PAUL SCHERRER INSTITUT

PSI

ETH zürich

WIR SCHAFFEN WISSEN – HEUTE FÜR MORGEN

Veronica Montanaro :: AMAS Group, LSM

Mixed precision in IPPL

Midterm presentation

November 16, 2023

Mixed precision updated timeline

- March & April
  - ☑ Work on mixed precision
    - ☑ Fix templates of basic classes
    - ☑ Work on solvers
    - ☑ Perform different memory tests on solvers
    - ☑ ~~Work on Alpine mini-apps such that data types can be user-determined~~ Write one test case for mixed precision in alpine
- May
  - ☑ Wrap-up mixed precision
  - ☑ Finalise reading documentation and literature for DCT-I
  - ☐ Write new `unit_tests`

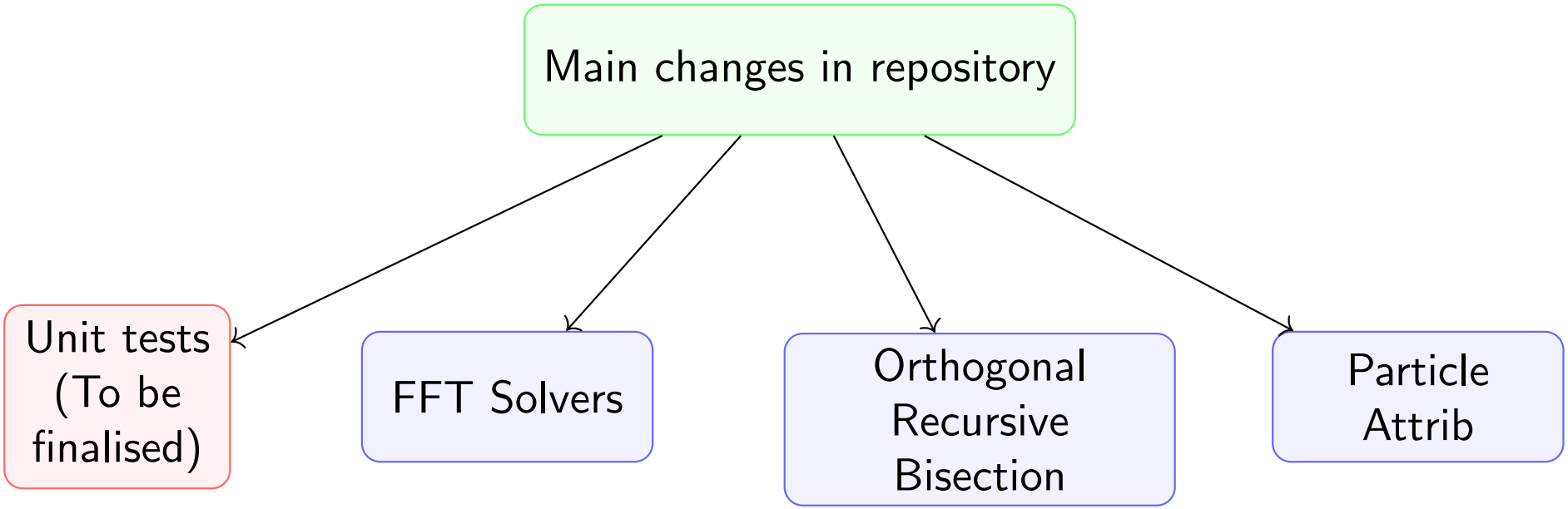
DCT-I updated timeline

- June & July
  - ☐ Present motivation and strategy
  - ☐ Implement the cuFFT DCT-I backend
  - ☐ Write cosine 1 transform class in `IPPL/FFT`
  - ☐ Test it with Vico
- August
  - ☐ Wrap-up and write report

