PAUL SCHERRER INSTITUT

PSI

ETH zürich

Veronica Montanaro :: AMAS Group, LSM

Mixed precision in IPPL

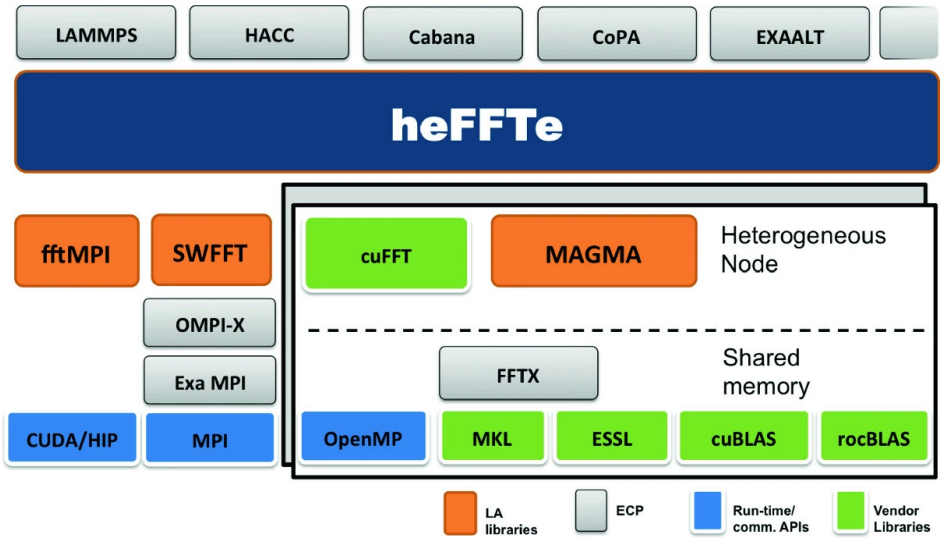
Three-weeks update presentation

March 29, 2023

Background

Highly Efficient FFT for Exascale (HeFFTe) [Ayala et al., 2020]

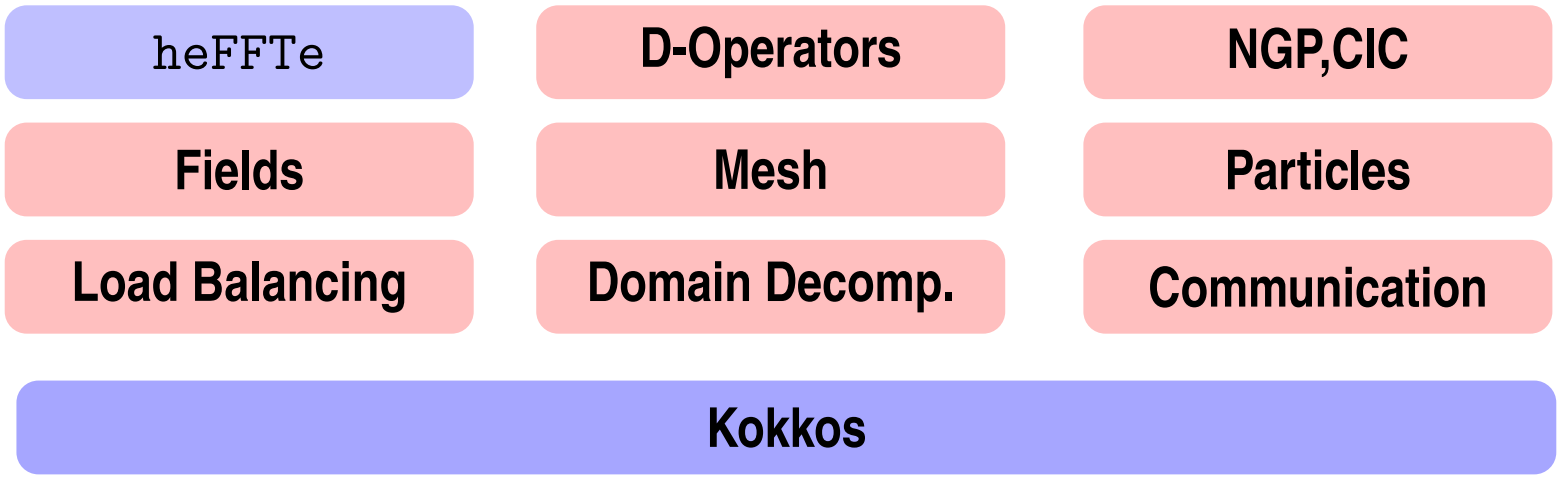
- Developed at UTK
- Part of the Exascale Project from the U.S. Department of Energy's Office of Science and National Nuclear Security Administration (NNSA)
- Delivers algorithms for distributed fast-Fourier transforms designed to run on exascale machines.



