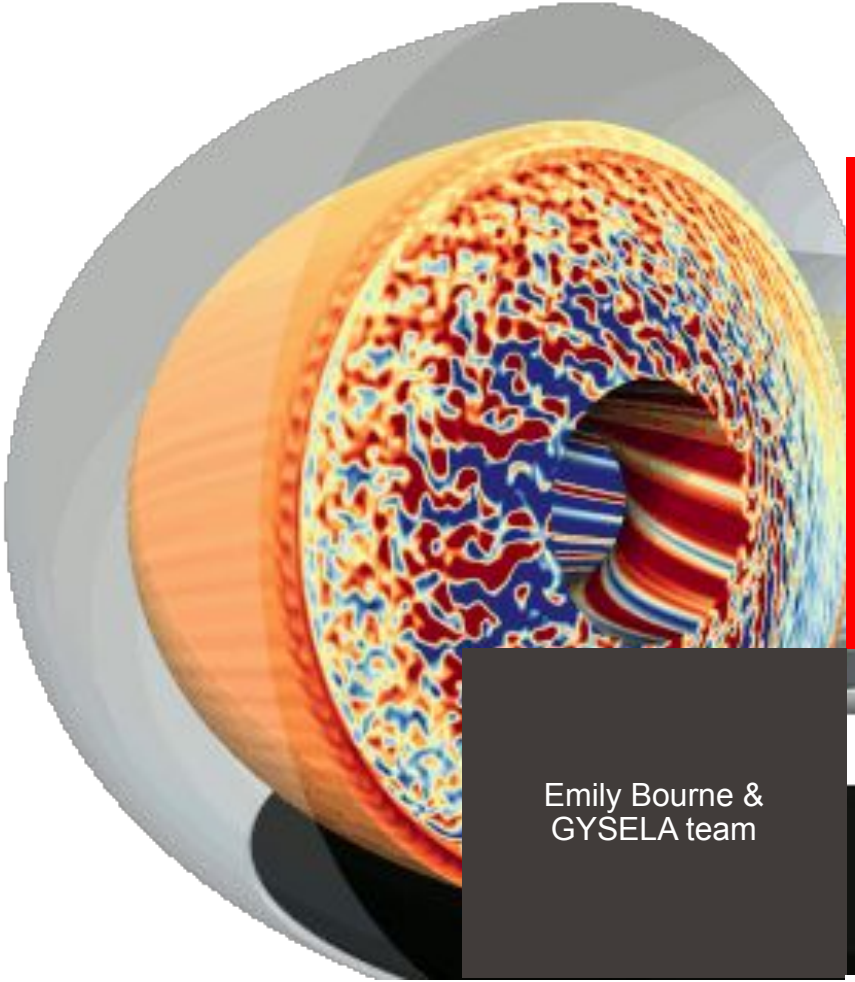
cea irfm

EPFL SCITAS
Scientific IT and Application Support

INES

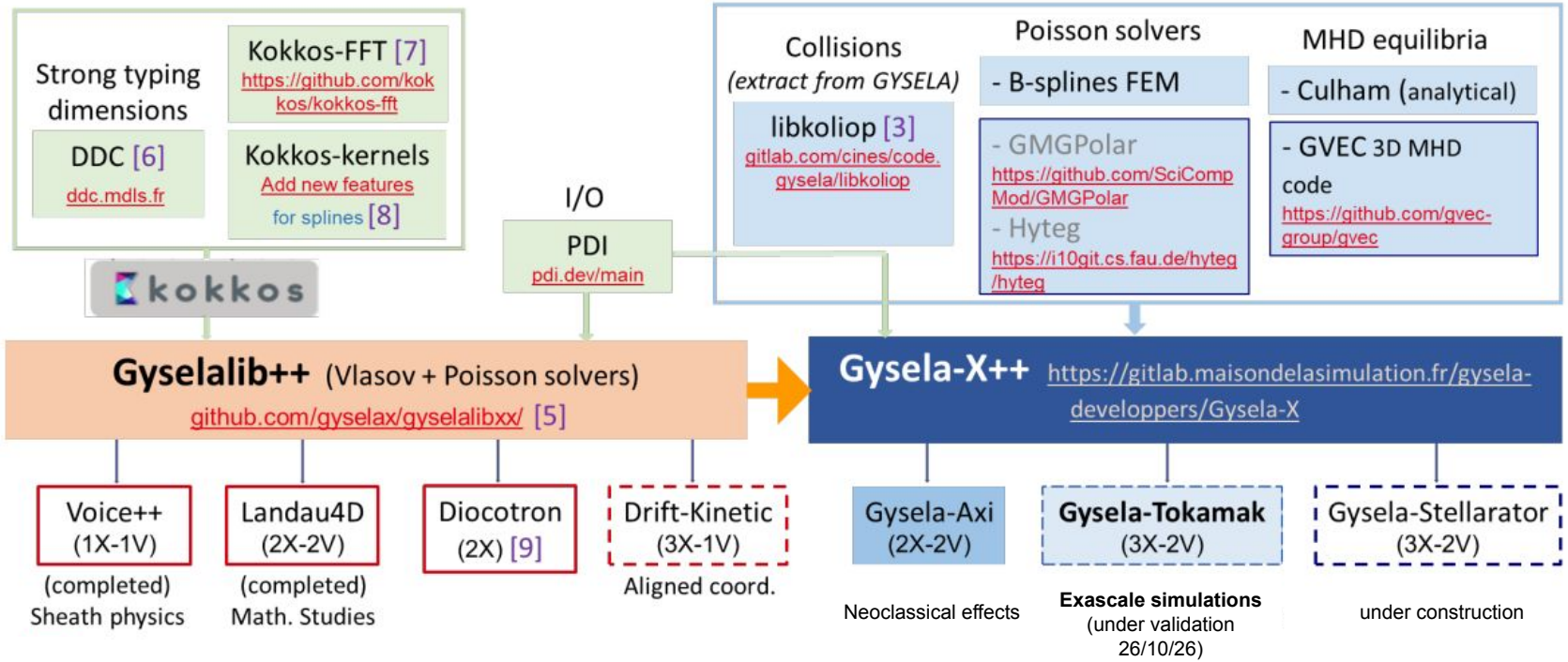
MAX PLANCK INSTITUTE FOR PLASMA PHYSICS



GyselaX++

Emily Bourne &
GYSELA team

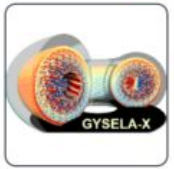
Gyselalib++ → GyselaX++



An open source ecosystem of mini-applications based on Gyselalib++ to prepare exascale