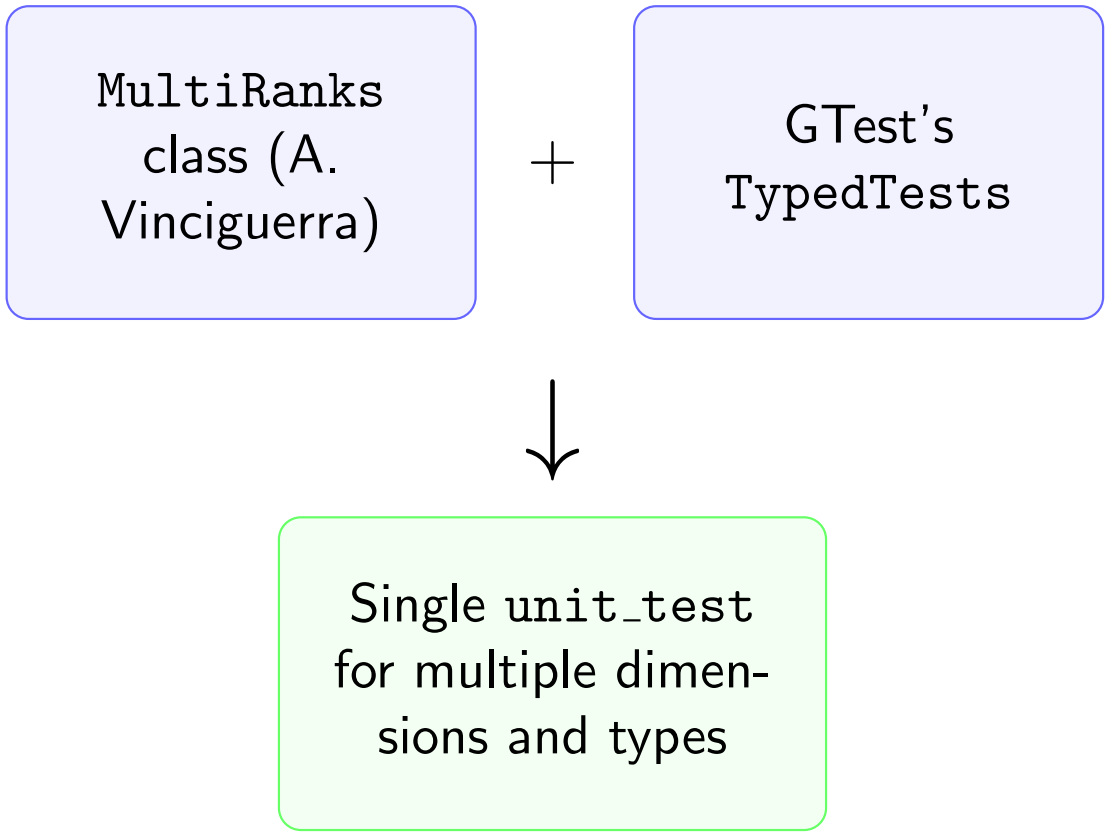
Mixed Precision

All hard-coded types succesfully removed.

→ Datatype of fields, meshes and particle attributes **can be chosen independently.**



Unit tests update



Currently, the GTest part is still to be merged.

ParticleAttrib and ORB update

ParticleAttrib

- Modified `scatter` function, that interpolates the attribute on the Field.
- Attribute and corresponding Field have no data dependency.
- Different attributes of a `ParticleBase` object can coexist.

ORB

- Receives a `Field` object and a `ParticleAttribute` object representing the particle's position.
- Separated the field's type and the position's type.
- Added compatibility with dimension independence.

FFTPoissonSolver update

Checkpoint from three-weeks presentation

Problem

- The class has more fields than just E_m and ρ .
- Green function, copies of ρ on doubled and quadrupled domain, etc. are all templated on the same type as ρ.
- Having just E_m in single precision does not affect memory in a significant way.

Class fields templating

Field name	Usage	Template type
`storage_field`	Field on doubled grid	Trhs
`rho2_mr`	Charge density field on doubled mesh	Trhs
`grn_mr`	Green's function (Hockney)	Trhs
`rho2tr_m`	Transformed ρ	complex Trhs
`grntr_m`	Transformed Green's fct (Hockney)	complex Trhs
`temp_m`	Temporary field for E	complex Tlhs
`grnL_m`	Green's fct (Vico)	complex Tlhs

- **Trhs**=Data type of right-hand side field (ρ).
- **Tlhs**=Data type of left-hand side field (E).
- `rho2_mr` and `grn_mr` are both references to `storage_field`.