Background

Independent Parallel Particle Layer (IPPL)

- Developed in C++.
- Contains the implementations of main data structures (particles, meshes) and operations for PIC codes. [Muralikrishnan et al., 2021]
- Used for simulations of charged particle optics in accelerators.
- 2.0 main feature: **Performance portability** and **dimension independence** of data structures.



Goals

At PSI:

Work on the IPPL code to make it type-independent and allow simulations in mixed precision.

At UTK:

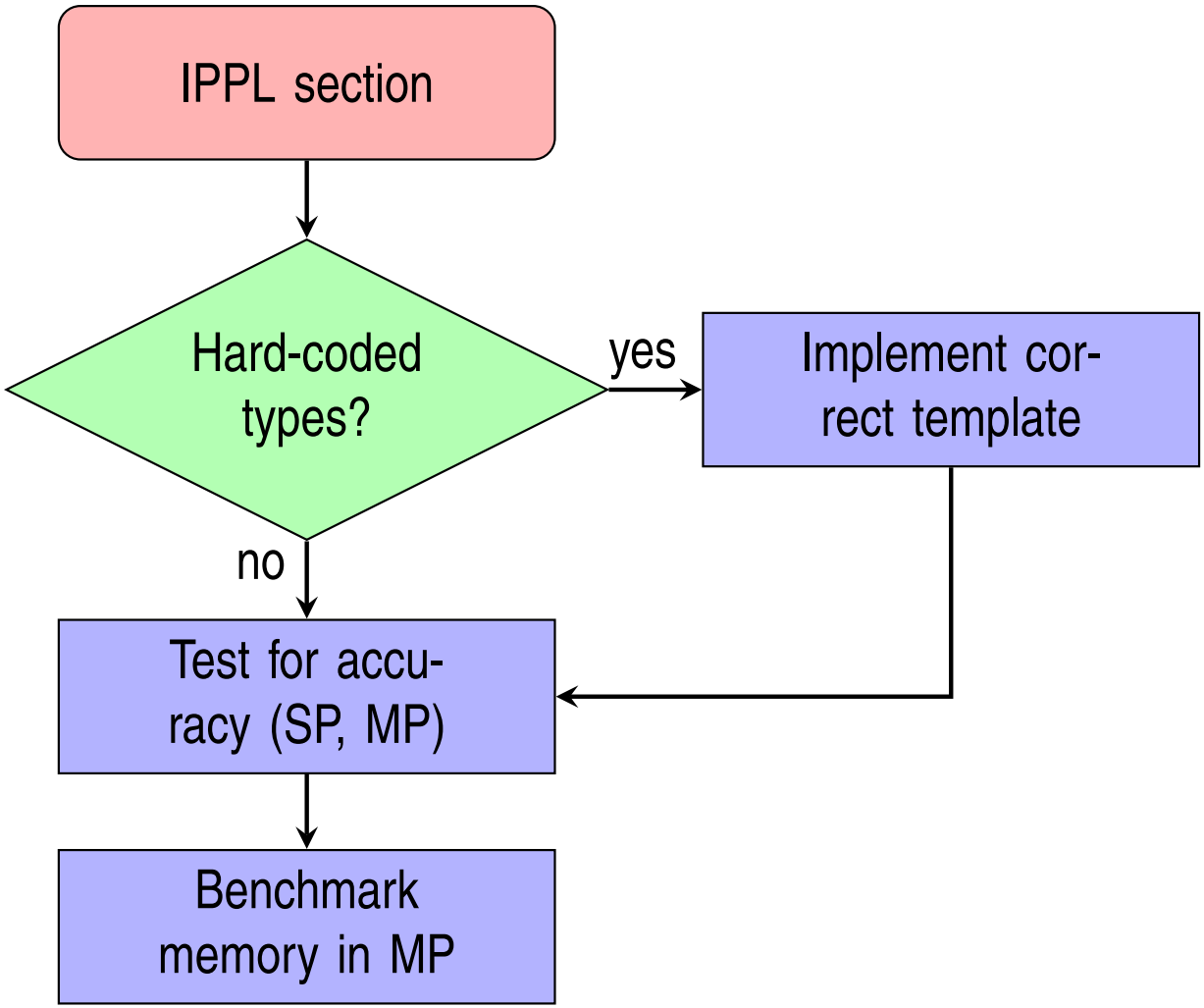
Work with the developers of the HeFFTe library to develop a GPU and CPU compatible implementation of the type 1 Discrete Cosine Transform (DCT-I).

