27–28 Oct. 2026

HYBRID STELLA

A STEP CLOSER TO MULTISCALE GYROKINETIC SIMULATIONS ON STELLARATORS

Valentin Seitz valentin.seitz@bsc.es
Joan Vinyals Ylla-Català joan.vinyals@bsc.es
Marta Garcia Gasulla marta.garcia@bsc.es

4th Annual Meeting of EUROfusion HPC ACHs

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.

Topics

- Hybrid Stella
- Improving the initialisation
- Maintainability with better performance
  - (Linear algebra libraries, and Fortran 2008)
- Results

Hybrid Stella

Hybrid Stella

- **Our problem**: Memory limitations are set by
  - Replication of data across MPI processes.
  - and, very large response matrices
    - Which are also stored in every MPI process.

- **Original solution**: response matrices are shared within:
  - MPI Shared memory: 1 copy per node
  - or MPI Shared memory (SPLIT_BY_NUMA): 1 copy per NUMA domain

- **Our contribution**: use OpenMP to parallelize within the NUMA domain
  - Still keeps 1 copy of the response matrices per NUMA domain
  - But, because we can run with less processes, we save on replicated data per MPI process.

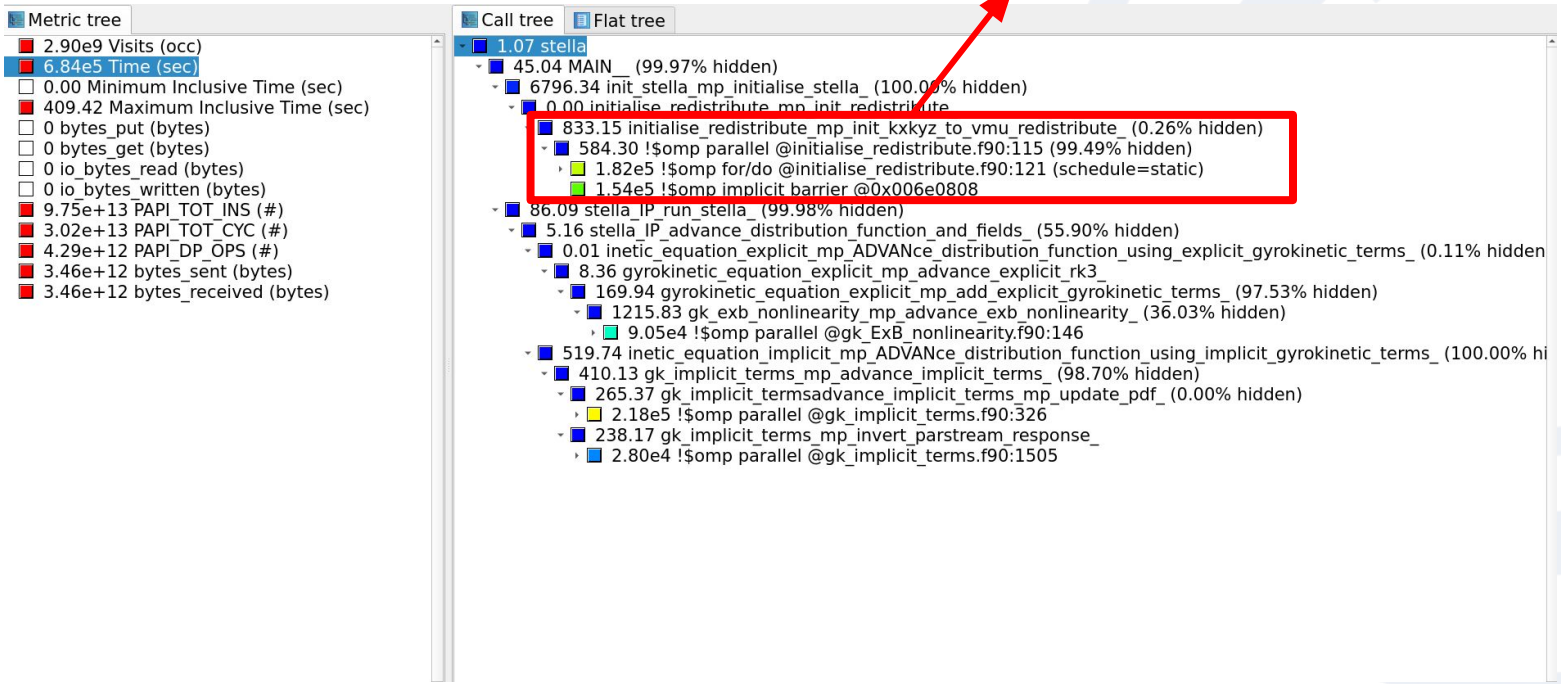
Goal: Enabling multiscale simulations

- High resolution (GOAL):
  - mu=12, nvpa=48, nzed=64; nx=1673 ny=1672, b=stellarator
- Low resolution, but still physical (SCIENCE STARTS HERE):
  - mu=12, nvpa=24, nzed=48; nx=743 ny=742, b=stellarator
- Low resolution, “physics hack”:
  - mu=12, nvpa=24, nzed=48; nx=743 ny=742, **b=periodic**

Improving the initialisation

Initialization improvements: Redistribute index maps

We observed high times spent in the initialization of the redistribute function.



Initialization improvements: Redistribute index maps

- **The objective of** *Redistribute index maps*: precompute the communication neighbours to remap the distribution function from one decomposition to another.

