New advances to prepare GYSELA-X code for exascale global gyrokinetic plasma turbulence simulations: porting on GPU and ARM architectures

Virginie Grandgirard, Kevin Obrejan, Dorian Midou, Y Asahi, P-E Bernard, J Bigot, E Bourne, J Dechard, G Dif- Pradalier, P Donnel, et al.

New Advances to Prepare GYSELA-X Code for Exascale Global Gyrokinetic Plasma Turbulence Simulations: Porting on GPU and ARM Architectures

Virginie GRANDGIRARD, Kevin OBREJAN, Dorian MIDOU

Y. Asahi[2], P-E. Bernard[3], J. Bigot[4], E. Bourne[1], J. Dechard[5], G. Dif-Pradalier[1], P. Donnel[1], X. Garbet[1], A. Gueroudji[4], G. Hager[6], H. Murai[7], Y. Ould-Ruis[8], T. Padioleau[4], L. Nguyen[9], M. Peybernes[10], Y. Sarazin[1], M. Sato[7], M. Tsuji[7], P. Vezolle[3]

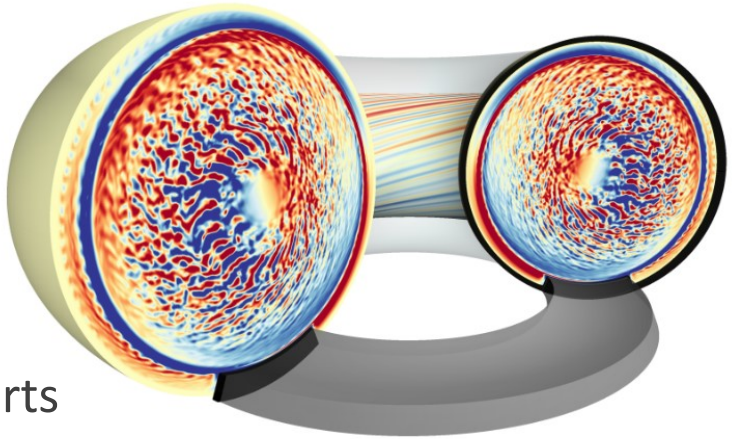
[1] CEA, IRFM, 13108 Saint-Paul-lez-Durance Cedex, France.
[2] Japan Atomic Energy Agency, CCSE, 178-4-4 Wakashiba, Kashiwa, Chiba, Japan.
[3] Hewlett Packard Enterprise (HPE), SENSE Building, 92800 Puteaux, France.
[4] Maison de la Simulation, CEA, CNRS, Univ. Paris-Sud, UVSQ, Université Paris-Saclay, 91191 Gif-sur-Yvette, France.
[5] AS+ Groupe EOLEN, 37-39 Rue Boissière, 75116 Paris.
[6] FAU Erlangen-Nürnberg, Erlangen National High-Performance Computing Center (NHR@FAU).
[7] RIKEN Center for Computational Science, Japan.
[8] INRIA-Grenoble, France.
[9] CEA, DAM/DIF, France.
[10] EPFL, SCITAS, CH-1015 Lausanne, Switzerland.

DE LA RECHERCHE À L'INDUSTRIE

GYSELA-X = GYrokinetic SEmi LAgrangian code for plasma turbulence simulation

[Grandgirard et al., CPC 2016]

- GYSELA-X strength:
  - **Global**: simulate entire tokamak → boundary conditions (SOL-like, **limiter**)
  - **Full-f**: multi-scale physics
  - **Flux-Driven** (heat, momentum, … sources)
  - Multi-ion species → impurity transport
  - Multi-species collision operator → synergy neoclassical & turbulent transports
  - **3 electron modes** : adiabatic (AE), full kinetic (FKE) or trapped kinetic electrons (TKE)
  - Electromagnetic effects (under validation)

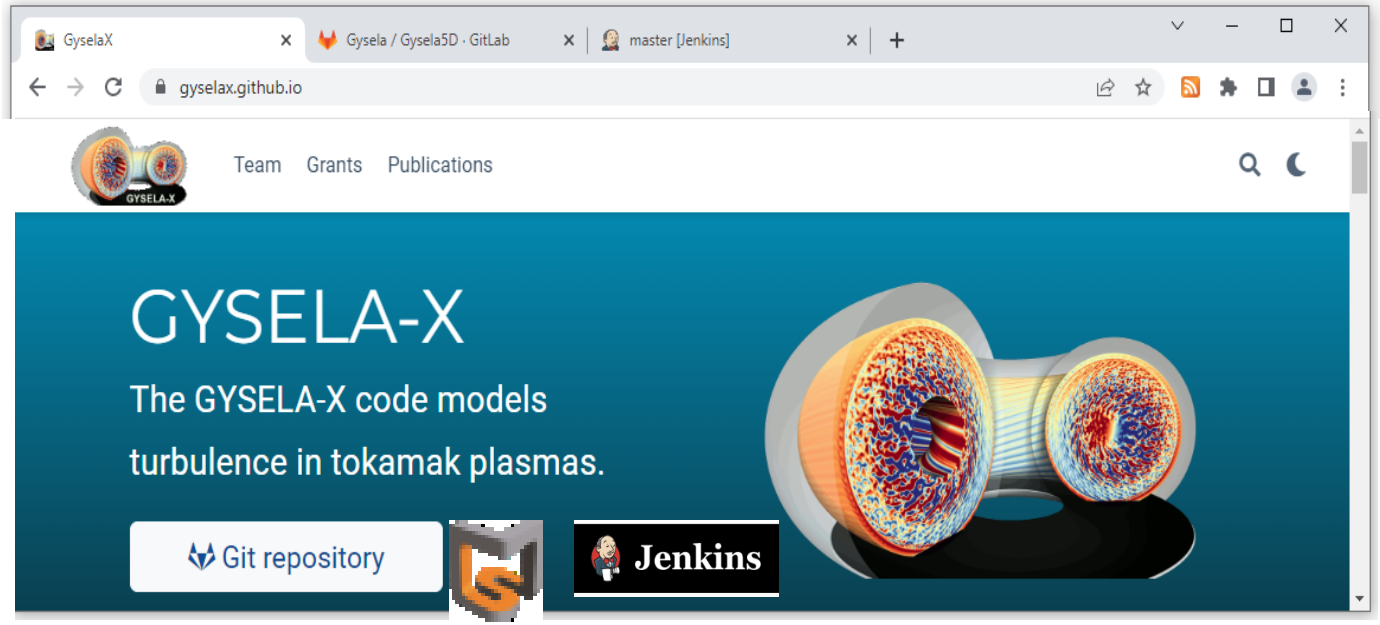


- GYSELA-X current physics interests:
  - Core-edge turbulence → Evidences of core-edge interplay *[Y. Sarazin , PPCF 2021 ; G. Dif-Pradalier, Comm. Phys. 2022]*
  - Ripple effects → Edge flows: impact of non axi-symmetry quantified *[R. Varennes, PRL 2022]*
  - Impurity transport *[G. Lo-Cascio, submitted to NF]*
  - Impact of magnetic configuration on turbulence

GYSELA-X code currently uses Petascale facilities and requires Exascale

- GYSELA-X is based on a **backward semi-lagrangian** scheme
- 25 years-old Fortran code with some C routines
  - Use Blas/Lapack + HDF5 for I/O
- Hybrid MPI/OpenMP parallelism
- GITLAB + JENKINS non-regression tests
- Spack installation available
- SLACK : https://gysela.slack.com

https://gyselax.github.io



Intensive use of petascale resources

- > 100 millions of hours / year
  - (GENCI + PRACE + HPC Fusion resources)
- Optimised for up to 500,000 cores

1 simulation:
- **100 billion points** (5D mesh: 3D space + 2D velocity)
- **> 8 million hours** (20 days / 18 432 cores),
- **10 PBytes of data** (3 Tbytes saved)

Exascale needs for ITER plasma turbulence simulation with electromagnetic effects

How to prepare GYSELA-X to HPC exascale architectures ?

