



**HAL**  
open science

## 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.

### ► To cite this version:

Virginie Grandgirard, Kevin Obrejan, Dorian Midou, Y Asahi, P-E Bernard, et al.. New advances to prepare GYSELA-X code for exascale global gyrokinetic plasma turbulence simulations: porting on GPU and ARM architectures. PASC22 - Conderence on The Platform for Advanced Scientific Computing, the Association for Computing Machinery (ACM); the Swiss National Supercomputing Centre (CSCS), Jun 2022, Bâle (virtual event), Switzerland. pp.1-21. cea-03740685

**HAL Id: cea-03740685**

**<https://cea.hal.science/cea-03740685v1>**

Submitted on 29 Jul 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

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

DE LA RECHERCHE À L'INDUSTRIE



*[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.*

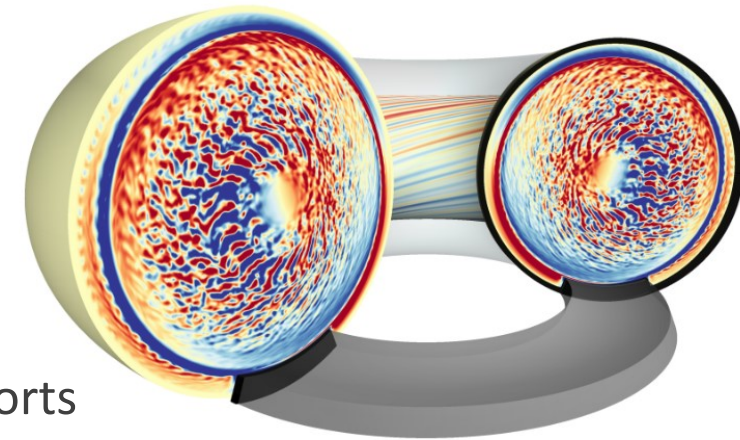
### GYSELA-X strength:

- **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
- Ripple effects → Edge flows: impact of non axi-symmetry quantified
- Impurity transport
- Impact of magnetic configuration on turbulence

[Grandgirard et al., CPC 2016]



[Y. Sarazin , PPCF 2021 ;  
G. Dif-Pradalier, Comm. Phys. 2022]

[R. Varennes, PRL 2022]

[G. Lo-Cascio, submitted to NF]

► GYSELA-X is based on a **backward semi-lagrangian** scheme

<https://gyselax.github.io>

► 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>



**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

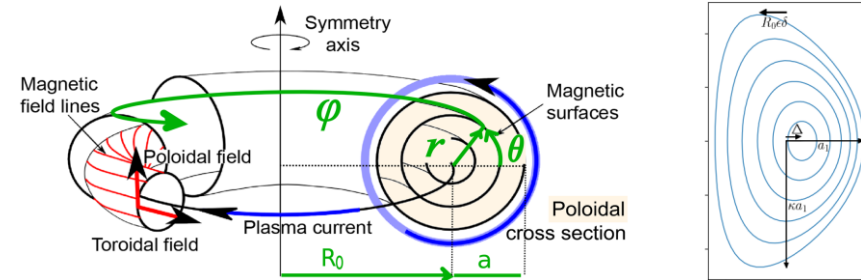
## ► First steps in the rewriting of the code for exascale

Rank	System	TOP 500 - JUNE 2022 The List.	Cores	Rmax (PFlop/s)	Rpeak (PFlop/s)	Power (kW)
2	<b>Supercomputer Fugaku</b> - 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	<b>CEA-HF</b> - 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	<b>Adastra</b> - 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

geometry

mesh  
(equidistant in  $(r, \theta, \varphi)$ )

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



Vlasov

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

+

kinetic electrons

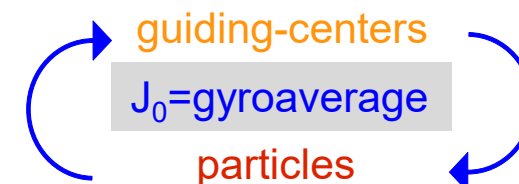
$$B_{\parallel s}^* \frac{\partial \bar{F}_s}{\partial t} + \nabla \cdot \left( \frac{d\mathbf{x}_G}{dt} B_{\parallel s}^* \bar{F}_s \right) + \frac{\partial}{\partial v_{G\parallel}} \left( \frac{dv_{G\parallel}}{dt} B_{\parallel s}^* \bar{F}_s \right) = C(\bar{F}_s) + S + \mathcal{K}_{\text{buff}}(\bar{F}_s)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

with the equations of motion:

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

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

where  $\mathbf{B}^* = \mathbf{B} + (m_s v_{G\parallel} / e) \nabla \times \mathbf{b}$  and  $\Lambda = e J_0 \phi + \mu B$ ;