- **Original implementation**: iterates over all the grid space to decide its mapping twice
  - and its replicated by all MPI processes.

- **Our improvement:** each MPI process iterates over the part of the grid that it owns, only once.
  - This implementation showed a 30x improvement with a smaller grid
    - (naky=43, nakx=85, nzed=48, nspec=2, nvgrid=24)
  - This implementation scales with MPI processes.

Initialization improvements: Port to OpenMP

- **Our OpenMP initial implementation:** did not port the response matrices computation to OpenMP

- This implementation ports the computation of the response matrices over its columns with OpenMP.

```
!$omp parallel do default(none) schedule(dynamic, 1)
do icol = 1, nresponse
     call get_velocity_response(response_matrix(:,icol))
end do
!$omp end parallel do

call MPI_Allreduce(respone_matrix)

!$omp parallel do default(none) schedule(static)
do icol = 1, nresponse
     call fill_response_matrix_column(icol)
end do
!$omp end parallel do
```

- With the “Low resolution (b=stellarator)” run we improve the initialization time from 60 to 5 min.

Linear algebra libraries

Linear algebra improvements

- **Original solution for linear algebra**:
  - Hand-written solvers for dense LUs.
  - LAPACK for banded LUs.

- **Our contribution:** unified the solver interfaces for banded and dense LU to use
  - PLASMA (https://icl.utk.edu/plasma/) or LAPACK.

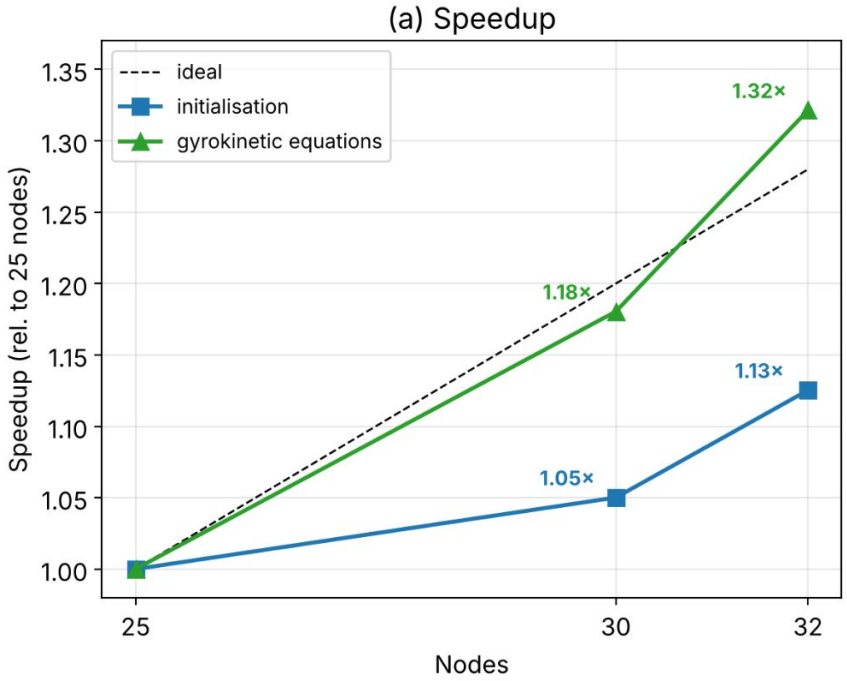
- **Benefits:**
  - Better code maintainability.
  - Better performance:
    - (Low resolution “b=periodic”) **Global** LU decomposition go from **60 minutes** to **12 seconds**.
    - (Low resolution “b=periodic”) **Local** LU decomposition go from **20 seconds** to **12 seconds**.

