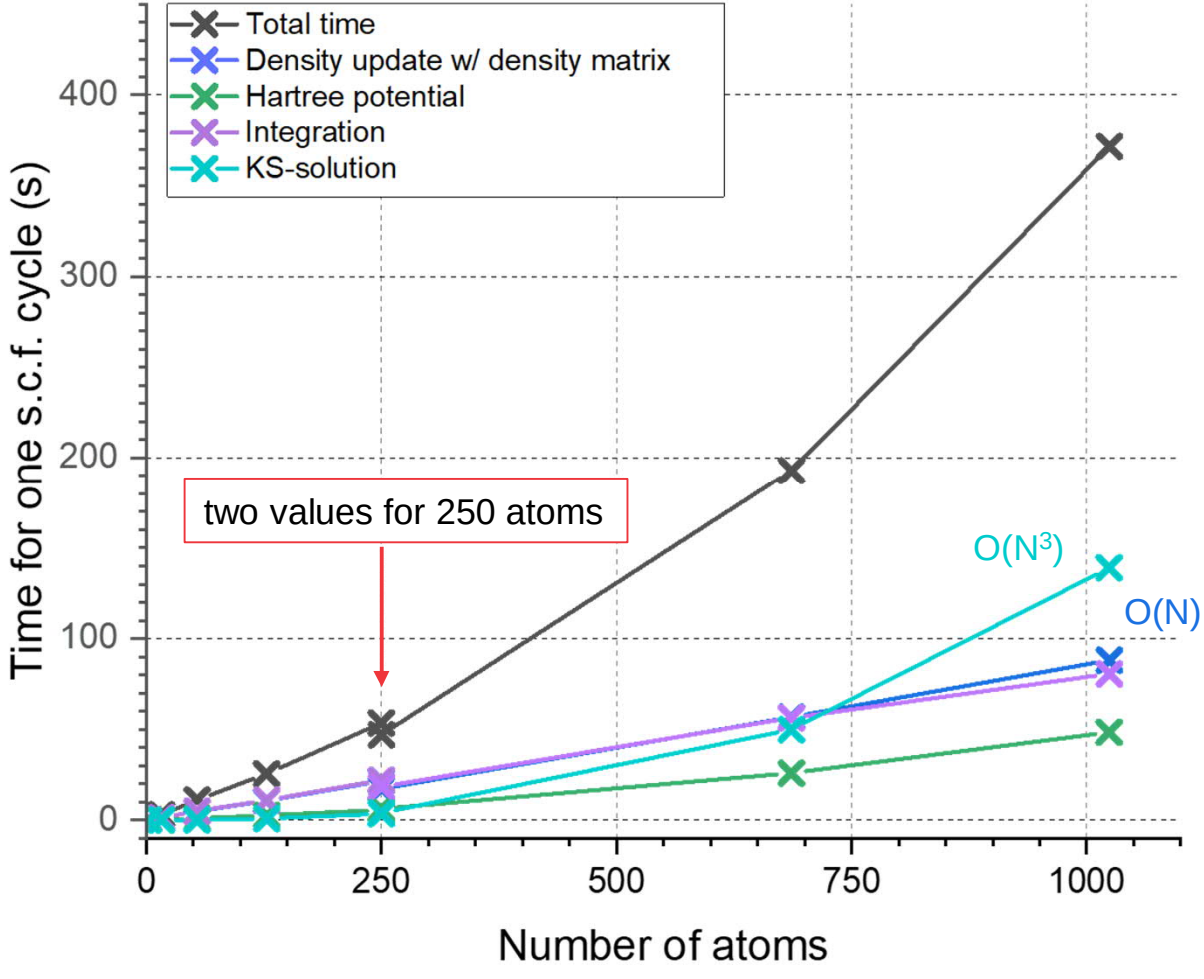
Aalto University
School of Science

First look

Bulk Si, tight settings, 2 – 1024 atoms in a unit cell, 1 k-point, 128 MPI-tasks (2 x 64 AMD Rome), 256 GB memory

gcc, OpenMPI, OpenBLAS, ScaLAPACK

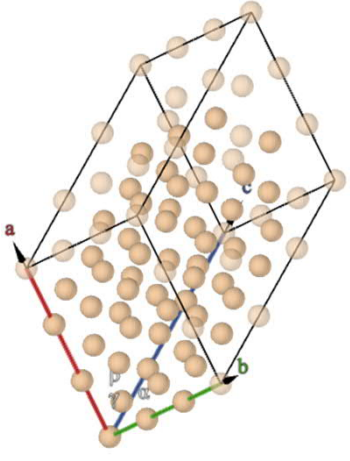


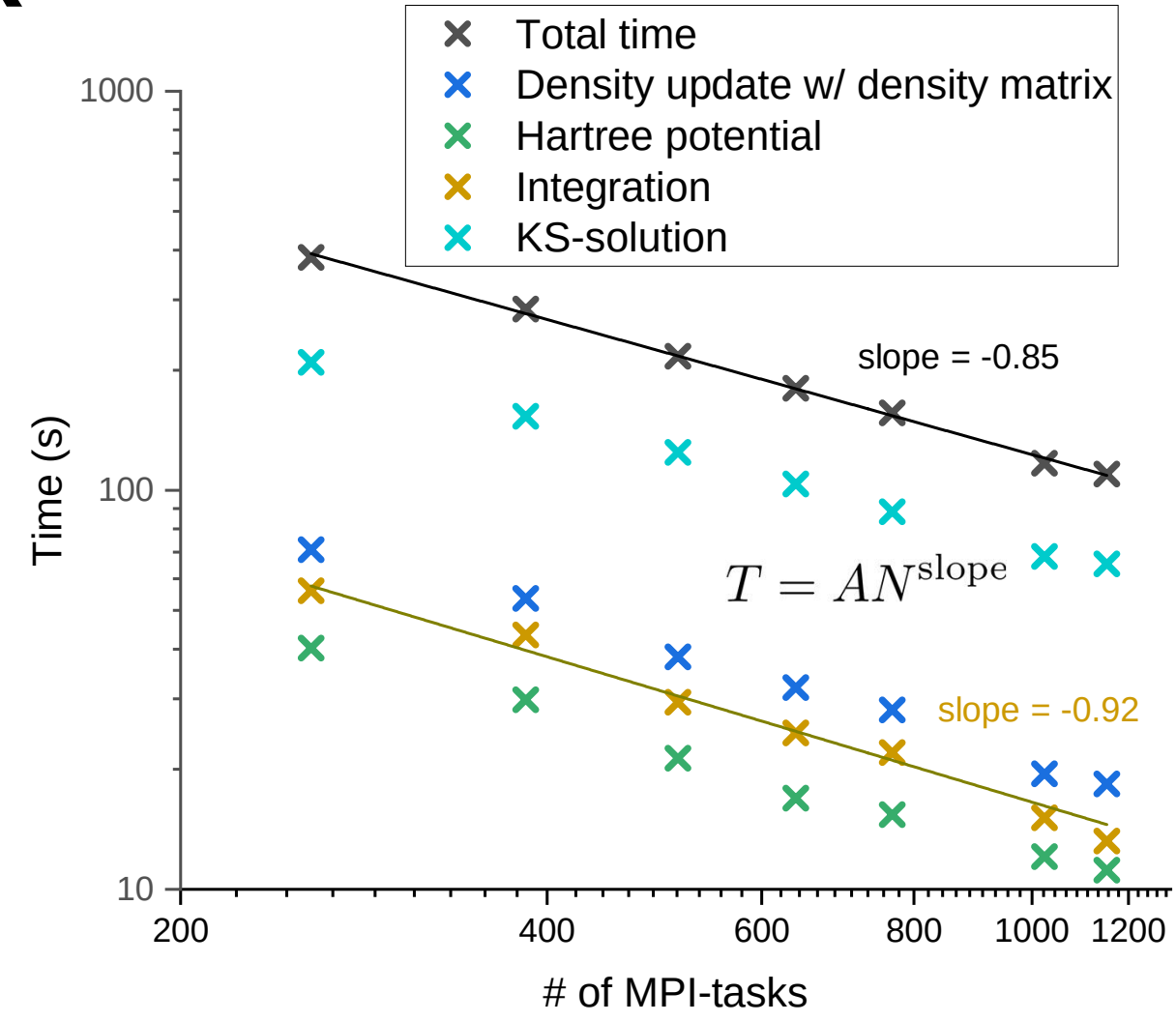


Aalto University
School of Science

Second look

Bulk Si, tight settings, 1458 atoms
in a unit cell
256 - 1152 MPI-tasks





Aalto University
School of Science

Integration in DFT

Integration – calculation of matrix elements

$$h_{ij} = \int d\mathbf{r}\, \varphi_i(\mathbf{r}) h_{KS}(\mathbf{r}) \varphi_j(\mathbf{r}) \qquad h_{KS}(\mathbf{r}) = -\frac{1}{2}\Delta + V_{\text{eff}}(\mathbf{r})$$