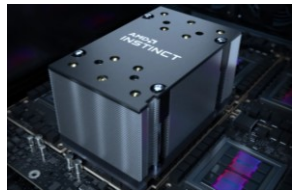
→ Target architectures : 3 in the top 20

- ARM-A64FX porting
- AMD CPU : Scaling up to > 500k cores
- GPU porting in progress on AMD-GPU

GYSELA-X strategy : a unique CPU/GPU code

TOP 500 The List. - JUNE 2022

Rank	System	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
2	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,630,848	442.01	537.21	29,899
17	**CEA-HF** - BullSequana XH2000, AMD EPYC 7763 64C 2.45GHz, Atos BXI V2, Atos Commissariat a l'Energie Atomique (CEA) France	810,240	23.24	31.76	4,959
10	**Adastra** - HPE Cray EX235a, AMD Optimized 3rd Generation EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-11, HPE Grand Equipement National de Calcul Intensif - Centre Informatique National de l'Enseignement Suprieur (GENCI-CINES) France	319,072	46.10	61.61	921

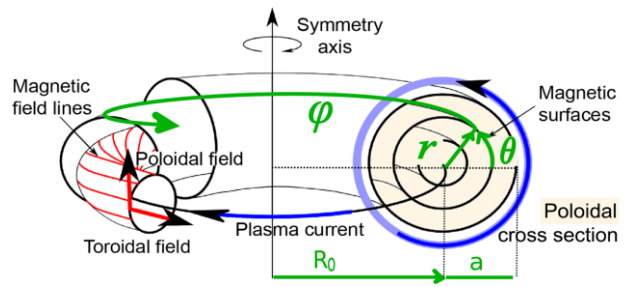


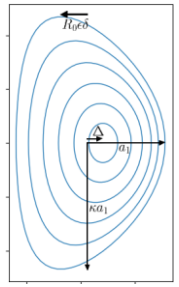
First steps in the rewriting of the code for exascale

geometry

mesh
(equidistant in (r,θ,φ))

magnetic configuration
(circular cross-section or D-shape)





Vlasov

5D Vlasov solver for D + W
(semi-lagrangian scheme)

+

kinetic electrons

$$B^*_{\|s}\frac{\partial \bar{F}_s}{\partial t} + \nabla\cdot\left(\frac{d\mathbf{x_G}}{dt}B^*_{\|s}\bar{F}_s\right) + \frac{\partial}{\partial v_{G\|}}\left(\frac{dv_{G\|}}{dt}B^*_{\|s}\bar{F}_s\right) = C(\bar{F}_s) + S + \mathcal{K}_{\text{buff}}(\bar{F}_s) + \mathcal{D}_{\text{buff}}(\bar{F}_s)$$

with the equations of motion:

$$B^*_{\|s}d_t\mathbf{x}_G = v_{G\|}\mathbf{B}^* + \frac{1}{e}\mathbf{b}\times\nabla\Lambda$$

$$B^*_{\|s}m_s d_t v_{G\|} = -\mathbf{B}^*\cdot\nabla\Lambda$$

where $\mathbf{B}^* = \mathbf{B} + (m_s v_{G\|}/e)\nabla\times\mathbf{b}$ and $\Lambda = eJ_0\phi + \mu B$;

guiding-centers
J_0=gyroaverage
particles

Poisson

3D Poisson solver (Finite Differences in r + Fourier in (θ,φ))

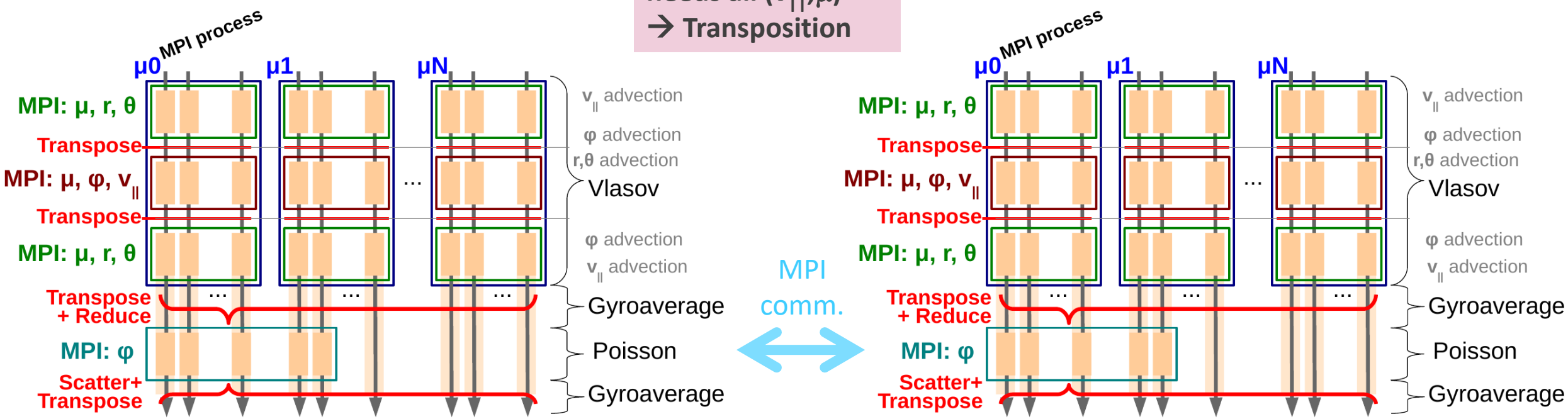
$$\frac{e}{T_{e,\text{eq}}}\left(\phi - \langle\phi\rangle\right) - \frac{1}{n_{e_0}}\sum_s Z_s\nabla_\perp\cdot\left(\frac{n_{s,\text{eq}}}{B\Omega_s}\nabla_\perp\phi\right) = \frac{1}{n_{e_0}}\sum_s Z_s\int J_0\cdot\left(\bar{F}_s - \bar{F}_{s,\text{eq}}\right)d^3v$$

GYSELA – MPI parallelization
→ MPI communicator in μ + domain decomposition in (r,θ) + MPI communicator in species

