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.

Solving the generalized eigenvalue problem in the JOREK free boundary and resistive wall extension: a progress report from the CIEMAT-BSC ACH about the assessment of using ELPA

F. Cipolletta, N. Isernia, S. Ventre, M. Hoelzl, N. Schwarz, G. Rubinacci, A. Soba, E. Cabrera

25/11/2025

3rd Annual Meeting of EUROfusion HPC ACHs

Outline

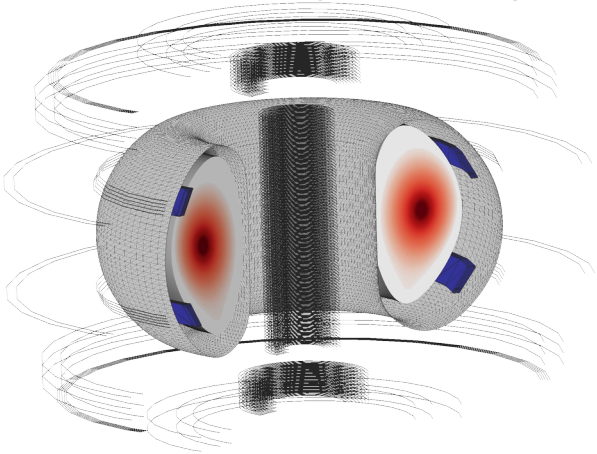
- **Old work**
  - Recap on J-S and J-C couplings
  - What has been attempted
- **New work**
  - The generalized eigenvalue problem
  - Problems and ideas
  - Available tools
  - Methodologies
  - Results
- **Conclusions**
  - Takeaways and Perspectives

Old Work

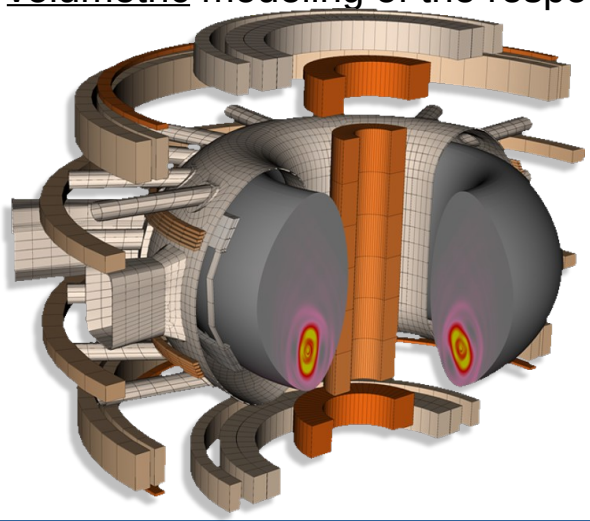
Recap on J-S and J-C couplings

- The *free-boundary and resistive wall extension* considers the **interactions** between the conducting structures and the plasma
- Calculations are done via coupling of JOREK to STARWALL (**J-S**) or CARIDDI (**J-C**) by means of response matrices
- Those couplings consist of memory-intensive works to be performed, directly related to the accuracy adopted in 3D modeling
- Representing "big" geometries (like ITER) with high accuracy may easily become **untreatable** on modern CPU architectures

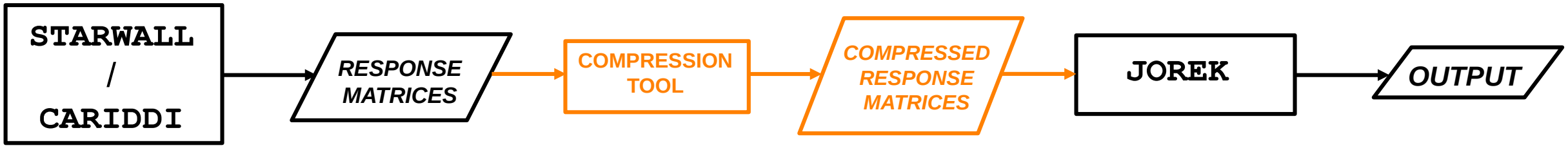
Geometry used in **STARWALL** adopting a 3D thin wall modeling of the response



Geometry used in **CARIDDI** adopting a 3D volumetric modeling of the response



What has been attempted



- A **compression tool** has been implemented:
  - Compress the response matrices using **SVD**
  - Produce a new response file holding compressed matrices
  - Adapt JOREK to deal with the compressed matrices format
  - Test the new implementation with **TM** and **VDE**
  - See 10.1088/1361-6587/ad728a for details
- Some **issues**:
  - The (theoretical) achievable compression is somehow **limited**
  - The compressed matrices can be **not effective** in all the scenarios of applications (e.g. in VDE)
  - The novel implementation is quite **complicated** and merging into develop is still under consideration

New Work

The generalized eigenvalue problem

- Both STARWALL and CARIDDI rely on the solution of a generalized eigenvalue problem (**gEP**)
- The **3D passive structures** (STARWALL/CARIDDI) are interfaced with the **plasma** (JOREK)
- JOREK adopts a 2D Bezier discretization on the poloidal cut and a Fourier expansion on the toroidal direction
- STARWALL/CARIDDI instead models the walls in full 3D

$$L_w^* \, S_\lambda = \lambda \, R_w \, S_\lambda ,$$

L_w^* → Inductance (**dense**) matrix for the conductive structures

S → Basis of eigenvectors

R_w → Resistance (**dense**) matrix

λ → Eigenvalue

- Once this problem is solved, the system of equations treated in STARWALL/CARIDDI becomes **diagonal**
- Both the inductance and resistance matrices have a **size of the walls' DoF squared**
- No cut can be done, because a **complete basis of eigenvectors** is needed

See 10.1063/5.0167271 for further details

20 YEARS BSC

Problems and ideas

- The gEP can become untreatable if considering ITER-like geometry, due to required memory
- Two ideas:

  1. Evaluate **alternative solvers** for the gEP:
     - Test new libraries and optimization and try to reach the maximum treatable number of DoF
     - Test also the capabilities to exploit GPUs for the calculation
  2. Evaluate Model Order Reduction (**MOR**):
     - Perform a first assessment of MOR to limit the DoF on the walls without losing precision

- The rest of the talk will report what has been done so far for 1.

