



HAL
open science

How to prepare the GYSELA-X code to future exascale edge-core simulations

Virginie Grandgirard, Y Asahi, J Bigot, E Bourne, Guilhem Dif-Pradalier, P Donnel, X Garbet, Ph. Ghendrih, Yaman Güçlü, K Kormann, et al.

► To cite this version:

Virginie Grandgirard, Y Asahi, J Bigot, E Bourne, Guilhem Dif-Pradalier, et al.. How to prepare the GYSELA-X code to future exascale edge-core simulations. PASC 2021 - The Platform for Advanced Scientific Computing Conference, Association for Computing Machinery (ACM); the Swiss National Supercomputing Centre (CSCS), Jul 2021, Genève - E-Conference, Switzerland. cea-03347970

HAL Id: cea-03347970

<https://cea.hal.science/cea-03347970v1>

Submitted on 17 Sep 2021

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.

How to Prepare the GYSELA-X Code to future Exascale Edge-Core Simulations

Virginie GRANDGIRARD



DE LA RECHERCHE À L'INDUSTRIE



Y. Asahi^[2], J. Bigot^[3], E. Bourne, G. Dif-Pradalier, P. Donnel^[4],
X. Garbet, Ph. Ghendrih, Y. Güçlü^[5], K. Kormann^[5], D. Midou,
Y. Munschy, K. Obrejan, Ch. Passeron, R. Varennes, Y. Sarazin

CEA, IRFM, 13108 Saint-Paul-lez-Durance Cedex, France

[2] *CCSE, Japan Atomic Energy Agency, 178-4-4 Wakashiba, Kashiwa, Chiba, Japan*

[3] *Maison de la Simulation, CEA, CNRS, Univ. Paris-Sud, UVSQ, Université Paris-Saclay, 91191 Gif-sur-Yvette, France*

[4] *SPC - Théorie. EPFL. PPB 315, CH-1015 Lausanne, Switzerland.*

[5] *IPP, Boltzmannstraße 2, 85748 Garching, Germany.*

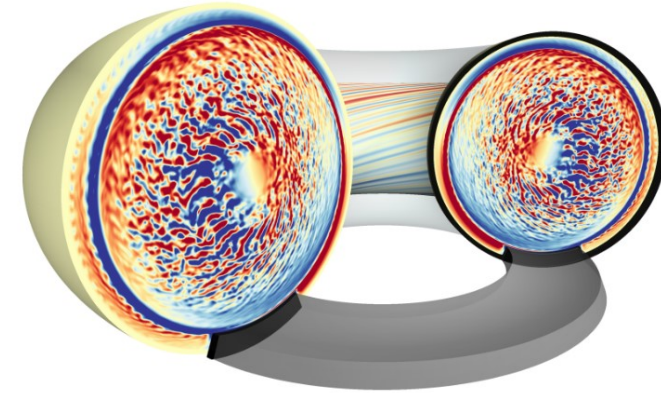
■ GYSELA-X a 5D gyrokinetic code based on a **backward semi-Lagrangian scheme**

■ 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)

[Grandgirard et al., CPC 2016]

<https://gyselax.github.io/>



HPC needs: 100 Mh/year

■ GYSELA-X current physics interests:

- Impact of magnetic configuration on turbulence
- **Adding of a more realistic geometry in GYSELA-X**
- Core-edge turbulence:
 - **How to numerically treat more realistic SOL**
- Ripple
- Impurity transport

[post-doc: K. Obrejan 2020-2021] +
 [post-doc: P. Donnel sept 2021-2022]
 [PhD: E. Bourne 2019-2022] +
 [PhD: Y. Munsch 2021-2024]
 [PhD: R. Varennes 2019-2022]
 [PhD: K. Lim 2019-2022]

- ▶ New numerical methods
 - Adding of more realistic geometry in the GYSELA-X code
 - Validation of semi-Lagrangian scheme with non-equidistant splines in VOICE (GYSELA mini-application)

 - ▶ Improvement of the parallelisation
 - Optimisation of the collision operator
 - Optimisation of I/O : implementation of PDI in GYSELA-X

 - ▶ Porting on new architecture for preparing to exascale simulations
 - First test on GPU with mini-applications (OpenACC/OpenMP or KOKKOS)
 - First feedback for GYSELA-X porting on ARM machines (FUGAKU/Japan + TGCC/France)
- Do we need to rewrite GYSELA-X ? If yes, which choice ?

