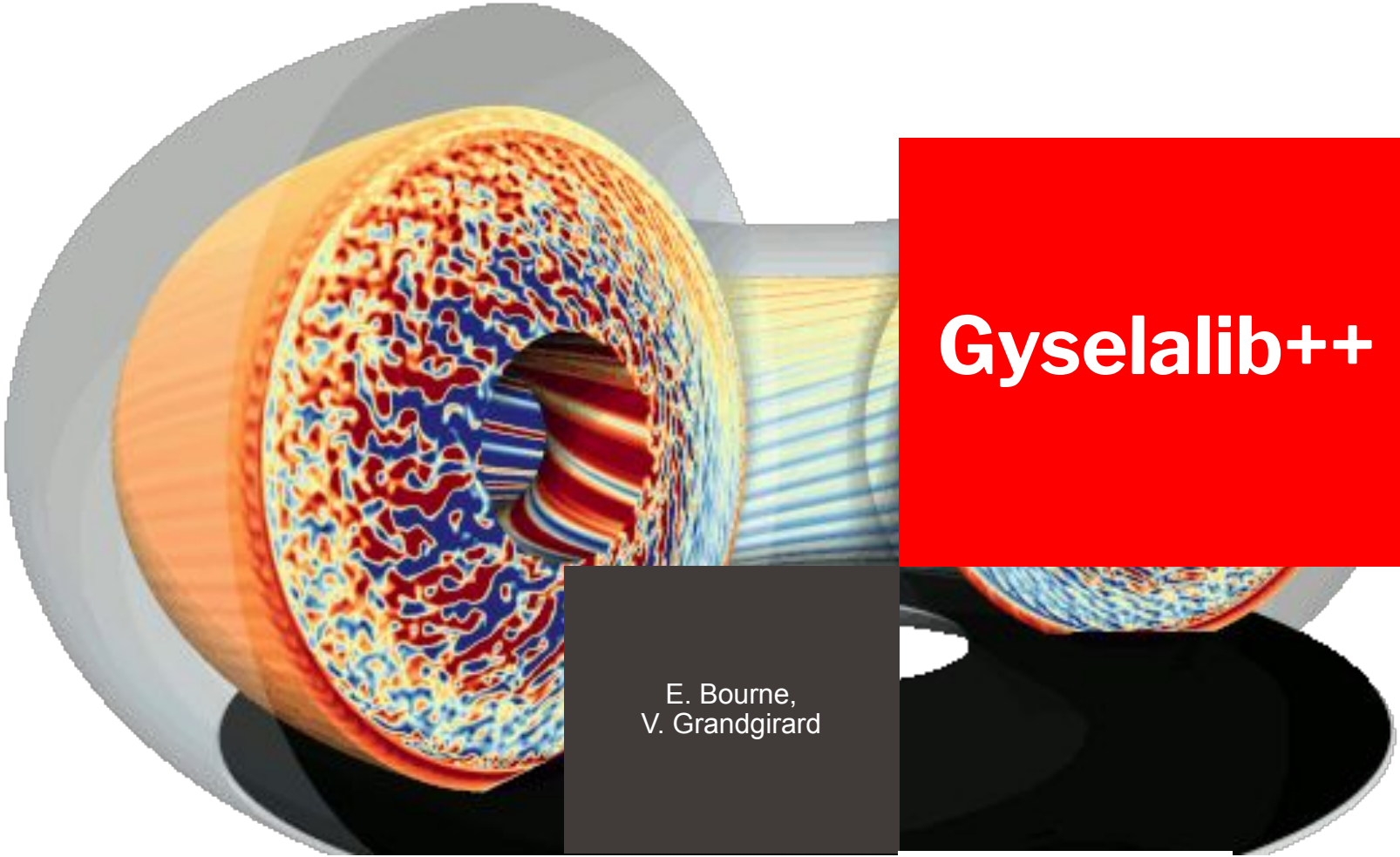
EPFL



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)

