MAX-PLANCK-INSTITUT
FÜR PLASMAPHYSIK

STRUPHY

domain cloning and profiling with pylikwid

Max Lindqvist
Max Planck Institute for Plasma Physics, Germany

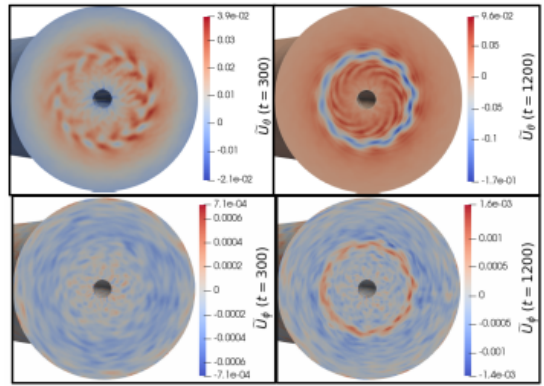
EUROfusion

This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

Table of Contents

StruPhy: Structure-Preserving Hybrid Codes

- Solution of PDEs using Geometric Finite Element Exterior Calculus (FEEC) and Particle-in-Cell (PIC) methods in 3D
- Features a growing list of PDE models applicable to a variety of mapped domains
- Open-source and can be installed on any architecture
- Written in **Python**, kernels traslated to C/Fortran with `pyccel`



Zonal flow components of a toroidal Alfvén eigemode

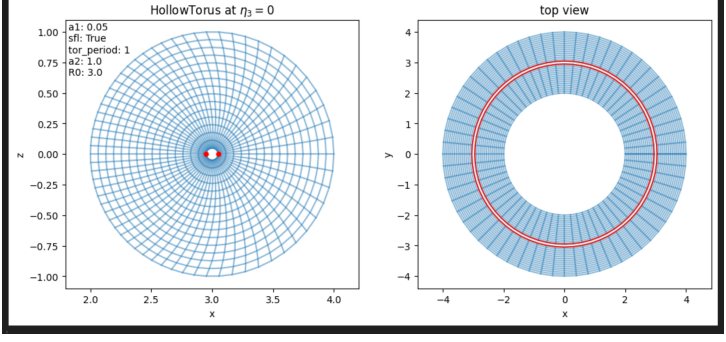
Why Python? - Philospohy of Struphy development

1. **Clarity, usability and flexibility first**.
2. Be easily accessible/usable by all scientists, for free.
3. Seamless integration with `Python` ecosystem:
   - `psydac`: MPI-distributed high-order spline library
   - `pyccel`: translate and compile in C or Fortran
   - `mpi4py`: use MPI from Python
   - `numpy`: fast linear algebra
   - `desc-opt`: Stellarator equilibria
   - `pytest`: unit testing
   - and countless more ...
4. Provide an abstract framework for adding **new physics models, quickly**.