$$S_{ij} = \int d\mathbf{r}\, \varphi_i(\mathbf{r}) \varphi_j(\mathbf{r})$$

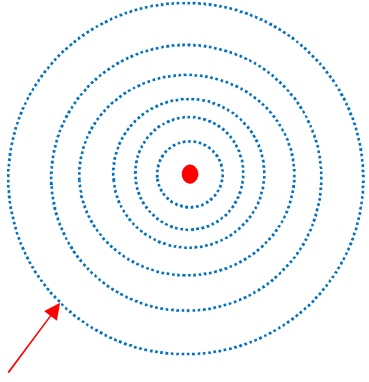
Gaussian basis functions → many analytic formulae available

Plane wave basis functions → fast Fourier transform does the trick

Numerical atom centered basis functions → need atom centered grids

$$h_{ij} = \sum_k \varphi_i(\mathbf{r}_k) h_{KS}(\mathbf{r}_k) \varphi_j(\mathbf{r}_k) w_k$$

$$s_{ij} = \sum_k \varphi_i(\mathbf{r}_k) \varphi_j(\mathbf{r}_k) w_k$$

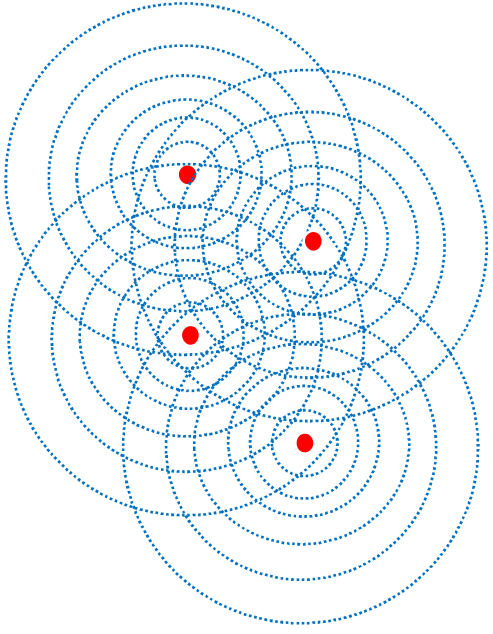


Grid points around the atom, location $\mathbf{r}_k$, weight w_k

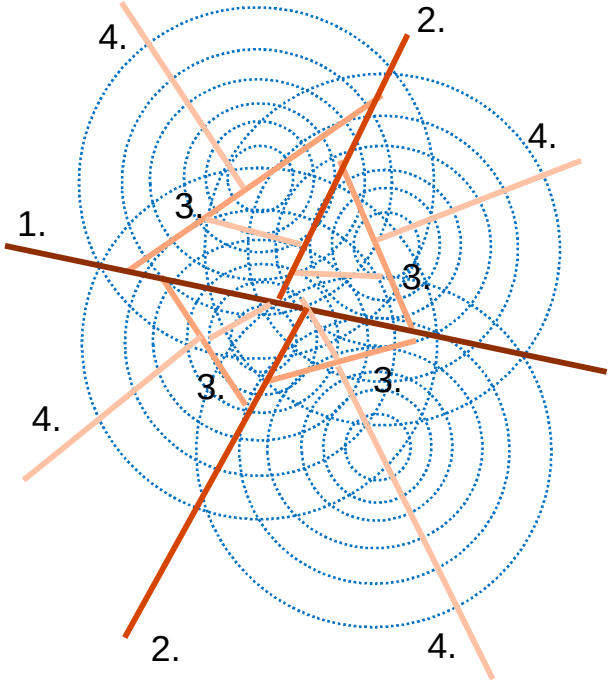
Grid-based integration

Many atoms, large overlapping grids, need to consider integration points in batches

Overlapping grid



Split recursively into batches of equal size



In each batch:

1. Compute the matrices

$$M_{jk} = \varphi_j(\mathbf{r}_k)\sqrt{w_k}$$

$$N_{ik} = h_{KS}(\mathbf{r}_k)\varphi_i(\mathbf{r}_k)\sqrt{w_k}$$

2. Sum up the local contribution

$$h_{ij,loc} = \sum_k M_{jk}N^*_{ik}$$

When all batches done sum up:

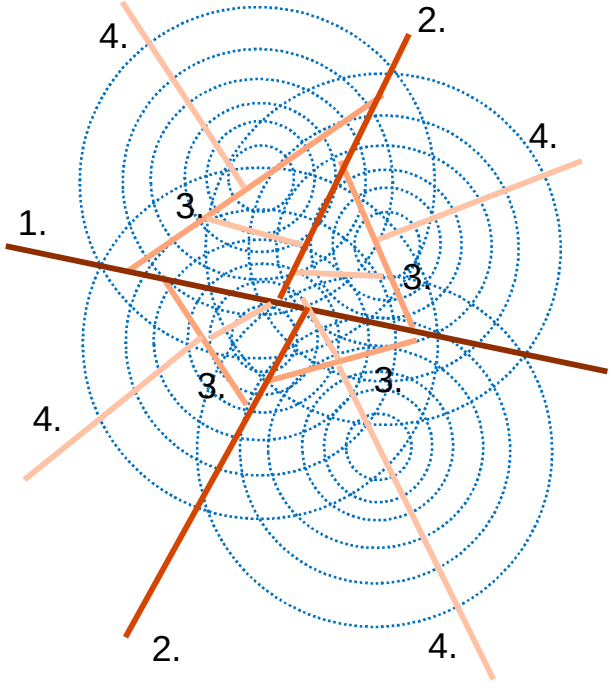
$$h_{ij} = \sum h_{ij,loc}$$

Aalto University
School of Science

Havu et. at. *Journal of Computational Physics* **228**, 8367-8379

Grid-based integration

Many atoms, large overlapping grids, need to consider integration points in batches



<u>In each batch:</u>

1. Compute the matrices

$$M_{jk} = \varphi_j(\mathbf{r}_k)\sqrt{w_k}$$

$$N_{ik} = h_{KS}(\mathbf{r}_k)\varphi_i(\mathbf{r}_k)\sqrt{w_k}$$

Need to evaluate only non-zero basis functions in each batch

2. Sum up the local contribution

$$h_{ij,loc} = \sum_k M_{jk} N^*_{ik}$$

Matrix-matrix product (BLAS3 operation)

<u>When all batches done</u> sum up:

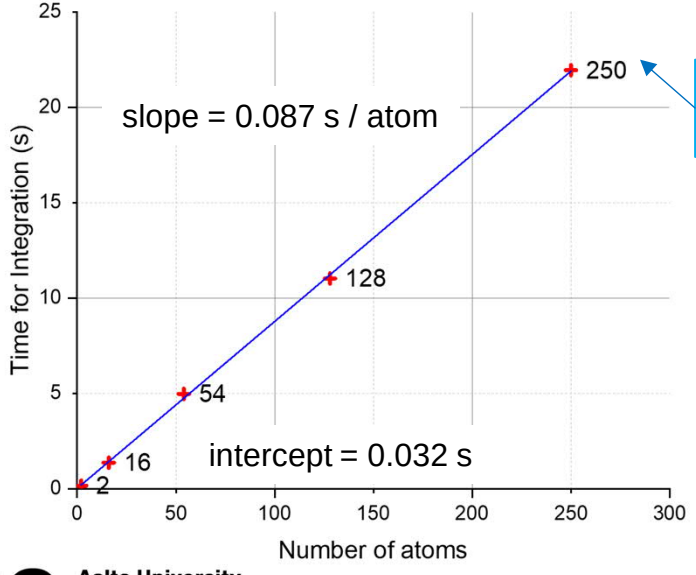
$$h_{ij} = \sum h_{ij,loc}$$

MPI_Allreduce makes operation parallel

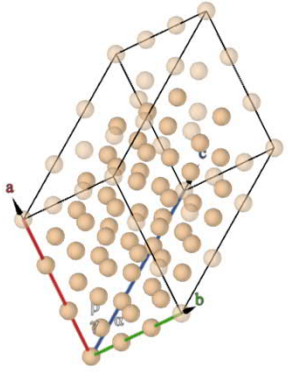
Complexity: O(*N*) points, O(1) functions / batch → O(*N*) in total