- ▶ New numerical methods
 - Adding of more realistic geometry in the GYSELA-X code
 - Validation of semi-Lagrangian scheme with non-equidistant splines in VOICE (GYSELA mini-application)

 - ▶ Improvement of the parallelisation
 - Optimisation of the collision operator
 - Optimisation of I/O : implementation of PDI in GYSELA-X

 - ▶ Porting on new architecture for preparing to exascale simulations
 - First test on GPU with mini-applications (OpenACC/OpenMP or KOKKOS)
 - First feedback for GYSELA-X porting on ARM machines (FUGAKU/Japan + TGCC/France)
- Do we need to rewrite GYSELA-X ? If yes, which choice ?

- **Culham equilibrium:** axisymmetric, second order accurate $O((r/R_0)^2)$ solution of the Grad-Shafranov equation

$$R = R_0 + \Delta(r) + (r - E(r) - P_{corr}(r))\cos(\theta) + T(r)\cos(2\theta)$$

$$Z = (r + E(r) - P_{corr}(r))\sin(\theta) - T(r)\sin(2\theta)$$

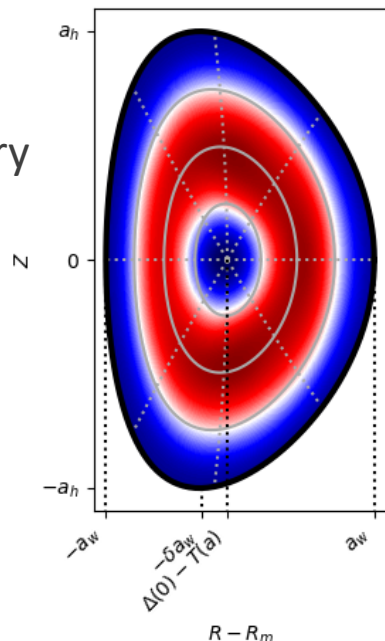
$E \rightarrow$ elongation, $T \rightarrow$ triangularity,
 Δ : Shafranov shift, P_{corr} : flux surface relabelling

- SELALIB^[1] Poisson solver recently coupled to GYSELA

^[1] <https://selalib.github.io/selalib/>

- First tests on **GAM simulations with full-kinetic electrons for ITER-like geometry** successful

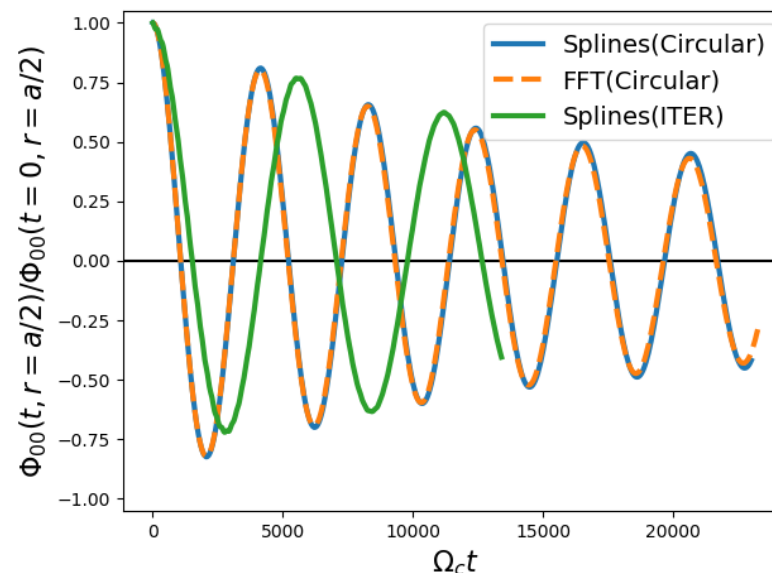
\rightarrow theoretical pulsation proportional to $\frac{1}{\sqrt{1+\kappa^2}}$ \rightarrow ratio of 0.72 coherent with measured 0.74