Species 1 : main ion

+ Collision operator
needs all ($v_{||}$,μ)
→ Transposition

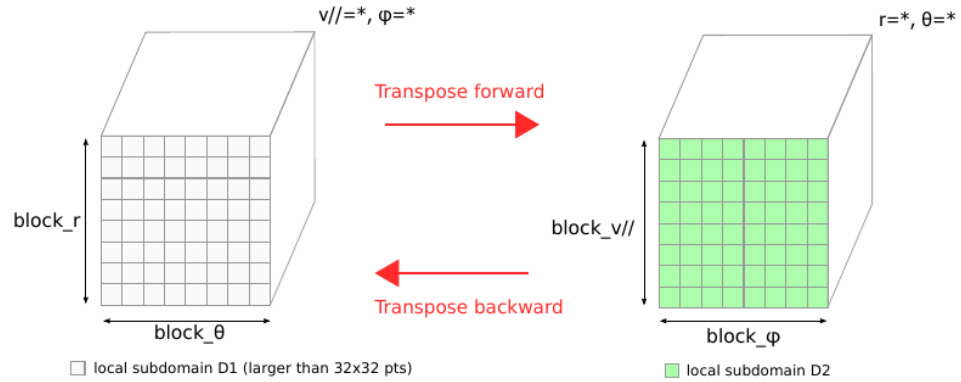
Species 2 : kinetic electrons



→ 3 transpositions per iterations:

- large communication amount ☹:

$$\Theta((N_r N_\theta) N_\varphi N_{v_\parallel} N_\mu)$$



GYSELA-X code optimisation
→ Improvement of the vectorisation

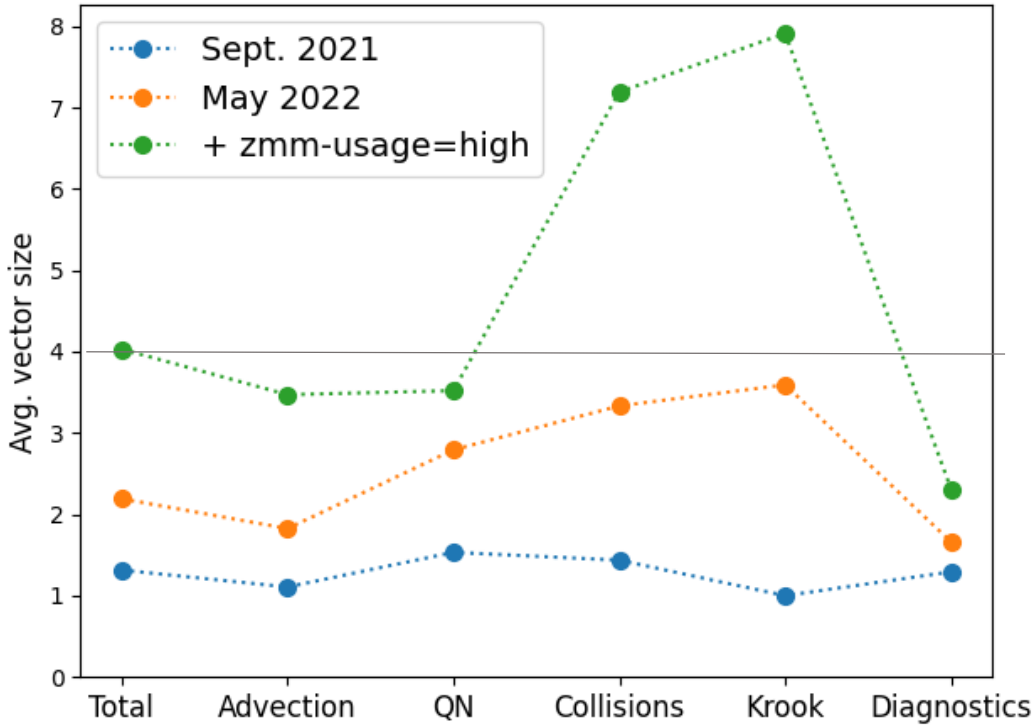
- Profiling of the code performed with several tools
  - → Sept 2021: Very low vectorisation could explain poor performance results on FUGAKU

- Strategy : **Improvement of vectorisation** would benefit to all architectures
  - → May 2022: All kernel vectorisation improved
  - Intel compiler uses preferentially AVX2 (able to activate turbo mode) while AVX512 reduces CPU freqency

- As a result, **hyperthreading** which was the standard mode for GYSELA-X when available **has no more benefit**

Work done in collaboration with FAU university *(G. Hager, M. Wittmann, T. Klöffel)* in EoCoE-II EU project and GENCI support with ATOS *(A. Morvan, S. Jaure)*

Tests performed on MARCONI-SKL partition



Kevin Obrejan, May 2022

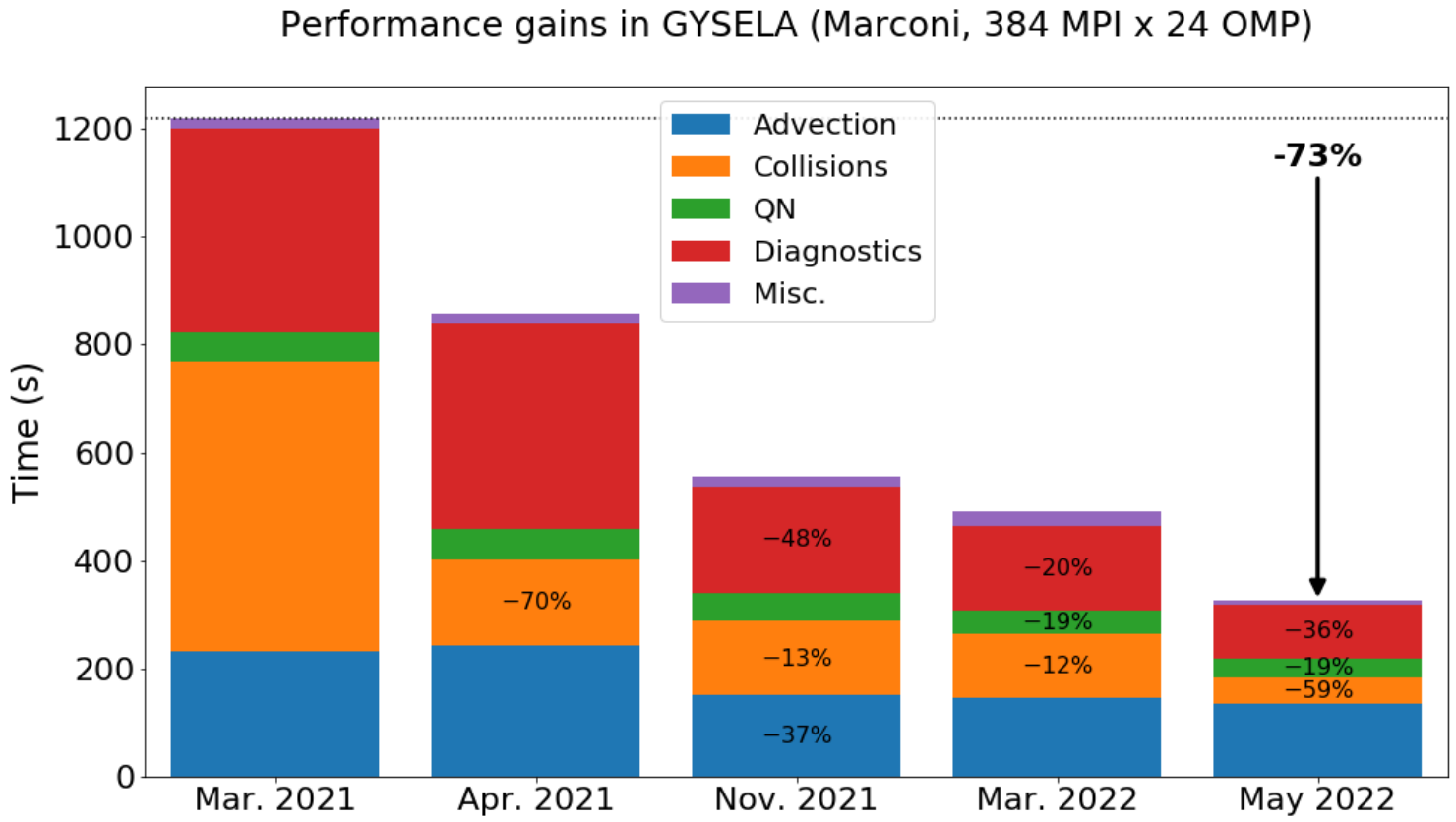
Strong effort of optimisation performed on all the kernels of GYSELA-X → gain of a factor > 70% from 03-2021 to may 2022