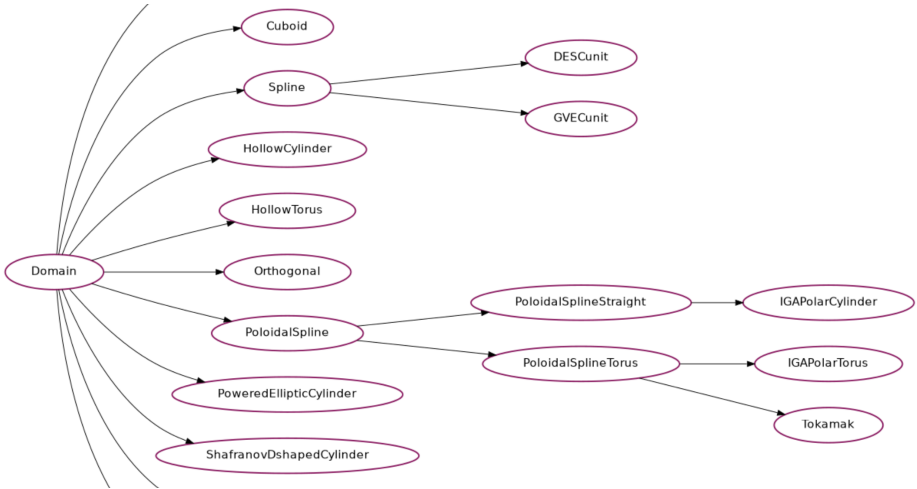
How Struphy works

1. Through the command line: `$ struphy run LinearMHD --mpi 8 -o DIR`
2. Through the Struphy API:

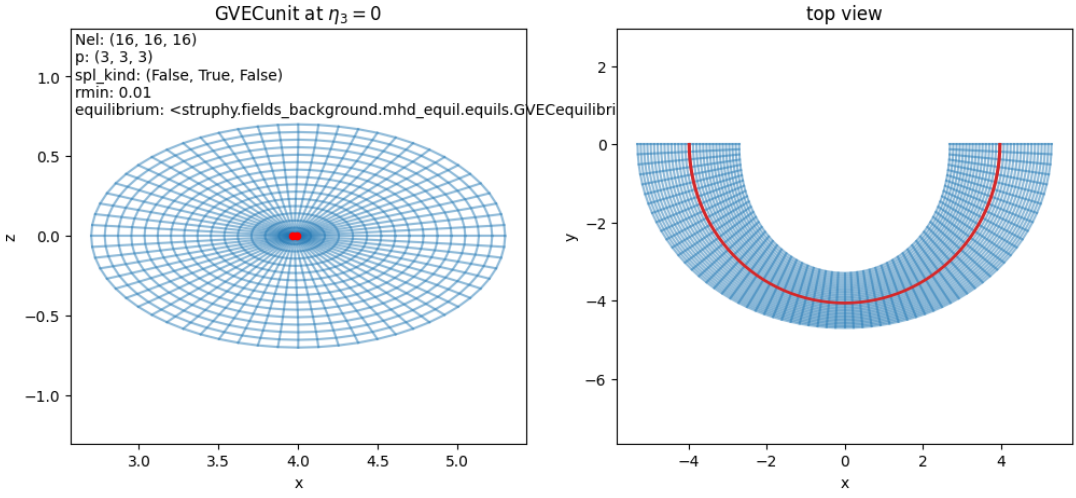
```
1 from struphy.geometry import domains
2 domain = domains.HollowTorus(a1=.05, sf1=True, tor_period=1)
3 domain.show()
✓ 1.0s
```



`StruphyModel` can be launched in curvilinear coordinates



Grid example: from GVEC output



Struphy Models

Struphy features a variety of plasma models for different physics scenarios

- **Fluid models**: `LinearMHD`, `LinearExtendedMHDuniform`, `ColdPlasma`, `VariationalMHD`, `ViscoresistiveMHD`, `ViscousFluid`
- **Kinetic models**: `VlasovAmpereOneSpecies`, `VlasovMaxwellOneSpecies`, `LinearVlasovAmpereOneSpecies`, `DriftKineticElectrostaticAdiabatic`
- **Fluid-kinetic hybrid models**: `LinearMHDVlasovCC`, `LinearMHDVlasovPC`, `LinearMHDDriftkineticCC`, `ColdPlasmaVlasov`
- **Toy models**: `Maxwell`, `Vlasov`, `GuidingCenter`, `ShearAlfven`, `VariationalPressurelessFluid`, `VariationalBarotropicFluid`, `VariationalCompressibleFluid`, `Poisson`, `DeterministicParticleDiffusion`, `RandomParticleDiffusion`

The `LinearMHDDriftkineticCC` model will be used for profiling

Struphy propagators

Propagators are the main building blocks of Struphy models, a large list are available