gyselax

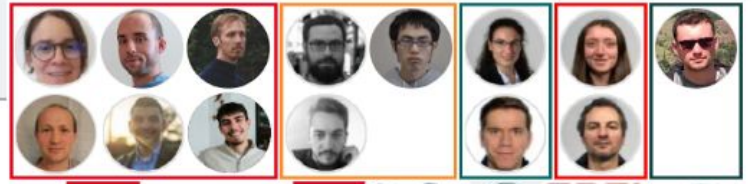
Overview | Repositories 13 | Projects 1 | Packages | Teams 4 | People 18 | Insights



GyselaX

github.com/gyselax/gyselalibxx/

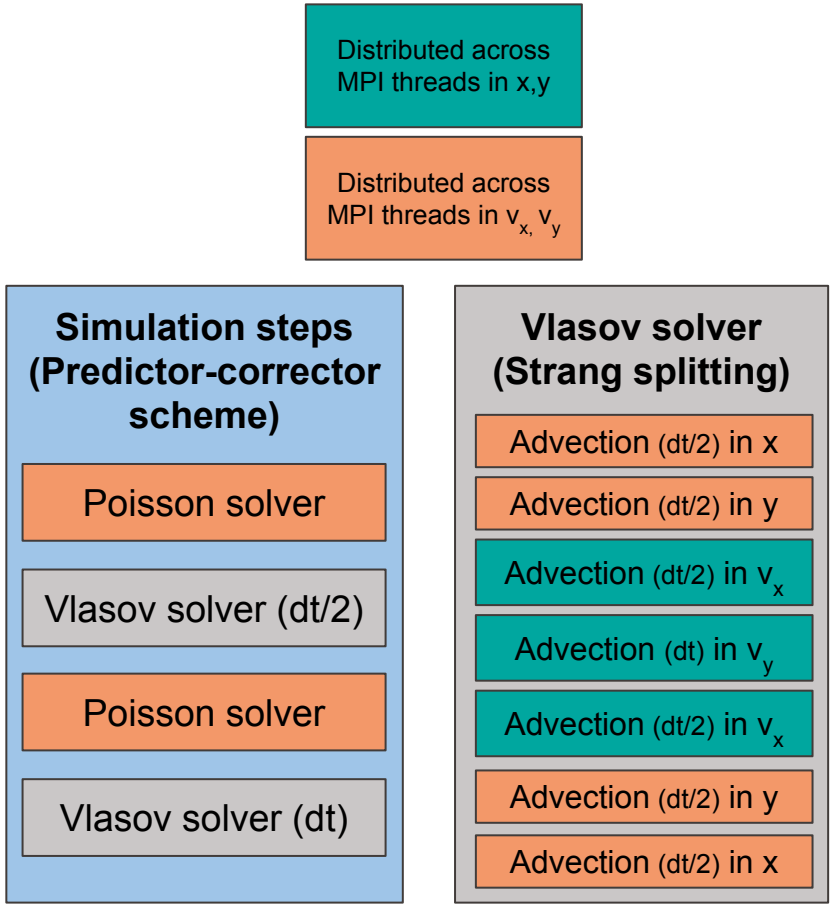
gyselax.github.io/gyselalibxx/



gyselalibxx	Gyselalib++ is a collection of C++ components for writing gyrokinetic semi-lagrangian codes and similar	*[E. Bourne, EPFL]*	C++	19	55
gysela-mini-app_poisson	Mini-application to benchmark Poisson solvers	*[E. Bourne, EPFL]*	C++	1	2
gysela-collisions	Collision operator	*[E. Malaboeuf, CINES]*	C++	0	1
gysela-mini-app_insitu	Mini-application for in-situ and in-transit tests	*[Ph. Krah, CEA-IRFM]*	Python	0	0
gysela-mini-app_io	Mini-application for IO testing and performance benchmarking	*[A. DeVora, CEA-IRFM]*	C++	8	6
gyselax.github.io	GyselaX website: https://gyselax.github.io/	*[GYSELA team, IRFM]*	TeX	0	0
spack	Provides extra Spack recipes for the Gyselalib++ library	*[T. Padioleau, CEA-MdlS]*		2	0

2X-2V Landau damping - algorithm

- The Landau damping scheme treats different operators in different memory layout patterns
- A transposition occurs at each change of MPI layout
- Poisson solver : FFT
- Advection : Semi-Lagrangian advection
  - For each grid point estimate where a particle at that point would be found at t-dt
  - Use an interpolation scheme to estimate the distribution function at that point



2X-2V Landau Damping:
One application multiple possible hardware

Interpolation	Precision	Backend	Hardware
Lagrange	double	CUDA	Nvidia GPU with FP64
Lagrange	float	CUDA	Nvidia GPU without FP64
Lagrange	double	OpenMP	CPU
Lagrange	double	HIP	AMD GPU
Spline	double	CUDA	Nvidia GPU with FP64

Hardware description

Kuma (GPUs)