Poisson

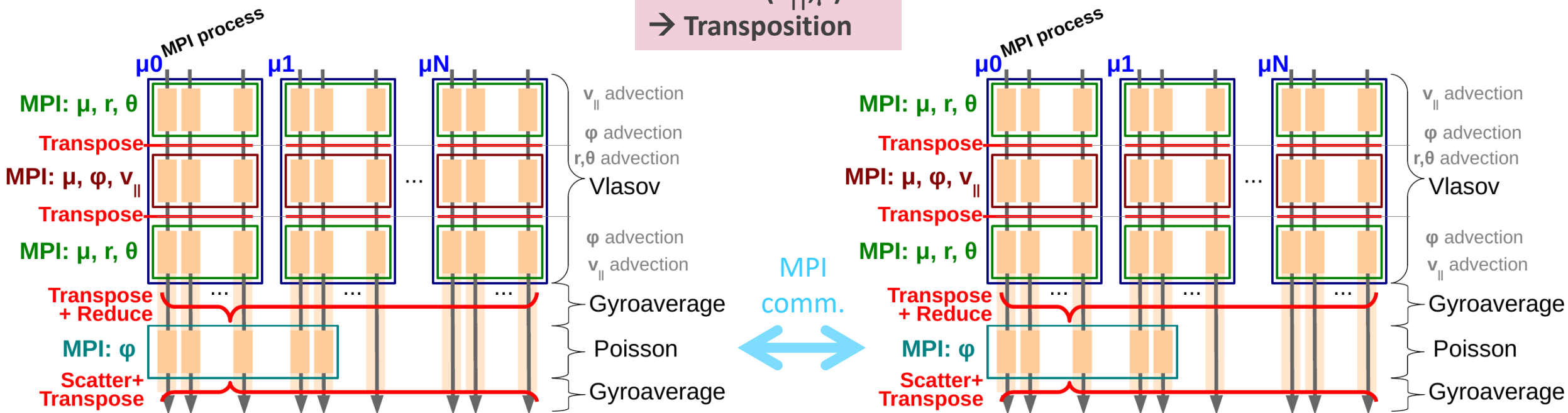
3D Poisson solver (Finite Differences in  $r$  + Fourier in  $(\theta, \varphi)$ )

$$\frac{e}{T_{e,\text{eq}}} (\phi - \langle \phi \rangle) - \frac{1}{n_{e0}} \sum_s Z_s \nabla_{\perp} \cdot \left( \frac{n_{s,\text{eq}}}{B \Omega_s} \nabla_{\perp} \phi \right) = \frac{1}{n_{e0}} \sum_s Z_s \int J_0 \cdot (\bar{F}_s - \bar{F}_{s,\text{eq}}) d^3 v$$

Species 1 : main ion

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

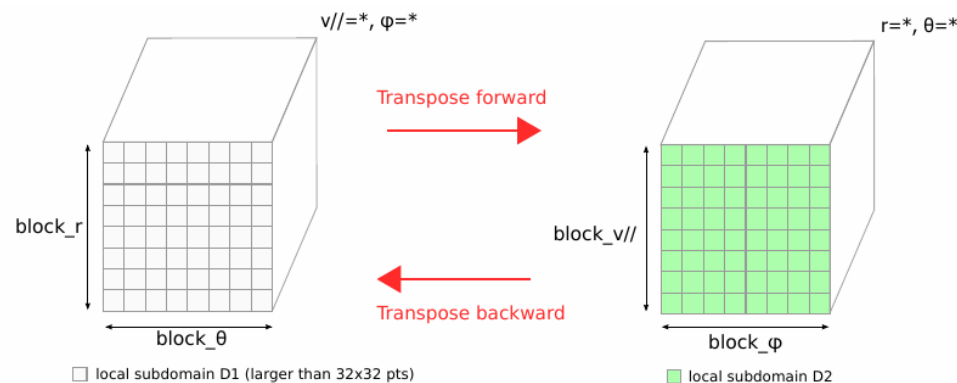
Species 2 : kinetic electrons



→ 3 transpositions per iterations:

▶ large communication amount 😞:

$$\Theta((N_r N_\theta) N_\phi N_{v_{||}} N_\mu)$$



- 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

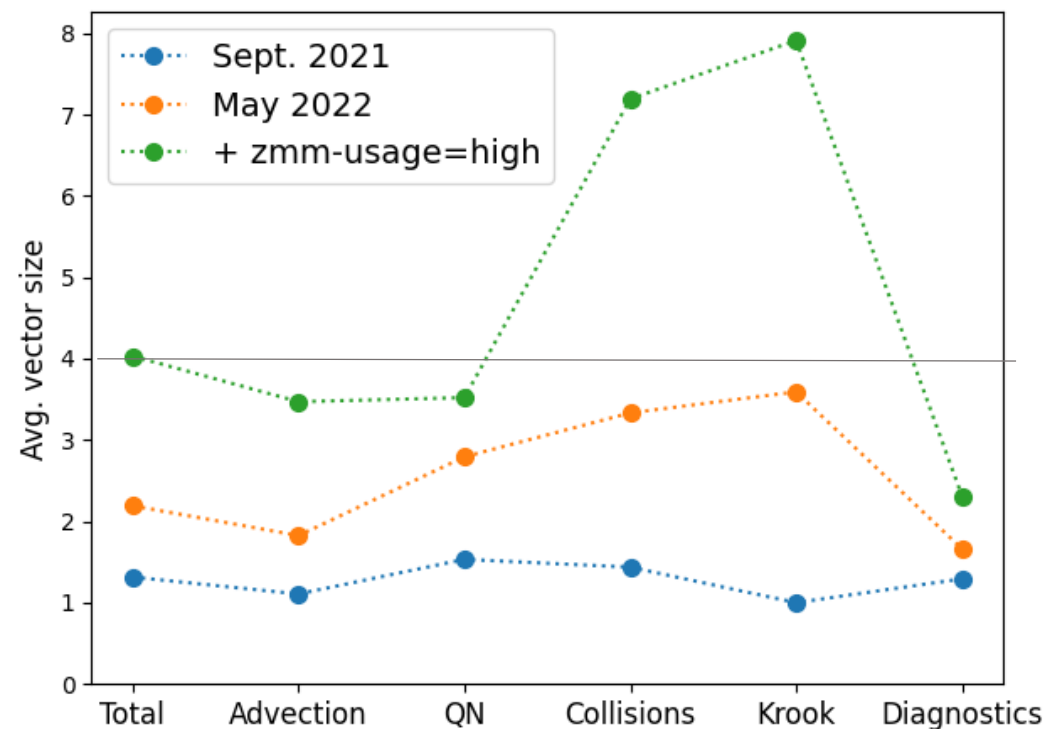
- **May 2022: All kernel vectorisation improved**

- Intel compiler uses preferentially AVX2 (able to activate turbo mode) while AVX512 reduces CPU frequency