- **Field propagators**: `Maxwell` , `OhmCold` , `JxBCold` , `ShearAlfven` , `ShearAlfvenB1` , `Hall` , `Magnetosonic` , `MagnetosonicUniform` , `FaradayExtended` , `CurrentCoupling6DDensity` , `ShearAlfvenCurrentCoupling5D` , `CurrentCoupling5DDensity` , `ImplicitDiffusion` , `VariationalMomentumAdvection` , `VariationalDensityEvolve` , `VariationalEntropyEvolve` , `VariationalMagFieldEvolve` , `VariationalViscosity` , `VariationalResistivity` , `TimeDependentSource` , `AdiabaticPhi`
- **Particle propagators**: `PushEta` , `PushVxB` , `PushEtaPC` , `PushGuidingCenterBxEstar` , `PushGuidingCenterParallel` , `PushVinEfield` , `StepStaticEfield` , `PushDeterministicDiffusion` , `PushRandomDiffusion`
- **Particle-field coupling propagators**: `VlasovAmpere` , `EfieldWeights` , `EfieldWeightsImplicit` , `EfieldWeightsDiscreteGradient` , `EfieldWeightsAnalytic` , `PressureCoupling6D` , `CurrentCoupling6DCurrent` , `CurrentCoupling5DCurlb` , `CurrentCoupling5DGradB`

The propagators for `LinearMHDDriftkineticCC` are marked in red

Table of Contents

LIKWID

- LIKWID (Like I Knew What I'm Doing) is a performance monitoring and benchmarking tool for multi-core processors.
- Developed to simplify the access to performance counters on modern processors.
- Selected tools
  - `likwid-topology` : print thread, cache and NUMA topology.
  - `likwid-perfctr` : configure and read out hardware performance counters.
  - `likwid-pin` : pin your threaded application.
  - `likwid-mpirun` : Wrapper to start MPI and Hybrid MPI/OpenMP applications.

Pylikwid - python marker API

- Python wrapper for LIKWID, access to LIKWID marker API directly in Python scripts.
- `likwid-perfctr -C 0 -g MEM_DP -m python test.py`

```
# test.py
import pylikwid
import numpy as np
pylikwid.markerinit()
pylikwid.markerthreadinit()
N = 100_000
a = np.random.random((N,1))
b = np.random.random((N,1))
c = np.random.random((N,1))
pylikwid.markerstartregion("RegionID")
d = a + b * c
pylikwid.markerstopregion("RegionID")
pylikwid.markerclose()
```

Pylikwid - Struphy implementation

- `ProfilingConfig`: Singleton class for managing global profiling settings.
- `ProfileManager`: Manages profiling regions, collects data across MPI processes, and saves profiling results for analysis.

Pylikwid - Struphy implementation

- `ProfilingConfig`: Singleton class for managing global profiling settings.
- `ProfileManager`: Manages profiling regions, collects data across MPI processes, and saves profiling results for analysis.

```
config = ProfilingConfig()
config.likwid = args.likwid # Turn on likwid profiling with --likwid
# ...
for propagator in self.propagators:
    prop_name = type(propagator).__name__
    with ProfileManager.profile_region(prop_name):
        propagator(dt)
```

Pylikwid - Struphy implementation

- `ProfilingConfig`: Singleton class for managing global profiling settings.
- `ProfileManager`: Manages profiling regions, collects data across MPI processes, and saves profiling results for analysis.

```
config = ProfilingConfig()
config.likwid = args.likwid # Turn on likwid profiling with --likwid
# ...
for propagator in self.propagators:
    prop_name = type(propagator).__name__
    with ProfileManager.profile_region(prop_name):
        propagator(dt)
```

```
struphy run LinearMHDDriftkineticCC --mpi 32 --likwid
```

- Profiling with `pylikwid` integration to the weekly scheduled CI using private runners (in progress).

Profiling on Raven - `LinearMHDDriftkineticCC`

Particle propagators

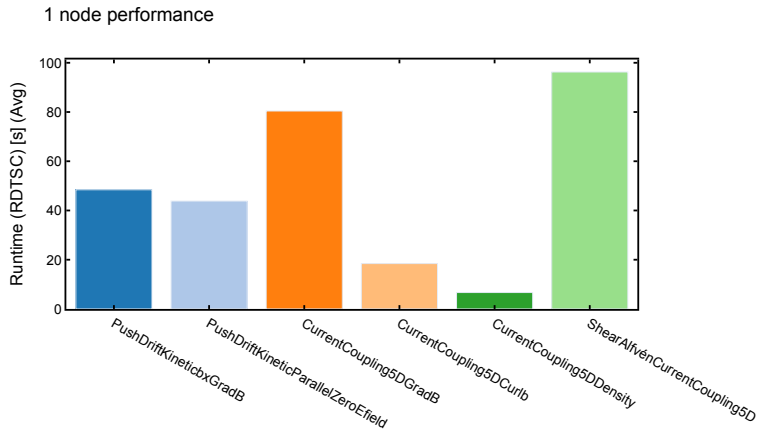
- `PushGuidingCenterBxEstar`
- `PushGuidingCenterParallel`