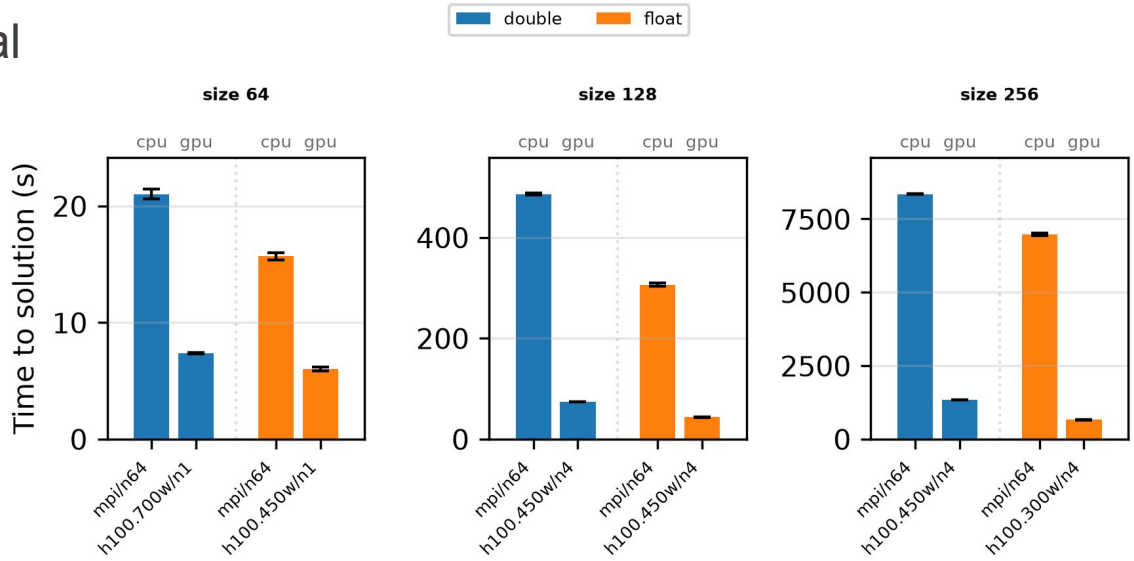
- 4 Nvidia H100
  - VRAM: 96GB
  - TDP: 700W
- 2 AMD EPYC 9334
  - cores: 32
  - RAM: 192GB
  - TDP: 240W
- “Total” TDP
  - 3280 W

Jed (CPU only)

- 2 Intel Xeon Platinum 8360Y
  - cores: 36
  - RAM: 256GB
  - TDP: 250W
- “Total” TDP
  - 500W
  - **~6.6x less**

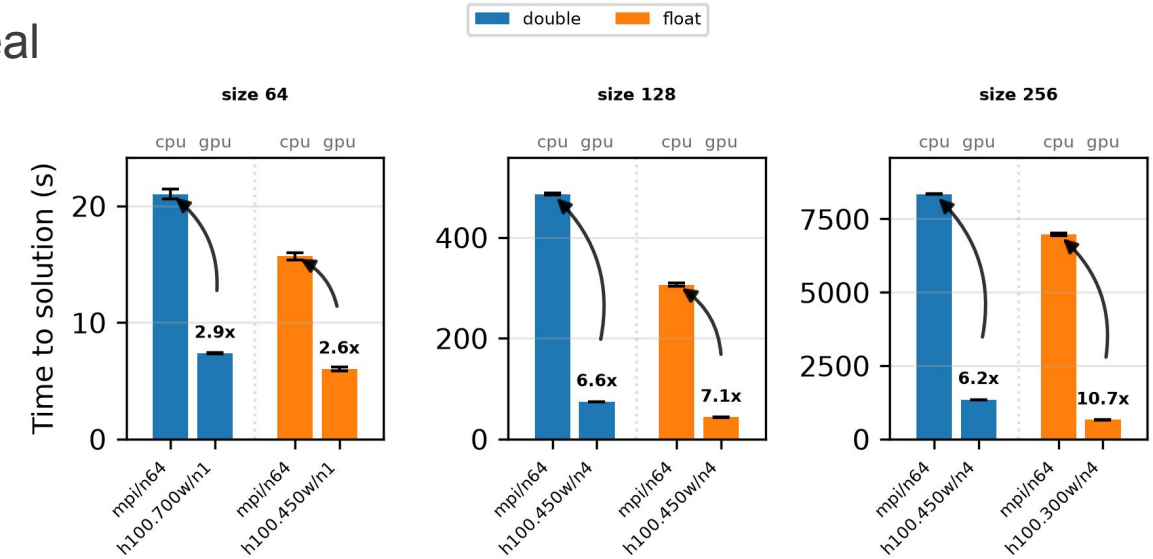
Time to solution

- Most cases GPU less than 6.6x Faster
- GPU scaling not ideal
  - might be a mpi scaling issue
- 300w marginally better than 700w