- 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



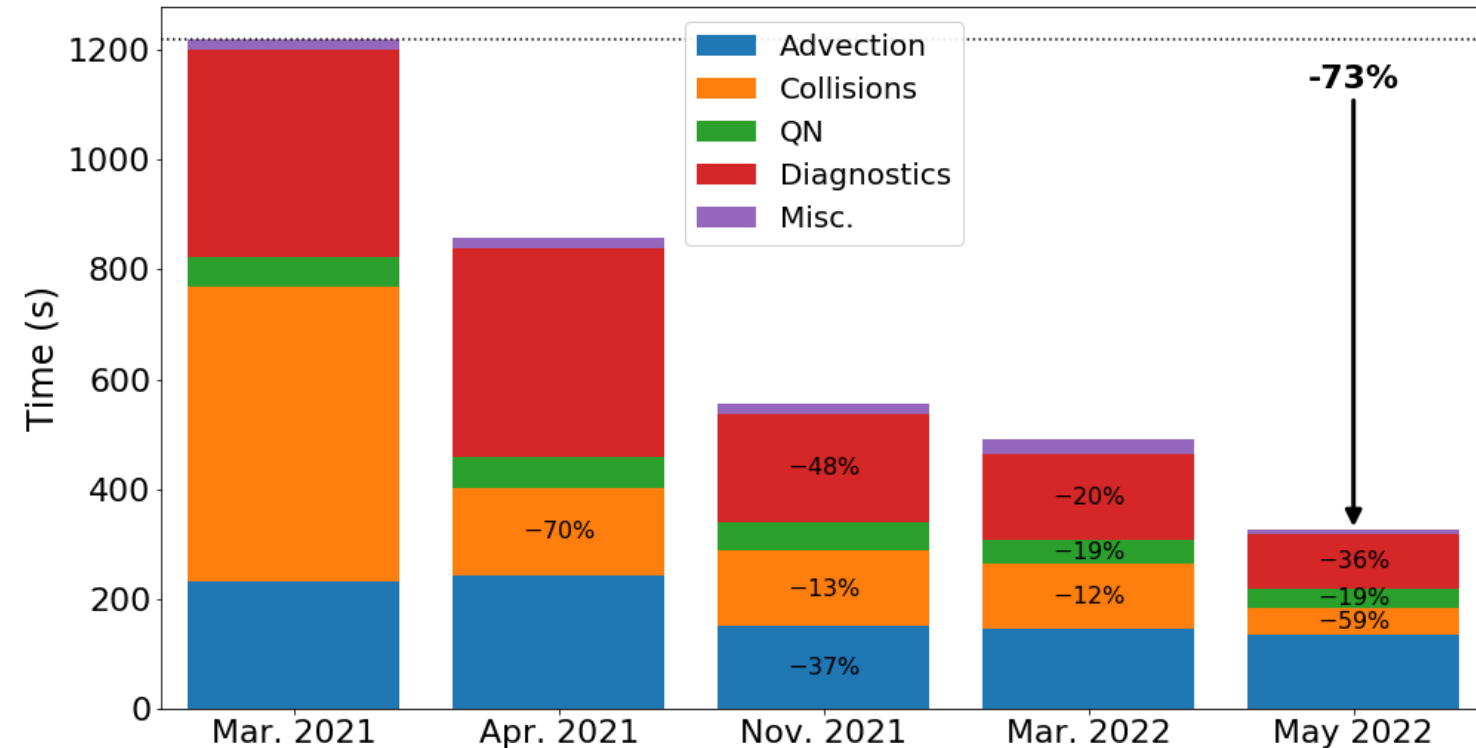
## Numerical improvements in 2021-2022

- ▶ Complete refactoring of collision operator
- ▶ More complex gyroaverage operator (to take 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

Performance gains in GYSELA (Marconi, 384 MPI x 24 OMP)



CPU gain around 70% percent

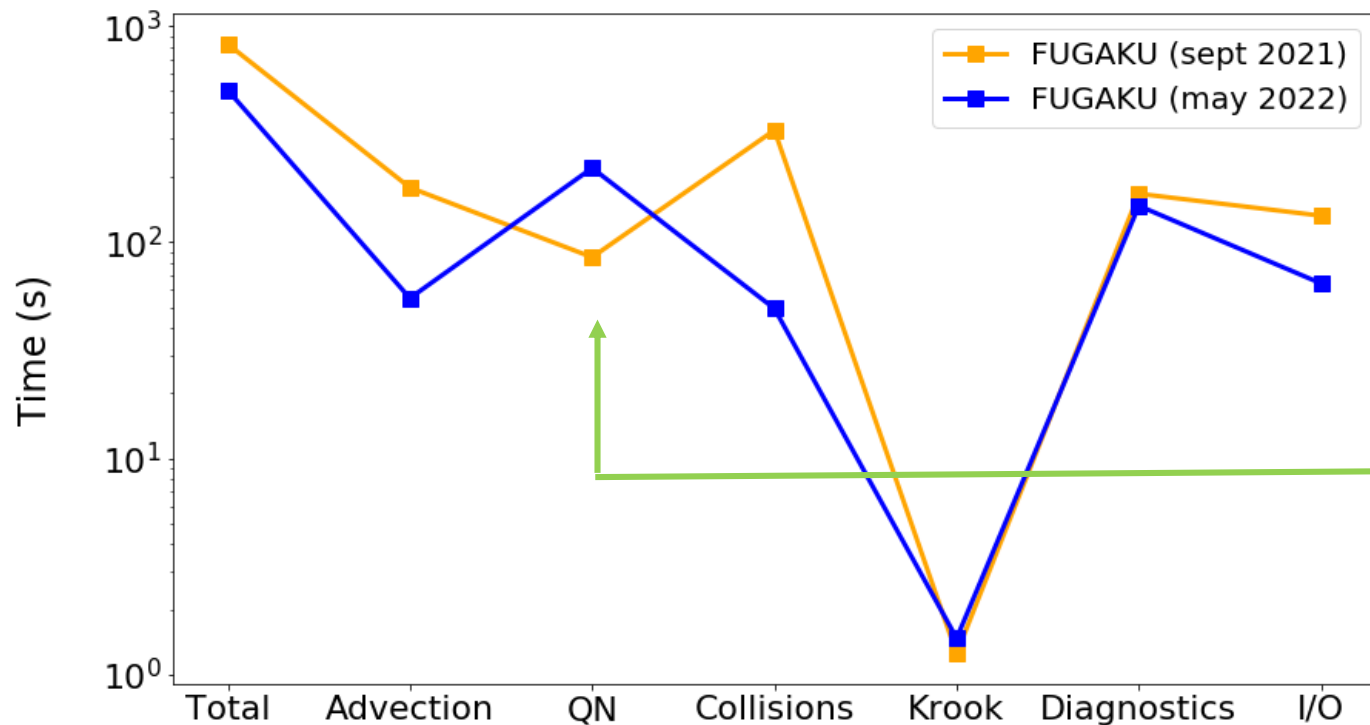
*Kevin Obrejan, May 2022*

► 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)

FUGAKU improvement from sept. 2021 to may 2022  
MPI = 512, Nthreads = 12, mesh5D = 512X1024X64X128X4