Class fields templating

Motivation

Vico

- Vico's Green function in frequency domain occupies a big portion of memory.
- However, it has no dependencies on $\rho \Rightarrow$ Good candidate to have same type as E_m.

Hockney

- Green's function has dependency on `storage_field` and, consequently, on `rho2_mr`.
- Moreover, HeFFTe does not support mixed precision $\Rightarrow$ we can't have a transformed Green's function with a different type.

Memory testing

Setup

Test case

Gaussian convergence test [Mayani, 2021]

- Runs one iteration of the solver (Vico or Hockney) for given mesh size.
- Problem size $= 128^3 \rightarrow$ maximum size for single GPUs/CPU node. [Mayani, 2021]
- For mixed precision case, E is set to `float`.

CPU

- Single node from Merlin6 cluster (88 CPUs, 380GB).
- Memory tracker: Kokkos Memory Highwater.

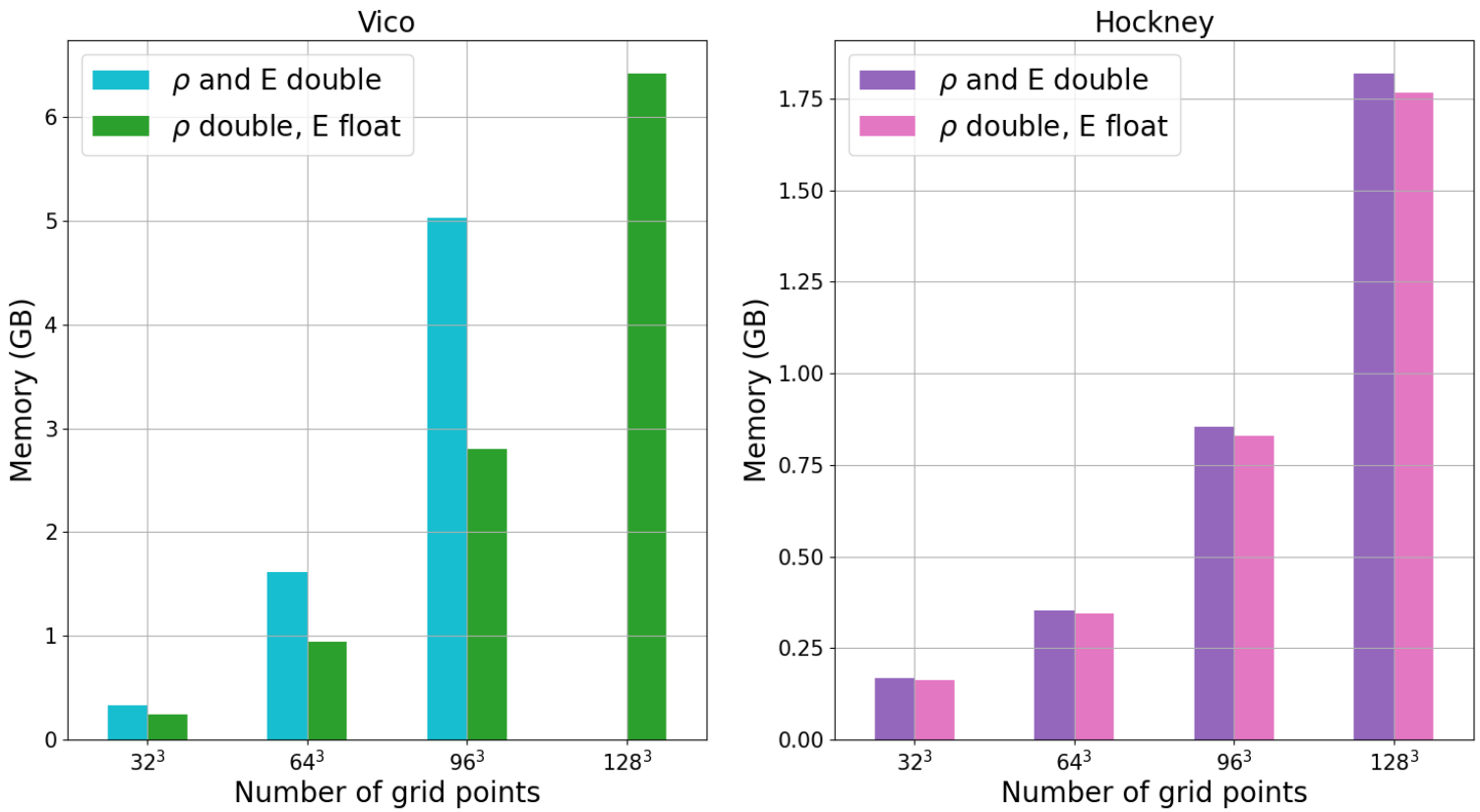
GPU

- Single node with one GPU on Gwendolen (16GB).
- Memory tracker: Kokkos Space Time Stack.

Memory testing

CPU Results

Vico exceeds memory bound for double precision.
Almost 50% less memory used.
Hockney's Green Function field is not affected ⇒ Less visible changes.