Available tools

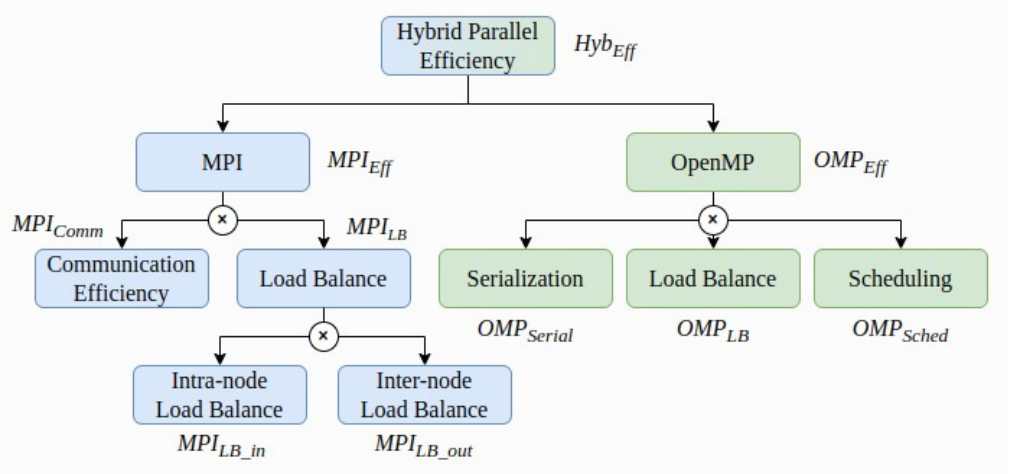
1. ScaLAPACK
   - This is currently used in STARWALL/CARIDDI
   - The implementation starts to be a bit old (mid 90s) but it is robust
   - It is still considered the **state-of-the-art reference** for parallel linear algebra on **CPUs**
2. MAGMA
   - It is developed by the same group of LAPACK/ScaLAPACK
   - It offers CPUs and GPUs capabilities
   - The gEP solver is not available on GPUs → **NOT AN OPTION**
3. ELPA
   - Developed at MPG
   - It offers CPUs (**ELPA2** solver) and GPUs (**ELPA1** solver) capabilities
   - **The results shown ahead are obtained with this library**
4. SLATE
   - Developed by the same group of LAPACK/ScaLAPACK
   - It offers CPUs and GPUs capabilities
   - Not clear if the gEP solver is available on multi-GPUs...**should still be investigated and tested**...

Methodologies used

1. Toy Fortran code to solve the gEP calling ELPA or ScaLAPACK
2. The matrices can be defined
   - Randomically (prescribed dimension)
   - Analytically (prescribed complexity)

   Test implementation
   - Read from STARWALL
   - Read from CARIDDI

   Results
3. Measure time for gEP with **WTIME**
4. Measure memory consumption per node with **free**, at a prescribed time interval
5. Measure memory consumption per GPU with **nvidia-smi**, at a prescribed time interval
6. Measure Parallel Efficiency with TALP metrics from the DLB library (Pure MPI)



Results - MareNostrum5

- Due to the initial unavailability of MARCONI, **MareNostrum5** was used for the tests
- 2 partitions available:

  1. ACC

  - 80 cores per node
  - 512 GiB of RAM per node
  - 4 NVIDIA H100 (64GB) GPUs per node

  2. GPP (Standard)

  - 112 cores per node
  - 256 GiB of RAM per node
  - Only CPUs (no GPUs)

Results – Matrices from STARWALL

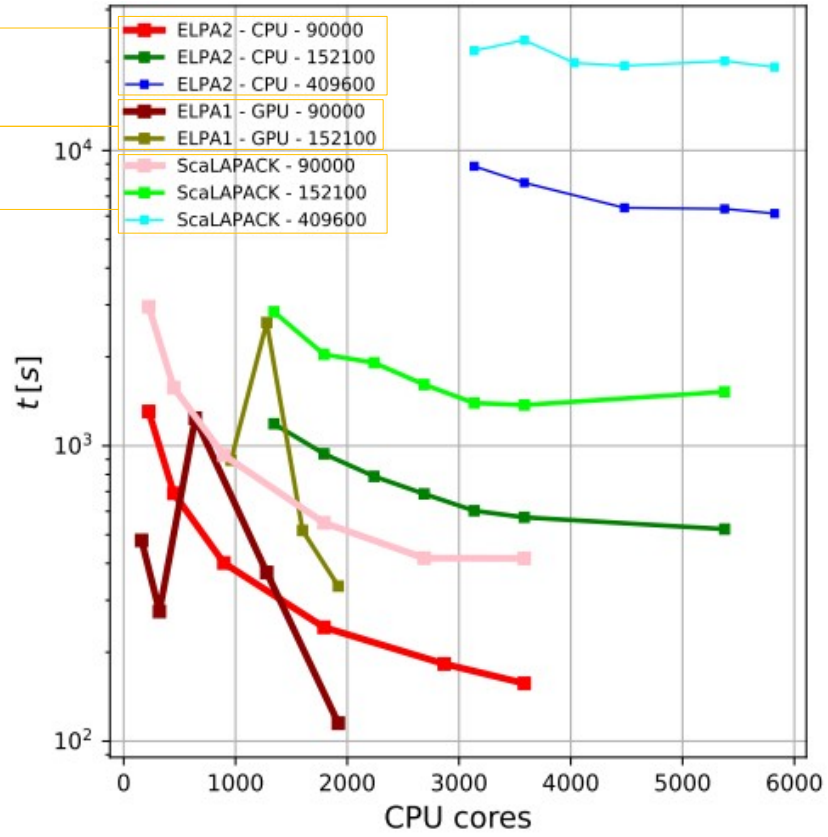
nvu x nwv	300 x 300	390 x 390	640 x 640
Matrix Size	$(90000)^2$ = 8.1 B	$(152100)^2 \approx$ 23.1 B	$(409600)^2 \approx$ 168 B