Memory limitation given by types

- GPUs have limited memory space and larger size problems may overcome the performance benefits.
- Limiting double precision data structures saves memory.
- At the same time, keeping **some** structures in double precision limits truncation errors.

Mixed precision

Workflow



Mixed precision

Memory profiling

Kokkos-tools

Kernel Inspection | **3D Party Profiling** | **Memory Analysis**

HighWater | **Events** | **Usage**

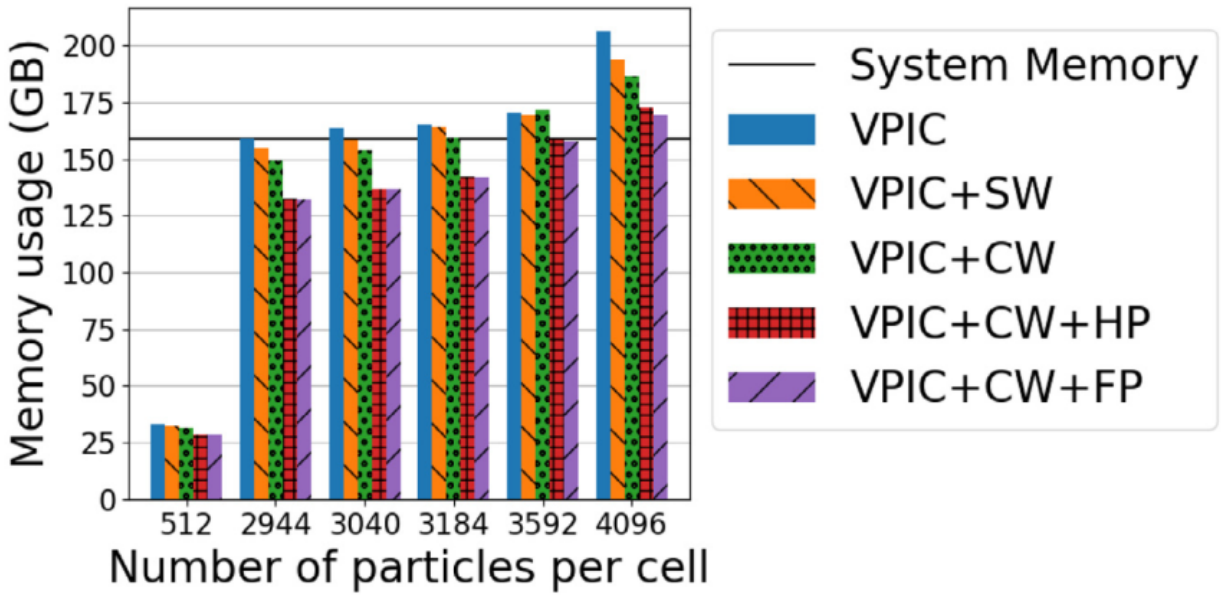


Image taken from [Tan et al., 2022].

Poisson problem with open boundary conditions

At each time step of the Particle In Cell loop, $\vec{E}(\vec{x})$ is computed by means of the Poisson equation

$$\vec{E}(\vec{x}) = -\vec{\nabla}\phi(\vec{x})$$

Consider open boundary conditions, s.t.:

$$\phi \to 0 \text{ as } |\vec{x}| \to 0$$

Then we can write ϕ as:

$$\phi(\vec{x}) = \int G(\vec{x} - \vec{x'})\rho(\vec{x})d\vec{x}$$

And G is called **Green's function for the Poisson problem**.

DCT-I

Methods for solving the Poisson equation currently implemented in IPPL:

Hockney-Eastwood [Eastwood and Brownrigg, 1979]

- Green's function needs to be transformed in frequency domain.
- Second-order accurate.
- Uses less memory.