Performance and scalability: basic algorithm

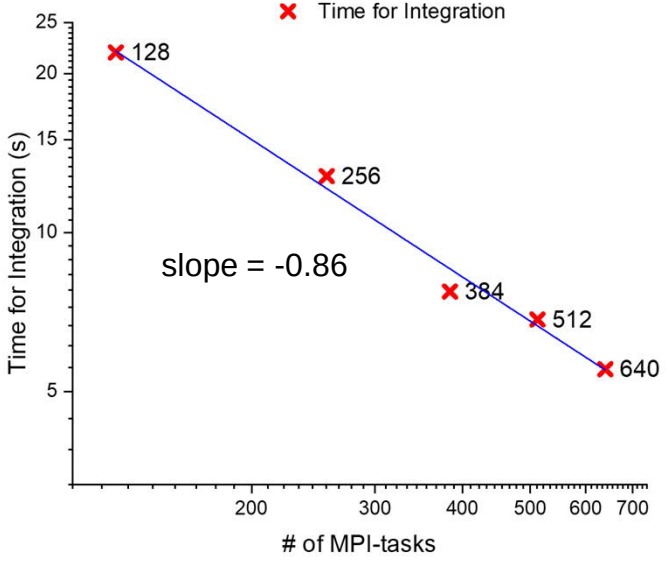
Bulk Si, tight settings, 2 – 250 atoms in a unit cell
128 MPI-tasks (2 x 64 AMD Rome), 256 GB memory, single s.c.f. iteration



out of memory for the next unit cell



Bulk Si, tight settings, 250 atoms in a unit cell
128 – 640 MPI-tasks, single s.c.f. iteration



Improving integration with local basis storage and load balancing

The basic algorithm works well in many cases but can be improved

Per batch for each MPI-task:

1. Compute the matrices

$$M_{jk} = \varphi_j(\mathbf{r}_k)\sqrt{w_k}$$
$$N_{ik} = h_{KS}(\mathbf{r}_k)\varphi_i(\mathbf{r}_k)\sqrt{w_k}$$

2. Sum up the local contribution

$$\tilde{h}_{ij,loc} = \sum_k M_{jk}N^*_{ik}$$

When all batches done sum up:

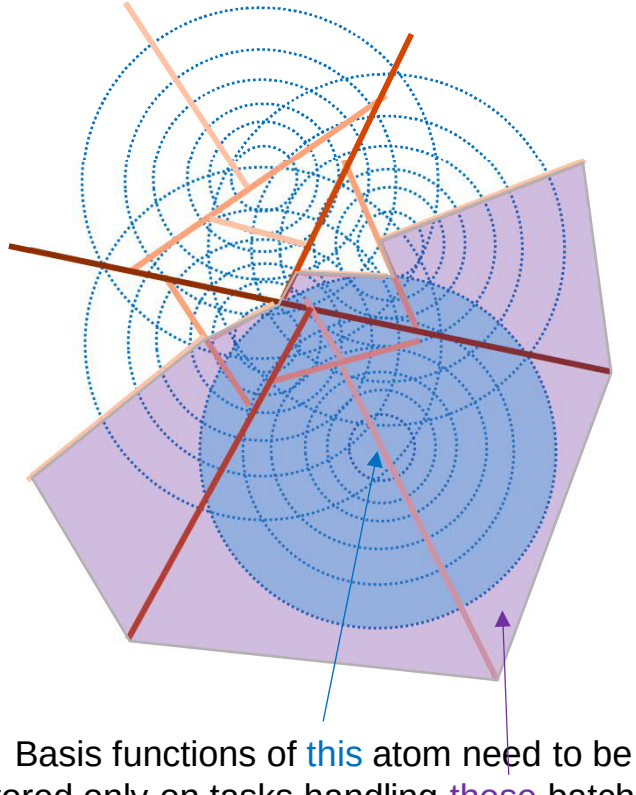
$$h_{ij} = \sum \tilde{h}_{ij,loc}$$

Aalto University
School of Science

Store only the elements relevant to the MPI-task doing the integration in $h_{ij,loc}$ → no global matrix storage. It follows:

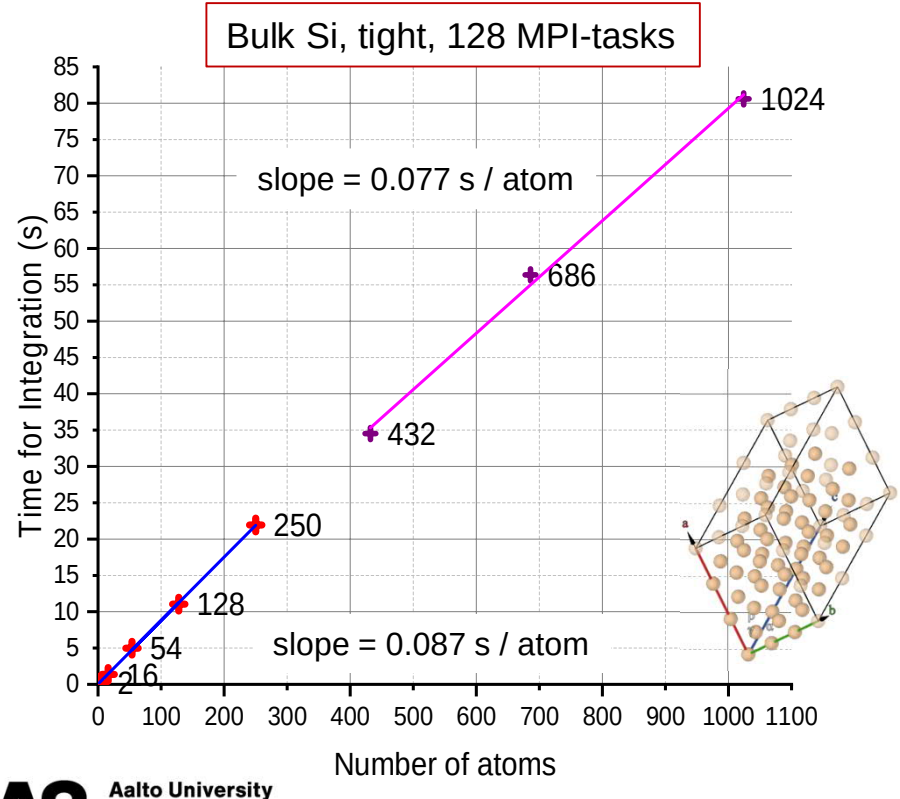
1. Batches need to be distributed to MPI-tasks **by location**, not by load
2. Load imbalance can follow to be compensated by **load balancing**
3. Drawback: some more exotic functionality may lack support