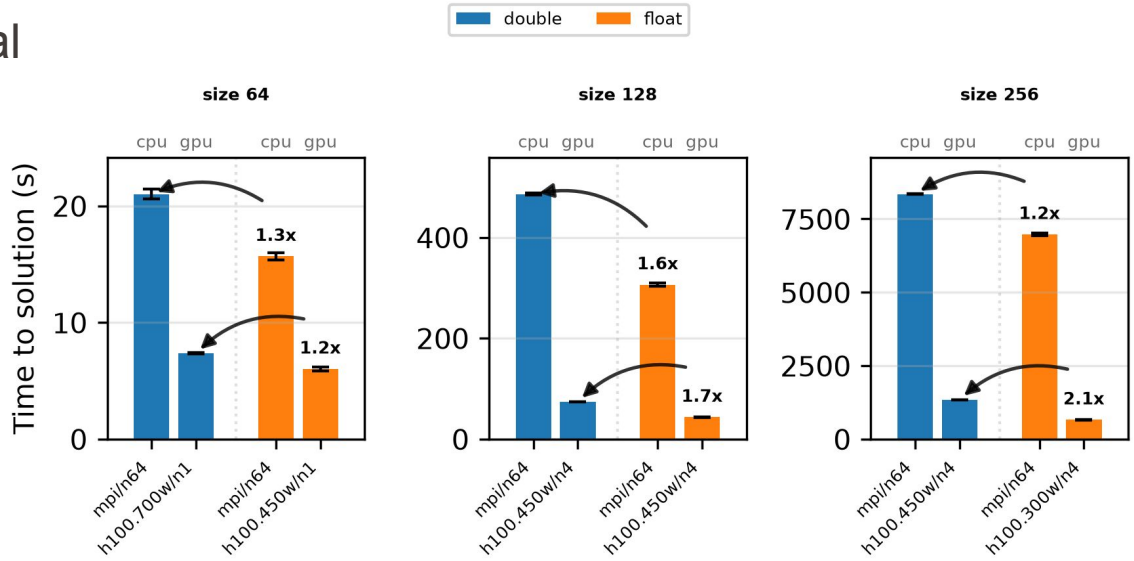
Time to solution

- Most cases GPU less than 6.6x Faster
- GPU scaling not ideal
  - might be a mpi scaling issue
- 300w marginally better than 700w



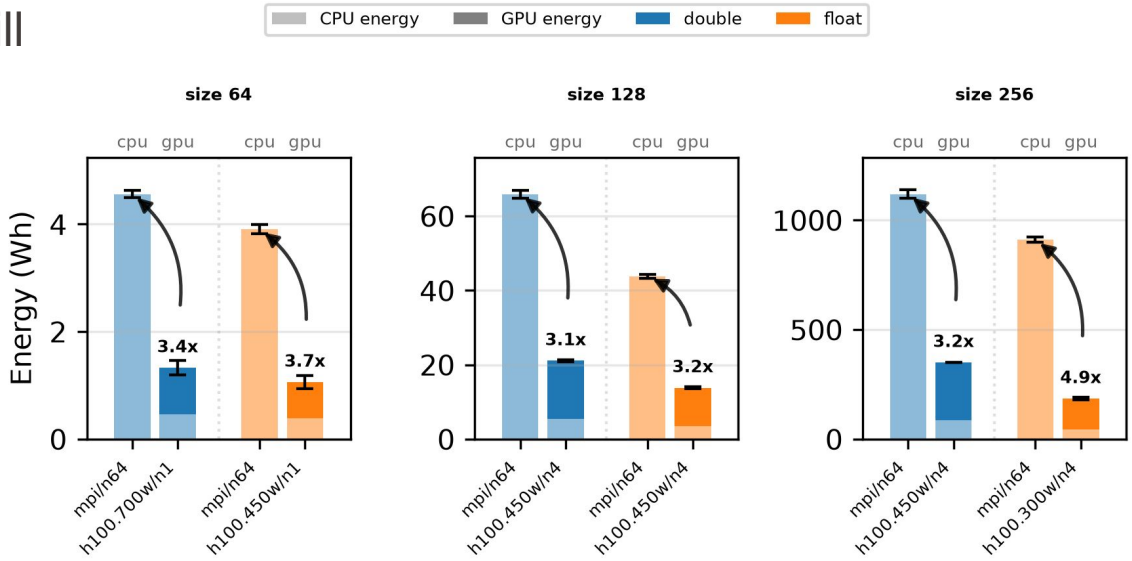
Time to solution

- Most cases GPU less than 6.6x Faster
- GPU scaling not ideal
  - might be a mpi scaling issue
- 300w marginally better than 700w
- Float 1.5 to 2x faster than double



Energy to solution

- GPU still better but ratio divided by 2
- Any idle hardware still accounted:
  ~10% cores