ELPA2 on CPUs

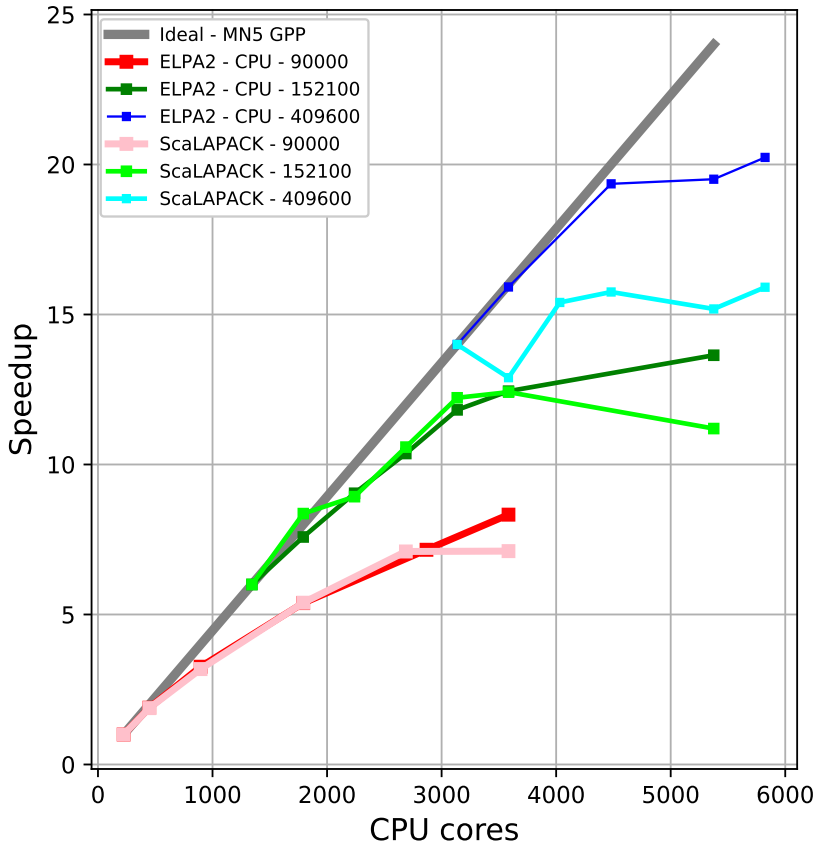
ELPA1 on GPUs

ScaLAPACK

Time to solution



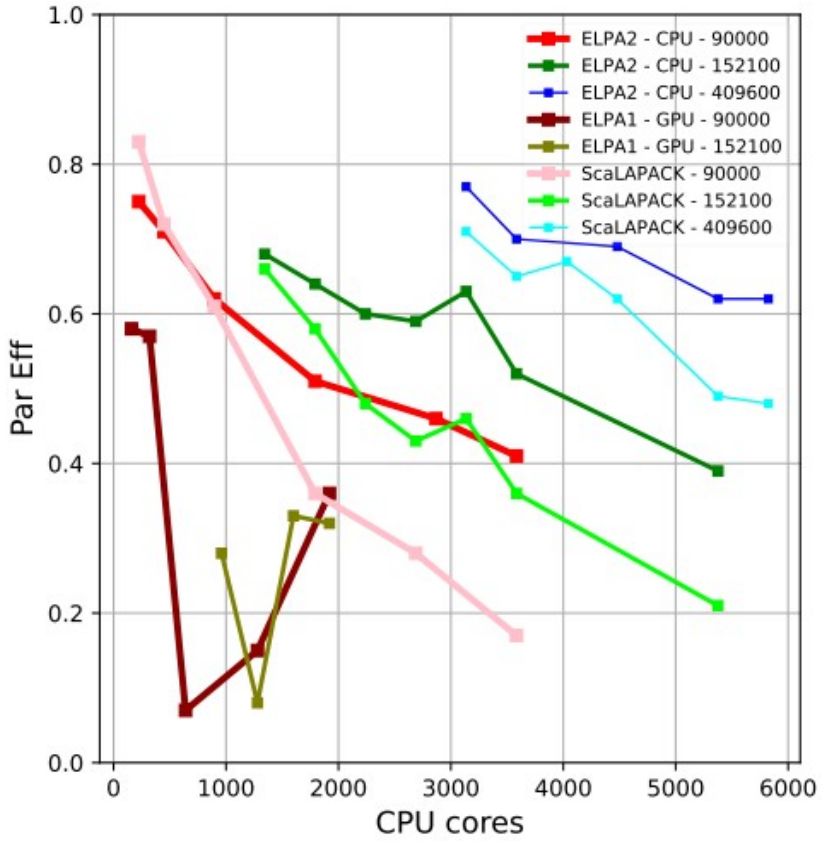
Speedup



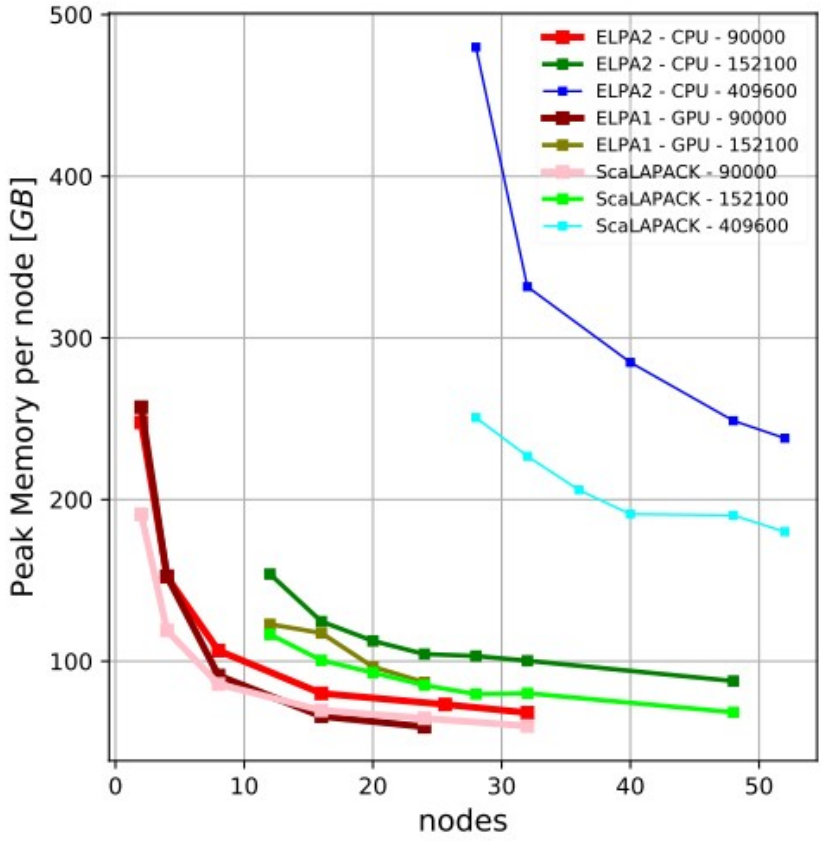
Results – Matrices from STARWALL

nvu x nwv	300 x 300	390 x 390	640 x 640
Matrix Size	$(90000)^2$ = 8.1 B	$(152100)^2 \approx$ 23.1 B	$(409600)^2 \approx$ 168 B