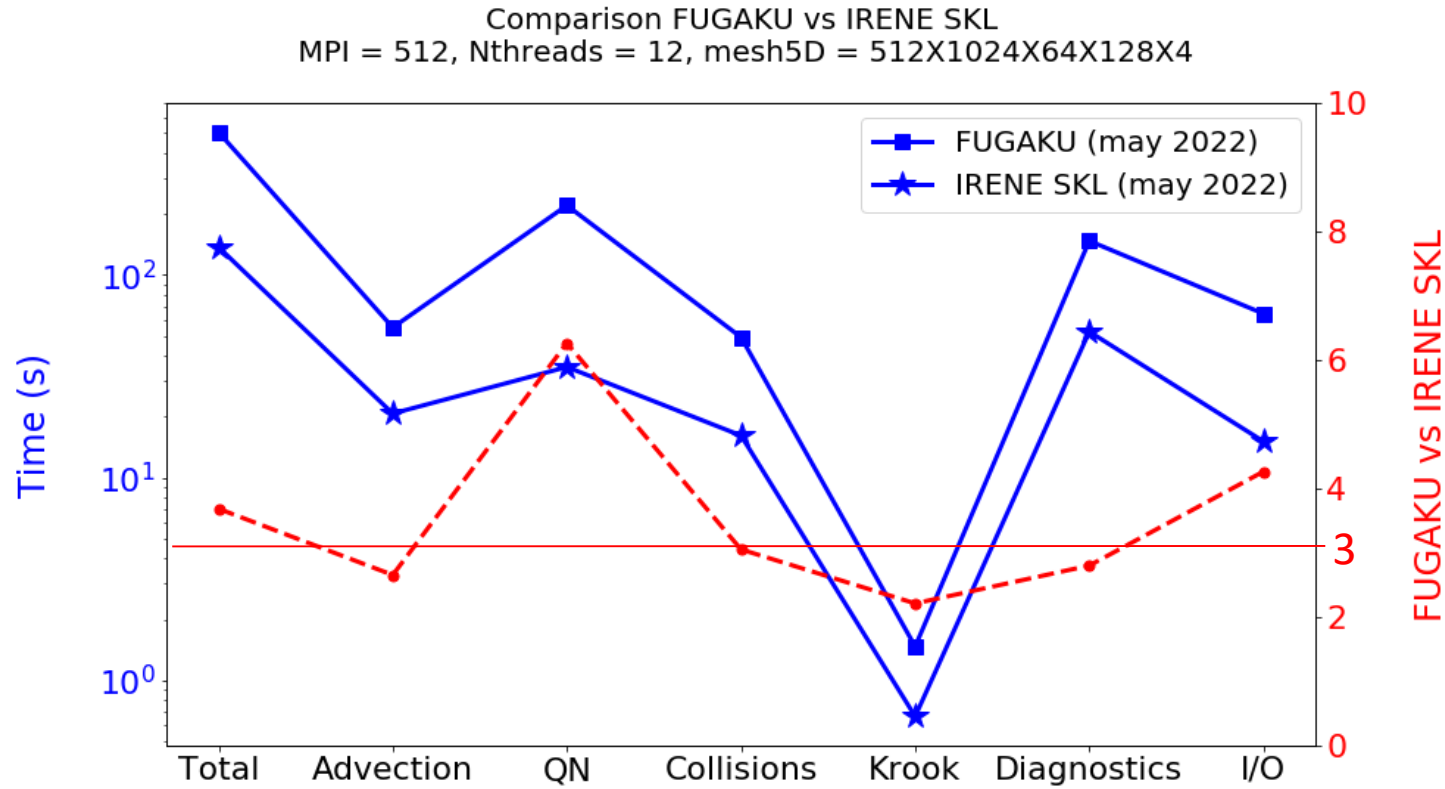


- CEA/Riken collaboration

*(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

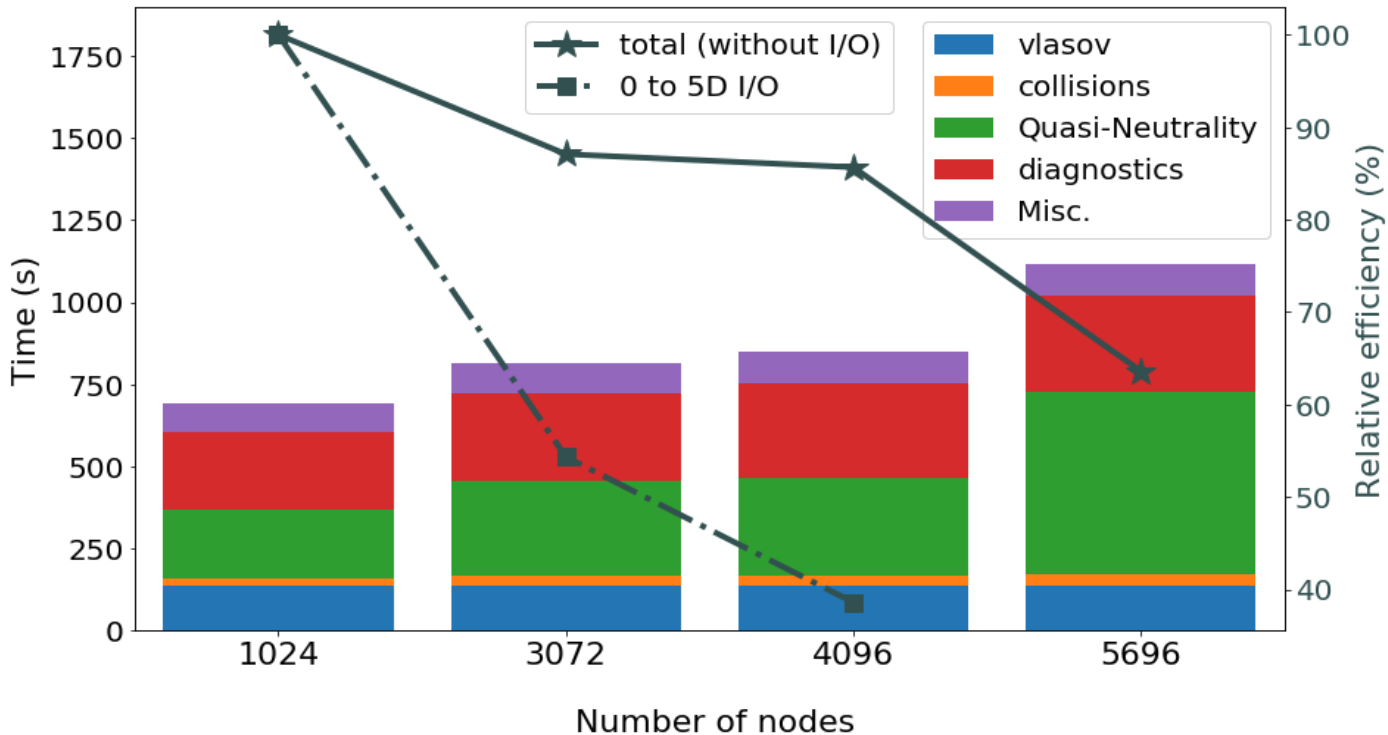


- ▶ Except for QN solver, results better compared to 2021 but still a **factor ~3 slower on FUGAKU vs IRENE SKL**
- ▶ Still lot of work to obtain good performance on Fujitsu A64FX
  - difficult without developing specific version of the kernels
- ▶ Preliminary results show **better performance on AWS Graviton3** (Arm Neoverse V1 cores)



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

Weak scaling of GYSELA on CEA-HF



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:  $\sim 2.75 \cdot 10^{11}$  points
- 2048 nodes  $\Rightarrow$  **262 144 CPU cores**
- 60 k iterations  $\Rightarrow$  **40 millions CPU hours**
- **30 TB of stored data** ( $\sim 110\ 000$  files)

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

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

*Laurent Nguyen, June 2022*

- ▶ In GYSELA-X, all diagnostics are saved in HDF5
- ▶ Huge amount of data that can be stored
  - **1 distribution function  $f_{\text{species}} \sim 1\text{TBytes}$**
  - Not possible to store the time evolution of  $f_{\text{species}}$
  - **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