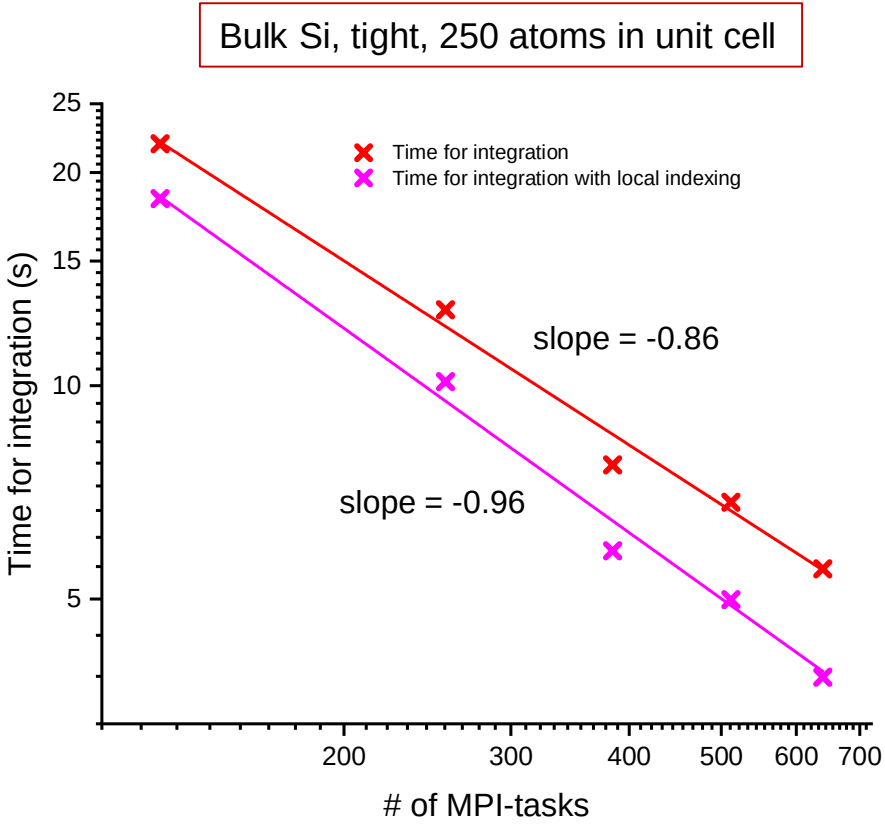
```
use_local_index
load_balancing
```



Basis functions of this atom need to be stored only on tasks handling these batches

Improving integration with local basis storage and load balancing





Aalto University
School of Science

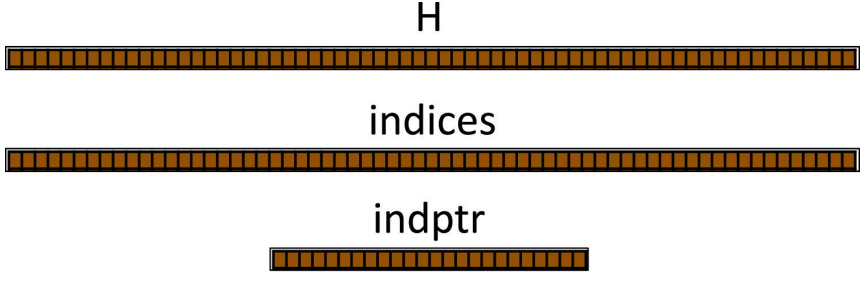
Local indexing storage schemes

Global sparse matrix storage

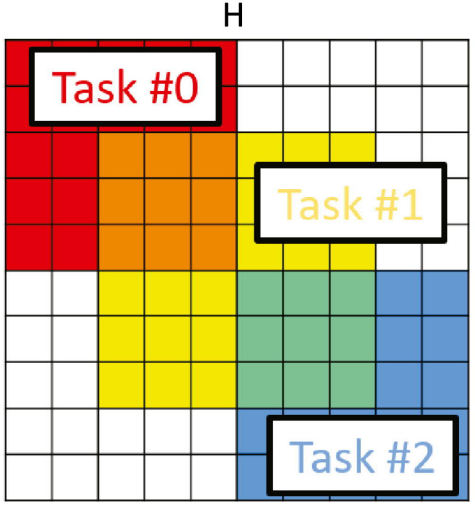
- All MPI-tasks store the same matrix
- Sparse storage format
  - Entries
  - Row indices of entries
  - Column pointers to row indices
- Storage cost O(*N*)
- Simply synchronization with MPI_AllReduce

Local dense storage

- MPI-tasks store only elements relevant to its batches
- Dense storage format
  - Larger system – storage cost up
  - More MPI-tasks – storage cost down
- Complex synchronization with point-to-point communication



(a) Globally-Indexed Sparse



(b) Locally-Indexed Dense

Huhn et al. *Comp. Phys. Comm.*, **254**, 107314

Density update

The density at each grid point follows from the single-particle orbitals: $n(\mathbf{r}_k) = \sum_n f_n |\psi_n(\mathbf{r}_k)|^2$

O(*N*) orbitals x O(*N*) grid points = O(N^2) if calculated directly

But $\psi_n(\mathbf{r}) = \sum_i c_n^i \varphi_i(\mathbf{r})$ and it follows

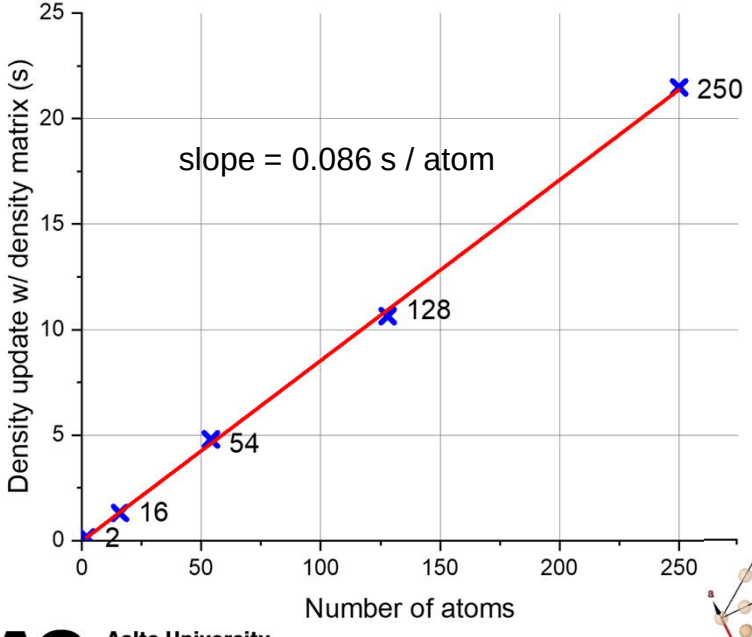
$$n(\mathbf{r}_k) = \sum_n f_n \sum_i c_n^{i*} \varphi_i(\mathbf{r}_k) \sum_j c_n^j \varphi_j(\mathbf{r}_k) = \sum_{ij} \varphi_i(\mathbf{r}_k)\varphi_j(\mathbf{r}_k) \sum_n f_n c_n^{i*} c_n^j$$

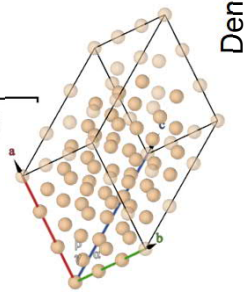
O(1) for localized basis $= D_{ij}$

Formally O(N^3) but
- a very small prefactor
- D_{ij} is sparse

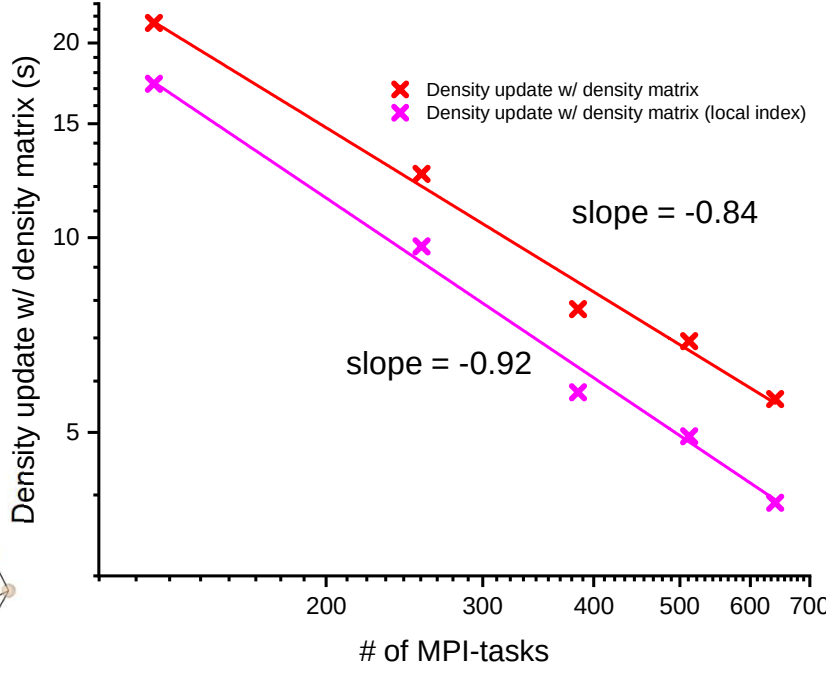
Performance and scalability

Bulk Si, tight settings, 2 – 250 atoms in a unit cell
128 MPI-tasks, single s.c.f. iteration