ITER-like geometry
 $(\kappa=1.7, \delta=0.4)$

$$E(a) \sim 0.26 * a$$

$$T(a) \sim 0.075 * a$$



Good agreement in circular geometry between new solver (2D splines) and old one (finite difference in r and FFT in θ)

- 2D spline solver slower due to conjugate gradient iterations
 \rightarrow Search of a better preconditionner still under investigation

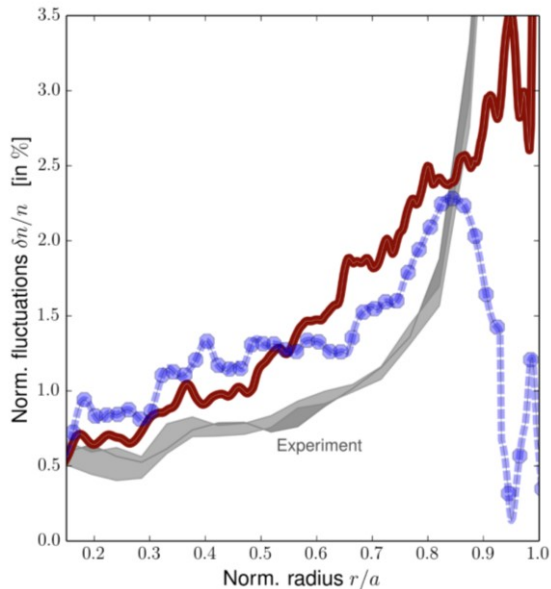
E. Bourne, K. Obrejan, X. Garbet, Ken Leleux + Y. Güçlü, K. Kormann (IPP Garching)

- ▶ **Flux-driven ITG turbulence** (adiab. electrons)
 $1/\rho^* = 250$, $\nu^* = 0.14$, global domain: $0 < \rho < 1.3$

- ▶ PRACE 2020 simulation (TGCC/Irene ROME):

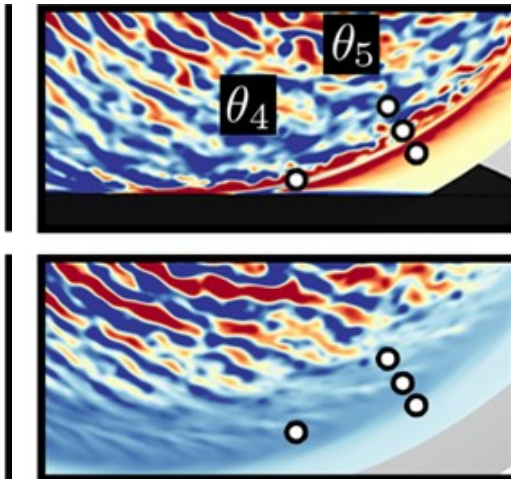
- ✓ 5D mesh \sim 100 billions of points
- ✓ \sim 6 millions of hours (\sim 20 days on 12k cores)
- ✓ Several Tbytes of data