Parallel Efficiency



Peak memory per node



NOTE 1:
All the simulations using CPUs were ran on GPP standard nodes; selecting the compilers toolchain as

INTEL 2024.2 +
MKL 2024.2 +
IMPI 2021.13

was needed to avoid a bug calling too much memory

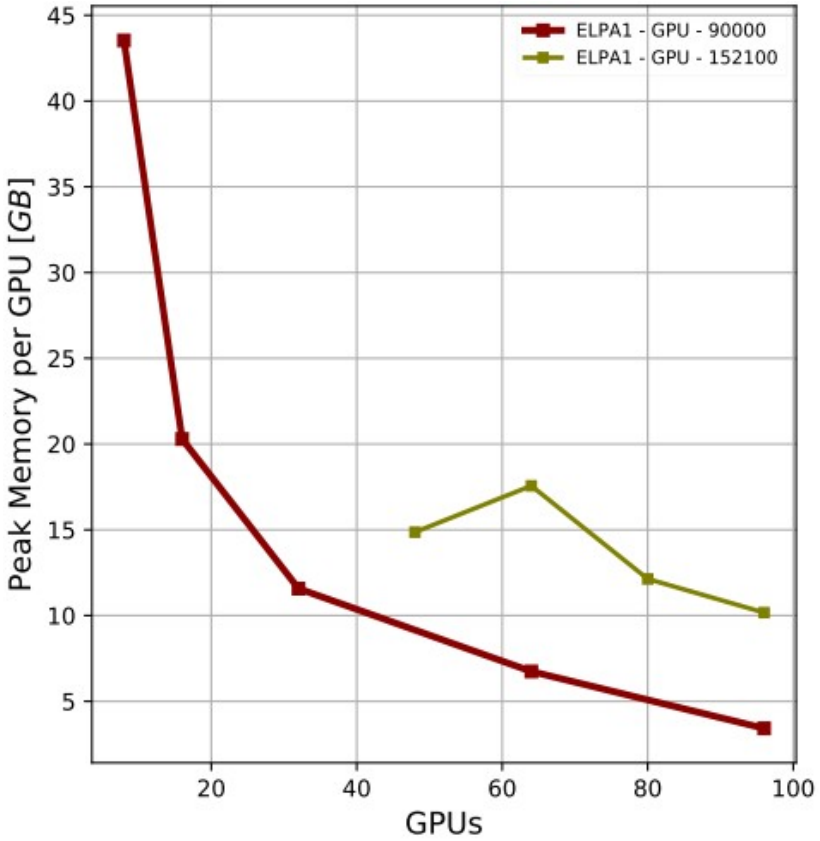
NOTE 2:
On ACC, the same compilers were used together with