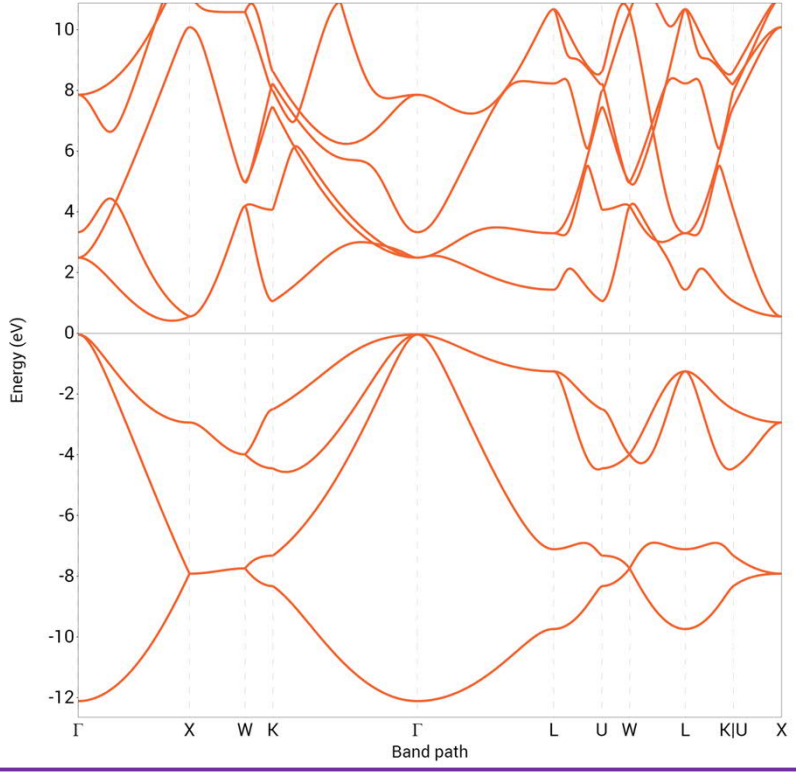
Bulk Si, tight, 250 atoms in unit cell, single s.c.f. iteration



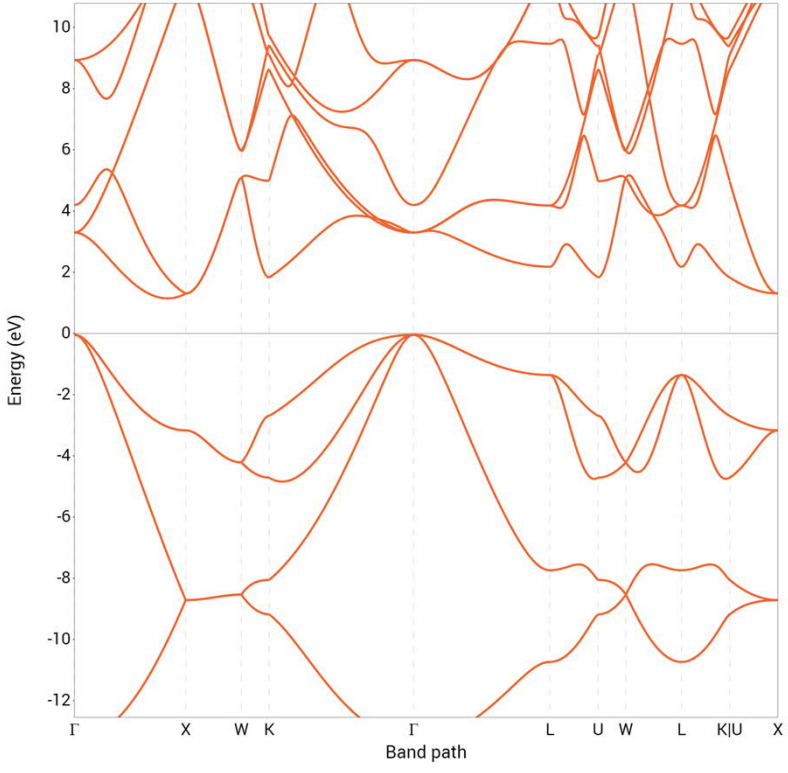
Aalto University
School of Science

Beyond DFT – local is not enough

Si-bulk, LDA, tight, 15x15x15 k-grid: E_g = 0.466 eV



Si-bulk, HSE06, tight, 12x12x12 k-grid: E_g = 1.188 eV



Aalto University
School of Science

Input files and output file analysis made with GIMS: https://gims.ms1p.org/

Beyond DFT – the exchange operator

To go beyond local and semilocal approximations one very often needs the exchange operator:

$$\hat{K}\psi_n(\mathbf{r}) = \sum_m f_m \int \frac{\psi_m^*(\mathbf{r}')\psi_n(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\, d\mathbf{r}'\psi_m(\mathbf{r})$$

In basis representation, that is:

$$K_{ij} = \sum_{kl}(ik|lj)D_{kl} \qquad (ij|kl) = \iint \frac{\varphi_i^*(\mathbf{r})\varphi_j(\mathbf{r})\varphi_k^*(\mathbf{r}')\varphi_l(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\, d\mathbf{r}d\mathbf{r}'$$

Resolution of identity

Four center integrals (ij | kl) are very expensive. Need to reduce costs. $$(ij|kl) = \iint \frac{\varphi_i^*(\mathbf{r})\varphi_j(\mathbf{r})\varphi_k^*(\mathbf{r}')\varphi_l(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\, d\mathbf{r}d\mathbf{r}'$$

Pair products are too many, expand in auxiliary functions (resolution of identity (RI), variational density fitting):

$$\varphi_i(\mathbf{r})\varphi_j(\mathbf{r}) = \sum_\mu C_{ij}^\mu P_\mu(\mathbf{r})$$

→

$$\begin{cases} (ij|kl) = \sum_{\mu\nu} C_{ij}^\mu V_{\mu\nu} C_{kl}^\nu \\ V_{\mu\nu} = \iint \frac{P_\mu(\mathbf{r})P_\nu(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} \end{cases}$$

Four centers becomes two indices. Need to:

1. Select the auxiliary basis functions
2. Define, how to compute the coefficients C_{ij}

Resolution of identity

$$\varphi_i(\mathbf{r})\varphi_j(\mathbf{r}) = \sum_\mu C_{ij}^\mu P_\mu(\mathbf{r})$$

Four centers becomes two indices. Need to:

1. Select the auxiliary basis functions
2. Define, how to compute the coefficients C_{ij}

2. To compute the coefficients, minimize:

$$\delta\rho_{ij}(\mathbf{r}) = \varphi_i(\mathbf{r})\varphi_j(\mathbf{r}) - \sum_\mu C_{ij}^\mu P_\mu(\mathbf{r})$$

Option a: minimize the norm $(\delta\rho_{ij}\,\delta\rho_{ij})$

$$C_{ij}^\mu = \sum_\nu (ij\nu) S_{\nu\mu}^{-1} \qquad (ij\nu) = \int \varphi_i(\mathbf{r})\varphi_j(\mathbf{r}) P_\nu(\mathbf{r})\, d\mathbf{r}$$

$$S_{\mu\nu} = \int P_\nu(\mathbf{r}) P_\mu(\mathbf{r})\, d\mathbf{r}$$

RI-SVS

Option b: minimize the norm $(\delta\rho_{ij}|\delta\rho_{ij})$

$$C_{ij}^\mu = \sum_\nu (ij|\nu) V_{\nu\mu}^{-1} \qquad (ij|\nu) = \iint \frac{\varphi_i(\mathbf{r})\varphi_j(\mathbf{r}) P_\nu(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\, d\mathbf{r} d\mathbf{r}'$$

$$V_{\mu\nu} = \iint \frac{P_\mu(\mathbf{r}) P_\nu(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}$$

