Accelerating plane-wave-based ab initio molecular dynamics by optimization of Fast-Fourier Transforms for modern HPC architectures

Christian L. Ritterhoff[1], Tobias Klöffel [1], Sagarmoy Mandal[1], Bernd Meyer[1]

[1] *Interdisciplinary Center for Molecular Materials (ICMM), Computer Chemistry Center (CCC), Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Germany*

❶ Fast-Fourier Transforms within plane-wave codes

- Many Kohn-Sham density functional theory based ab initio molecular dynamics (MD) codes use a Plane-Wave (PW) Basis Set[1–3]
  $\rightarrow$ satisfy the periodic boundary conditions efficiently
- Wavefunctions are represented in reciprocal space as a linear combination of PWs

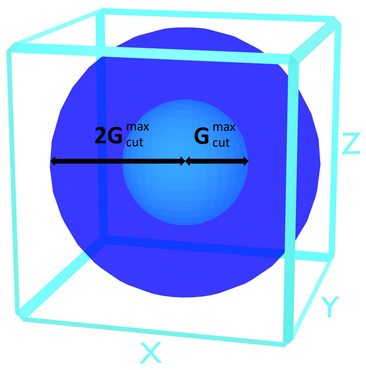
$$\psi_n(\mathbf{r}) = \frac{1}{\sqrt{\Omega}} \sum_{\mathbf{G}} c_n(\mathbf{G}) e^{i\mathbf{G}\cdot\mathbf{r}}$$

- PWs allow for quick and easy transition from real (R) space to reciprocal (G) space and vice versa
- Essential in e.g. the application of the kinetic and potential energy operators
  $\rightarrow$ reduces the overall scaling costs from N^2 to $NlogN$
- Modern MD simulation codes can spend up to 90% of the runtime doing Fast-Fourier Transformations (FFTs)
  $\rightarrow$ highly efficient FFT routines required

❷ Energy cutoff

In practice the sum is truncated

$$\frac{1}{2} |\mathbf{G}|^2 \leq E_{cut}$$



Cutoff for wavefunctions:

- $\mathbf{G}_{cut}^{max} \leq \sqrt{2E_{cut}}$

Cutoff for the density $\rho(\mathbf{r})$:

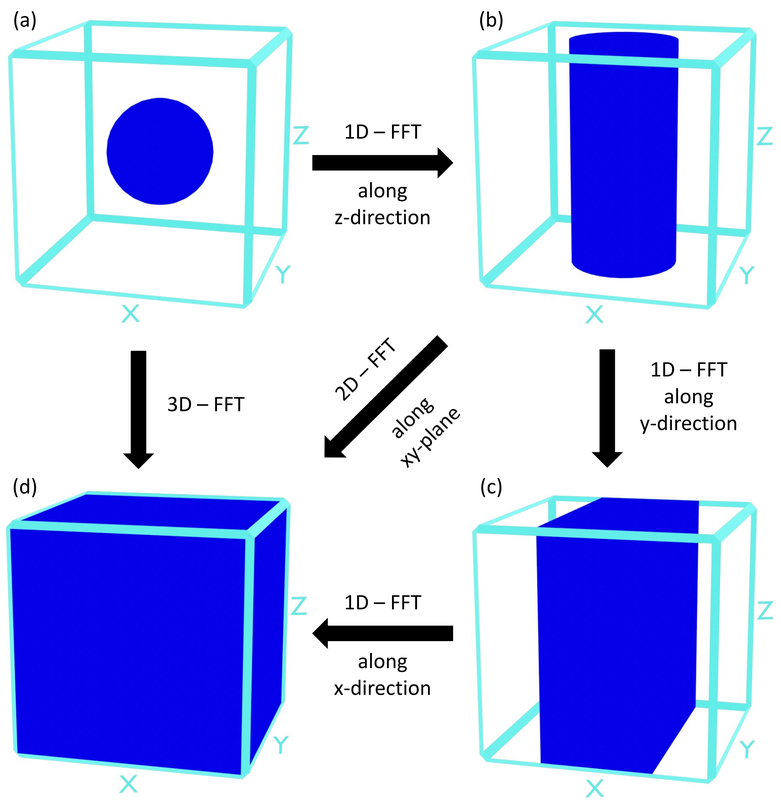
- Radius $2\mathbf{G}_{cut}^{max}$

Occupation of the FFT grid:

- G space: only grid points within spheres $\neq 0$
- R space: all grid points $\neq 0$

❸ 3D FFT strategy

- 3D FFT is done by subdividing into 2D/1D FFTs

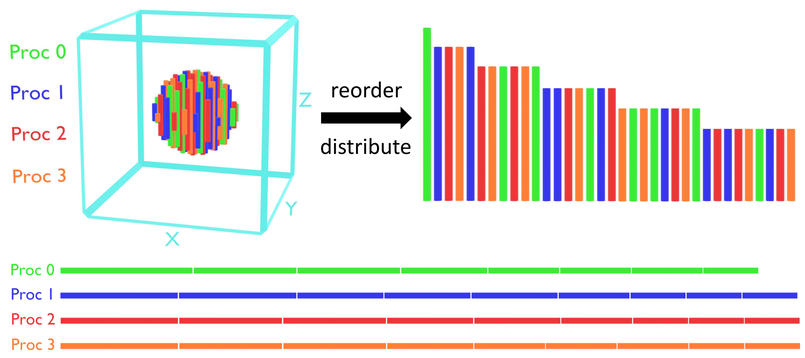


❹ Optimization strategies

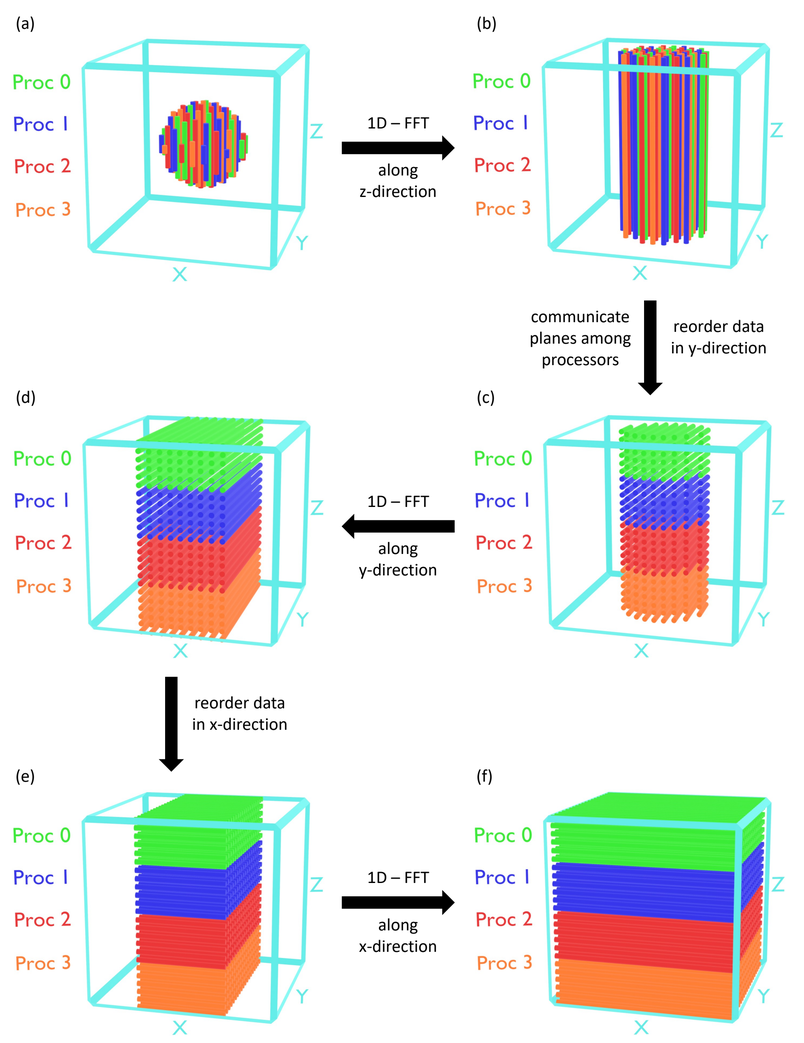
- 1D FFT already highly optimized
  $\rightarrow$ we use the freely available FFTW3 library
- Instead, the focus lies on the organization of data
  $\rightarrow$ reordering/communication between 1D FFTs
  $\rightarrow$ many 3D FFTs at the same time
- Improve the scaling to multiple nodes for all problem sizes
  $\rightarrow$ less MPI tasks; more OMP threads
  $\rightarrow$ able to choose dynamically depending on the problem
- Reduce communication bottlenecks
  $\rightarrow$ improve load balancing among all processors