Numerical improvements in 2021-2022

- Complete refactoring of collision operator
- More complex gyroaverage operator (to takle D-shape magnetic configuration)
- More complex QN solver for trapped kinetic electrons with limiter
- Source terms simplification

Performance optimisation at node level:

- Vectorisation
- Blocking → Cache optimisation
- Asynchronous MPI communications

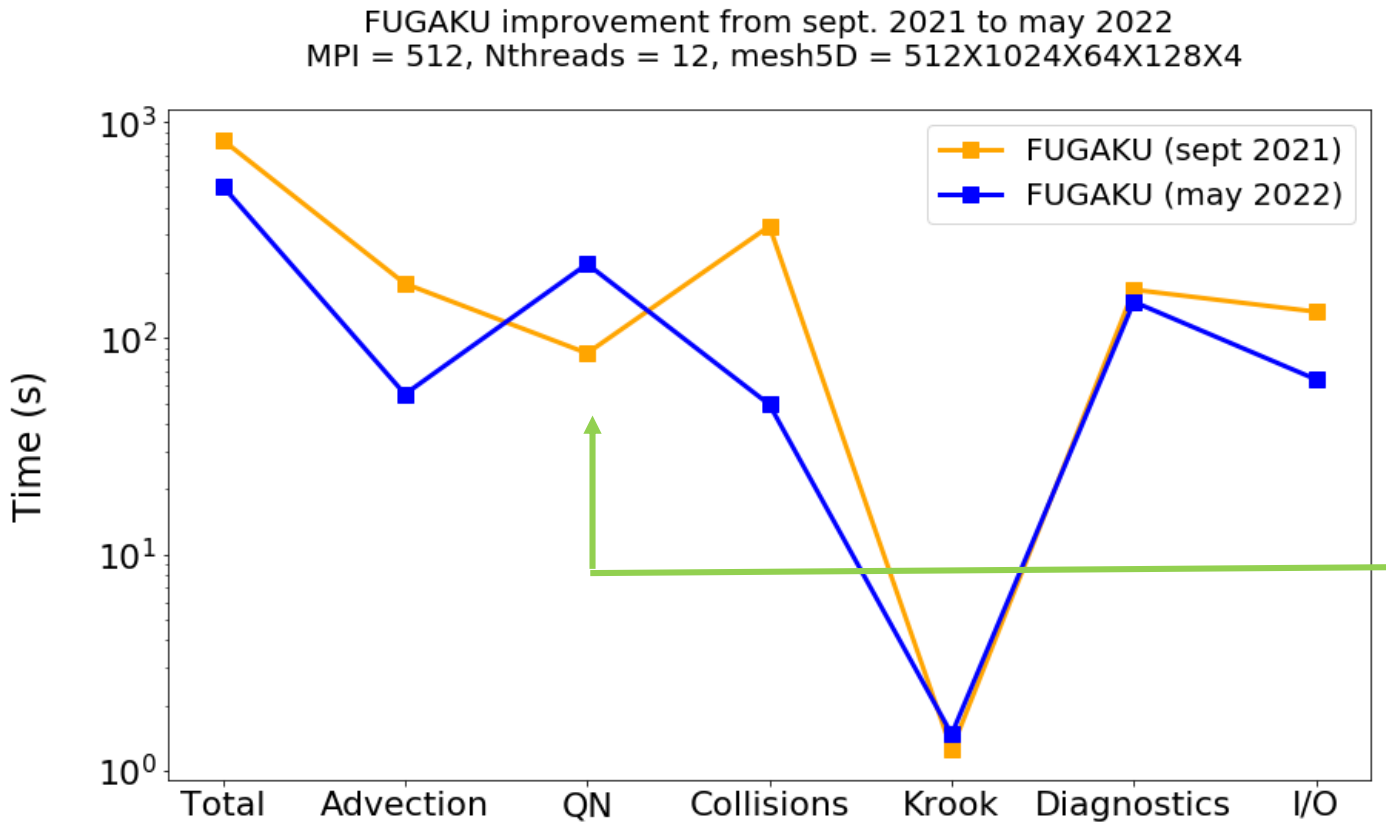


CPU gain around 70% percent

Kevin Obrejan, May 2022

Porting efficiently GYSELA-X on Fujitsu A64FX is not straightforward

- Porting issues on Fujitsu A64FX:
  - Pb for GYSELA-X with FUJITSU compiler optimization : vectorisation + inlining not taken into account
- Almost all improvements performed on SKL have had a beneficial impact on A64FX (FUGAKU computer)