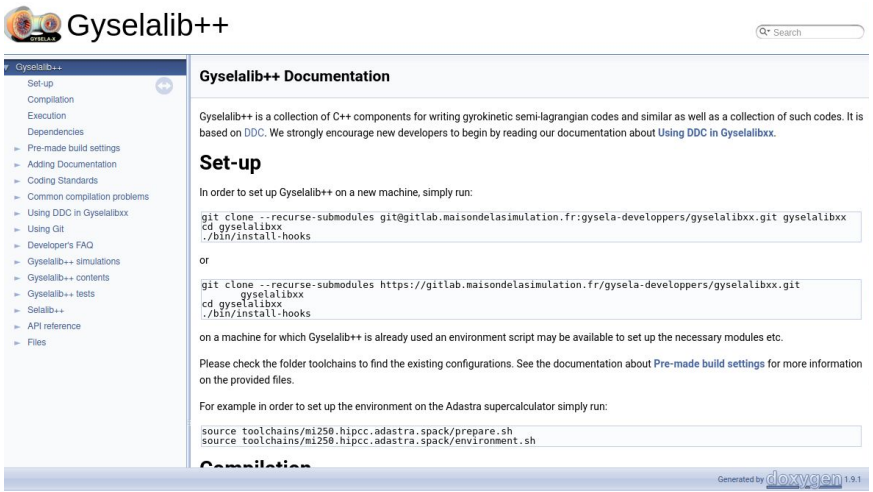
Gyselalib++ (C++) so far…

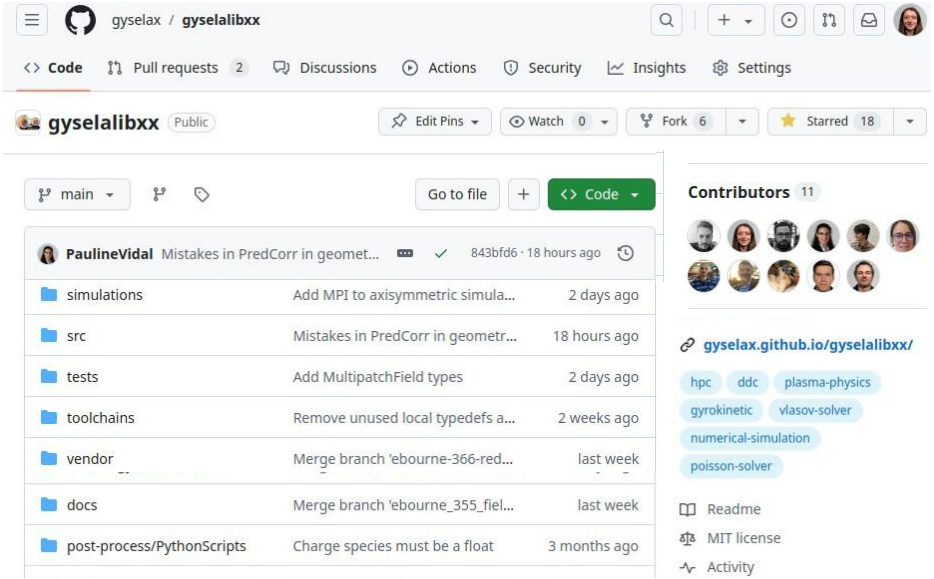
- 2D/4D
- MPI/Kokkos

Gyselalibxx