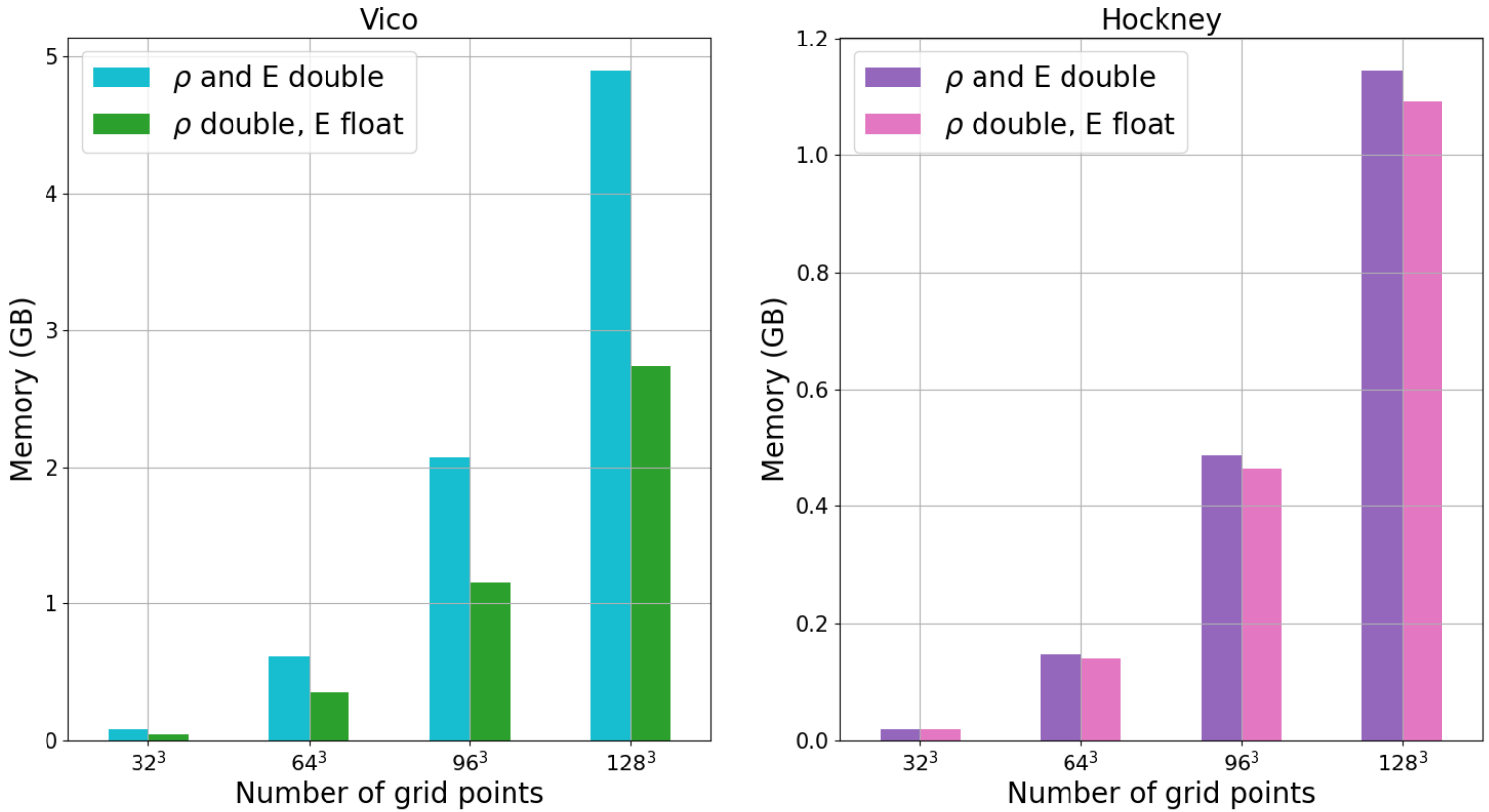


Memory testing

GPU Results

Only device space (= memory allocated on GPU) taken into account.

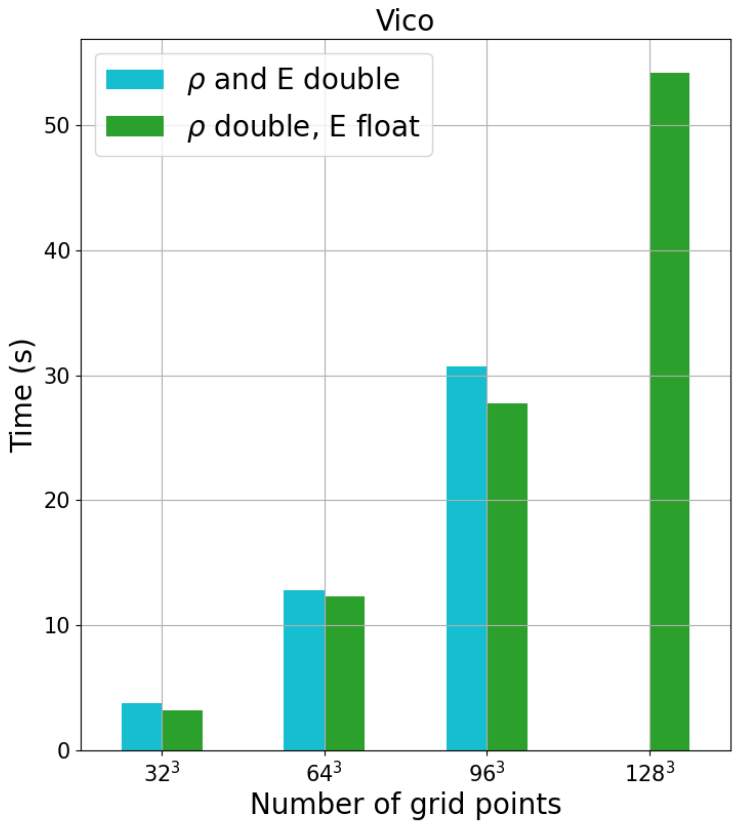
As expected, similar to the CPU results.