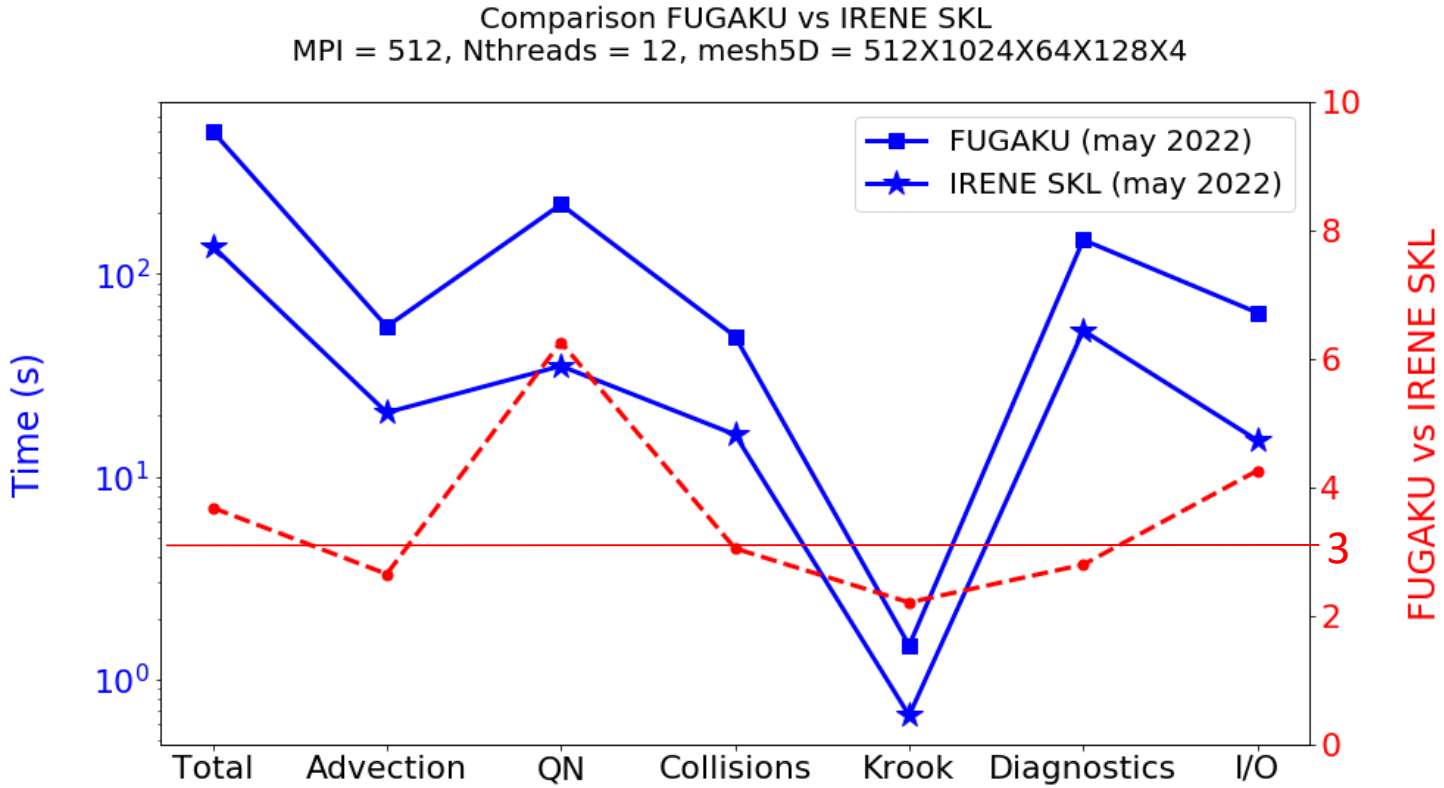
- CEA/Riken collaboration

  (H. Murai, M. Sato, M. Tsuji)

- GENCI support with ATOS *(A. Morvan, S. Jaure)* and Arm *(F. Dupros, C. Hillairet)*

QN solver more complex than in 2021 but degradation not observed on SKL is still not understood

Current results on FUGAKU supercomputer (A64FX)
→ Still 3 times slower than IRENE (Skylake CPU)



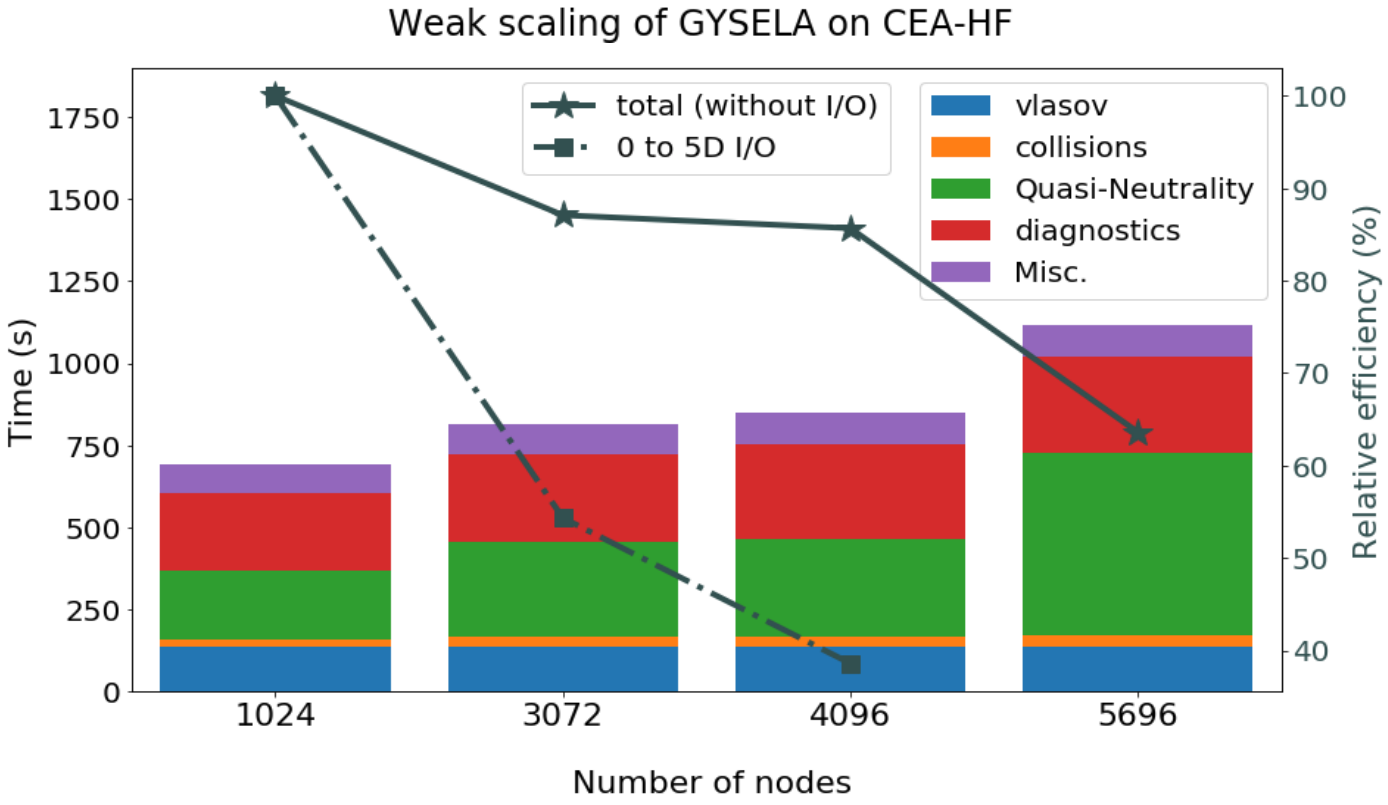
- Except for QN solver, results better compared to 2021 but still a **factor ~3 slower on FUGAKU vs IRENE SKL**
- Still lot of work to obtain good performance on Fujitsu A64FX

  → difficult without developing specific version of the kernels

- Preliminary results show **better performance on AWS Graviton3** (Arm Neoverse V1 cores)

GYSELA-X: weak scaling up to 729 088 cores

- **CEA-HF** : BullSequana XH2000, **AMD EPYC 7763** 64C 2.45GHz, Atos BXI V2, **810 240 cores** (= 6330 nodes)



Opportunity to run GYSELA-X on the new supercomputer CEA-HF at TERA center (France) during « Grand Challenge » campaign *(L. Nguyen)*