- ▶ Strong impact of boundary conditions on turbulence

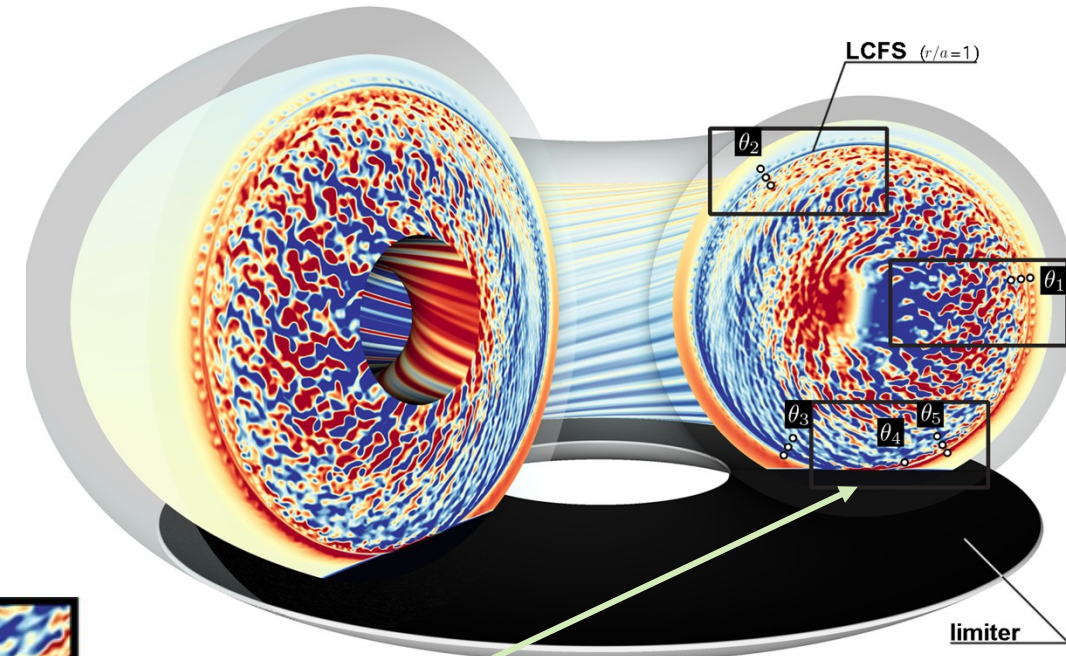


Flux-driven &
limiter b.c.
Case-1

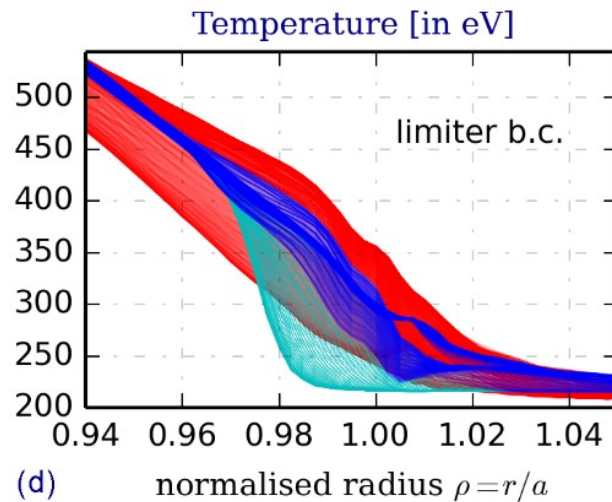
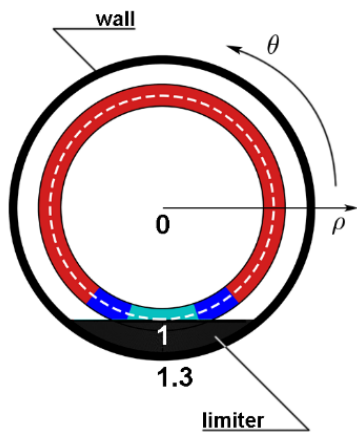
Flux-driven &
polo. symm b.c.
Case-2