Particle-field coupling propagators

- `CurrentCoupling5DCurlb`
- `CurrentCoupling5DGradB`

Field propagators

- `ShearAlfvenCurrentCoupling5D`
- `CurrentCoupling5DDensity`



Profiling on Raven - `LinearMHDDriftkineticCC`

Particle propagators

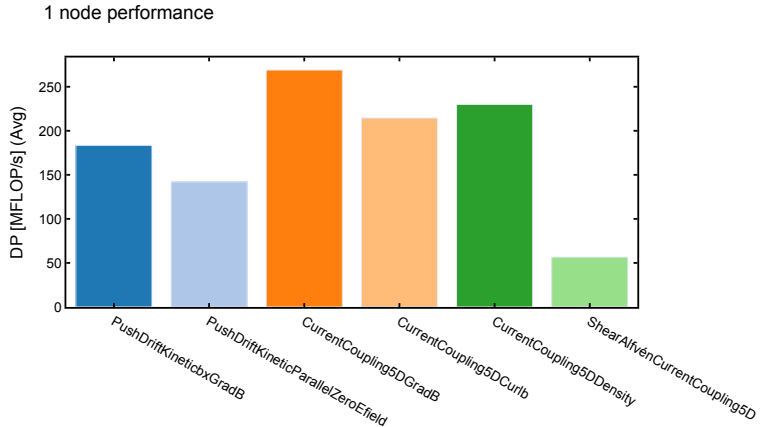
- `PushGuidingCenterBxEstar`
- `PushGuidingCenterParallel`

Particle-field coupling propagators

- `CurrentCoupling5DCurlb`
- `CurrentCoupling5DGradB`

Field propagators

- `ShearAlfvenCurrentCoupling5D`
- `CurrentCoupling5DDensity`



Profiling on Raven - `LinearMHDDriftkineticCC`

Particle propagators

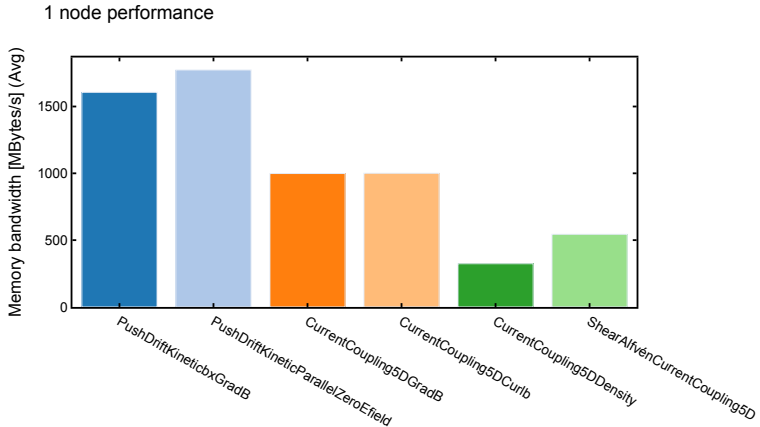
- `PushGuidingCenterBxEstar`
- `PushGuidingCenterParallel`

