

Gyselalib++

E. Bourne,
V. Grandgirard

Gysela Past and Future

Gysela (Fortran)

- 5D
- MPI/OpenMP
- Optimised up to 730k CPU cores
- Relative efficiency of 85% on more than 500k cores and 63% on 730k cores on CEA-HF (AMD EPYC 7763)
- Petascale resources: ~ 150 millions of hours / year (GENCI + PRACE + HPC Fusion resources)

Gyselalib++ (C++) so far...

- 2D/4D
- MPI/Kokkos

Gyselalibxx

- CI:
 - unit tests
 - code conventions
 - best practices
 - common bug detection
 - forced documentation

Gyselalib++ Documentation

Gyselalib++ is a collection of C++ components for writing gyrokinetic semi-lagrangian codes and similar as well as a collection of such codes. It is based on DDC. We strongly encourage new developers to begin by reading our documentation about [Using DDC in Gyselalibxx](#).

Set-up

In order to set up Gyselalib++ on a new machine, simply run:

```
git clone --recurse-submodules git@github.com:maisondelasimulation.fr:gyselax-developers/gyselalibxx.git gyselalibxx
cd gyselalibxx
./bin/install-hooks
```

or

```
git clone --recurse-submodules https://gitlab.maisondelasimulation.fr/gyselax-developers/gyselalibxx.git
cd gyselalibxx
./bin/install-hooks
```

on a machine for which Gyselalib++ is already used an environment script may be available to set up the necessary modules etc.

Please check the folder toolchains to find the existing configurations. See the documentation about [Pre-made build settings](#) for more information on the provided files.

For example in order to set up the environment on the Adastral supercalculator simply run:

```
source toolchains/m1250_hipcc_adastra.spack/prepare.sh
source toolchains/m1250_hipcc_adastra.spack/environment.sh
```

Generated by [doxygen](#) 1.9.1

<https://github.com/gyselax/gyselalibxx/>

gyselax / gyselalibxx

<> Code Pull requests 2 Discussions Actions Security Insights Settings

gyselalibxx Public Edit Pins Watch 0 Fork 6 Starred 18

main Go to file Code

PaulineVidal Mistakes in PredCorr in geomet... 843bfd6 · 18 hours ago

simulations	Add MPI to axisymmetric simula...	2 days ago
src	Mistakes in PredCorr in geomet...	18 hours ago
tests	Add MultipatchField types	2 days ago
toolchains	Remove unused local typedefs a...	2 weeks ago
vendor	Merge branch 'ebourne-366-red...	last week
docs	Merge branch 'ebourne_355_fiel...	last week
post-process/PythonScripts	Charge species must be a float	3 months ago