Ongoing "Grand Challenge" simulation:

- 5D mesh grid: ~ **$2.75\ 10^{11}$ points**
- 2048 nodes ⇒ **262 144 CPU cores**
- 60 k iterations ⇒ **40 millions CPU hours**
- **30 TB of stored data** (~ 110 000 files)

Relative efficiency of 85% on more than 500k cores and 63% on 729 088 cores

- Main bottlenecks : QN solver (MPI communication) + I/0

Laurent Nguyen, June 2022

I/O : strong bottleneck for exascale supercomputers

- In GYSELA-X, all diagnostics are saved in HDF5
- Huge amount of data that can be stored
  - **1 distribution function $f_{species}$~ 1TBytes**
  - Not possible to store the time evolution of $f_{species}$
  - **diagnostics** = 5D reduction to 3D to 0D data: 2D cross-section, fluid moments, Fourier projection, …
- **restart files** = 1 distribution function per species required

  → **One file saved per MPI process**
  - Advantage : Avoid MPI communications
  - Drawback: Pb of large number of inodes for large number of MPI process

    → file systems do not support an efficient simultaneous writing of more than 1000 files

    → I/O = strong bottleneck for exascale simulations

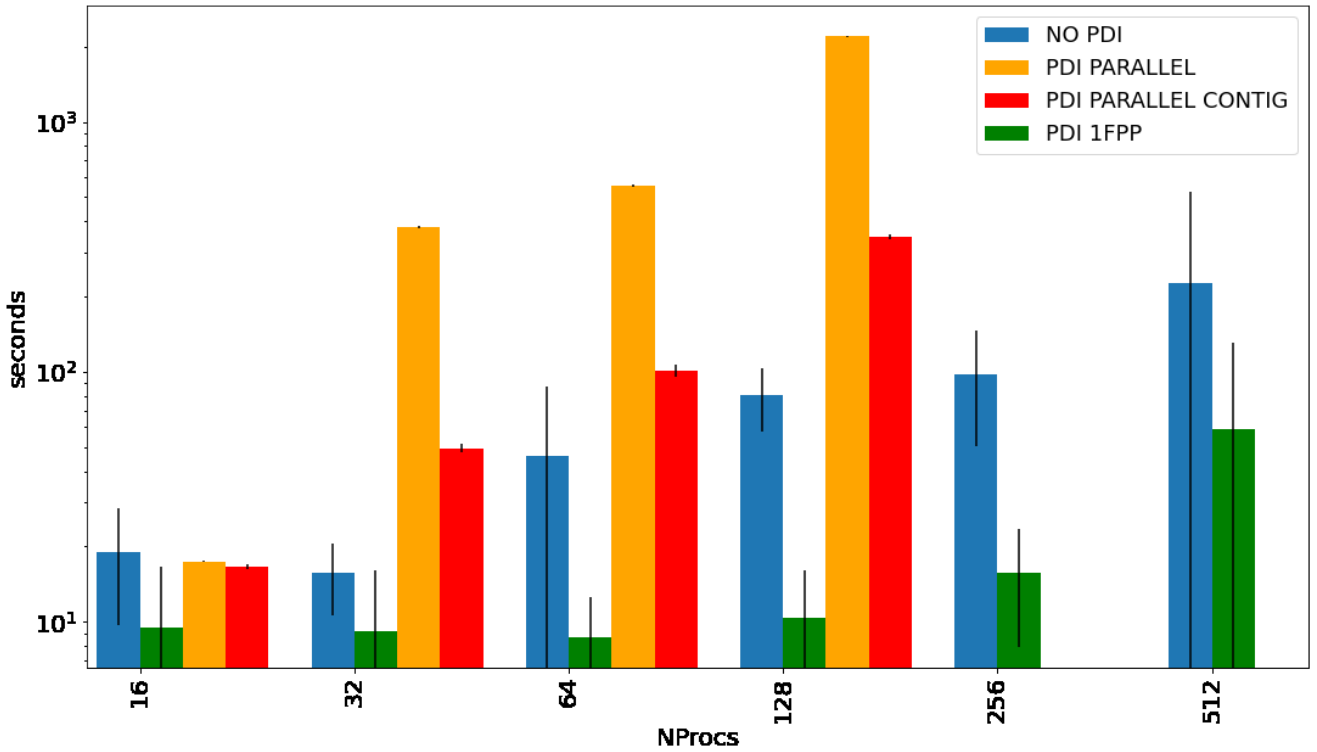
1 simulation:
- **100 billion points** (5D mesh: 3D space + 2D velocity)
- **> 8 million hours** (20 days / 18 432 cores),
- **10 PBytes of data** (3 Tbytes saved)

GYSELA-X is **coupled to PDI** Data Interface for handling I/O

https://pdi.julien-bigot.fr/master/

- PDI is a simple C/C++ API (also available for Fortran and Python) that offers to exchange data between the application code and various external data handlers : file-system for I/O or another code for code-coupling
- PDI is now fully implemented in GYSELA-X (i.e all diagnostics and restart files)
- Only need to modify a YAML configuration file to switch for sequential HDF5 to parallel HDF5

Weak scaling (Irene SKL) : 2GBytes per MPI process



- No PDI and PDI 1FPP = 1 file per MPI process
- PDI parallel = 1 file per mu value

- Large error bar due to strong dependence to the use of the file system by the others
- As expected HDF5 parallel slower than sequential
- Must take care of data contiguity

J. Bigot (MDlS/France) + Y. Ould-Ruis (INRIA-Grenoble/France)

- **Objective**: **Prepare GYSELA-X for running on** ADASTRA **GPU MI250X AMD** partition (CINES-France)

→ OpenMP while ensuring source code readability (unique CPU/GPU code) + CPU performance preservation

Work done with HPE *(P-E Bernard, P. Vezolles)* and EOLEN *(J. Dechard)* in the frame of ADASTRA Contrat de Progrès at CINES and with SCITAS-EPFL *(M. Peybernes)* in the frame of EuroFusion Advanced Computing Hub

Porting strategy:

- Exploit 2 levels of parallelism Teams/Threads to map algorithm to GPU architecture:
  - OpenMP teams mapped to Compute Unit (CU)
  - OpenMP team threads mapped to Scalar Unit / Part of Vector (SIMD) Unit
- Exploit parallelism by using loop blocking:
  - distribute work between teams and threads with a new parameter defining block size

Common routine features :
- Multiple nested calls for each routine
- BLAS/LAPACK calls on small matrices deep in the call-stack
- Multiple levels of loops (5D code)