Particle-field coupling propagators

- `CurrentCoupling5DCurlb`
- `CurrentCoupling5DGradB`

Field propagators

- `ShearAlfvenCurrentCoupling5D`
- `CurrentCoupling5DDensity`



Profiling on Raven - `LinearMHDDriftkineticCC`

Particle propagators

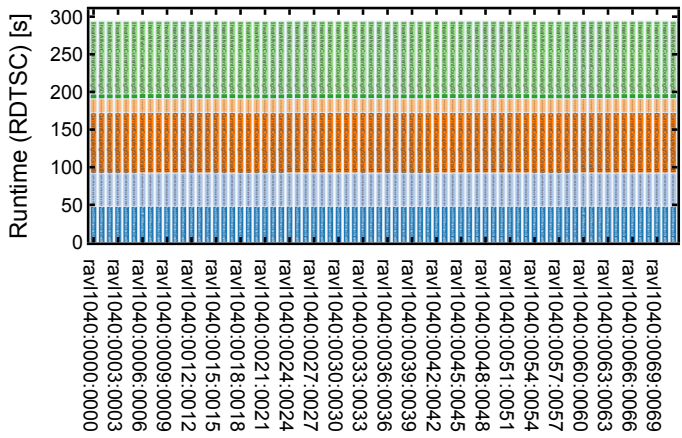
- `PushGuidingCenterBxEstar`
- `PushGuidingCenterParallel`

Particle-field coupling propagators

- `CurrentCoupling5DCurlb`
- `CurrentCoupling5DGradB`

Field propagators

- `ShearAlfvenCurrentCoupling5D`
- `CurrentCoupling5DDensity`



Profiling on Raven - `LinearMHDDriftkineticCC`

Particle propagators

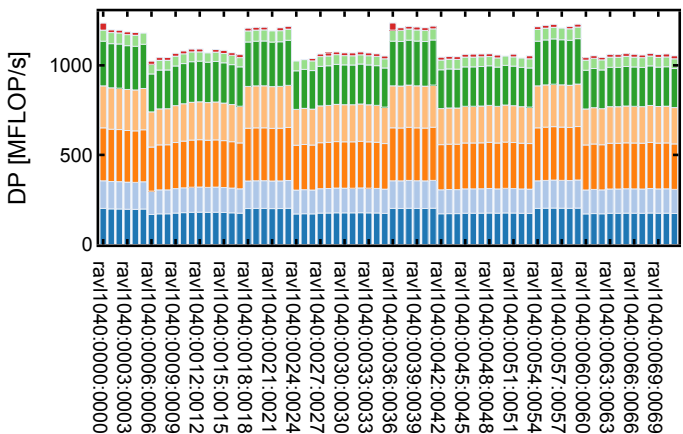
- `PushGuidingCenterBxEstar`
- `PushGuidingCenterParallel`

Particle-field coupling propagators

- `CurrentCoupling5DCurlb`
- `CurrentCoupling5DGradB`

Field propagators

- `ShearAlfvenCurrentCoupling5D`
- `CurrentCoupling5DDensity`



Profiling on Raven - `LinearMHDDriftkineticCC`

Particle propagators

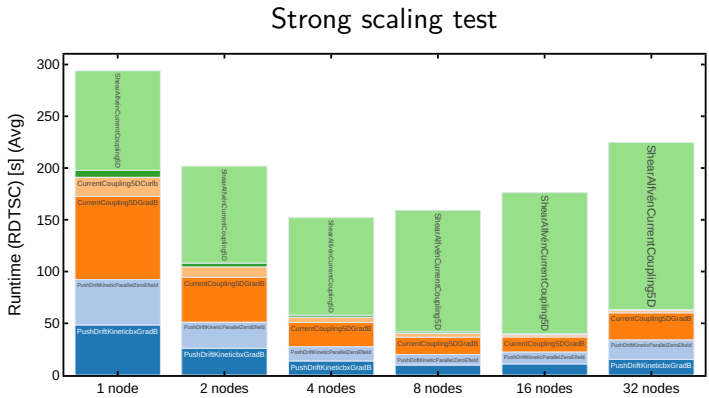
- `PushGuidingCenterBxEstar`
- `PushGuidingCenterParallel`