Vico-Greengard [Vico et al., 2016]

- Green's function is pre-computed in frequency domain and defined as

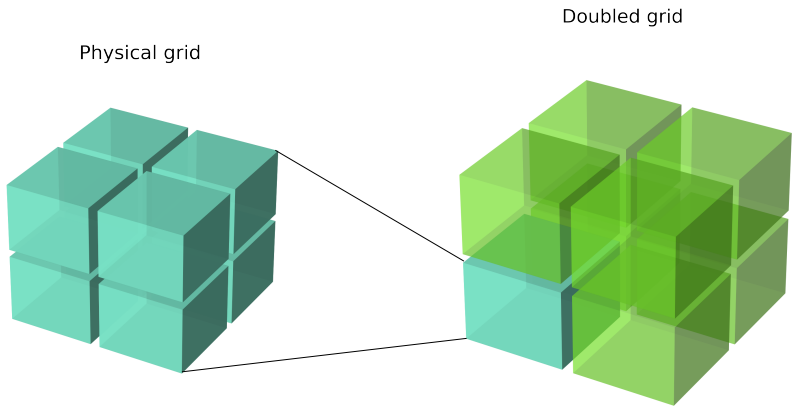
$$G(\vec{s}) = 2\left(\frac{\sin(L|\vec{s}|/2)}{|\vec{s}|}\right)^2$$

- Spectrally accurate.
- Requires more memory.

For implementation details, see [Mayani, 2021].

Domain expansion in Vico

- To avoid aliasing errors, need $4N^3$ grid to capture frequency content of Green's function
- In algorithm: create new `Field` and `Mesh` objects for the expanded domain, assign values of ρ and zero-pad it on the rest of the domain



Credits: [Mayani, 2021]

DCT-I

Vico solver

Problem

- Vico-Greengard converges spectrally, but memory occupation is of $\sim N_{field}(4N)^3$!
- Hockney needs "only" $(2N)^3$
- It could lead to memory overflows on GPU

DCT-I as a tool for a spectrallly accurate FFT Poisson solver

- The discrete cosine transform is equivalent to a Discrete Fourier Transform (DFT) of twice the length operating on periodic and symmetric coefficients.
- Instead of inverse-FFT on a domain of size $4N^3$, pre-compute G with the inverse DCT
- In this way, the domain size is reduced to $2N^3$ (same size used in Hockney)

Expected 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
- May
  - ☐ Wrap-up mixed precision
  - ☐ Finalise reading documentation and literature for DCT-I
- June & July
  - ☐ Work on DCT-I
  - ☐ Test with Vico
- August
  - ☐ Wrap-up and write report

Work done so far

Basic classes

Unit tests

- Created a folder for unit tests of basic classes to be run on single or mixed precision.
- Needed both for testing correctness of template and check accuracy.

Work done so far

Basic classes

Binary Balancer

- Part of the `FieldLayout` class, used in load balancing.
- Removed any dependency from `double`.

Orthogonal Recursive Bisection

- Used for domain repartition.
- Receives a `Field` object and a `ParticleAttribute` object representing the particle's position.
- Separated the field's type and the position's type.