RI-V

Method of choice due to accuracy and reliability

$$K_{ij} = \sum_{kl}\sum_{\mu\nu} (ik|\mu) V_{\mu\nu}^{-1} (\nu|jl) D_{kl} = \sum_\mu \sum_{kl} \left(\sum_\nu C_{ik}^\nu V_{\nu\mu}^{1/2} \right) \left(\sum_\nu C_{jl}^\nu V_{\nu\mu}^{1/2} \right) D_{kl}$$

Ren et al. *New J. Phys.* **14** 053020
Ihrig et al. *New J. Phys.*, **17,** 093020
Levchenko et al. *Comp. Phys. Comm.*, **192,** 60

Resolution of identity

$$\varphi_i(\mathbf{r})\varphi_j(\mathbf{r}) = \sum_\mu C_{ij}^\mu P_\mu(\mathbf{r})$$

Four centers becomes two indices. Need to:

1. Select the auxiliary basis functions
2. Define, how to compute the coefficients C_{ij}

1. Construct the auxiliary basis with the products of the orbital basis functions:

a. form on site-products of the radial functions $u_{sk_1l_1}(r)\,u_{sk_2l_2}(r)$
   - (optional) add radial functions to the construction of auxiliary basis only (`for_aux` keyword)

b. remove linear dependencies

c. add angular components with spherical harmonics $Y_{lm}(\phi,\theta)$

atom-centered auxiliary basis

$$\{P_\mu(\mathbf{r})\}$$

Ren et al. *New J. Phys.* **14** 053020
Ihrig et al. *New J. Phys.*, **17,** 093020
Levchenko et al. *Comp. Phys. Comm.,* **192,** 60

Localized resolution of identity

$$\varphi_i(\mathbf{r})\varphi_j(\mathbf{r}) = \sum_\mu C_{ij}^\mu P_\mu(\mathbf{r})$$

atom-centered auxiliary basis

$$\{P_\mu(\mathbf{r})\}$$

There is still a problem: localized basis functions are expanded over all atoms: C_{ij}^{μ} is dense

Fix with localized RI: require $C_{ij}^\mu = 0, \text{ for } \mu \notin \mathcal{P}(IJ)$

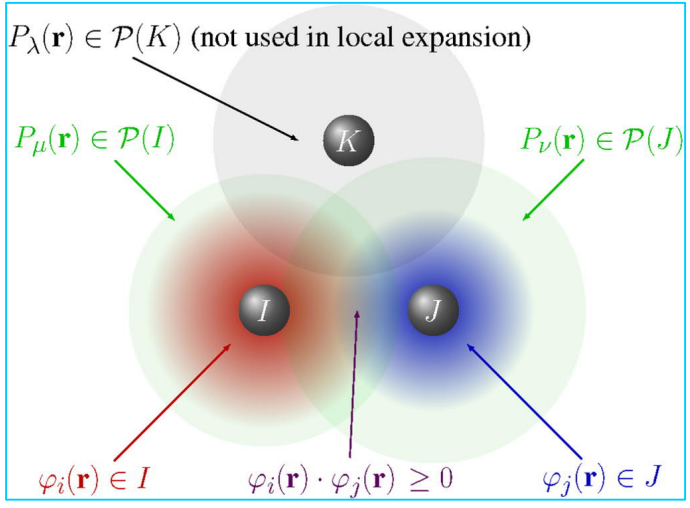
Expand the pair *(i,j)* using only auxiliary functions on the same atoms

$$C_{ij}^\mu = \sum_{\nu \in \mathcal{P}(IJ)} (ij|\nu) L_{\nu\mu}^{IJ}$$

$$L_{\nu\mu}^{IJ} = (V^{IJ})_{\nu\mu}^{-1}$$

RI-LVL

Better scaling implementation of RI-V



Ihrig et al. *New J. Phys.*, **17,** 093020

Aalto University
School of Science

`RI_method` defaults to
- `lvl` for HF and hybrid functionals
- `v` for MP2, RPA and GW

Periodic RI

Block-like basis functions $\varphi_{i\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{R}} e^{i\mathbf{k}\cdot\mathbf{R}} \varphi_i(\mathbf{r}-\mathbf{R})$

In reciprocal space, Fourier transform:

$$K_{ij}(\mathbf{k}) = \frac{1}{N_\mathbf{q}} \sum_{kl,\mathbf{q}} D_{kl}(\mathbf{q}) \iint \frac{\varphi^*_{i\mathbf{k}}(\mathbf{r})\varphi_{k\mathbf{q}}(\mathbf{r})\varphi^*_{l\mathbf{q}}(\mathbf{r}')\varphi_{j\mathbf{k}}(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r}d\mathbf{r}'$$

$$= \frac{1}{N_\mathbf{q}} \sum_{kl,\mathbf{q}} D_{kl}(\mathbf{q}) \sum_{\nu\mu} C^\mu_{ik}(\mathbf{k},\mathbf{q}) V_{\mu\nu}(\mathbf{k}-\mathbf{q}) C^\nu_{lj}(\mathbf{q},\mathbf{k})$$

$$P^\mathbf{k}_\mu(\mathbf{r}) = \sum_\mathbf{R} e^{i\mathbf{k}\cdot\mathbf{R}} P_\mu(\mathbf{r}-\mathbf{R})$$

$$\varphi^*_{i\mathbf{k}}(\mathbf{r})\varphi^*_{k\mathbf{q}}(\mathbf{r}) = \sum_\mu C^\mu_{ik}(\mathbf{k},\mathbf{q}) P^{\mathbf{q}-\mathbf{k}}_\mu(\mathbf{r})$$ RI-V

Using RI-LVL gives

$$C^\mu_{ik}(\mathbf{k},\mathbf{q}) = C^\mu_{ik}(-\mathbf{k},\mathbf{R}_0) + C^\mu_{ki}(\mathbf{q},\mathbf{R}_0)$$

but the matrix C^μ_{ik} is not sparse… still used for periodic GW

Ren et al. *Phys. Rev. Materials* **5**, 013807

In real space, sum up all periodic images:

$$K_{ij}(\mathbf{k}) = \sum_\mathbf{R} e^{i\mathbf{k}\cdot\mathbf{R}} X_{ij}(\mathbf{R}) \qquad D_{kl}(\mathbf{R}') = \frac{1}{N_\mathbf{q}} \sum_\mathbf{k} D_{kl}(\mathbf{k}) e^{-i\mathbf{k}\cdot\mathbf{R}'}$$

$$X_{ij}(\mathbf{R}) = \sum_{kl}\sum_{\mathbf{R}'} D_{kl}(\mathbf{R}') \sum_{\mathbf{R}''} \iint \frac{\varphi_i(\mathbf{r})\varphi_k(\mathbf{r}-\mathbf{R}')\varphi_j(\mathbf{r}'-\mathbf{R})\varphi_l(\mathbf{r}'-\mathbf{R}-\mathbf{R}'')}{|\mathbf{r}-\mathbf{r}'|} d\mathbf{r}d\mathbf{r}'$$

$$= \sum_{k\mathbf{R}'}\sum_{l\mathbf{R}''}\sum_{\mu\mathbf{Q}'}\sum_{\nu\mathbf{Q}''} C^{\mu(\mathbf{Q}')}_{ik(\mathbf{R}')} V_{\mu\nu(\mathbf{R}+\mathbf{Q}''-\mathbf{Q}')} C^{\nu(\mathbf{Q}'')}_{jl(\mathbf{R}'')} D_{kl}(\mathbf{R}+\mathbf{R}''-\mathbf{R}')$$

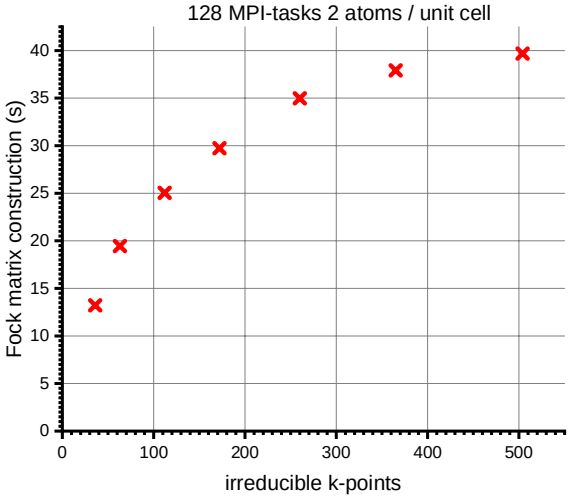
RI-LVL: **Q**' = **0** or **Q**' = **R**' and **Q**'' = **0** or **Q**'' = **R**''