Particle-field coupling propagators

- `CurrentCoupling5DCurlb`
- `CurrentCoupling5DGradB`

Field propagators

- `ShearAlfvenCurrentCoupling5D`
- `CurrentCoupling5DDensity`



Profiling on Raven - `LinearMHDDriftkineticCC`

Particle propagators

- `PushGuidingCenterBxEstar`
- `PushGuidingCenterParallel`

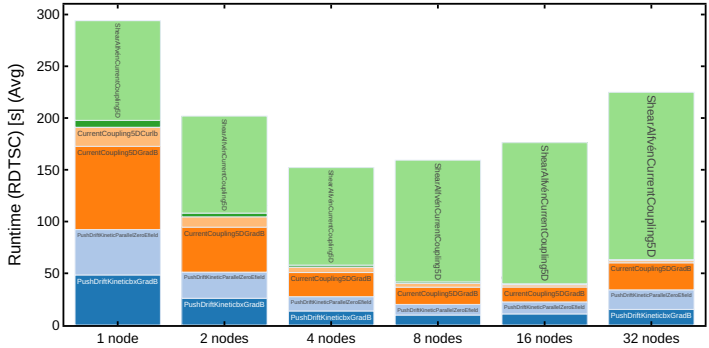
Particle-field coupling propagators

- `CurrentCoupling5DCurlb`
- `CurrentCoupling5DGradB`

Field propagators

- `ShearAlfvenCurrentCoupling5D`
- `CurrentCoupling5DDensity`

Strong scaling test

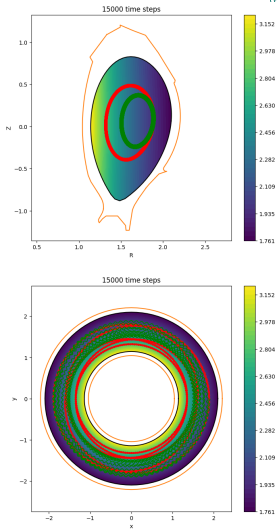


Next: Improve scaling of the particle propagators

Table of Contents

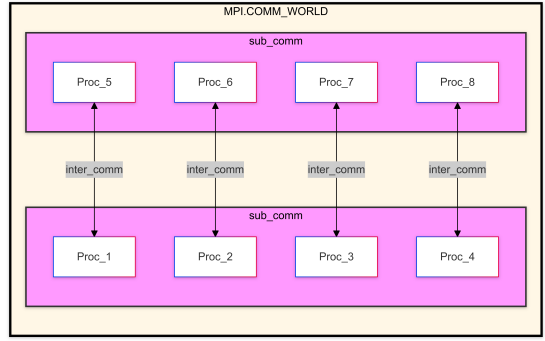
Domain Cloning

- $N_{\text{particles}}$ are evenly distributed among N_{clones} clones.
- Each clone independently calculates the charge density, summed across all clones.
- Field equations are solved separately within each clone.
- Particles propagate independently within the clones.
- Grid resolution decoupled from the number of processing elements, allowing for more MPI tasks than the number of toroidal grid points.



Domain Cloning - Communicators

- Additional communicators are initialized.
- `sub_comm`: within clones.
- `inter_comm`: between clones.
- Implementation: `sub_comm` replaces `MPI.COMM_WORLD` in the model, `inter_comm` is called after particle accumulation.




```
inter_comm.Allreduce(MPI.IN_PLACE, data_array, op=MPI.SUM)
data_array /= Nclones
```