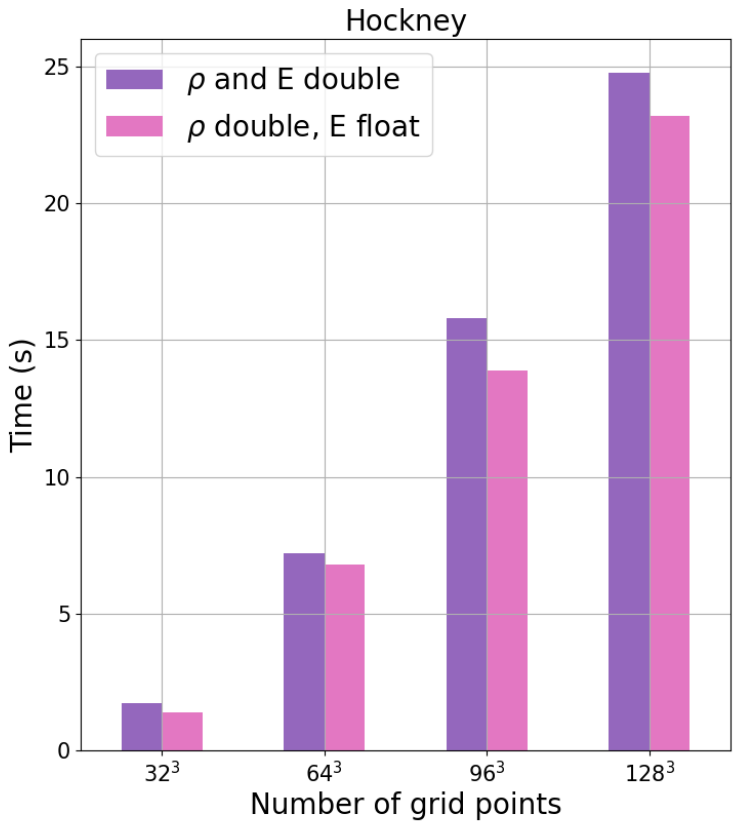


Runtime analysis

CPU Results

Minimal effect, but no negative consequences.