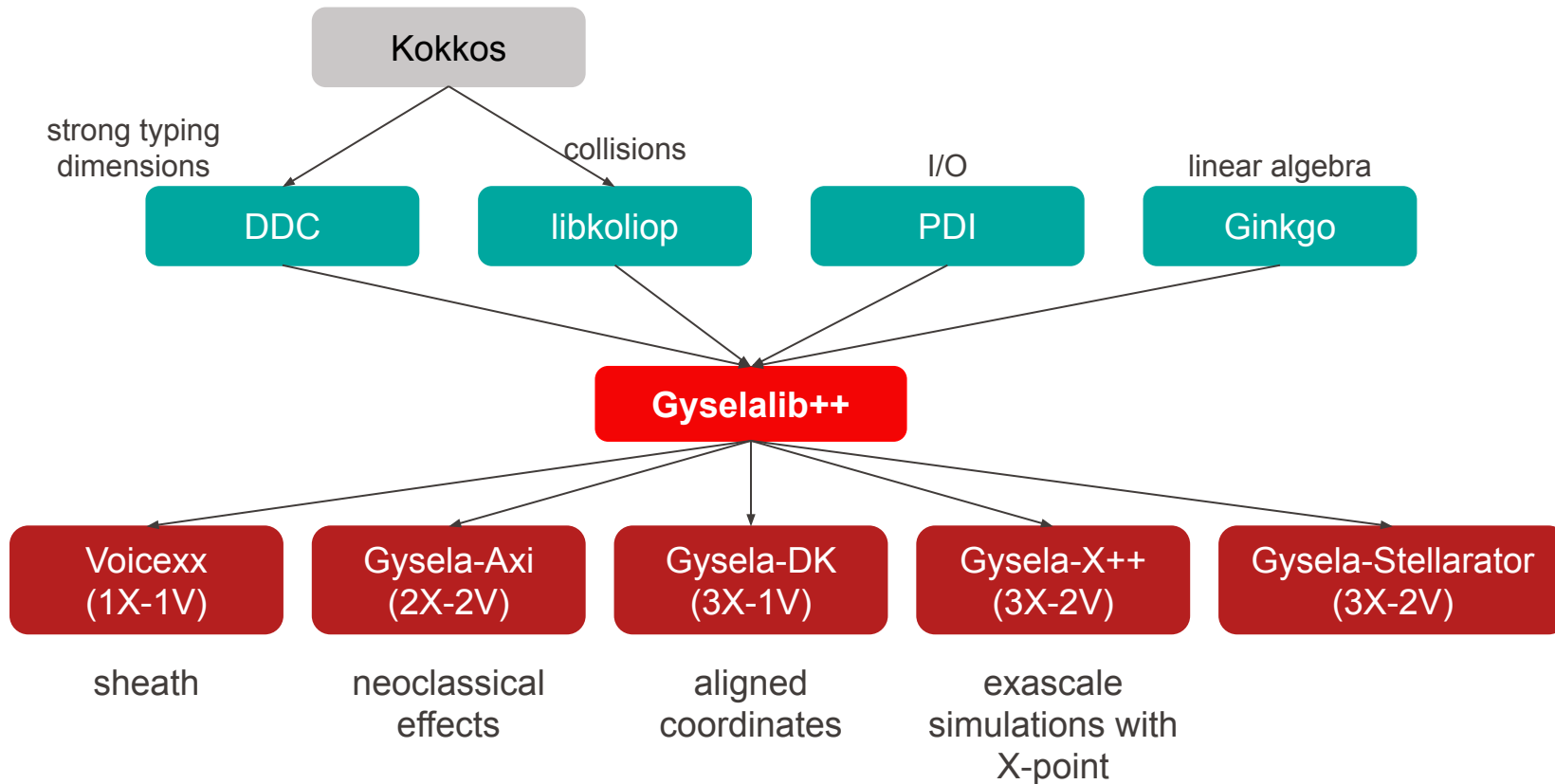
Contributors 11

gyselax.github.io/gyselalibxx/

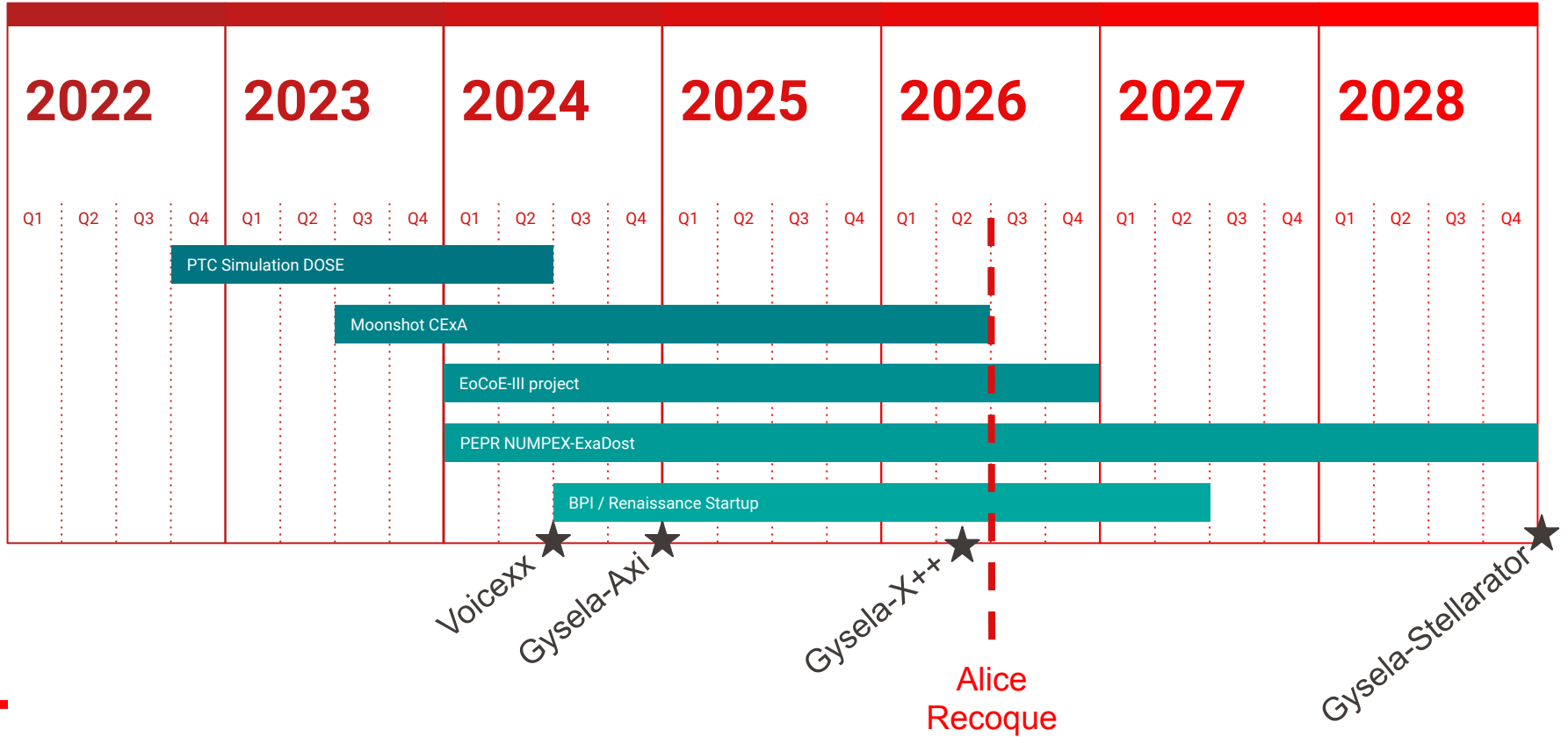
hpc ddc plasma-physics gyrokinetic vlasov-solver numerical-simulation poisson-solver

Readme MIT license Activity

Organisation



Roadmap



Work In Progress: Gysela-Axi

(2X-2V) semi-Lagrangian code for axisymmetric neoclassical simulations

- All the pieces of the puzzle are being put together
 - 2D advection in (r, θ) + 1D advection in v_{\parallel}
 - Non-uniform 1D and 2D splines including special treatment of the O-point
 - 2D poisson solver in (r, θ)
 - 2D collision operator (v_{\parallel}, μ)
 - translated from GYSELA F90 into C++ & Kokkos (3.79x speed up between 1 Genoa node and 1 AMD node)
 - libkoliop with an interface for both GYSELA F90 and Gysela-X++
 - MPI transposition
- Objective: end 2024 (EoCoE-III milestone)
 - Designed to work on multi-GPU optimization but also for physics:
 - Neoclassical effects with shaping and impurities

[PhD L. De Gianni]

VOICE - Vlasov Open boundary Ion Coupling to Electrons

GYSELA

5D Vlasov Solver

$$B_{\parallel s}^* \frac{\partial F_s}{\partial t} + \nabla \cdot \left(\frac{dx_G}{dt} B_{\parallel s}^* F_s \right) + \frac{\partial}{\partial v_{G\parallel}} \left(\frac{dv_{G\parallel}}{dt} B_{\parallel s}^* F_s \right) \\ = C(F_s) + S + K_{buffer}(F_s) + D_{buffer}(F_s)$$

- Backward semi-Lagrangian Advection on **uniform cubic splines**
- Penalisation for walls
- Collision operator