Domain Cloning - `LinearMHDDriftkineticCC`

Particle propagators

- `PushGuidingCenterBxEstar`
- `PushGuidingCenterParallel`

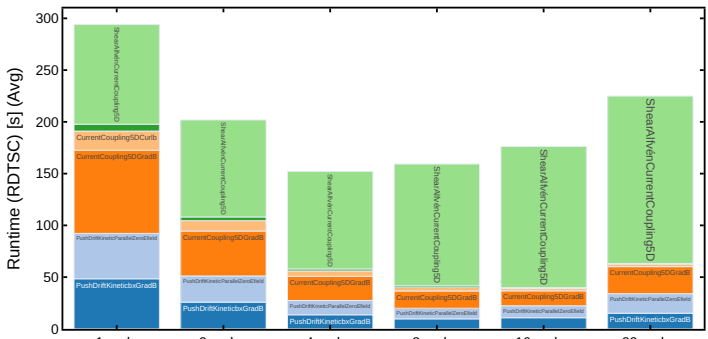
Particle-field coupling propagators

- `CurrentCoupling5DCurlb`
- `CurrentCoupling5DGradB`

Field propagators

- `ShearAlfvenCurrentCoupling5D`
- `CurrentCoupling5DDensity`

Scaling without domain cloning



Domain Cloning - `LinearMHDDriftkineticCC`

Particle propagators

- `PushGuidingCenterBxEstar`
- `PushGuidingCenterParallel`

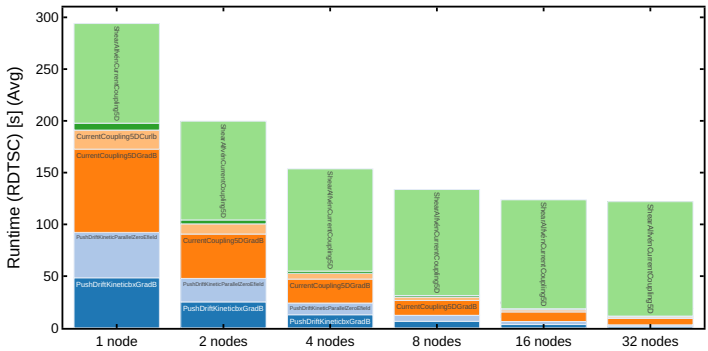
Particle-field coupling propagators

- `CurrentCoupling5DCurlb`
- `CurrentCoupling5DGradB`

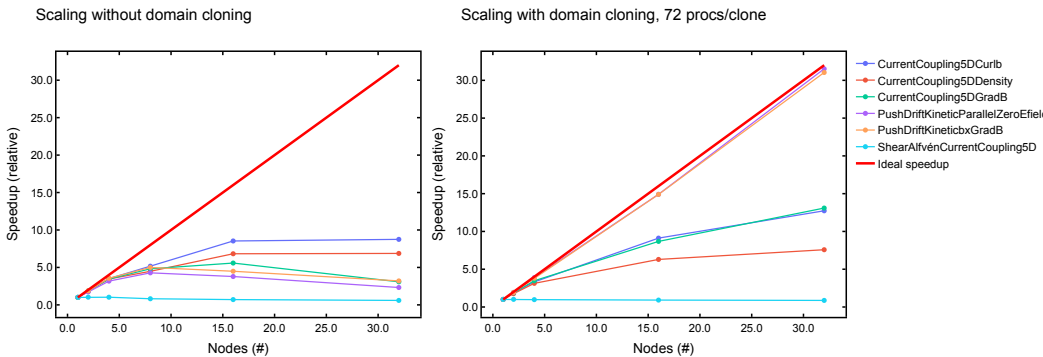
Field propagators

- `ShearAlfvenCurrentCoupling5D`
- `CurrentCoupling5DDensity`

Scaling with domain cloning, 72 procs/clone



Domain Cloning - `LinearMHDDriftkineticCC`



Conclusion: Particle propagators scale well using domain cloning.

Summary

- **Struphy**: Python-based, structure-preserving hybrid simulation framework for plasma physics.
- **Profiling**: Pylikwid integration provides detailed performance metrics, easily accessible for users.
- **Domain Cloning**: Enables scalable simulations of the particle propagators.
- **In progress**: Interface to PETSc

Installing Struphy

- Virtual environment:

```
python -m venv env
source env/bin/activate
```

- Current (stable) version:

```
pip install struphy
struphy compile
```

https://struphy.pages.mpcdf.de/struphy/index.html

- Install from source:

```
git clone https://gitlab.mpcdf.mpg.de/struphy/struphy.git
cd struphy
pip install -e .
struphy compile
```

Get in touch

- Documentation: `https://gitlab.mpcdf.mpg.de/struphy/struphy`
- Repository: `https://gitlab.mpcdf.mpg.de/struphy/struphy`
- Ticket system: `https://gitlab.mpcdf.mpg.de/struphy/struphy/-/issues`
- Developer's channel: `https://chat.gwdg.de/channel/struphy-developers`
- Follow Struphy on LinkedIn: `https://www.linkedin.com/company/struphy/`
- Main contacts:
  - Stefan Possander
  - Eric Sonnendrücker
  - Xin Wang