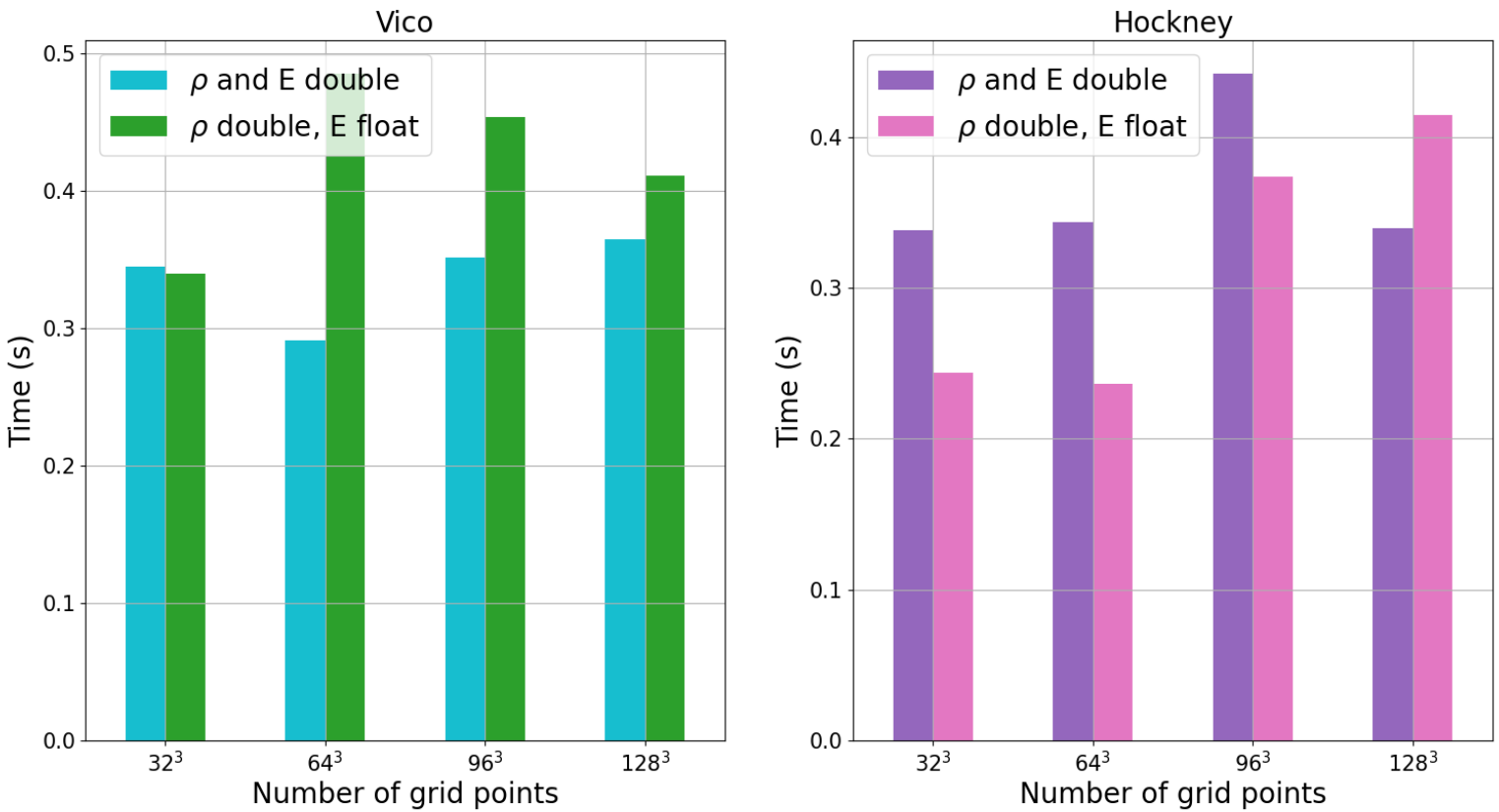




Runtime analysis

GPU Results

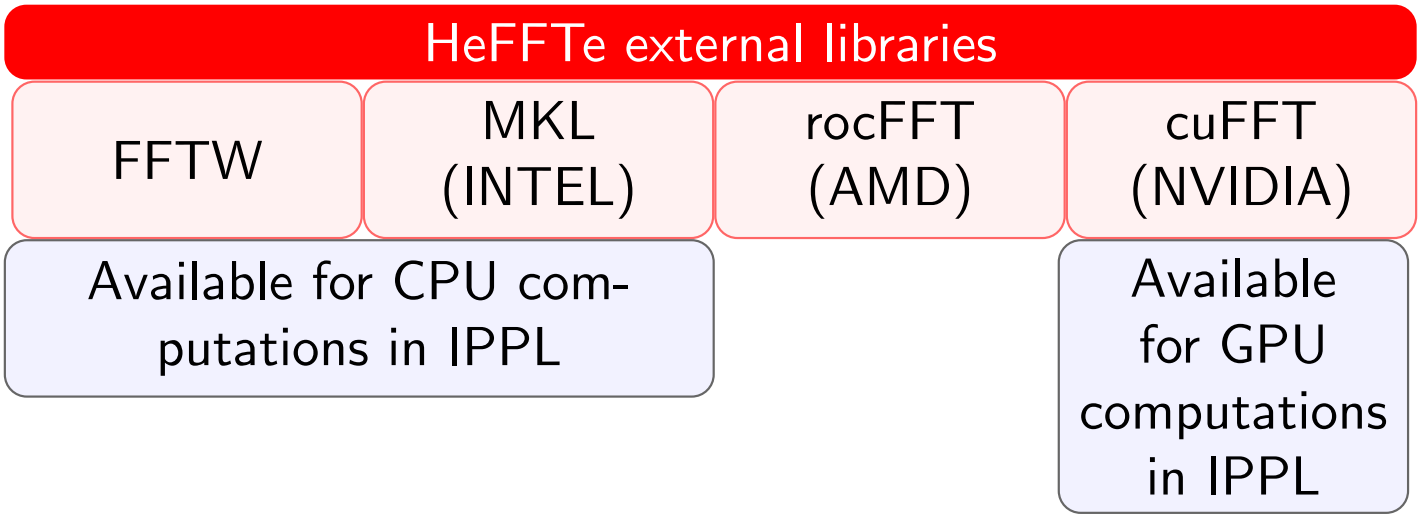
Vico seem to take longer in mixed precision.
More GPUs required to check whether the DP time will eventually overcome the MP run.



DCT-I

HeFFTe overview

HeFFTe relies on other FFT libraries as back-ends for computation.
Backend = library used + type of transform to compute.



FFT in HeFFTe

- My main focus: `executor` class.
- Contains method for performing the transform.
- Depends on back-end and type of input and output (**real to real** , real to complex, complex to complex).

Algorithm kernels [Ayala et al., 2020]

Reshape
Transposes data to a 1-D pencil or 2-D slab.

Execute
Computation of a 1-D or 2-D FFT. Handled through an `executor` object, templated on the back-end.