3D Poisson Solver

$$\frac{e}{T_{e,eq}} (\phi - \langle \phi \rangle) - \frac{1}{n_{e0}} \sum_s Z_s \nabla_{\perp} \cdot \left(\frac{n_{s,eq}}{B \Omega_s} \nabla_{\perp} \phi \right) \\ = \frac{1}{n_{e0}} \sum_s Z_s \int J_0 \cdot (F_s - F_{s,eq}) d^3v$$

- Finite Elements in (r, θ)
- Fourier Transform in φ

VOICE

2D Vlasov Solver

$$\partial_t f_s(t, x, v) + v \partial_x f_s(t, x, v) - \frac{q_s}{m_s} \partial_x \phi(t, x) \partial_v f_s(t, x, v) \\ = C_{ss}(t, x, v) + S_{s,w_1} + S_{s,w_2} + S_{s,k}$$

- Backward semi-Lagrangian Advection on arbitrary degree splines (SeLaLib)
- Penalisation for walls
- Collision operator

1D Poisson Solver

$$-\partial_x^2 \phi(t, x) = \frac{1}{\epsilon_0} \sum_x Z_s \int f_s(t, x, v) dv$$

- Finite Elements

VOICE - Vlasov Open boundary Ion Coupling to Electrons

$$\partial_t f_s(t, x, v) + v \partial_x f_s(t, x, v) - \frac{q_s}{m_s} \partial_x \phi(t, x) \partial_v f_s(t, x, v) = S_s(t, x, v) + C_{ss}(t, x, v)$$

$$\partial_x^2 \phi(t, x) = -\frac{\rho_q(t, x)}{\epsilon_0} \quad n_s(t, x) = \int f_s(t, x, v) dv \quad \rho_q(t, x) = \sum_s q_s n_s(t, x)$$

$$C_{ss}(t, x, v) = \partial_v \left[D_v(t, x, y(x, v)) \partial_v f_s(t, x, v) + f_s(t, x, v) D_v(t, x, y(x, v)) m_s \frac{v - V_M(t, x)}{T_M(t, x)} \right]$$

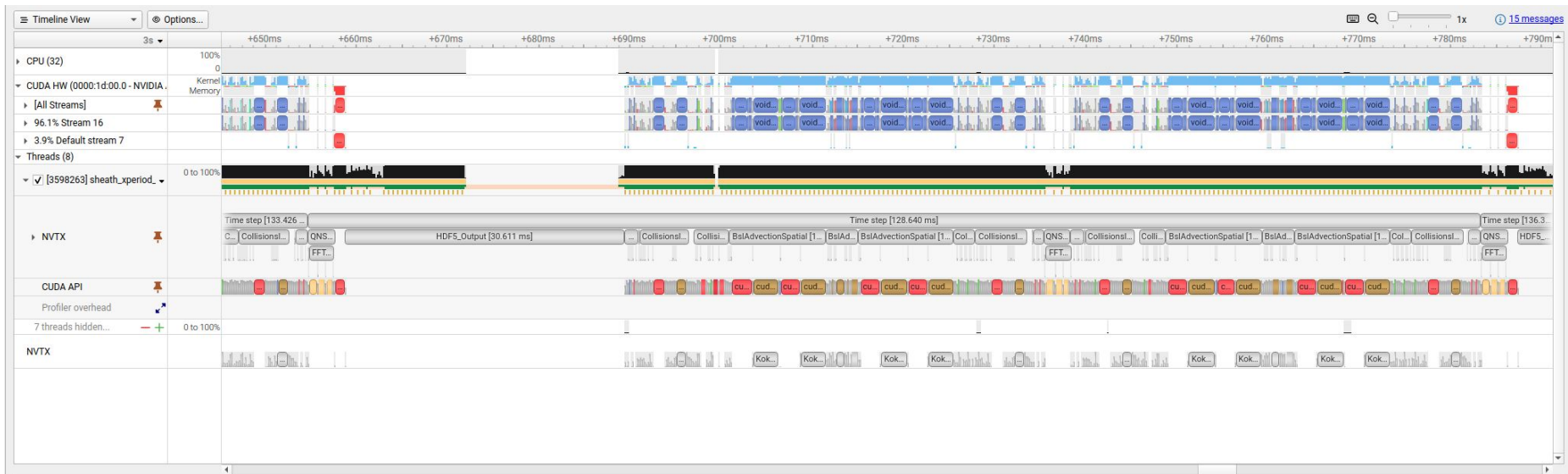
$$S_{s,w_1}(t, x, v) = -v_{s,w_1}(t, x) \mathcal{M}_{w_1}(x) [f_s(t, x, v) - g_{s,w}(n_w, T_{w_1}, v)]$$