EPFL SCITAS Scientific IT and Application Support

cea irfm

Maison de la Simulation

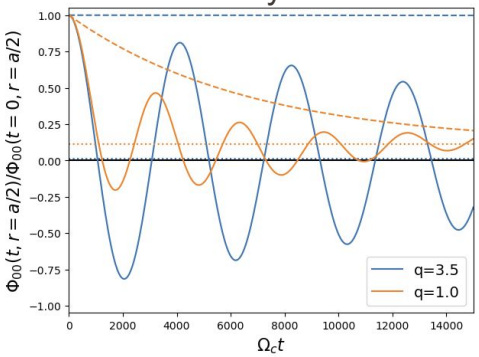
MAX-PLANCK-INSTITUT FÜR PLASMAPHYSIK

CINES

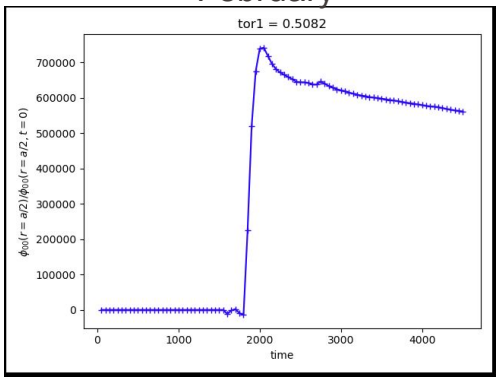
4D axi-symmetric Tokamak simulations

Debugging a 4D simulation

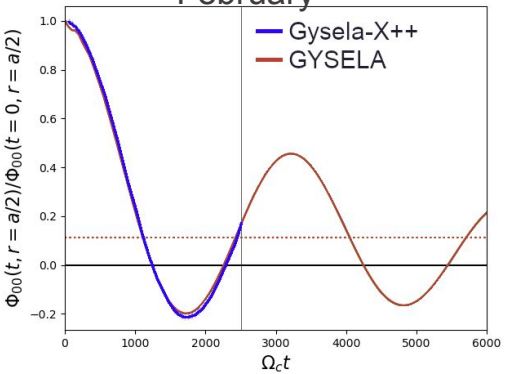
January



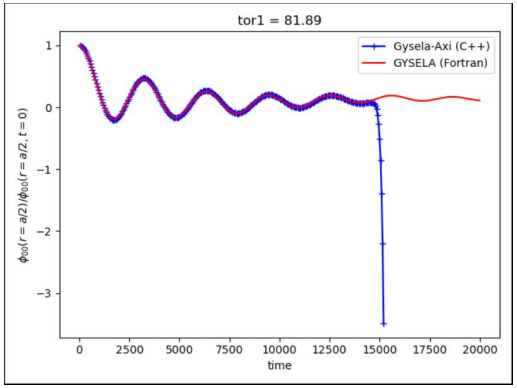
February



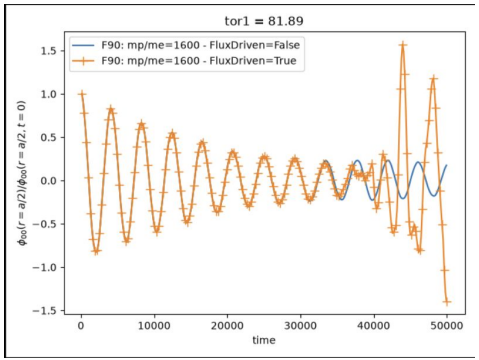
February



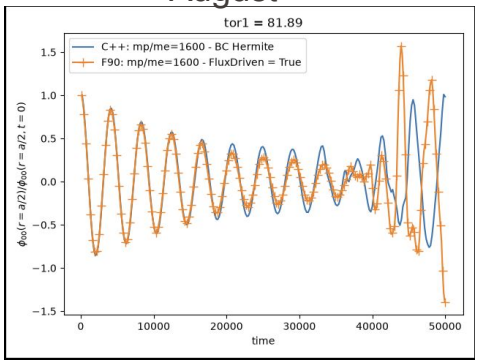
February



July



August



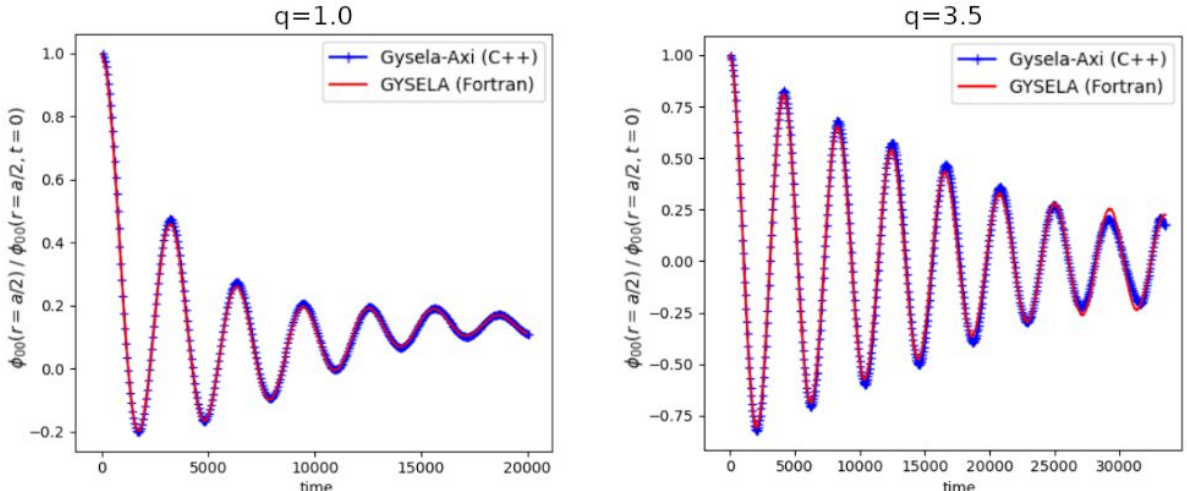
We realised that the issue we had at the O-point in C++ was also present in Fortran code if we took Flux_Driven = .true. instead .false. condition

Debugging a 4D simulation

- Bad initialisation of theta grid points
- Choice of boundary conditions at r=r_max
  - Greville vs Hermite
  - derivative approximation for fdistribu vs other fields (e.g. advection coefficients)
- Choice of boundary conditions at r=r_min
  - Greville vs Hermite
  - Centred vs decentred derivative approximation
- Choice of extrapolation conditions at r=r_min
  - Constant vs cubic

Current progress — Rosenbluth–Hinton tests

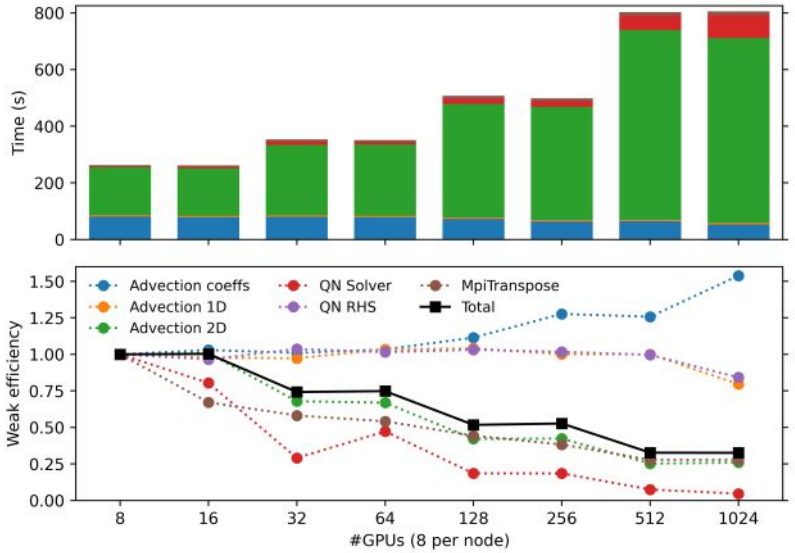
- Gysela-Axi is the axisymmetric (2X+2V) code showcasing the general code architecture and most of the physics functionalities required for Gysela-X++
  - Used for physics result validation, benchmarking, and as test-bed for functionalities modules (multi-patch & X-point) and external collaborations (Poisson solvers, I/O framework, AI workflows)
- Rosenbluth–Hinton test (GAM damping) with fully kinetic electrons as a first target
  - Physics cross-validation VS GYSELA
  - Realistic test case for performance optimisation and identification of bottlenecks