Work done so far

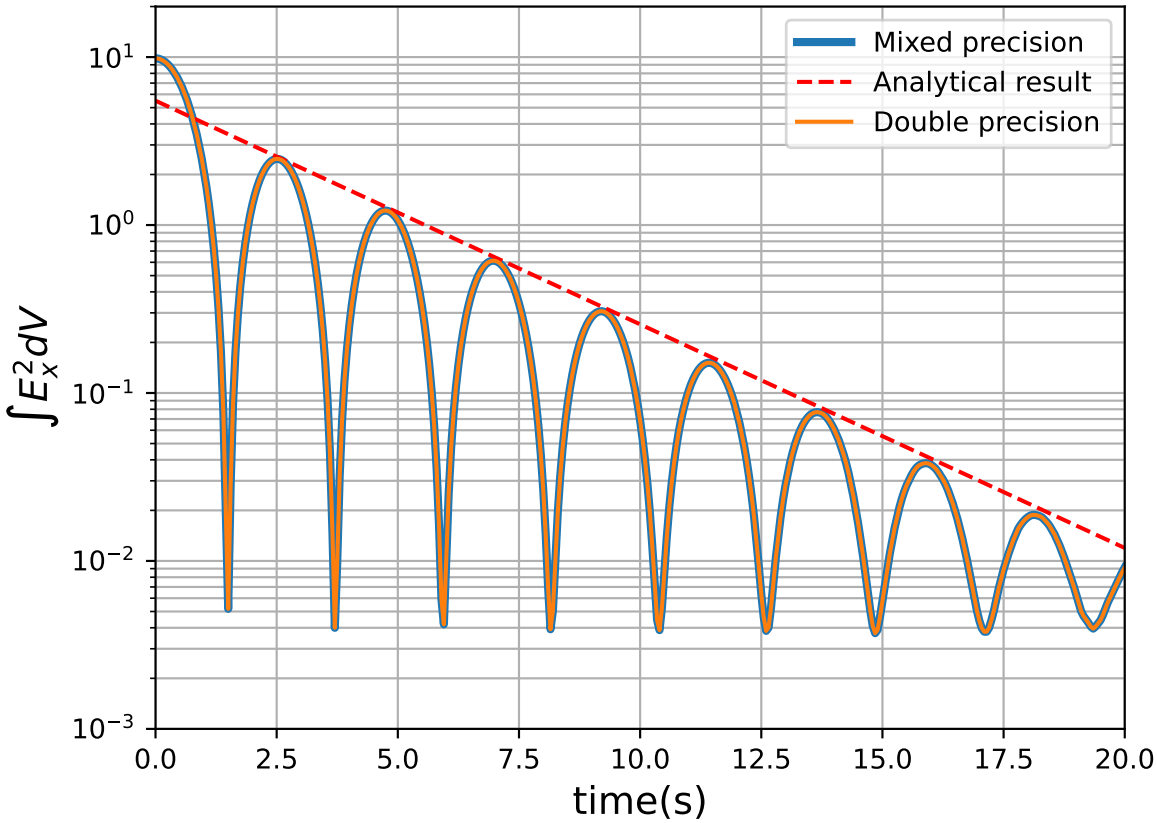
FFTPeriodicPoissonSolver

Dependency with Alpine

- Alpine mini-apps ([Muralikrishnan et al., 2022]): based on the class `ChargedParticles.hpp`
- This class is based on an FFT periodic Poisson solver.