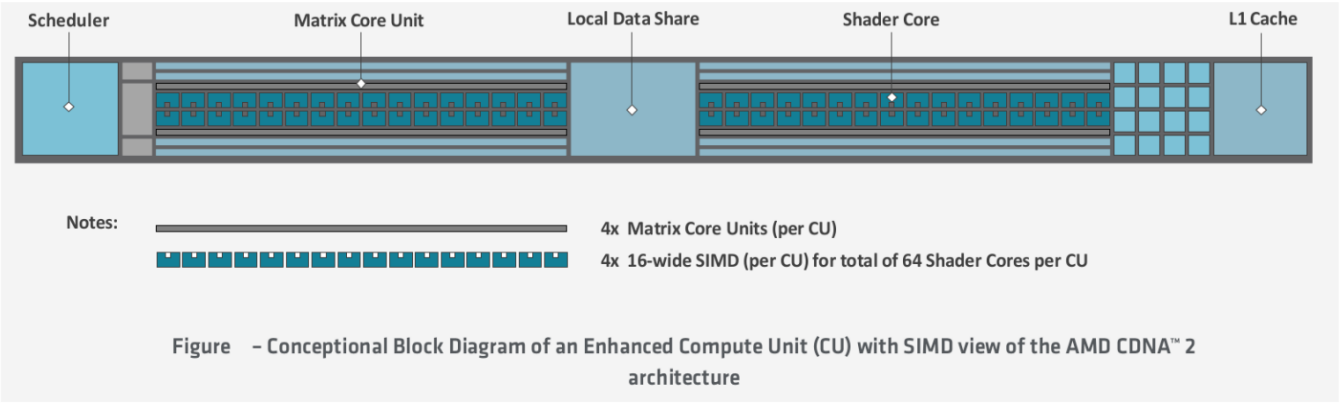


Figure – Conceptional Block Diagram of an Enhanced Compute Unit (CU) with SIMD view of the AMD CDNA™ 2 architecture

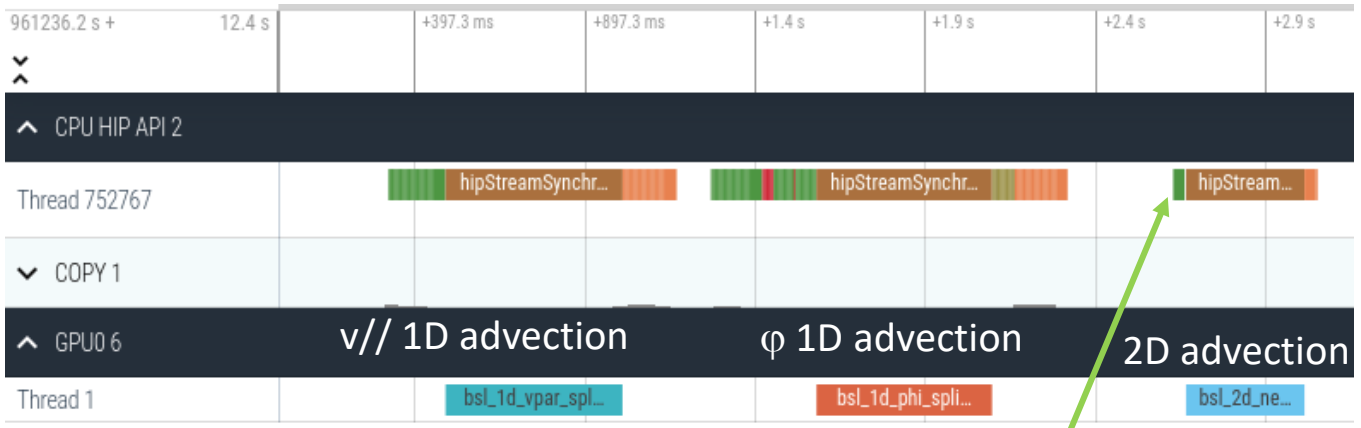
Porting GYSELA on AMD GPUs : ongoing work

- Use of **HPE Cray Compiling Environment** on CPU AMD EPYC 32 + **GPU AMD** MI100 with CRAY CCE/13.0.1
- Main kernels for Vlasov equation ported on GPU (advections with 1D and 2D splines, collisions, source)

→ 2 levels of parallelism teams/threads implemented, wait for CCE/14.0 for efficient parallelism at threads level

Preliminary results: 1MPI, mesh (256x256x32x32x1). Use of 480 teams and 256 threads per team

Kernels	Time CPU - 24 OMP threads (sec)	Time GPU - (sec)
splines2D	6.4	3.3
splines1D_vpar	3.8	11.2
splines1D_phi	3.8	11.6



Work in progress

- Optimize CPU/GPU data transfer → encouraging results with 2D splines
  - Use of runtime CRAY_ACC_DEBUG variable environment and rocprof
- Improve collisions + source (more complicate because lots of routine calls)
  - Use of roctx to focus on specific kernels

J. Dechard,
M. Peybernes, June 2022

Outline

- **How to prepare GYSELA-X to HPC exascale architectures ?**

→ Target architectures : 3 in the top 20

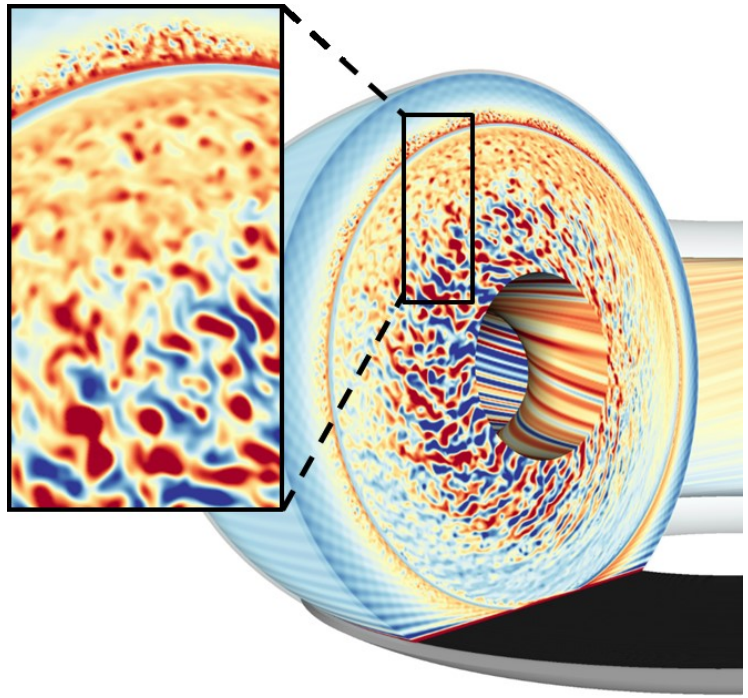
- ARM-A64FX porting
- AMD CPU : Scaling up to > 500k cores
- GPU porting in progress on AMD-GPU

GYSELA-X strategy : a unique CPU/GPU code

- **First steps in the rewriting of the code for exascale**

Why do we choose to rewrite the code ?

- 25 years-old code written in Fortran with hybrid MPI/OpenMP parallelism
- Possible to keep a unique code for CPU (AMD milan or ARM-A64FX) and GPU with OpenMP directives but extremely difficult to obtain good performance on such disparate architectures.
- Non-equidistant mesh would be more adapted to core-edge turbulence
  → Modifying splines in GYSELA-X = rewrite most of the kernels
- Lots of global variables in GYSELA-X → difficult to define interface to couple C++ kernels



↓

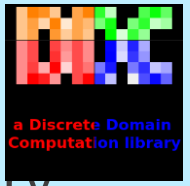
Simpler to rewrite main kernels in modern C++ from scratch and to couple Fortran modules for physics operators (collisions, gyroaverage, sources, …)

New GYSELA-X code → Project structure is almost defined

GYSELA-X | **Voice++**

Gyselalibxx
physics/maths related operators

DDC
A Discrete Domain Computation library
metadata : discretizations, distributions...
https://github.com/Maison-de-la-Simulation/ddc



See T. Padioleau presentation
MS1G at Monday 13:30
for more details

Kokkos
parallel loops

mdspan
arrays

...