Current progress — Performance benchmarks

Weak scaling benchmark performed on Adastra MI250X based on an RH case

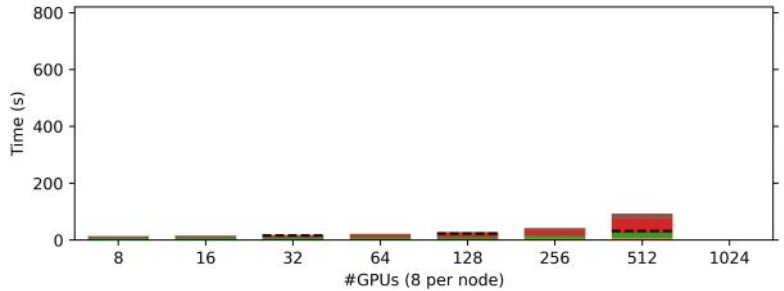
April 2026



Mesh $256 \times 512 \times 128 \times 64 \rightarrow 1024 \times 1024 \times 512 \times 256$

[E. Boella *et al*, 23rd ACM Int. Conf. on Comput. Frontiers]

September 2026



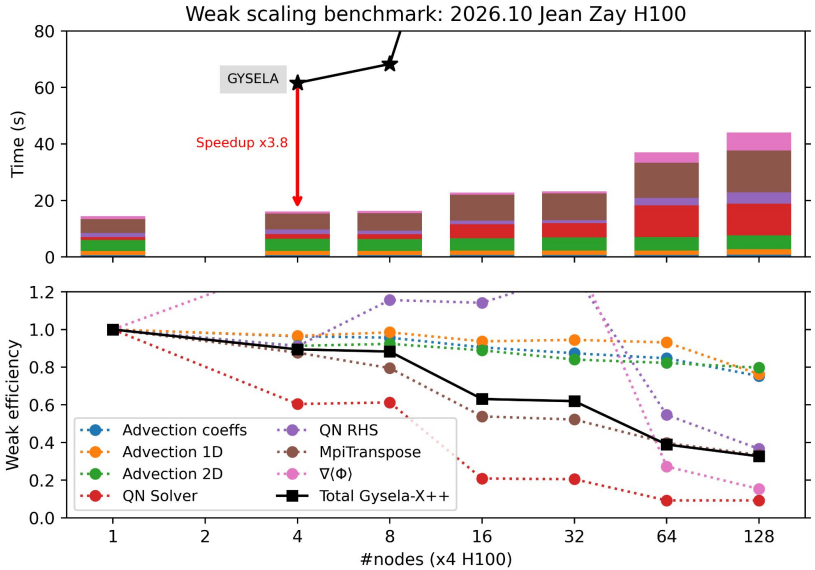
In 6 months, the total run time reduced by > 90%

- Reduction of overall memory footprint
- MpiTranspose = fewer transpositions
- Advection 2D = removed conversion to Cartesian coordinates
- Advection coefficients = precomputing the metric

Current progress — Performance benchmarks

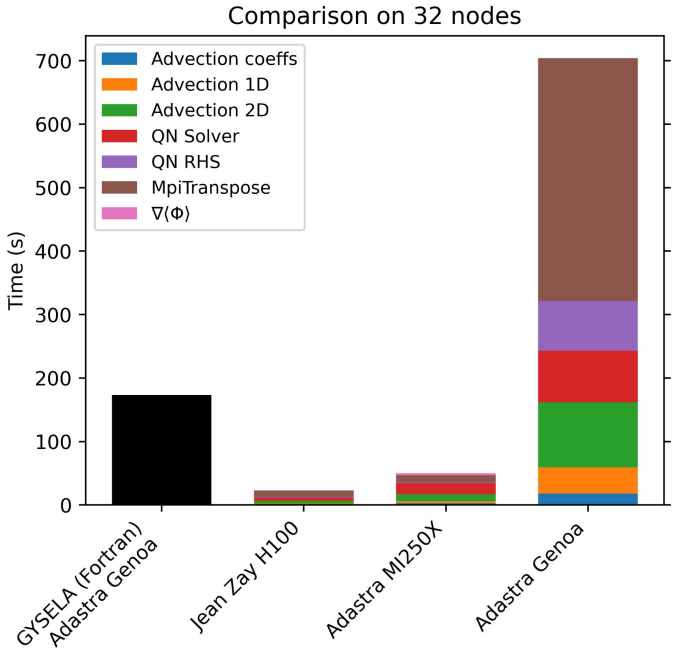
Performance optimisation continues in the background, as the code expands towards new functionalities

- Gyroaveraging operator ⟨·⟩
  - Increased dimensionality in some operators (dep. in species and μ) and new communications
- Re-integration of the collision operator, addition of a Krook operator for immersed BC
  - New bottlenecks / constraints?
- Addition of the 5th dimension ϕ throughout the code...



- Advection 2D = implementation of memory pools in Kokkos to reduce alloc/deallocs
- MpiTranspose = decomposition in μ first, then over sub-communicators
- QN Solver = add MPI decomposition
  - scaling over ϕ should reduce % of total
- Continued work on memory footprint to fit more data in each GPU

Current progress - Cross machine comparisons



Conclusions

- Gyselalib++ provides a flexible testbed for comparing methods
- 4D simulation gives physical results
- 4D simulation is being optimised
  - >90% speedup after first improvements
- 5D simulation is newly constructed
- 4D already shows several avenues for improvements