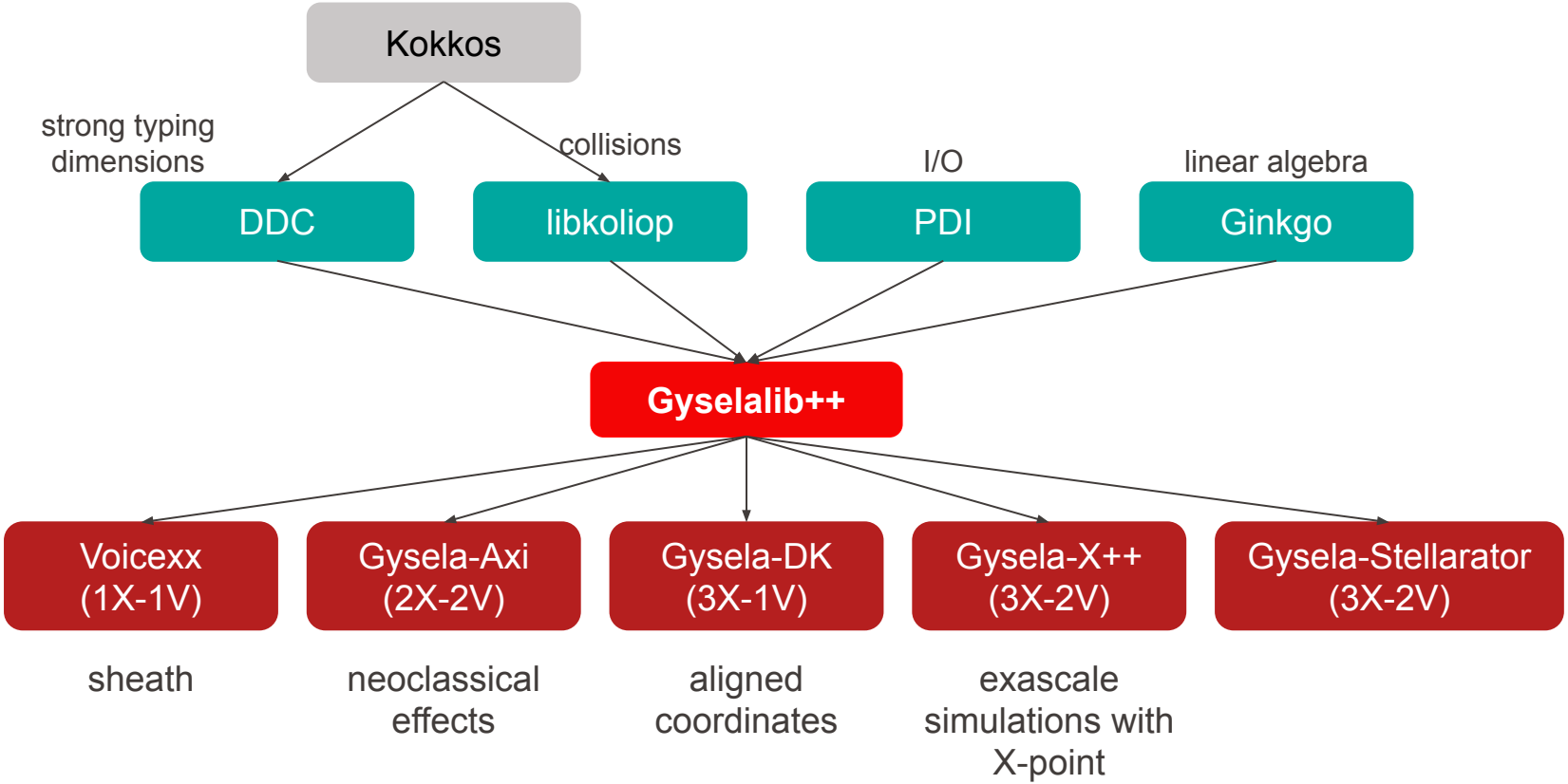
- CI:
  - unit tests
  - code conventions
  - best practices
  - common bug detection
  - forced documentation