$$S_{s,w_2}(t, x, v) = -v_{w_2} \mathcal{M}_{w_2}(x) [f_s(t, x, v) - g_{s,w}(n_s(t, x), T_{w_2}(t), v)]$$

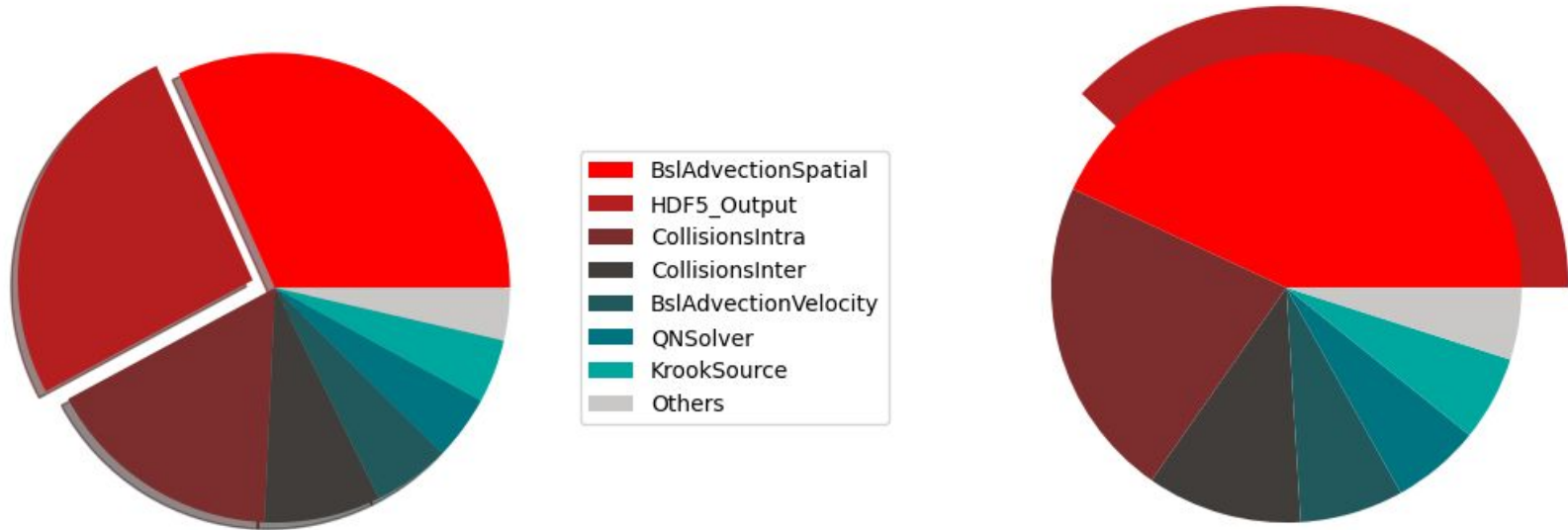
$$S_{s,k}(t, x, v) = \frac{\mathcal{M}_k(x)}{\int_0^{L_x} \mathcal{M}_k(x') dx'} \frac{s_k \sqrt{m_s}}{\sqrt{2\pi T_k}} e^{-\frac{m_s v^2}{2T_k}}$$

- See [E. Bourne et al., 2023] for numerics & [Y. Munsch et al., 2023] for physics