- Fixed the solver's template ➔ Mini apps can be run in mixed precision
- Candidate mini-app for test runs and benchmarking: **Landau damping in weak regime**.

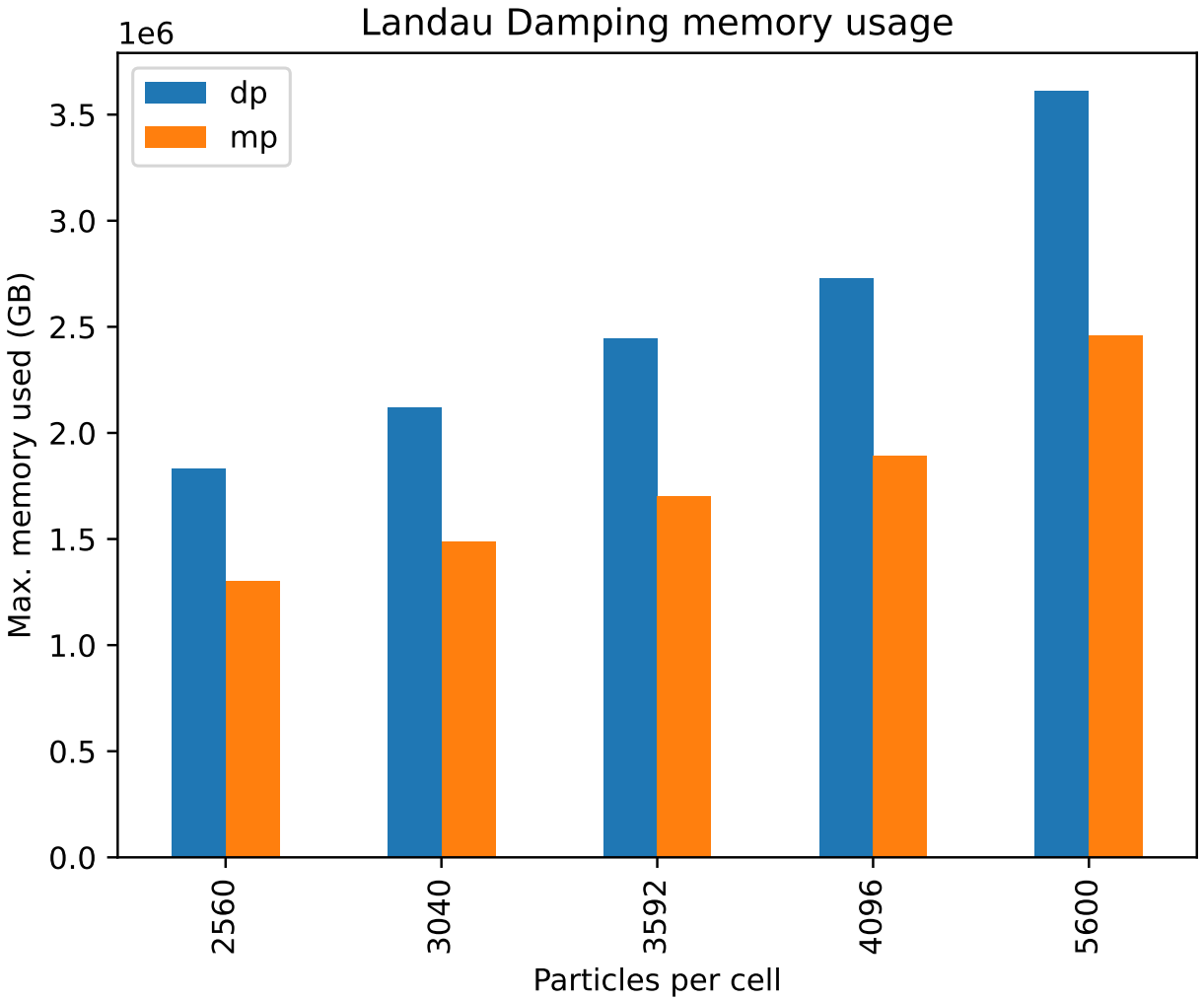
Type	Simulation parameters				
	E_m	ρ	Charge	Position	Velocity
`double`		✔	✔		
`float`	✔			✔	✔