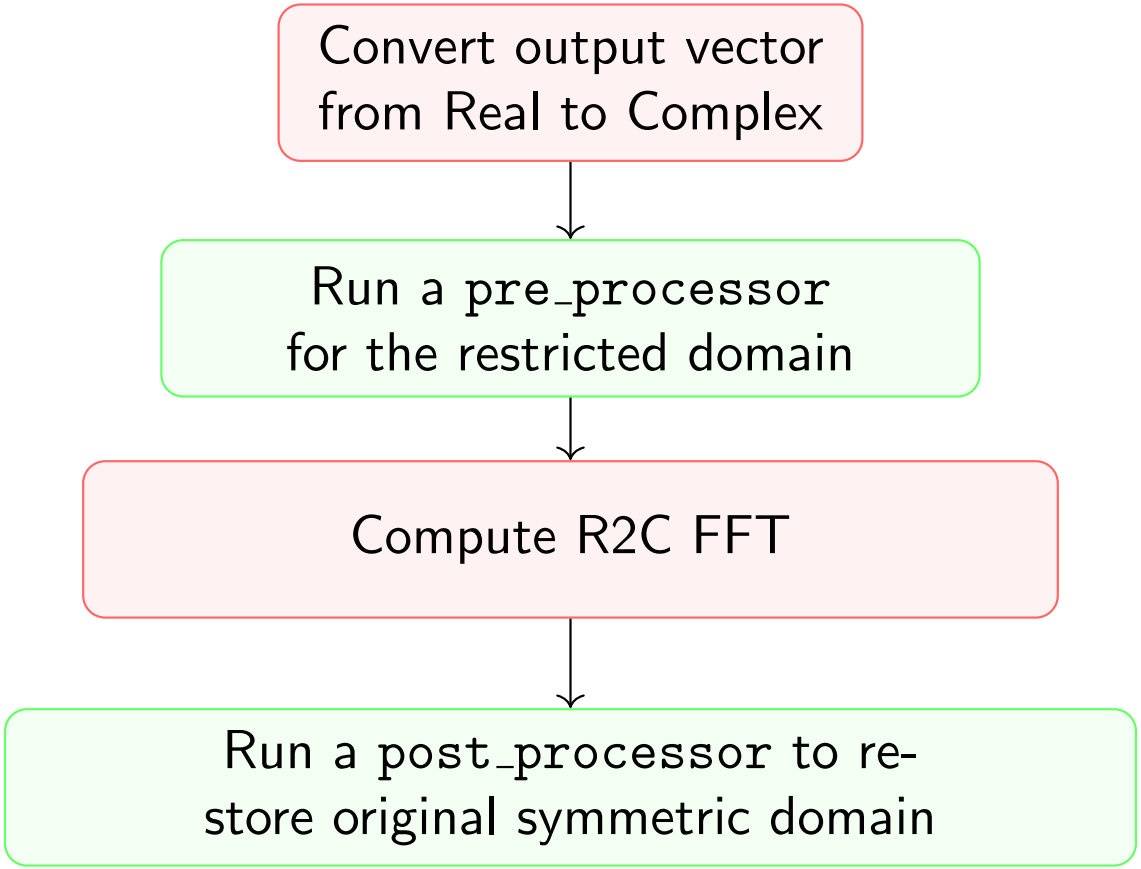
Current library status

Real to Real executor

- Only fftw has native DCTs of type I,II and III.
- CuFFT does not provide Real-to-Real transforms $\rightarrow$ Template class `r2r_executor`
- The class contains methods for Sine and Cosine transform.
- Currently: only in 1-D and only type II and III Cosine transform.

Current library status

R2R transforms for non-fftw backends



CuFFT Plan

- Add `cufft_cos1` back-end
- Add `gpu_cos1_pre/post_processor` in a similar way as already done in fftw
- Add scaling factor of DCT-I for CuFFT to be used with cuda::scale.
- Repeat for 1-D, 2-D and 3-D

Thanks

Thanks for your attention.
Open for questions!

References

Ayala, A., Tomov, S., Haidar, A., and Dongarra, J. (2020).

heffte: Highly efficient fft for exascale.

In Krzhizhanovskaya, V. V., Závodszky, G., Lees, M. H., Dongarra, J. J., Sloot, P. M. A., Brissos, S., and Teixeira, J., editors, *Computational Science – ICCS 2020*, pages 262–275, Cham. Springer International Publishing.

Mayani, S. (2021).

A performance portable poisson solver for the hose instability.