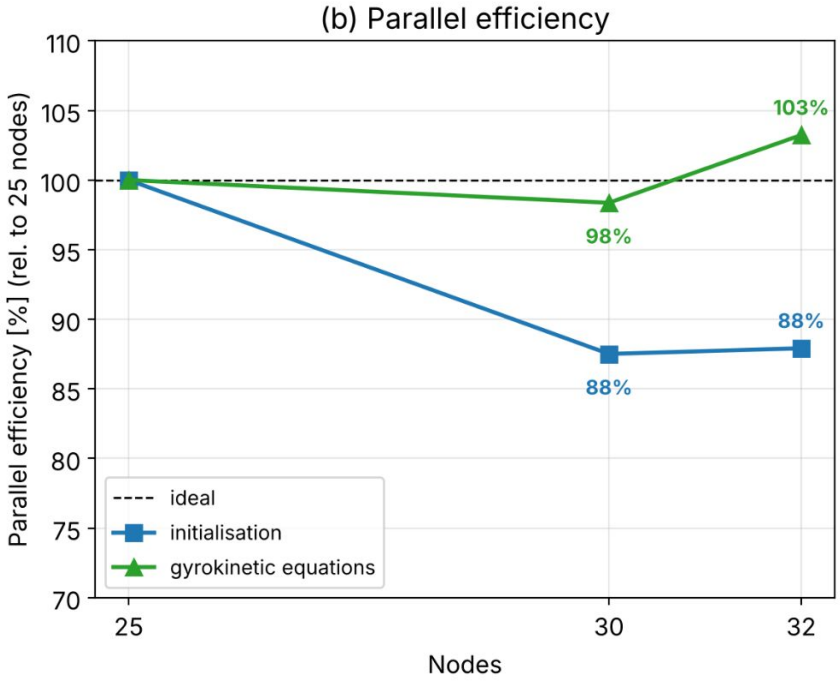
Maintainability

- We modified the code and build processes to comply with Fortran 2008
- This should help in compiler portability

Results

“Scalability” of Hybrid Stella (Low resolution b=stellarator)





Summary Improvements

Baseline

Our contribution

		b=stellarator		b = periodic	
		LU=local	**LU=global**	**LU=local**	**LU=global**
Stella v1.0	Initialization	Out Of Memory Killed		33 min	109 min
+ bug fixes	- LU Decomp			20s	60 min
2800MPI	Timestep			36s	36s
	Memory			115GB/node	180GB/node

		b=stellarator	b = periodic
Hybrid Stella	Initialization	5 min	9 min
	- LU Decomp	130s	12s
50 MPI	Timestep	360s	360s
56 Thread/MPI	Memory	216GB/node	103GB/node

Conclusions and next steps

- **Conclusions**
  - **Hybrid STELLA:** to bring down the memory requirements.
    - **To enable** “Low resolution with b=stellarator” simulation.
  - **Improve mapping computation:** from O(grid size) to O(grid partition)
  - **Generic linear algebra interface:** more maintainable, and faster.

- **To finish this year:**
  - Check numerical robustness with longer time evolutions.
  - Upstream changes

- **Possible continuation work:**
  - Assess the performance of the new version
  - Investigate OpenMP performance gap in the time-step.

27–28 Oct. 2026

HYBRID STELLA

A STEP CLOSER TO MULTISCALE GYROKINETIC SIMULATIONS ON STELLARATORS

Valentin Seitz valentin.seitz@bsc.es
Joan Vinyals Ylla-Català joan.vinyals@bsc.es
Marta Garcia Gasulla marta.garcia@bsc.es

4th Annual Meeting of EUROfusion HPC ACHs

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.