PDI
I/O

T. Padioleau (CEA/MDIS), J. Bigot (CEA/MDIS), E. Bourne (CEA/IRFM)

New GYSELA-X code → Where are we ? :
1. Backward semi-Lagrangian validated for non-equidistant mesh

GYSELA-X | **Voice++**

Gyselalibxx

physics/maths related operators

https://github.com/gyselax/gyselalibxx

DDC

A Discrete Domain Computation library

metadata : discretizations, distributions...

https://github.com/Maison-de-la-Simulation/ddc

Kokkos	**mdspan**	...	**PDI**
parallel loops	*arrays*		*I/O*

= 2D (x-v) Vlasov + 1 D poisson

- Successfull BSL on non-equidistant mesh to tackle large temperature gradient

[E. Bourne, in preparation 2022]

- Kinetic sheath physics is recovered

[Y. Munschy, in preparation 2022]

T. Padioleau (CEA/MDIS), J. Bigot (CEA/MDIS), E. Bourne (CEA/IRFM)

New GYSELA-X code → Where are we ?: 2. Finding a good parallelization (MPI+X) strategy

GYSELA-X | **Voice++**

Gyselalibxx
physics/maths related operators

DDC
A Discrete Domain Computation library
metadata : discretizations, distributions...
https://github.com/Maison-de-la-Simulation/ddc

Kokkos
parallel loops

mdspan
arrays

...

PDI
I/O

A 4D vlasov + 2 Poisson mi-app

developed by Y. Asahi to test a good parallelization (MPI+X) strategy

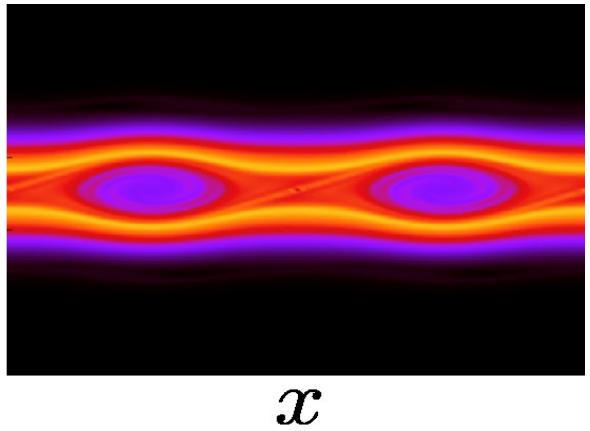
- X can be:

OpenACC/OpenMP/Kokkos/stdpar

T. Padioleau (CEA/MDIS), J. Bigot (CEA/MDIS), E. Bourne (CEA/IRFM)

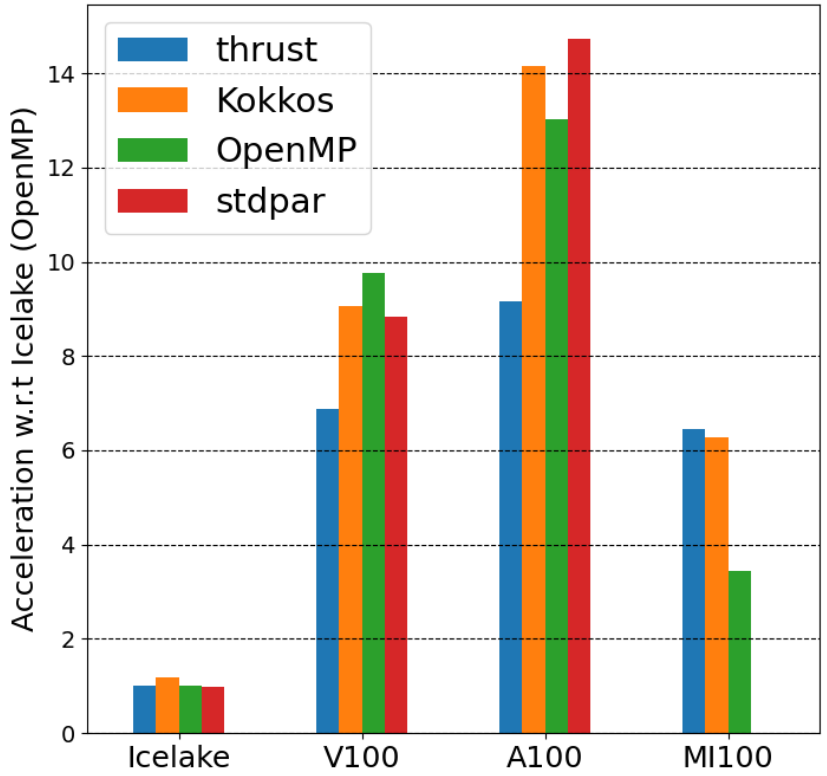
GYSELA mini-app in MPI + X: ready to go with C++ parallel algorithm (stdpar) stdpar version shows quite competitive performance with others

4D vlasov + 2 Poisson mi-app



Icelake: Intel Xeon Platinum 8360 x 2 (icpc 2021. 2, nvc++ 22.5)
V100: NVIDIA V100 x 2 (nvc++ 22.3)
A100: NVIDIA A100 x 2 (nvc++ 22.5)
MI100: AMD MI100 x 2 (rocm 5.0.2)

Problem size: 128^4, #Iterations: 40, 2MPI (1 node)



- stdpar version is quite **competitive** on Nvidia GPUs if **D2D communication enforced** (available with nvc++22.5)
- **mdspan** improves the **readability** and **productivity** with some performance overheads
- **Tiling** with Kokkos is critical for CPUs, which is non-trivial for stdpar

MPI + **stdpar** + **mdspan** is a candidate to keep in mind for GYSELA-X

[NOTE] stdpar is currently available only on NVIDIA GPUs

[Y. Asahi et al, to be submitted to P3HPC 2022]

Y. Asahi (JAEA/Japan), T. Padioleau (CEA/MDlS), G. Latu (CEA/DES), J. Bigot (CEA/MDlS)

Conclusion

- GYSELA-X uses efficiently Petascale ressources (Intel-SKL, AMD-rome) :
  - Relative efficiency of 85% on more than 500 000 cores and 63% on 729 088 cores

  (test performed on 90% of the CEA-HF – AMD EPYC 7763 nodes)
  - more than 100 millions of CPU/hours per year (GENCI, PRACE, Marconi-Fusion partition)
- Strong efforts of optimisation (vectorization, blocking) to port GYSELA-X both on ARM-A64FX and GPU architectures
  - Leads to a gain of 70% on Intel Skylake or AMD-rome CPU
  - Not sufficient to obtain good performance on FUGAKU computer (need dedicated rewritting of kernels)
  - Porting on GPU started 9 months ago and is still in progress
- GYSELA-X rewritting started in modern C++ based on DDC