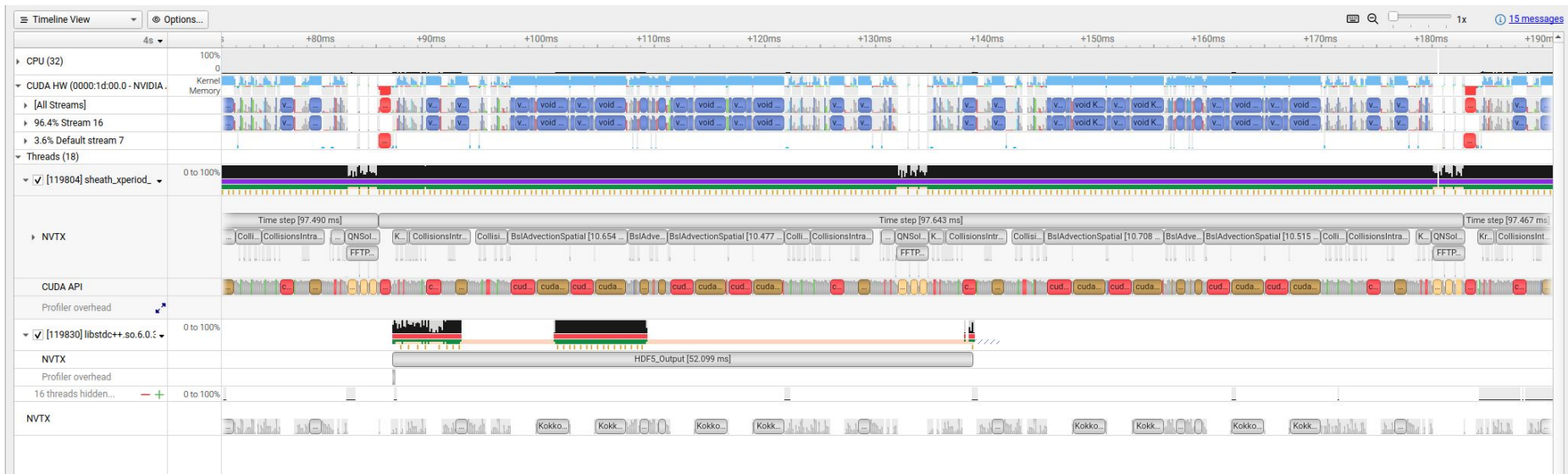
Voicexx first GPU implementation



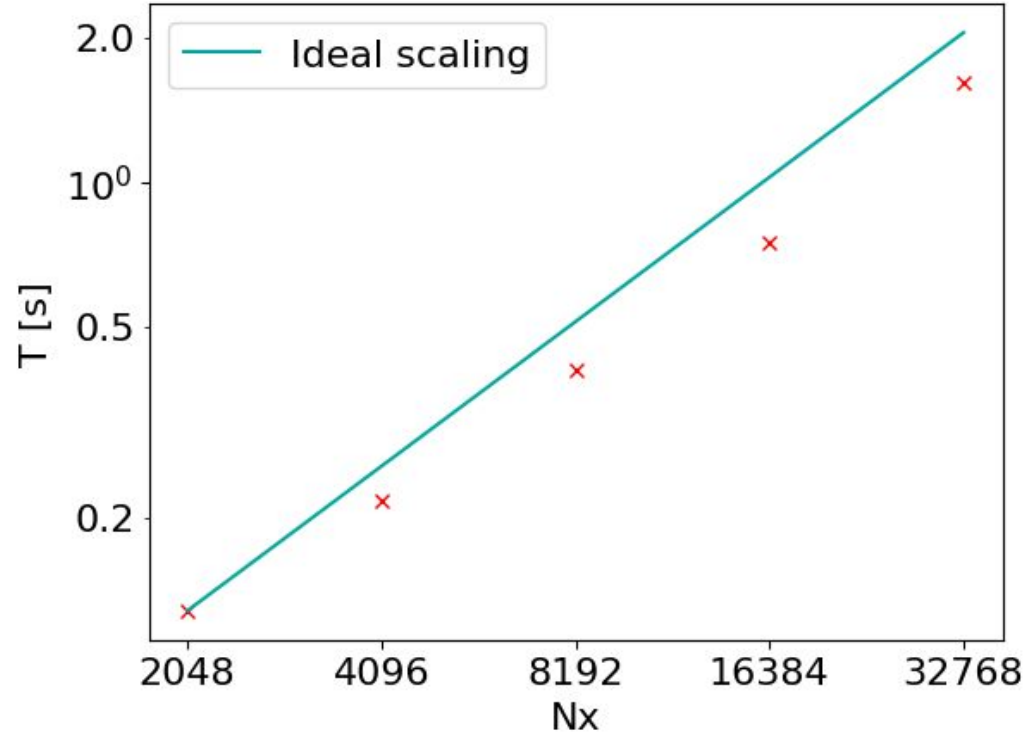
Overlapping I/O and computations