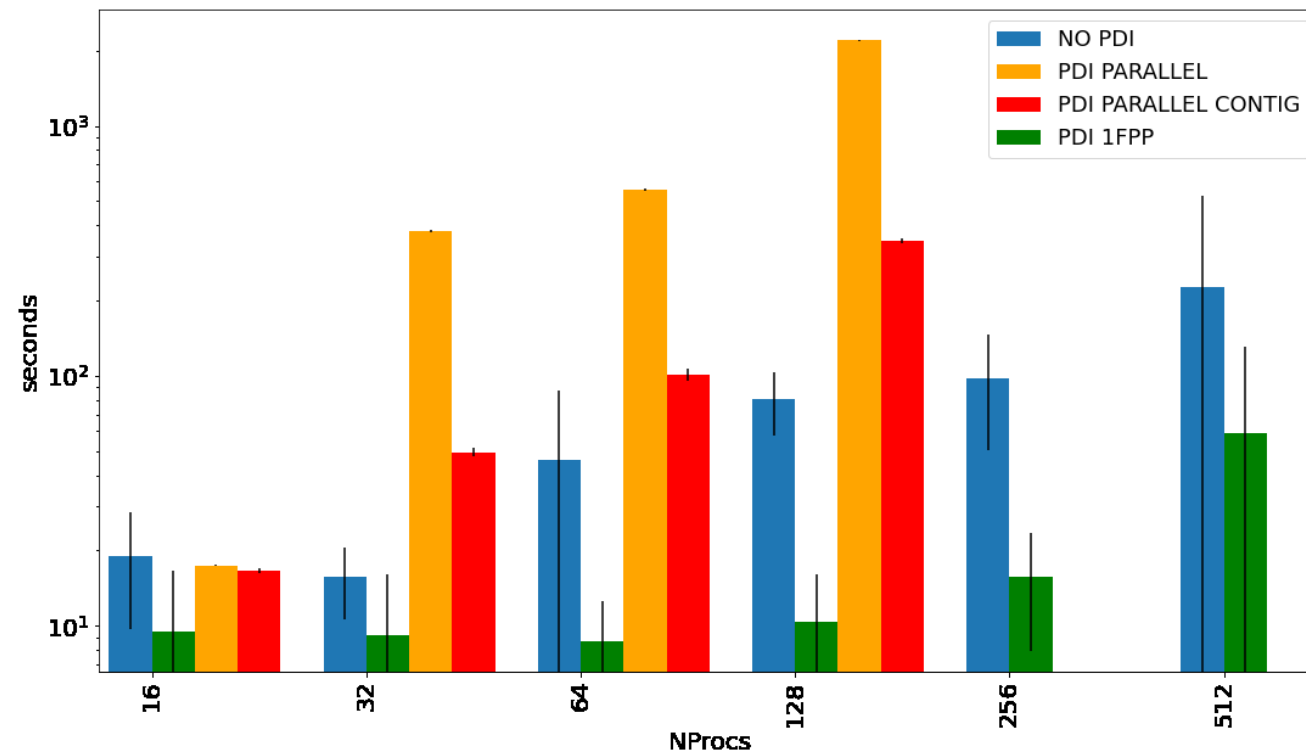
**1 simulation:**

- **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

- ▶ 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

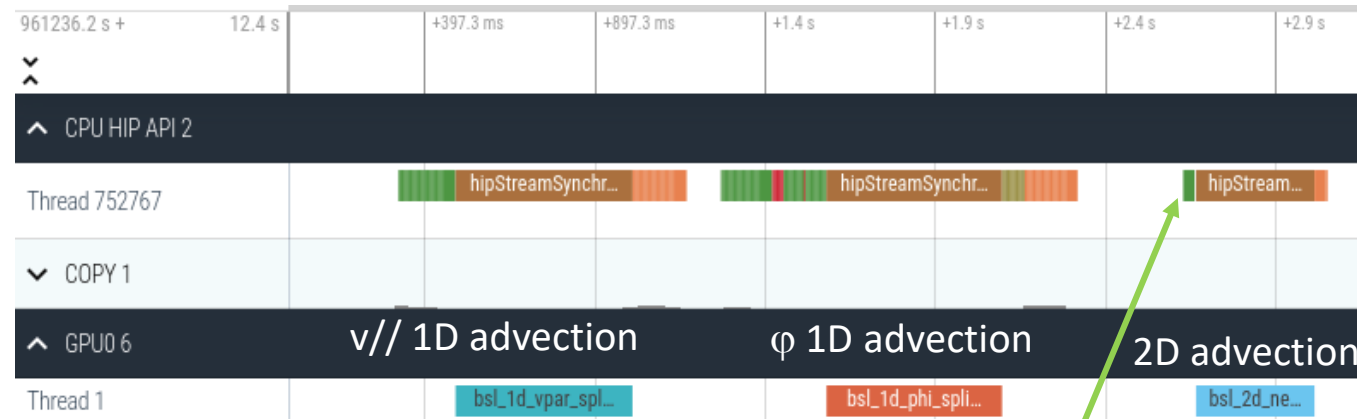
*J. Bigot (MDIS/France) + Y. Ould-Ruis (INRIA-Grenoble/France)*



- ▶ 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*



► 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

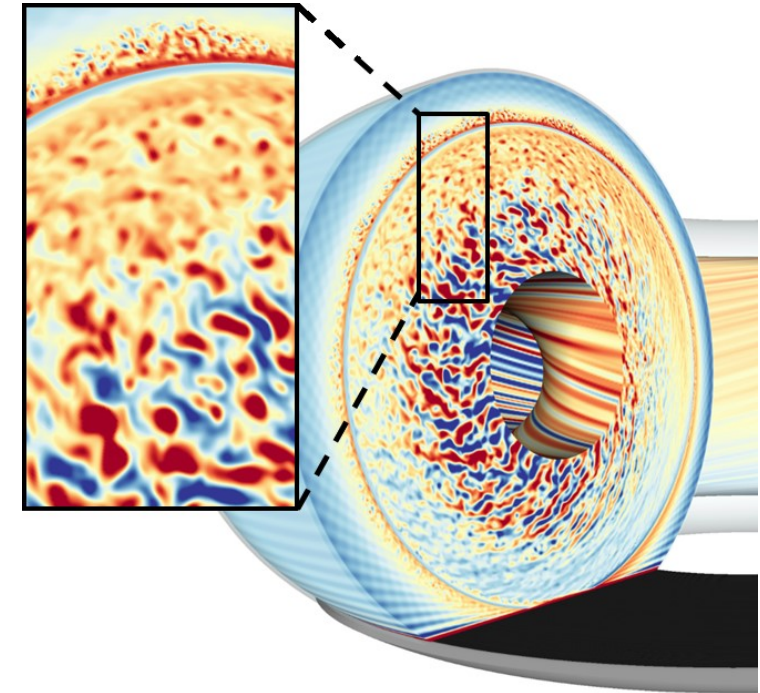
GYSELA-X strategy : a unique CPU/GPU code

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

- ▶ 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, ...)**



GYSELA-X

Voice++

Gyselalibxx

*physics/math 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)*



GYSELA-X

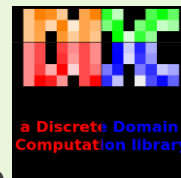
Voice++

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

*parallel loops*

mdspan

*arrays*

...