Accuracy check



Comparison between simulation damping ratio of the energy norm and analytical damping ratio, for grid size 32^3, $\sim 10^7$ total particles, and timestep 0.05s/step

Memory benchmarking



Memory Highwater results with grid size 32^3, 8 nodes, 16 MPI ranks. Averaged on 16 runs.

Work done so far

FFTPoissonSolver

pack and unpack

- Communication specific routines that assign data from the MPI buffer to the field data view (or viceversa).
- Field buffer and data view types are now independent from eachother.
- This means scalar and vector field can have different types

```
template <typename Tb, typename Tf>
void pack(const ippl::NDIndex<3> intersect, Kokkos::View<Tf***>& view,
    ippl::detail::FieldBufferData<Tb>& fd,...)


template <typename Tb, typename Tf>
void unpack(const ippl::NDIndex<3> intersect,const Kokkos::View<Tf***>&
    view, ippl::detail::FieldBufferData<Tb>& fd,..)
```

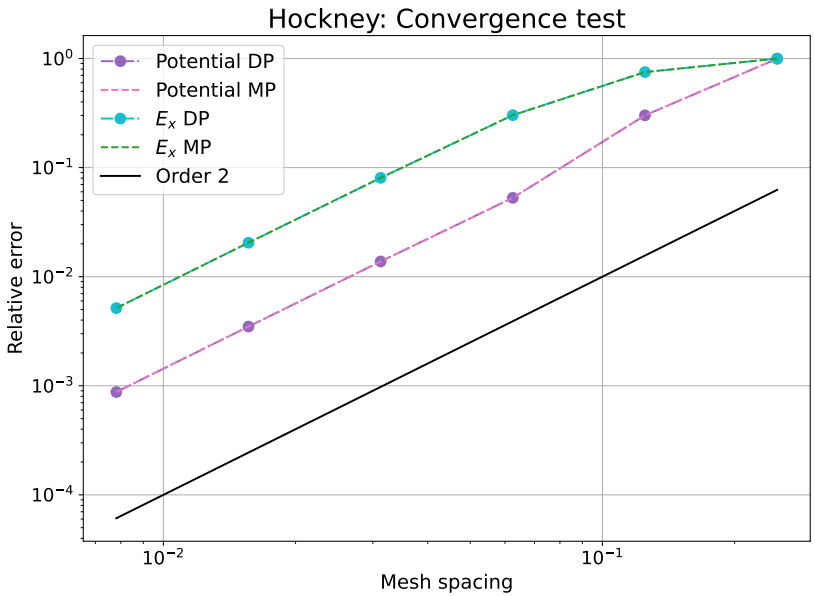
Convergence study and memory profiling

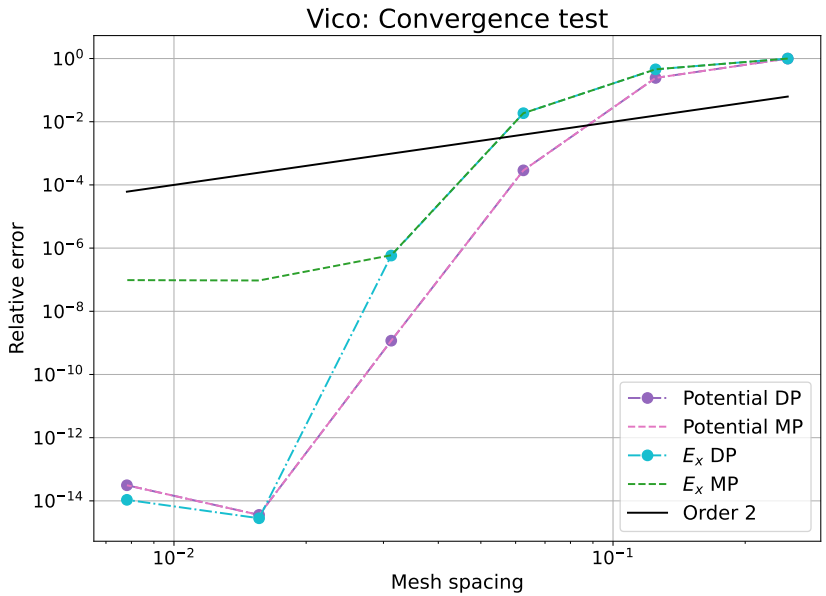
Simulation details

- With ρ in double precision, E in single precision
- Ran `TestGaussian_convergence.cpp` For both Vico and Hockney and then ran it again with E=`double`
- 8 nodes, 16 MPI ranks

	Mesh sizes	
	Convergence study	Memory profiling
Hockney	$\{4^3, 8^3, 16^3, ..., 128^3\}$	$\{64^3, 128^3, 256^3, ..., 1024^3\}$
Vico	$\{4^3, 8^3, 16^3, ..., 128^3\}$	$\{32^3, 64^3, 128^3, ..., 512^3, \}$