❺ Parallelization strategies

- Data within a sphere in G space is divided in sticks, sorted and distributed among the MPI tasks:



- Efficient load balancing is achieved through the even splitting of $\mathbf{G}$ vectors and sticks across MPI tasks
- Utilizing the 1D+1D+1D approach, the FFT proceeds as follows:



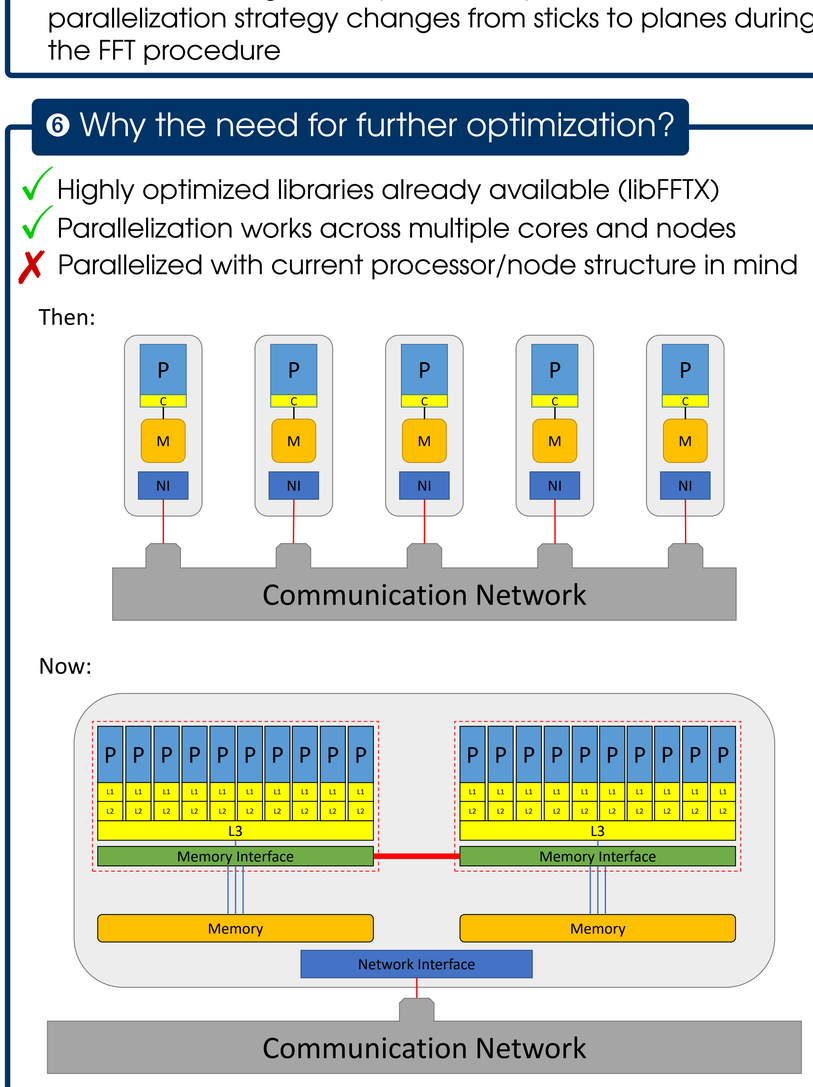
- Due to the change in shape from a sphere to a cube, the parallelization strategy changes from sticks to planes during the FFT procedure

❻ Why the need for further optimization?

✓ Highly optimized libraries already available (libFFTX)
✓ Parallelization works across multiple cores and nodes
✗ Parallelized with current processor/node structure in mind

Then:

Now:



⇒ Increase of complexity in HPC architectures results in new challenges for efficient FFT parallelization