CUDA 12.2

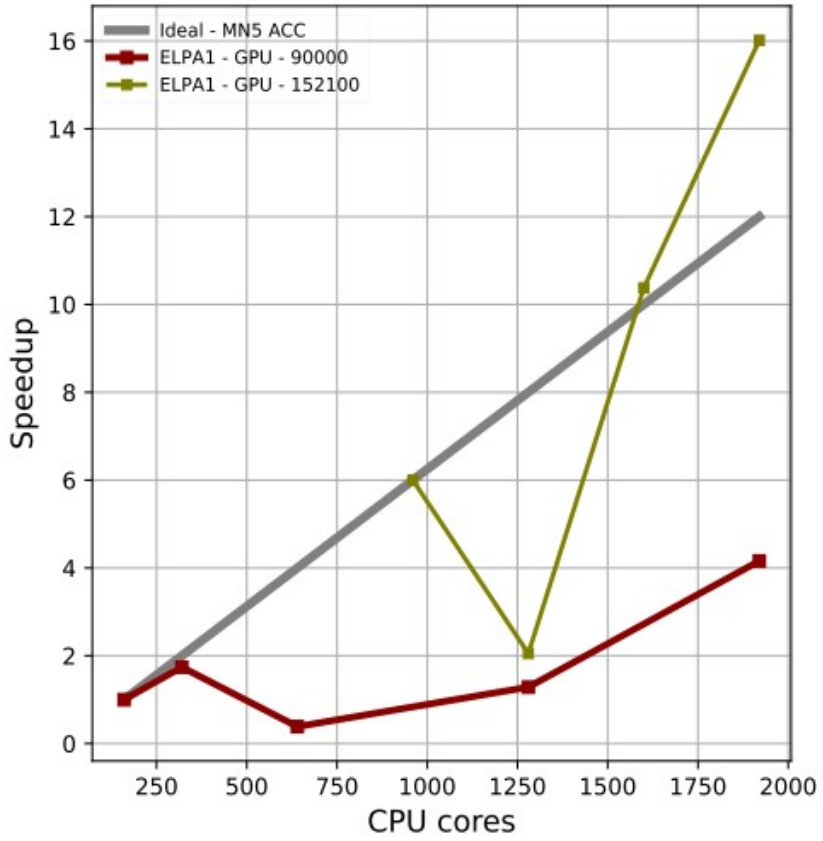
Results – Matrices from STARWALL

nvu x nwv	300 x 300	390 x 390	640 x 640
Matrix Size	$(90000)^2$ = 8.1 B	$(152100)^2 \approx$ 23.1 B	$(409600)^2 \approx$ 168 B

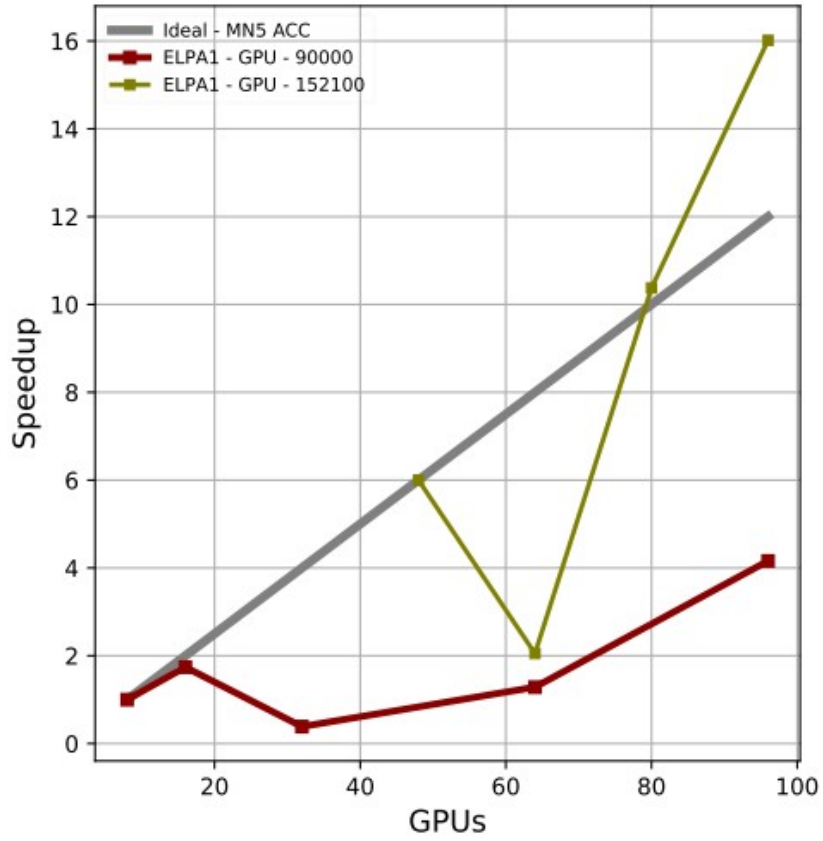
Peak memory per GPUs



Speedup on CPUs



Speedup on GPUs



Results - PITAGORA

- To avoid moving too big matrices, the tests with CARIDDI ones were done on PITAGORA
- 2 partitions available:

1. DCGP

- 256 cores per node
- 768 GiB of RAM per node
- Nodes based on AMD architecture

2. Booster

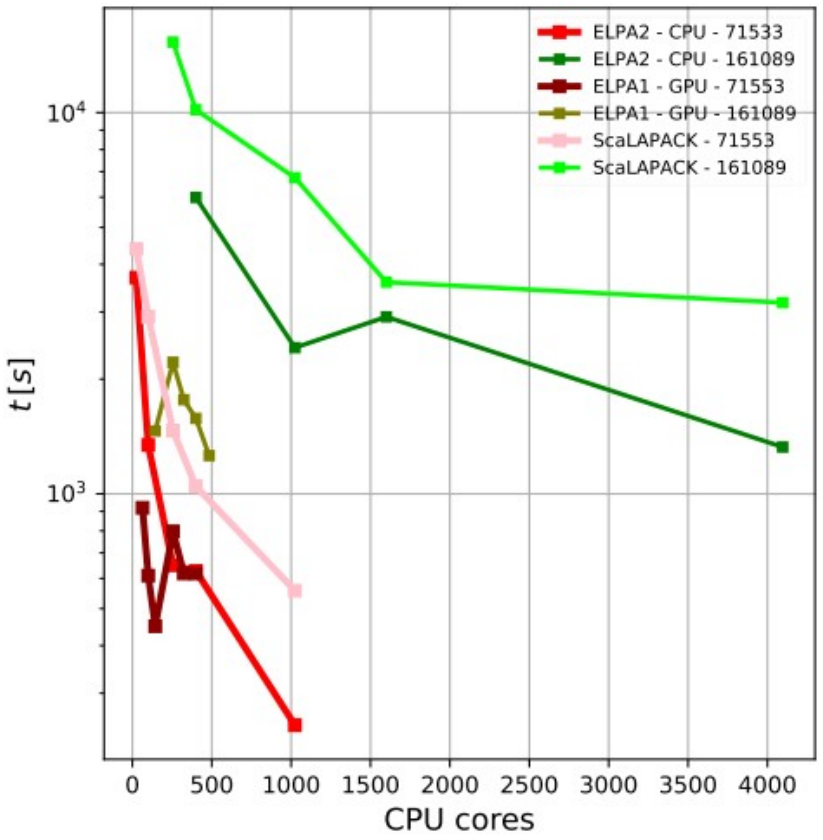
- 64 cores per node
- 512 GiB of RAM per node
- 4 NVIDIA H100 (80GB) GPUs per node

Results – Matrices from CARIDDI

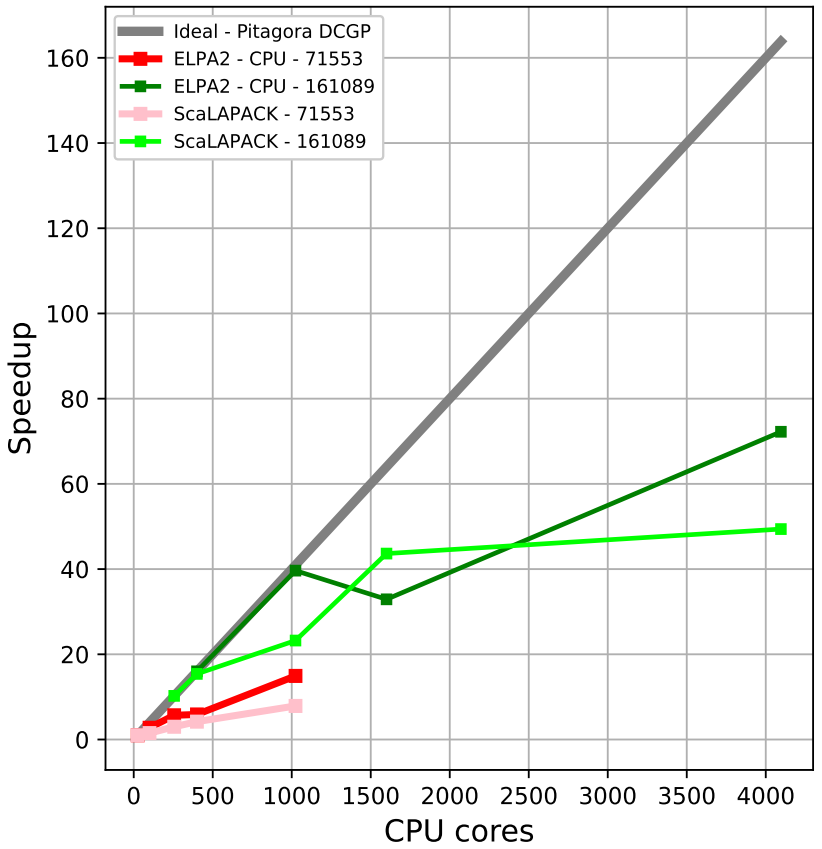
- Some notes on the following results:
- CARIDDI requires CAD models to start the computation → Scaling tests are not trivial
- CARIDDI needs squared grid processors to run → the results are obtained with the same constraint
- Tried compiling ELPA, DLB, and the Eigensolver toy code with
  1. INTEL
  2. GNU
  3. AOCC
- Only 1. and 2. works, while 3. returns wrong eigenvalues from ScaLAPACK → **Ticket never solved**
- CARIDDI was compiled with INTEL compilers, therefore the same are used for the results