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

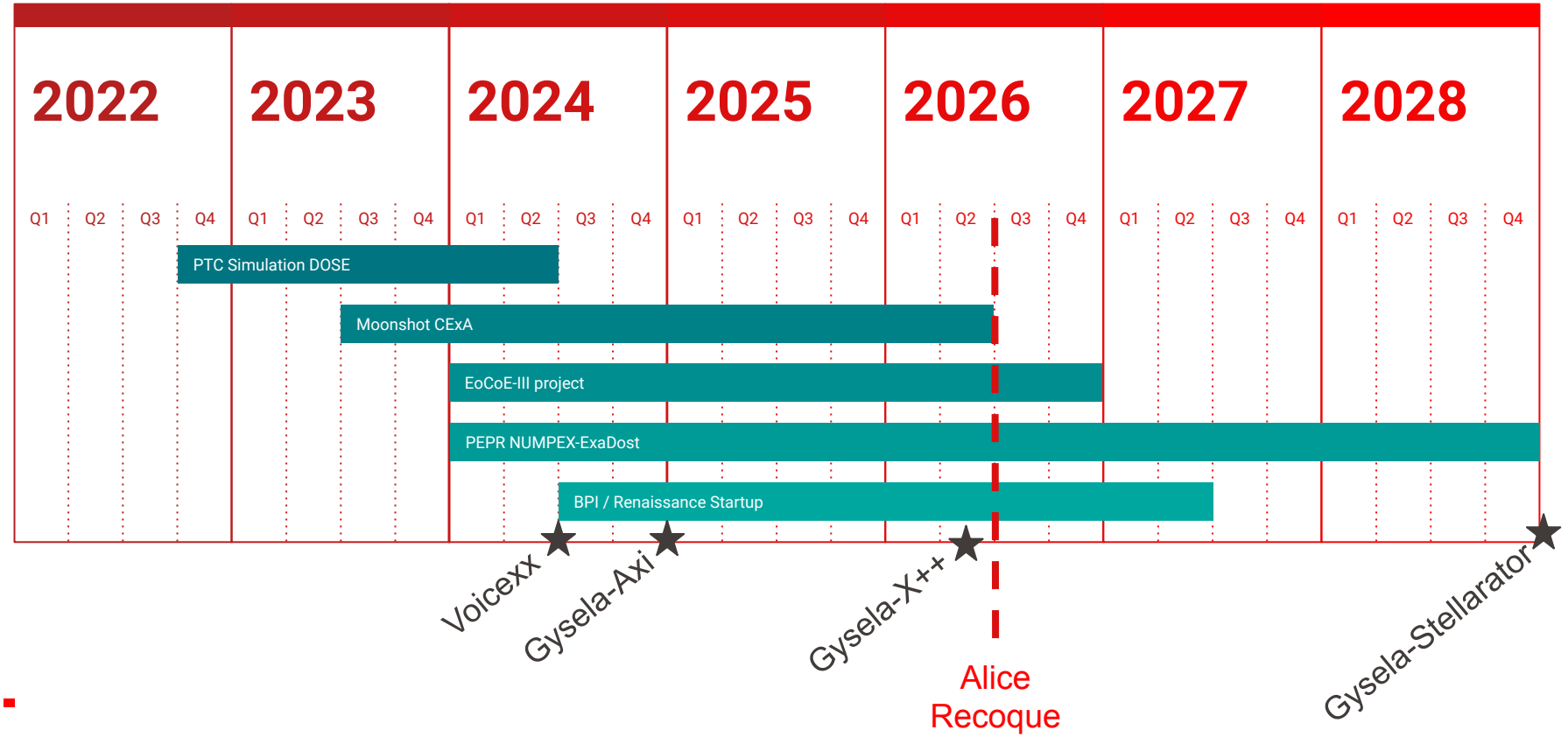




Organisation



Roadmap


2022
2023
2024
2025
2026
2027
2028
Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
PTC Simulation DOSE
Moonshot CExA
EoCoE-III project
PEPR NUMPEX-ExaDost
BPI / Renaissance Startup
Voicexx
Gysela-Axi
Gysela-X++
Alice Recoque
Gysela-Stellarator

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 $v_{//}$
    - Non-uniform 1D and 2D splines including special treatment of the O-point
  - 2D poisson solver in (r , θ)
  - 2D collision operator ($v_{//}$, 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

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

- 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

$$-\partial_x^2\phi(t,x) = \frac{1}{\varepsilon_0}\sum_x Z_s\int f_s(t,x,v)dv$$

- Finite Elements

[https://github.com/selalib/selalib]

VOICE - Vlasov Open boundary Ion Coupling to Electrons

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

$$\partial_x^2 \phi(t,x) = -\frac{\rho_q(t,x)}{\varepsilon_0} \qquad n_s(t,x) = \int f_s(t,x,v)dv \qquad \rho_q(t,x) = \sum_s q_s n_s(t,x)$$