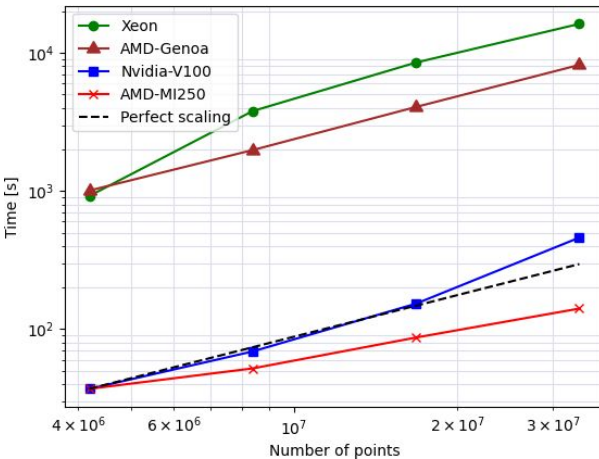
Voicexx first GPU optimisations



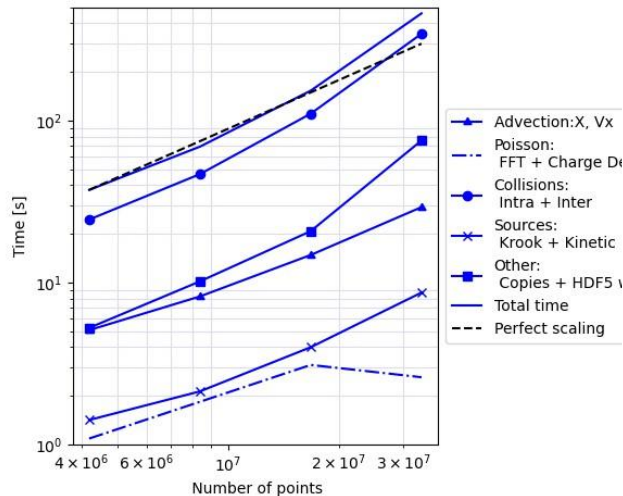
Strong Scaling on Leonardo (A100)