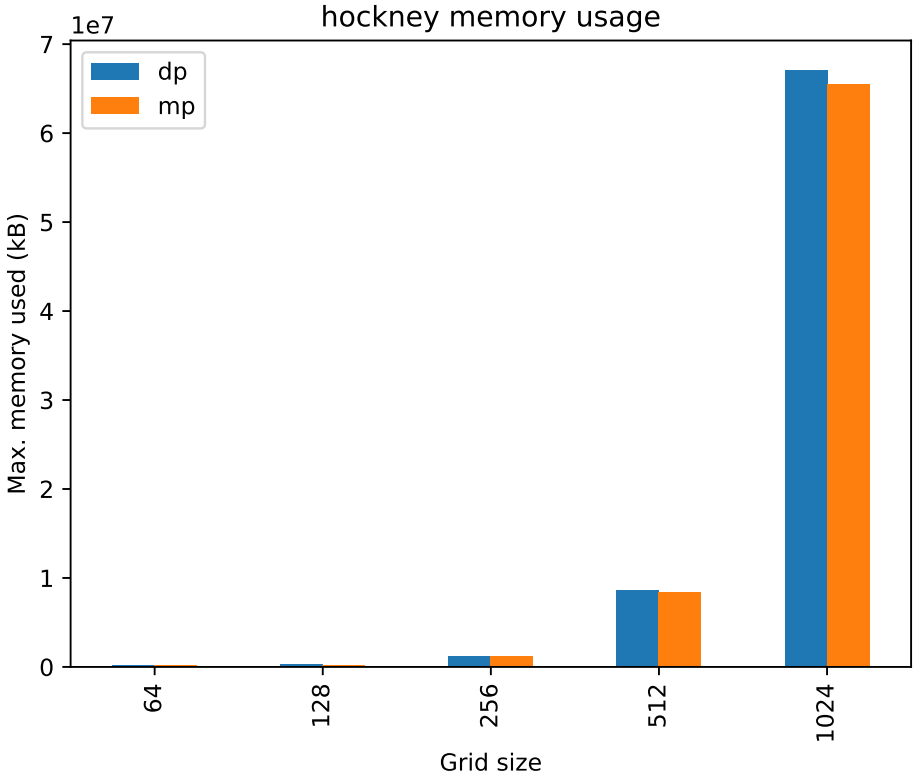
Convergence results

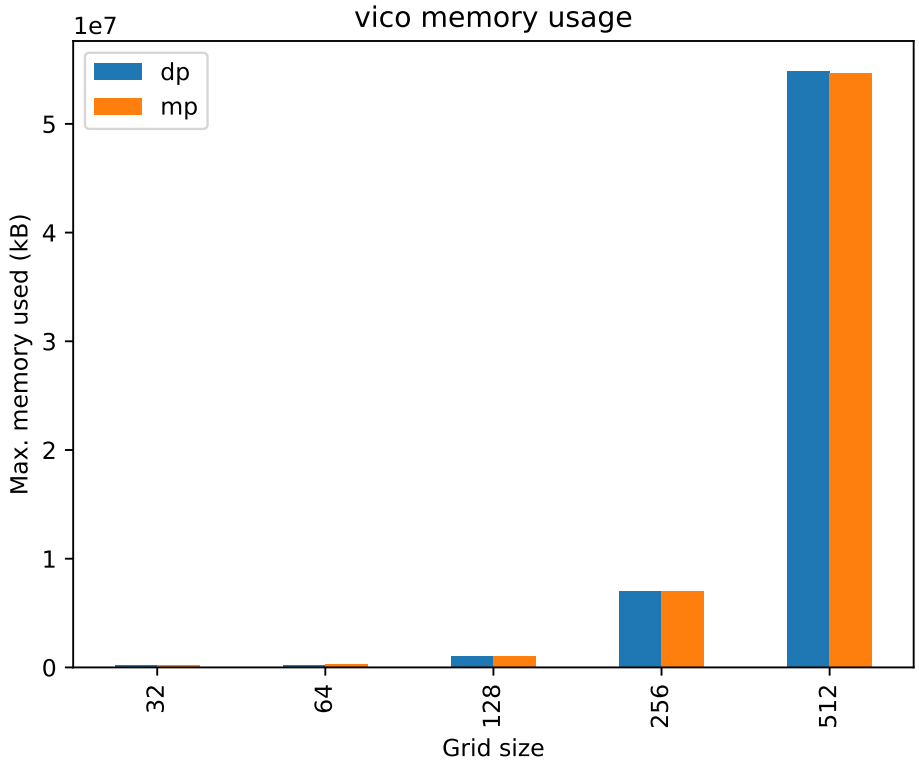




Convergence study for the solvers in mixed precision, versus double precision.

Memory benchmarking results





Memory highwater results for 8 runs, with Vico and Hockney solvers, obtained by increasing mesh size

Next step

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 doesn't affect memory in a significant way.

Possible solution

- Make a table of all the field in `FFTPoissonSolver` in order to keep track of where they're used in the code
- Analyse which fields would be worth to make independent from ρ's type.
- Starting by making the type of the Green function independent could be a good idea.

Personal goals

What do I wish to learn?

- Increase skills in C++ and GPU computing
- Learn more about optimization techniques
- Finalise what i started in my semester project
- Coding in an academic enviroment
- Interact with researchers from other universities

Thanks

Thanks everyone for your attention.
Open to 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.

Eastwood, J. and Brownrigg, D. (1979).

Remarks on the solution of poisson's equation for isolated systems.

Journal of Computational Physics, 32(1):24–38.

Mayani, S. (2021).

A performance portable poisson solver for the hose instability.

Muralikrishnan, S., Frey, M., Vinciguerra, A., Ligotino, M., Cerfon, A. J., Stoyanov, M., Gayatri, R., and Adelmann, A. (2021).

IPPL GitLab wiki.

`https://gitlab.psi.ch/OPAL/Libraries/ippl/-/wikis/home`.

Muralikrishnan, S., Frey, M., Vinciguerra, A., Ligotino, M., Cerfon, A. J., Stoyanov, M., Gayatri, R., and Adelmann, A. (2022).

Scaling and performance portability of the particle-in-cell scheme for plasma physics applications through mini-apps targeting exascale architectures.

`https://arxiv.org/abs/2205.11052`.

Tan, N., Bird, R. F., Chen, G., Luedtke, S. V., Albright, B. J., and Taufer, M. (2022).

Analysis of vector particle-in-cell (vpic) memory usage optimizations on cutting-edge computer architectures.

Journal of Computational Science, 60:101566.

Vico, F., Greengard, L., and Ferrando, M. (2016).

Fast convolution with free-space green's functions.