What was our initial question

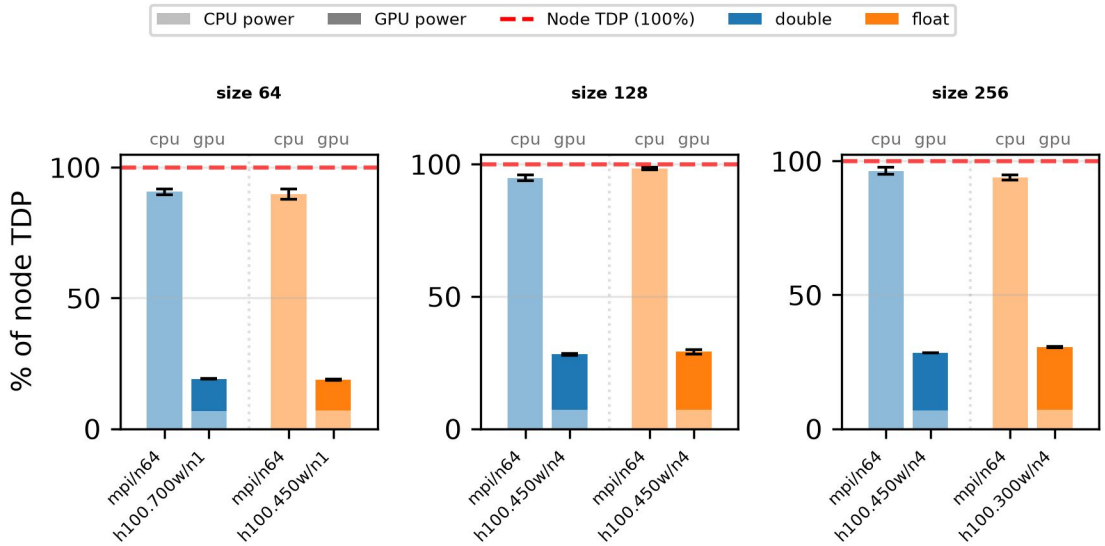
- New GPUs get higher Thermal Design Powers (TDP) but not as much more flop/J
- Simulation where ported on GPUs based on “older” machines
- Measure usually considered for optimization: Time To Solution
- Does better TTS also mean better energy to solution / environmental impact ?
- Need for a production code that works on all architectures
  - Gyselalib++

Measure methodology

- Hardware counters (1Hz) - Only “users” accessible ones
  - CPU: Running Average Power Limit (RAPL)
  - GPU: nvidia-smi
- Always measured one “full” node
- What type of run:
  - precisions: double, float
  - grid sizes: 64^4, 128^4, 256^4
  - CPU: MPI or OpenMP or Hybrid
  - GPU: MPI for different power capping
- For the results only the fastest for each configuration was kept

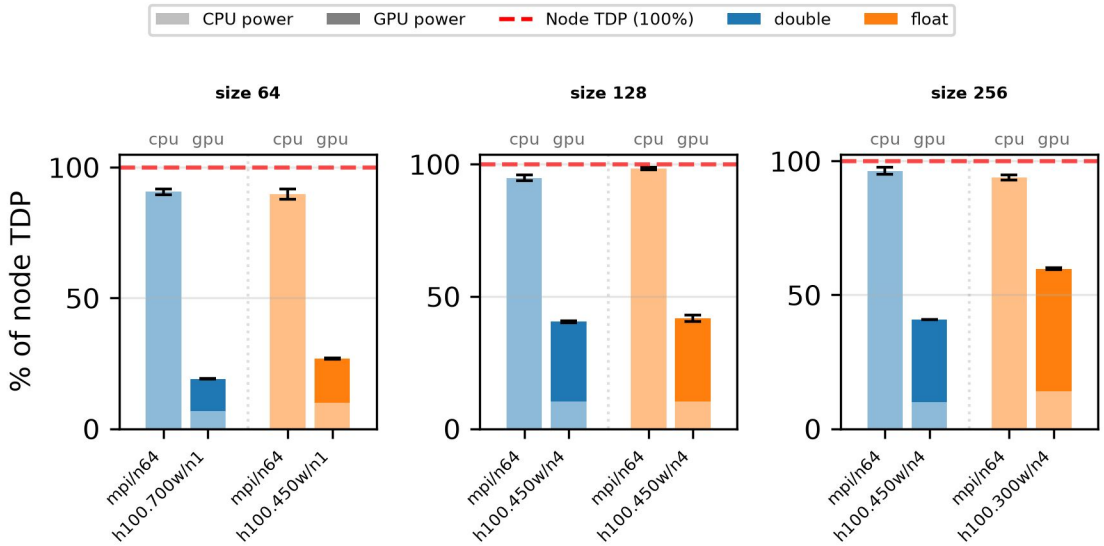
Percentage of the TDP

- Better usage of the CPUs ?
- Different power profiles on CPU vs GPU



Percentage of the TDP

- Better usage of the CPUs ?
- Different power usage profiles CPU vs GPU
- Should rescale TDP to account for capping

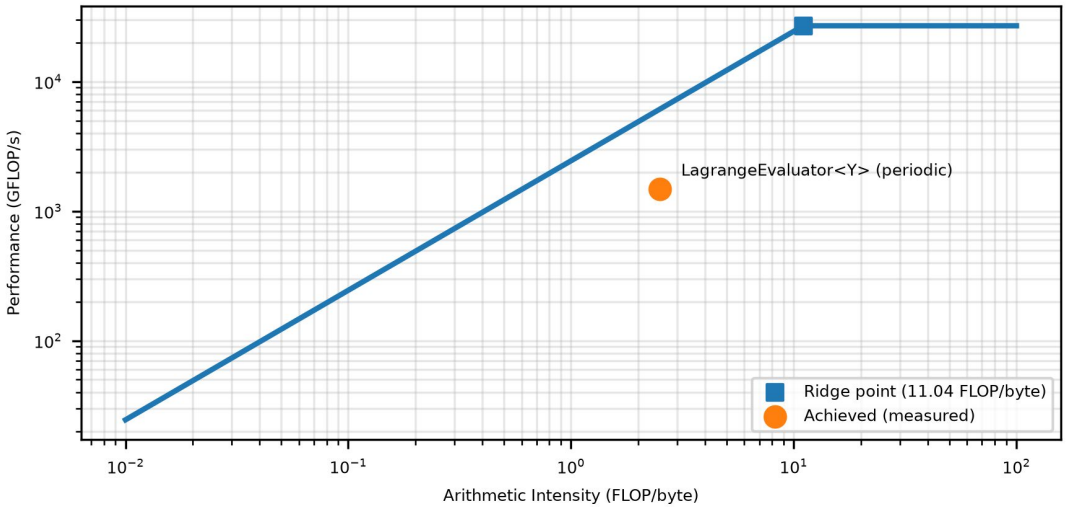


“Real” max power usage

- HPL run on 1 full GPU node
  - peak power: 3082 W
  - avg power: 2847 W
  - 87% TDP
  - compute bound