Results – Matrices from CARIDDI

Time to solution

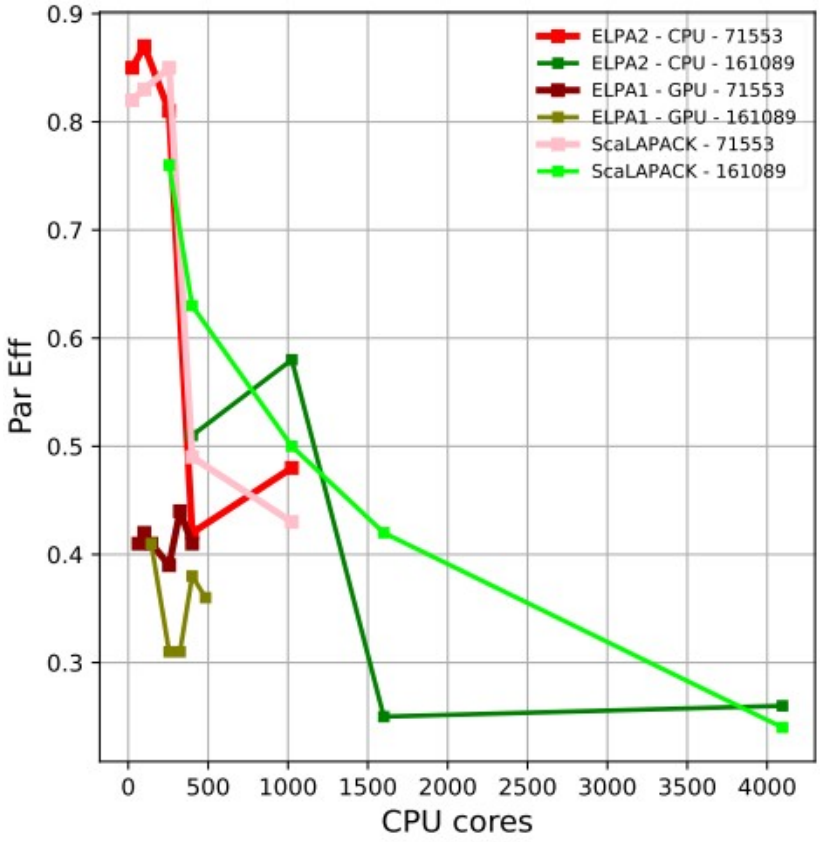


Speedup

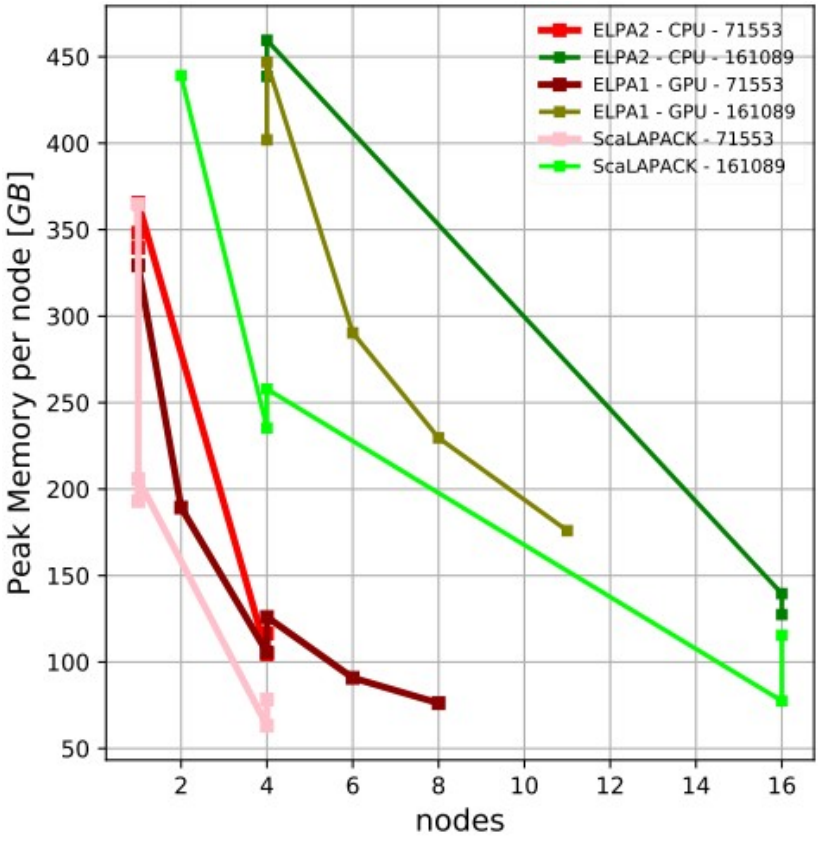


Results – Matrices from CARIDDI

Parallel Efficiency



Peak memory per node



NOTE 3:
All the simulations using CPUs were ran on DCGP nodes; selecting the compilers toolchain as

INTEL 2024.1 +
MKL 2024.0 +
IMPI 2021.12.1

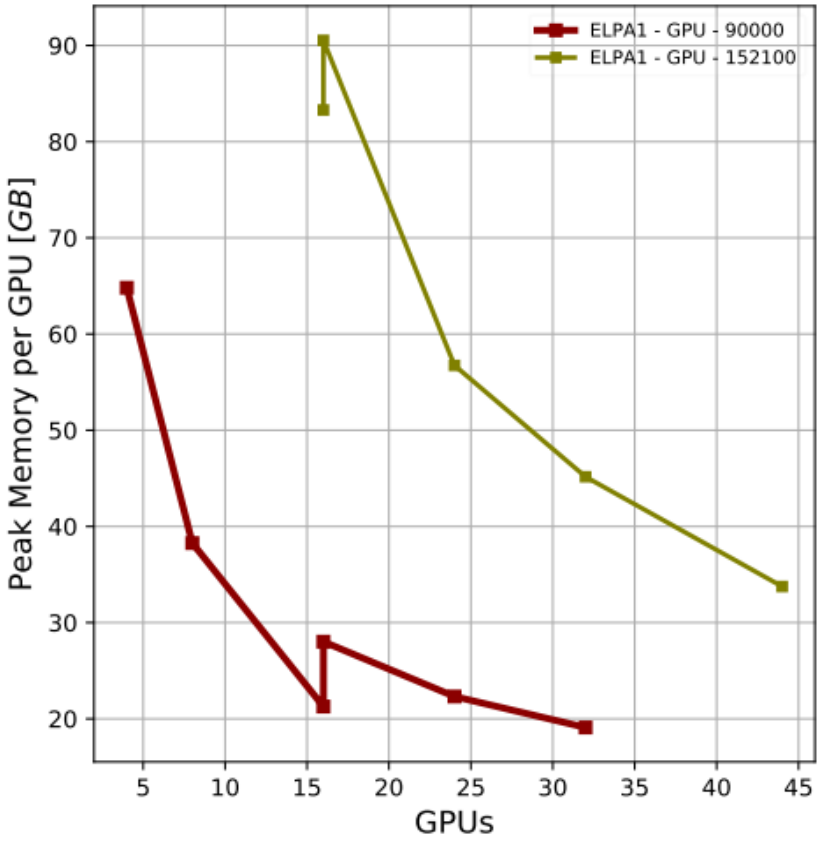
was needed to avoid a bug calling too much memory