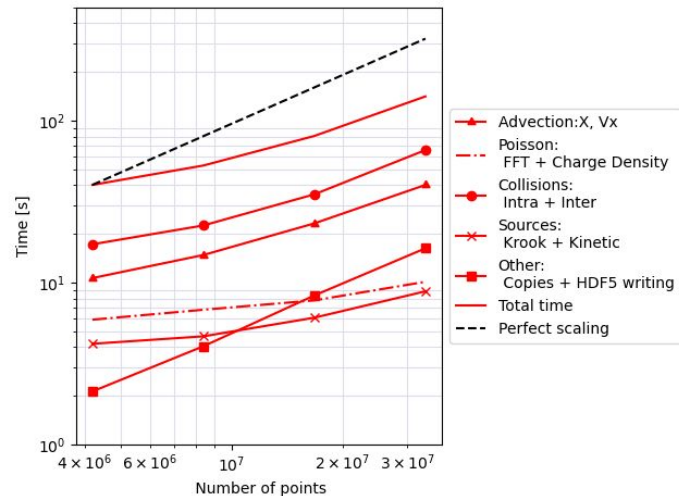
Strong Scaling comparisons



CPU versus GPU

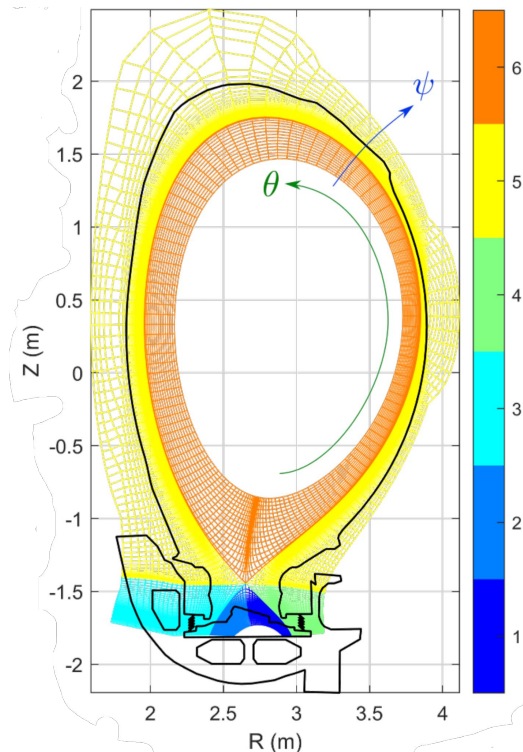


GPU: NVIDIA V100



GPU: AMD-MI250X

Gysela-X++ : Challenges



H Bufferand et al 2022 *Plasma Phys. Control. Fusion* 64 055001

Patches are required to manage the geometry

- Poisson over patches
 - SL scheme for multi-patches
 - [P. Vidal (IPP) PhD 2022-2025]*
- Advection over patches
 - 2D multi-patch Poisson solver
 - [A. Hoffmann (IPP) PhD 2023-2026]*
- MPI load balancing

Conclusions

- First results from Voicexx are very promising
 - Good scaling
 - Large number of points in X is possible
 - Portable on both Nvidia and AMD GPUs
 - Detailed comparisons with the Fortran version should be carried out
- Gysela-Axi is almost ready
 - Larger simulation will give more pertinent performance analyses
 - First simulation with wide interest for physicists
- Work on GyselaX++ is progressing well
 - Patches are non-trivial but will allow new physical studies

VOICE - Vlasov Open boundary Ion Coupling to Electrons

- Poisson solver : FEM (spline basis)
- Advection : semi-Lagrange (spline basis)
- Collision operator : FDM
- Source terms : Runge-Kutta
- Output for diagnostics

Preliminary performance (only collisions in Vpar)

- What do I measure ?

- I use a modified **CLK_begin_collisions_vpar** timer
- With and without the MPI transposition (before and after the collision computation)
 - Aka, `collypamu_general_main_routine` duration or only the computation duration
- Koliop block size: 128x32x4

- Speedup

- Run it optimally on one Genoa node (using master: c9203580)
 - Parallelization characteristics: 1 node, 32 ranks per node, 6 OMP thread/Rank.
 - time for collisions in vpar=11,8 s (no MPI transposition)
 - time for collisions in vpar=17,5 s
- Run it optimally on one MI250 node (using gysela: 8a932eb and koliop: 34e6ffc)
 - Parallelization characteristics: 1 node, 8 ranks per node, 8 OMP thread/rank, 1 GCD/rank.
 - time for collisions in vpar=3,11 s (no MPI transposition)
 - time for collisions in vpar=20,1 s
- Measured speedup: x3,79 (no MPI transposition)
- Measured speedup: x0,87
 - Significant issue with the transposition

- Energy consumption

- **TODO: I need to exclude the energy spent on the CPU part of Gysela when running the GPU simulation.**

Preliminary performance (only collisions in Vpar)

- Choose a large test case (at the node level)
 - Must fit in memory of the *smallest* node
 - Lots of CPU memory taken by the operator (in fortran, not koliop) due to the transposition stuff
 - `fdistribu5d_t%values -> f5d_s_seqx4x5`
 - Leads to a reduced test case.
 - Based on `bench/forGPU/MPI8_Nthread32_512x512x32x64x1_TKE`
 - `NPROC_MU=4/Nmu=3`
 - `Nr=255`
 - `Ntheta=256`
 - `Nphi=32`
 - `Nvpar=127`
 - `Nbiter=8` (4 collision iterations)
 - No diag except Ni
 - Note that a low Nmu count is disadvantageous to the operator (Nmu x Nvpar loops in most kernels), *real* workloads may perform better.