- Kernel for Advection in Y

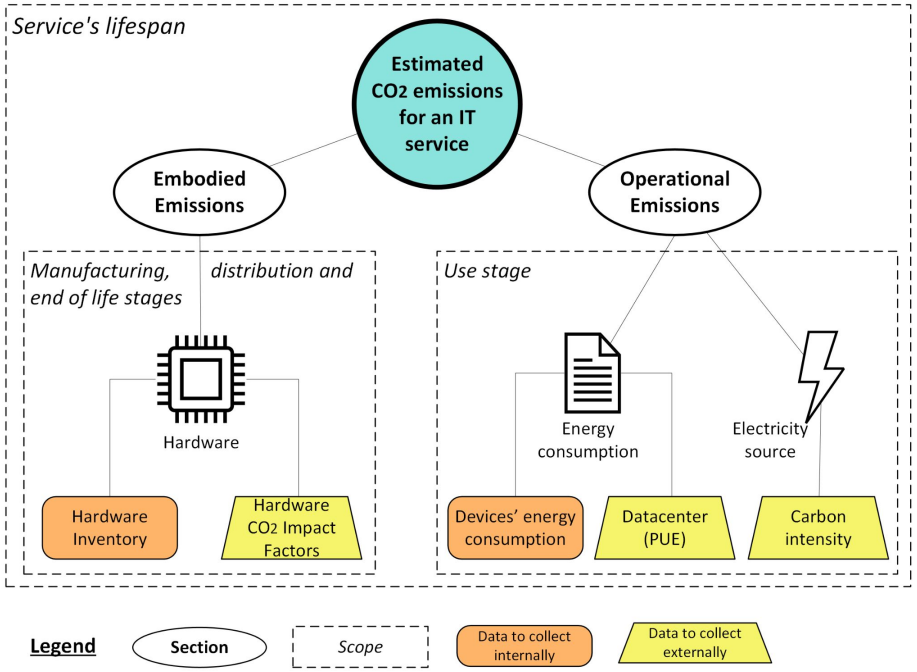
- Memory bound (AI: 2.5)



Methodology for IT Services environmental Impact assessment (MITSI)

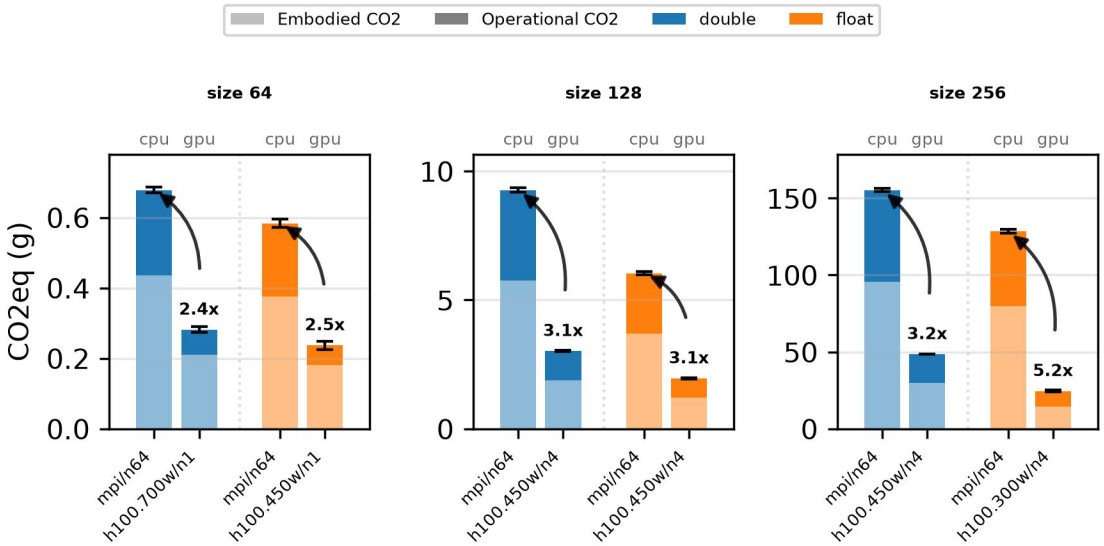
- Embodied emissions
  - What is accounted
    - CPUs
    - GPUs
    - RAM
    - PSU
    - Storage
    - Network
  - Kuma
    - 387 tCO_2eq (84 nodes)
    - 88 $gCO_2eq/nodeh$ (6 years)
  - Jed
    - 906 tCO_2eq (419 nodes)
    - 41 $gCO_2eq/nodeh$ (6 years)

- Usage emissions
  - Average power usage over a representative time period
    - Real power usage for Gysela
  - PUE
    - 1.3
  - Carbon Intensity (CI)
    - 41 gCO_2eq/kWh (2026)



Carbon impact comparaisons

- Mainly embodied is contributing
  - up to ⅔
  - scales with TTS
- Operational worst on CPU
  - scales with energy to solution



ECCOMAS Conclusion

- GPUs are still better, but margin are not so big
  - Faster enough to compensate:
    - Energy draw
    - Embodied carbon

- Do the same tests different hardware
  - Newer CPUs
  - AMD GPUs
  - Nvidia B200, TDP: 1000W, FP64: 37 TFlop/s, 8 GPU/node

- Better profiling: not limited to CPU + GPU, full nodes

- Test CPU dynamic frequency scaling