PDI

*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. Munsch, in preparation 2022]*

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

**GYSELA-X**

**Voice++**

**Gyselalibxx**

*physics/math related operators*

**DDC**

A Discrete Domain Computation library

*metadata : discretizations, distributions...*

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



**Kokkos**  
*parallel loops*

**mdspan**  
*arrays*

...

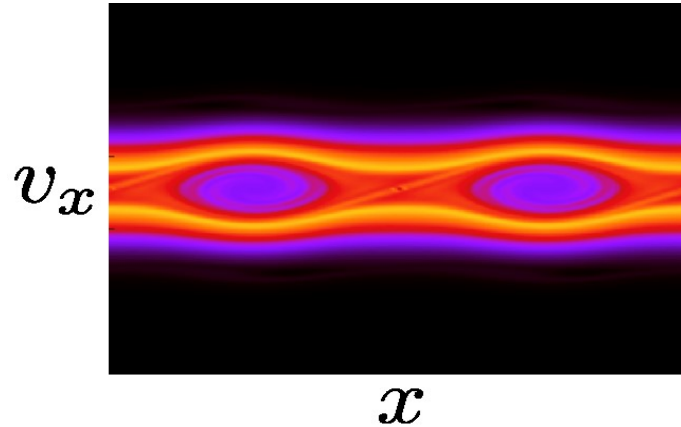
**PDI**  
*I/O*

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

## 4D vlasov + 2 Poisson mi-app



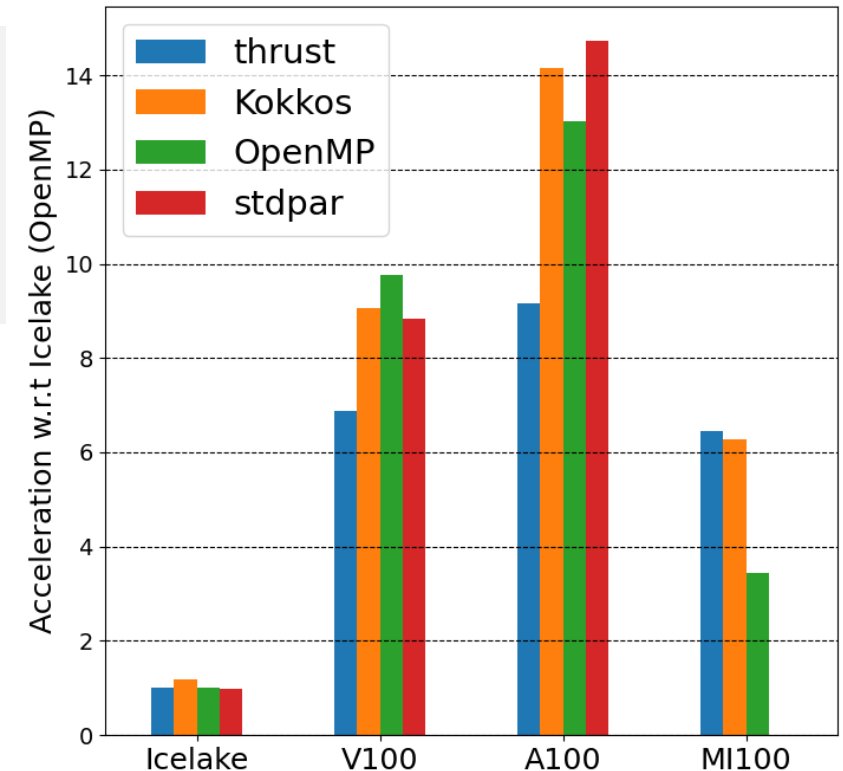
Icelake: Intel Xeon Platinum 8360 x 2 (icpc 2021. 2, nvc++ 22.5)  
 V100: NVIDIA V100 x 2 (nvc++ 22.3)  
 A100: NVIDIA A100 x 2 (nvc++ 22.5)  
 MI100: AMD MI100 x 2 (rocm 5.0.2)

- ▶ 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

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



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

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

- ▶ 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 rewriting of kernels)
  - Porting on GPU started 9 months ago and is still in progress
  
- ▶ GYSELA-X rewriting started in modern C++ based on DDC