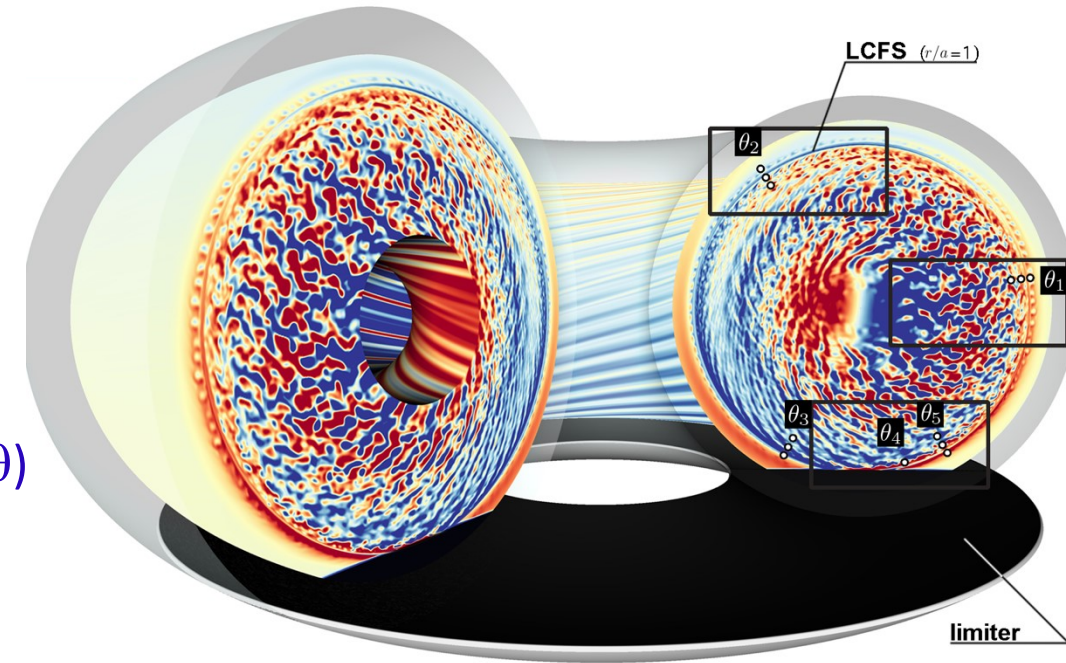
[Dif-Pradalier 2020]



- ▶ **Flux-driven ITG turbulence** (adiab. electrons)
 $1/\rho^* = 250$, $v^* = 0.14$, global domain: $0 < \rho < 1.3$
- ▶ PRACE 2020 simulation (TGCC/Irene ROME):
 - ✓ 5D mesh **~ 100 billions of points**
 - ✓ **~ 6 millions of hours** (~20 days on 12k cores)
 - ✓ Several Toctets of data
- ▶ **Strong density & temperature gradients close to the LCFS ($\nabla\theta$)**



[Dif-Pradalier 2020]



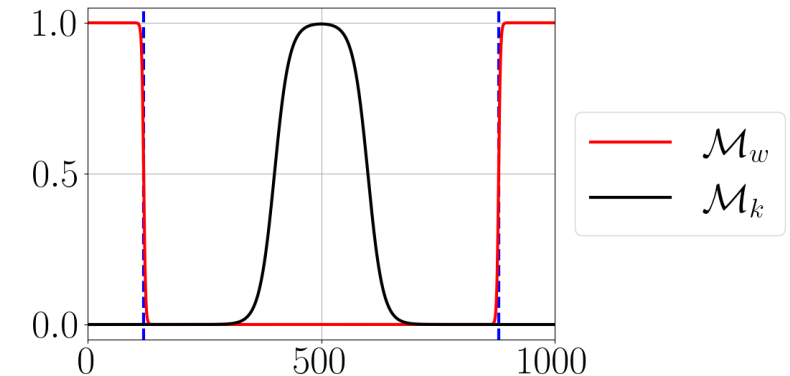
- ▶ **How to treat more realistic temperature gradients ? (2 order of magnitude) → non-equidistant mesh**