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

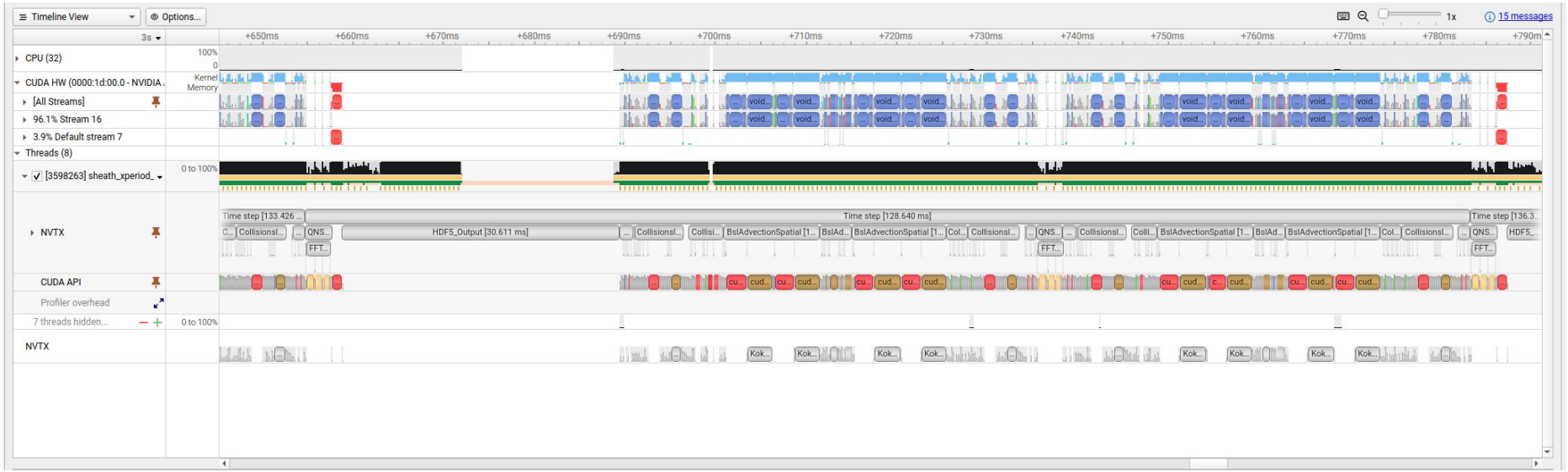
$$S_{s,w_1}(t,x,v) = -\nu_{s,w_1}(t,x)\mathcal{M}_{w_1}(x)[f_s(t,x,v) - g_{s,w}(n_w, T_{w_1}, v)]$$

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

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

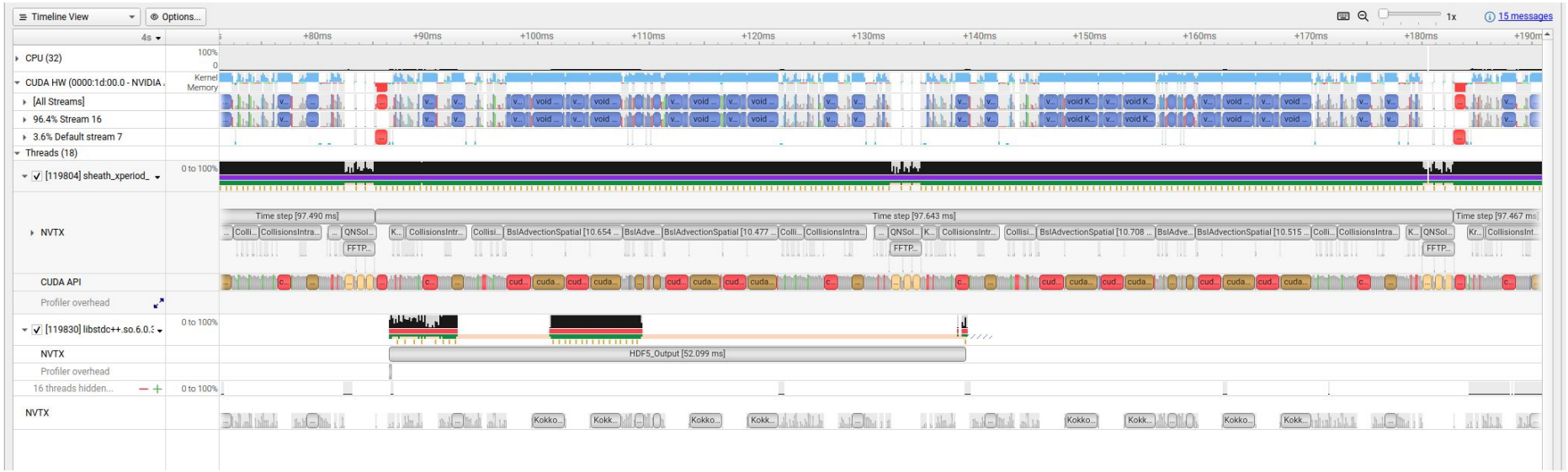
- See *[E. Bourne et al., 2023]* for numerics & *[Y. Munschy et al., 2023]* for physics