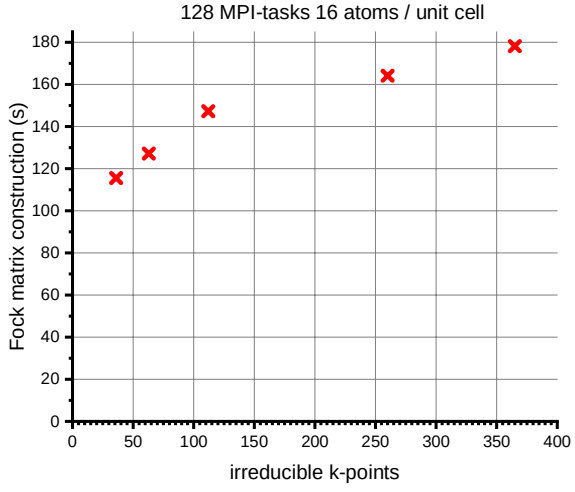
Plus:

- Born-von Karman periodic $V_{\mu\nu}$
- Sparsity of X_{ij} (**R**)
- Screening of small elements in X_{ij} (**R**)

Levchenko et al. *Comp. Phys. Comm.*, **192,** 60

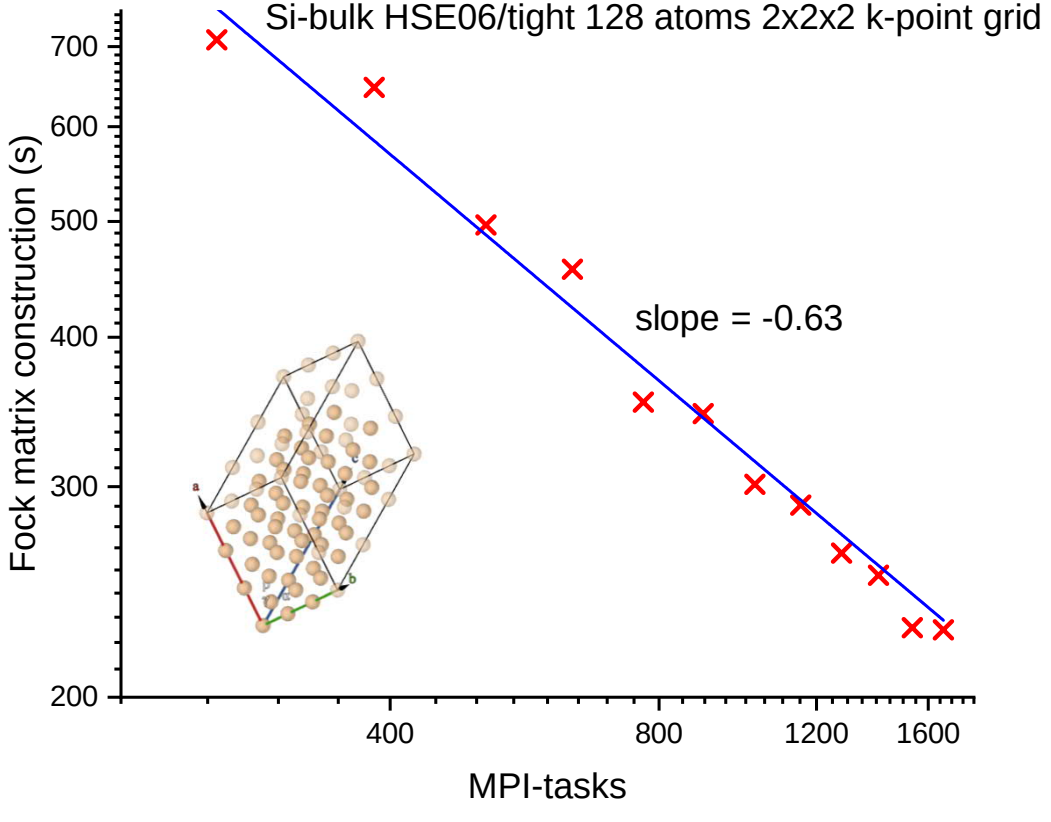
Performance and scalability HSE06



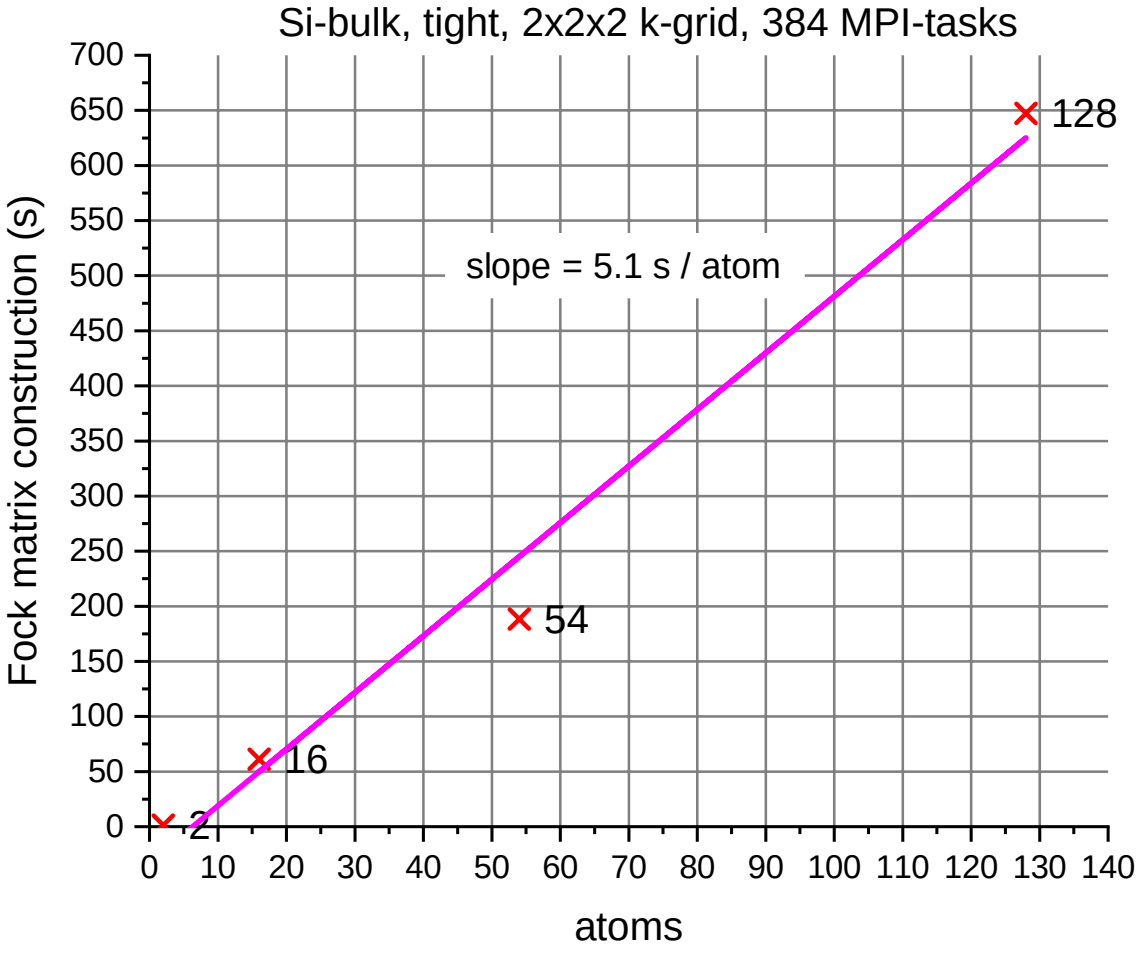


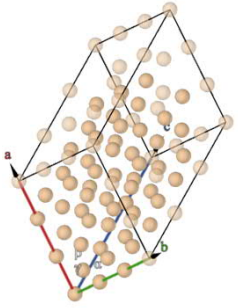
HSE → screened exchange:

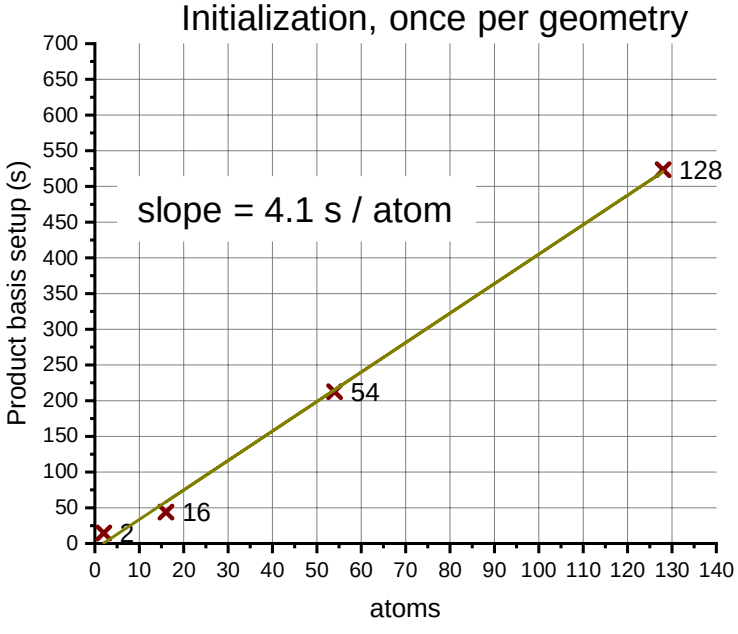
$$(ij|kl) = \iint \frac{\varphi_i^*(\mathbf{r})\varphi_j(\mathbf{r})\,\mathrm{erfc}(\omega|\mathbf{r}-\mathbf{r}'|)\,\varphi_k^*(\mathbf{r}')\varphi_l(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\,d\mathbf{r}d\mathbf{r}'$$



Performance and scalability HSE06

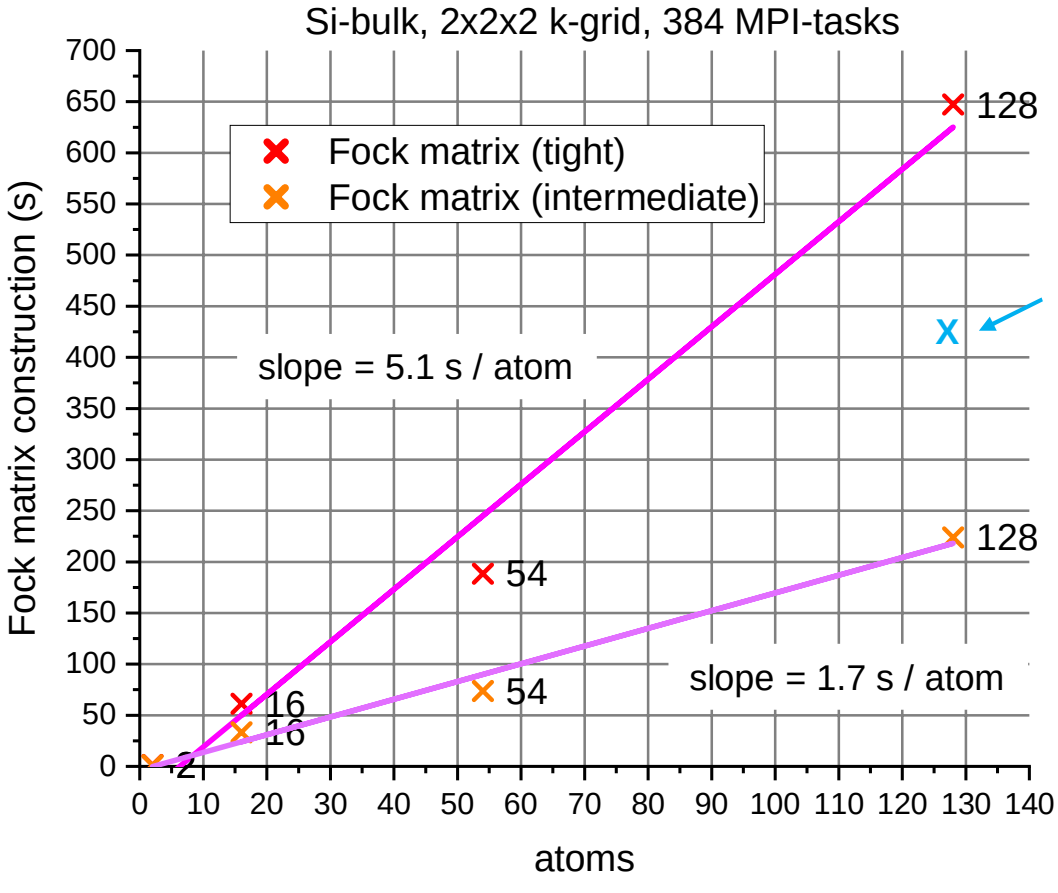


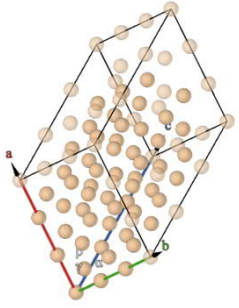




HSE06: tight → intermediate

For practical calculations *intermediate* settings are often sufficient





HF (no HSE-screening), intermediate

Si-tight:

```
hydro 3 d 9
hydro 5 g 9.4
```

Si-intermediate:

```
# hydro 3 d 9
for_aux hydro 5 g 9.4
```

GPU computing

Certain operations can be offloaded to GPUs. The results are best when the operations:

- are vectorizable
- contain no branching statements
- have minimal communication between CPU and GPU

E.g., integration: <u>In each batch:</u>

1. Compute the matrices on CPU

$$M_{jk} = \varphi_j(\mathbf{r}_k)\sqrt{w_k}$$

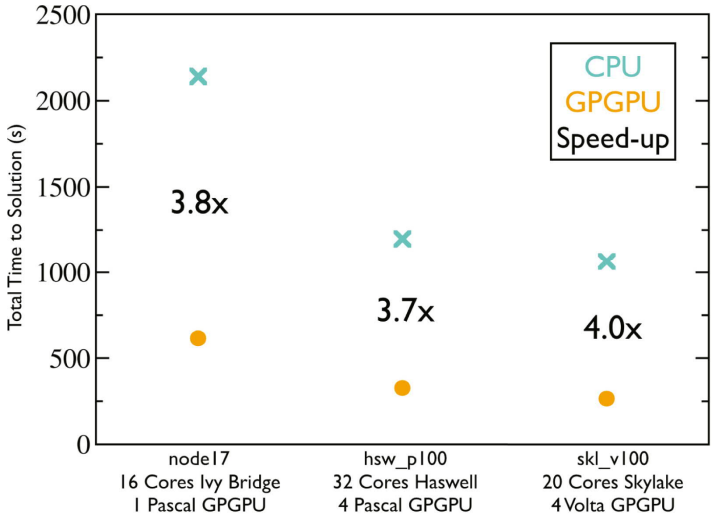
$$N_{ik} = h_{KS}(\mathbf{r}_k)\varphi_i(\mathbf{r}_k)\sqrt{w_k}$$

2. Copy data to GPU
3. Sum up the local contribution on GPU

$$h_{ij,loc} = \sum_k M_{jk} N^*_{ik}$$

works best with local dense storage

<u>When all batches done</u>
copy data to CPU



Si-bulk, 54 atoms / unit cell, tight, single k-point. Time for full calculation including forces and stress tensor.

Huhn et al. *Comp. Phys. Comm.*, **254**, 107314

How to get it? Build a GPU enabled binary and set `use_gpu .true.`

Conclusions

Grid based operations

- O(N) scaling
- Modest prefactor
- Use
  - o `use_local_index` when possible
  - o `load_balancing` when needed
- GPU support available
  - experiment with `points_in_batch`

RI operations

- Also O(N) scaling
- Much larger prefactor and cost of initialization
- Use the default method (RI-V or RI-LVL) or select with `RI_method`
- Test *intermediate* settings
- Experiment with `for_aux` to check the accuracy of the auxiliary basis

Thank you for your attention

Ville Havu

Wednesday, 27 October 2021

Aalto University
School of Science