NOTE 4:
On Booster, the same compilers were used together with

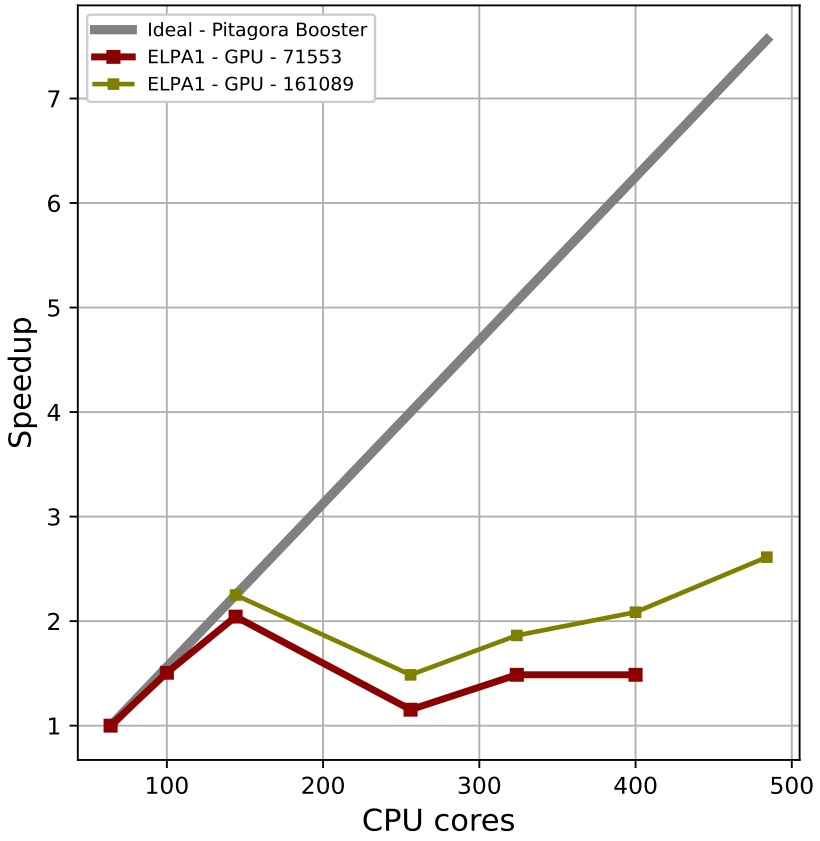
CUDA 12.6

Results – Matrices from CARIDDI

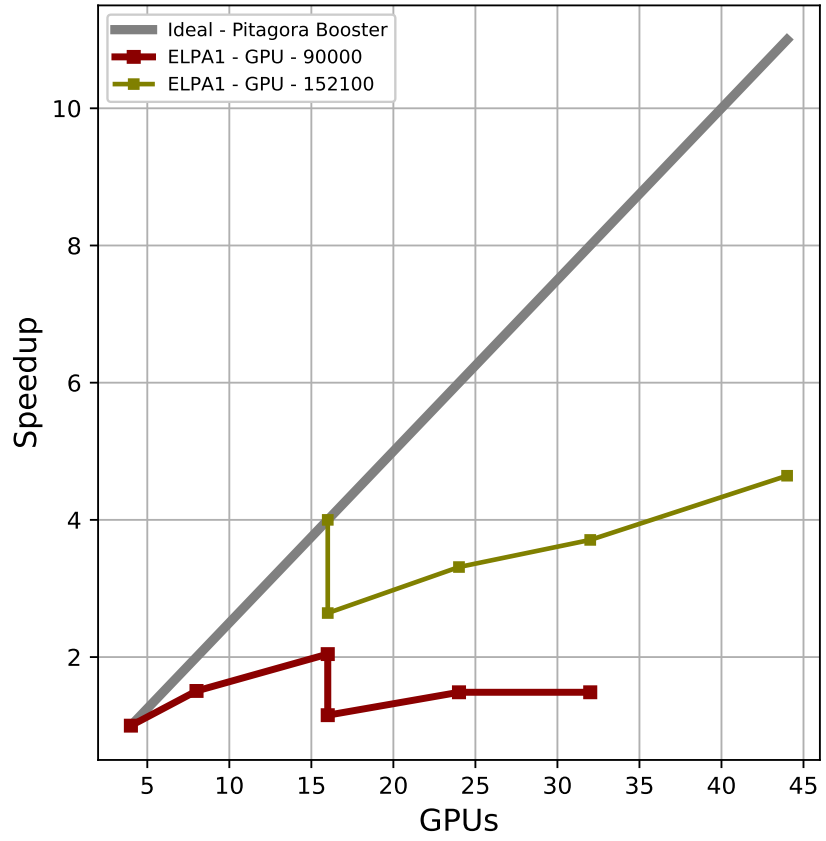
Peak memory per GPUs



Speedup on CPUs



Speedup on GPUs



Conclusions

Takeaway and perspectives

- ELPA offers a very promising solvers for the gEP, in terms of time-to-solution (kind of 3X)
  - CARIDDI team is interested in implementing the library in the code
  - It also offers GPU capabilities → almost automatic introduction within CARIDDI
- The memory utilization of ELPA is slightly worse (kind of 2X) wrt to ScaLAPACK
  - It might not be the best choice to grow with treatable matrix dimensions
- Some more studies and trials might be needed for exploiting GPUs
- Alternative libraries (SLATE) could be investigated
- Shall we consider trying different compilers to exploit AMD architectures?

Thank you

fusion.bsc.es

@Fusion_BSC

fusion-bsc

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.

3rd Annual Meeting of EUROfusion HPC ACHs