Voicexx first GPU implementation

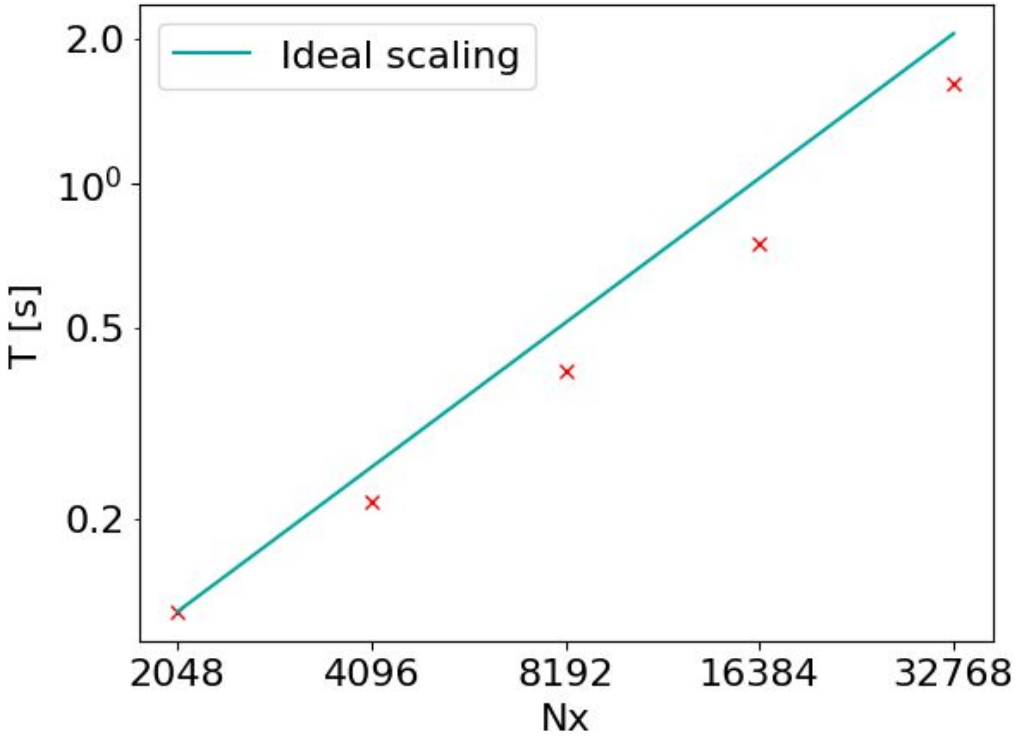


Overlapping I/O and computations

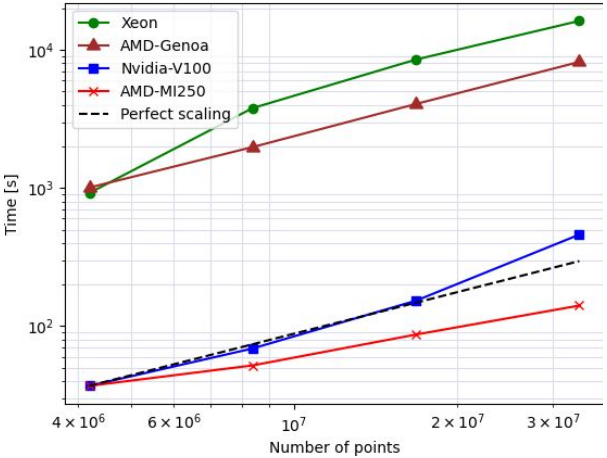
Voicexx first GPU optimisations



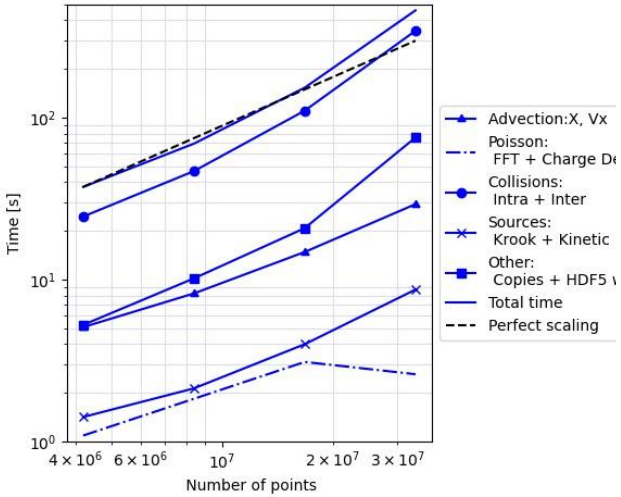
Strong Scaling on Leonardo (A100)