❼ R space to G space algorithm

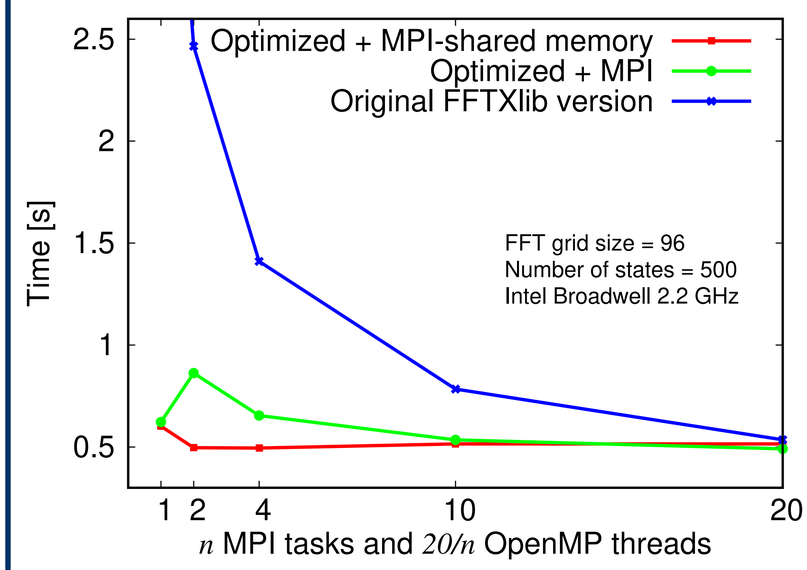
Algorithm of the decomposition of 3D FFT in 1D+1D+1D FFTs

(1) Reorder $\mathbf{G}$ vectors for FFT along z direction
(2) Perform 1D FFT along z direction
(3) Prepare data for communication across nodes
(4) Communication
(5) Reorder received data along y direction
(6) Perform 1D FFT along y direction
(7) Reorder data along x direction
(8) Perform 1D FFT along x direction

❽ Implemented optimization and results

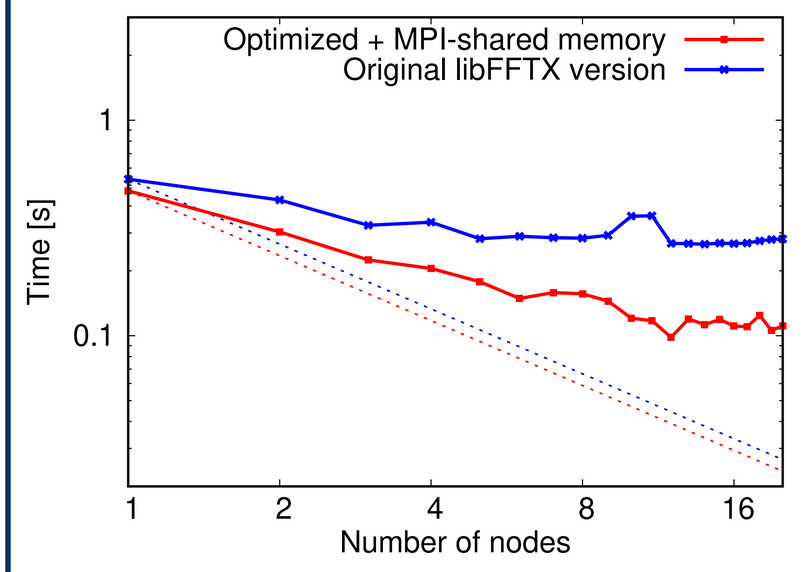
Single node optimizations

- Include OMP directives (e.g. vectorization)
- Remove unnecessary zeroing and allocations
- Refactor loop structures to be write-consecutive
- Introduce MPI shared memory



Multi node optimizations

- Switch from ALL2ALL to Send/Receive
- Communicate batches of multiple states
- Overlap calculation and communication



- We are currently implementing this optimized version of the libFFTX into our already optimized CPMD code[4,5]
- Current results indicate a substantial increase in performance in e.g. the calculation of $v(\mathbf{r})\Psi_n(\mathbf{r})$ for most core counts; in some cases of more than 100%

❾ References

[1] D. Marx and J. Hutter (Cambridge University Press, Cambridge, 2009).
[2] M. C. Payne et al., Rev. Mod. Phys. **64**, 1045–1097 (1992).
[3] R. M. Martin (Cambridge University Press, Cambridge, 2004).
[4] `http://www.cpmd.org`, Copyright 2000-2021 jointly by IBM Corp. and by Max Planck Institute, Stuttgart. (2021).
[5] T. Klöffel, G. Mathias, and B. Meyer, Comput. Phys. Commun. **260**, 107745 (2021).