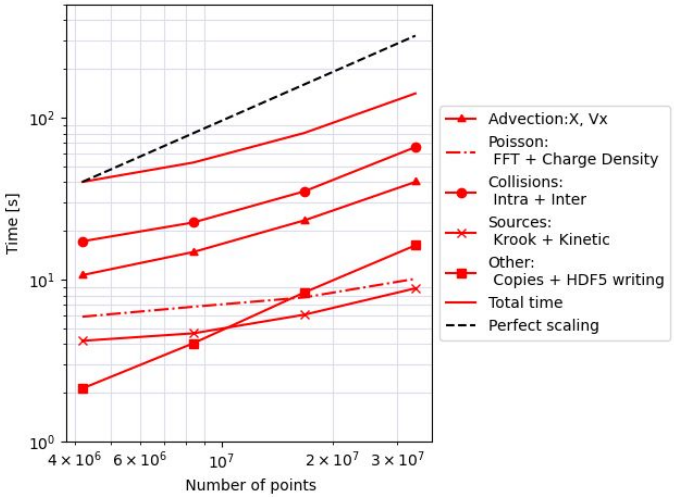
Strong Scaling comparisons



CPU versus GPU

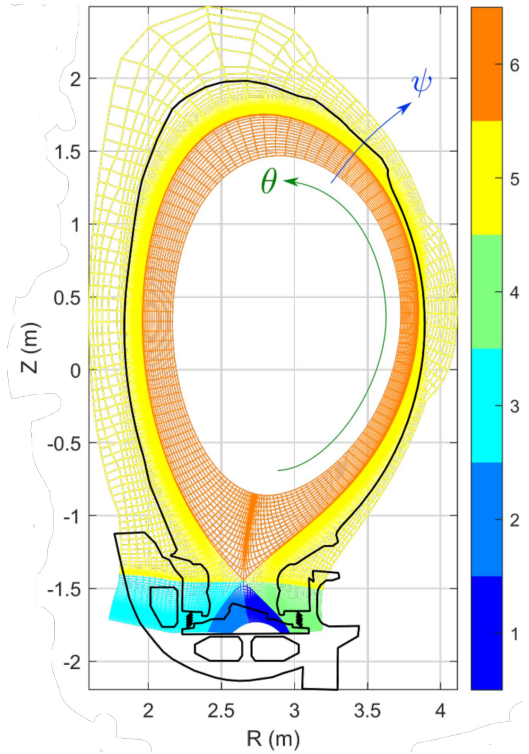


GPU: NVIDIA V100



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, collvparmu_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.