- Solves normalised 1D-1D Vlasov-Poisson equation for particles of species s , mass m_s

$$\frac{\partial f_s}{\partial t} + \sqrt{\frac{m_e}{m_s}} (v_s \partial_x f_s + q_s (E_{ext} - \partial_x \phi) \partial_{v_s} f_s) = \nu_w \mathcal{M}_w (f_s - g_w) + \mathcal{M}_k S_v(v_x)$$

$$\partial_x^2 \phi = \frac{e}{\epsilon_0} (n_i - n_e)$$

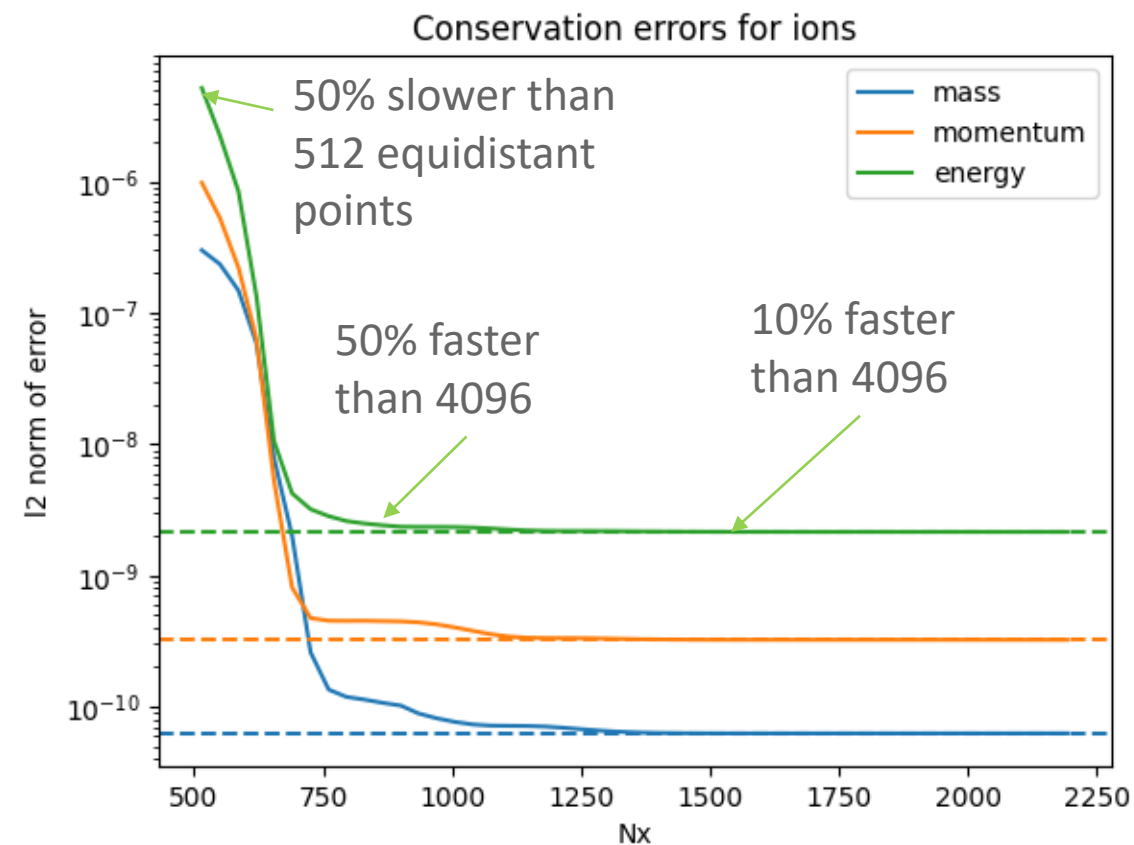
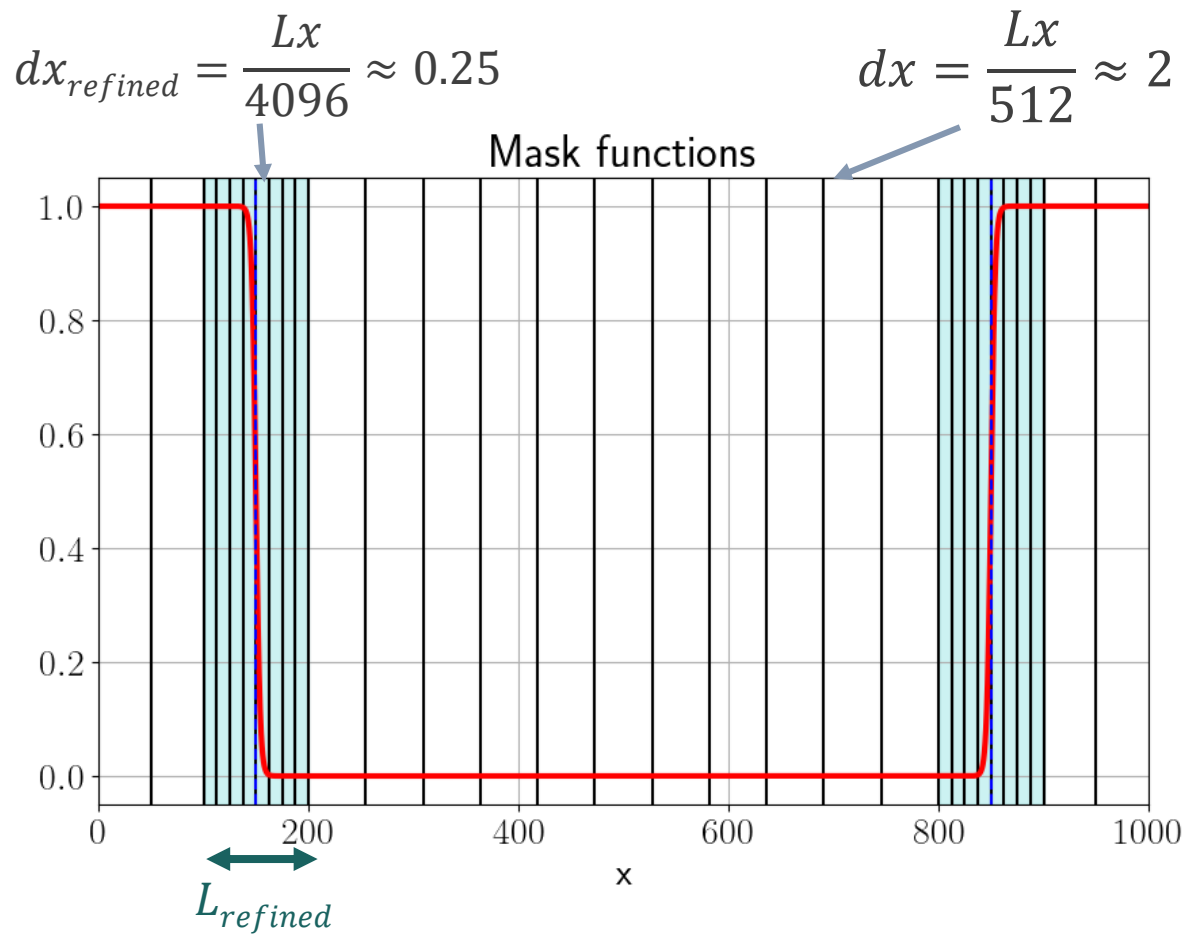
ν_w is a measure of the restoring force which is localised at the mask \mathcal{M}_w
 g_w is a Maxwell-Boltzmann distribution function



- Periodic in x , non-periodic in v
- Construct with GYSELA modules → Same numerical schemes
 - Backward semi-Lagrangian scheme (BSL)
 - 1D advection
 - Spline interpolation
 - Poisson solver (FEM or FD or FFT)
 - **Same penalization technique**

VOICE	GYSELA
2D (1D in x + 1D in v)	5D (3D in x + 2D in v)
No parallelisation	MPI + OpenMP
Fortran + C++	Fortran + few C modules
10k lines	50k lines

- Testbed for BSL with non-equidistant splines before implementation in GYSELA-X code



$dx_{refined}$, dx fixed and $0.5 < L_{refined} < 235$.

- Accuracy: same error with 1500 non-equidistant points and 4096 equidistant points.
- Non-equidistant points are slower as a binary search is needed to find the relevant basis spline and calculations cannot be simplified.

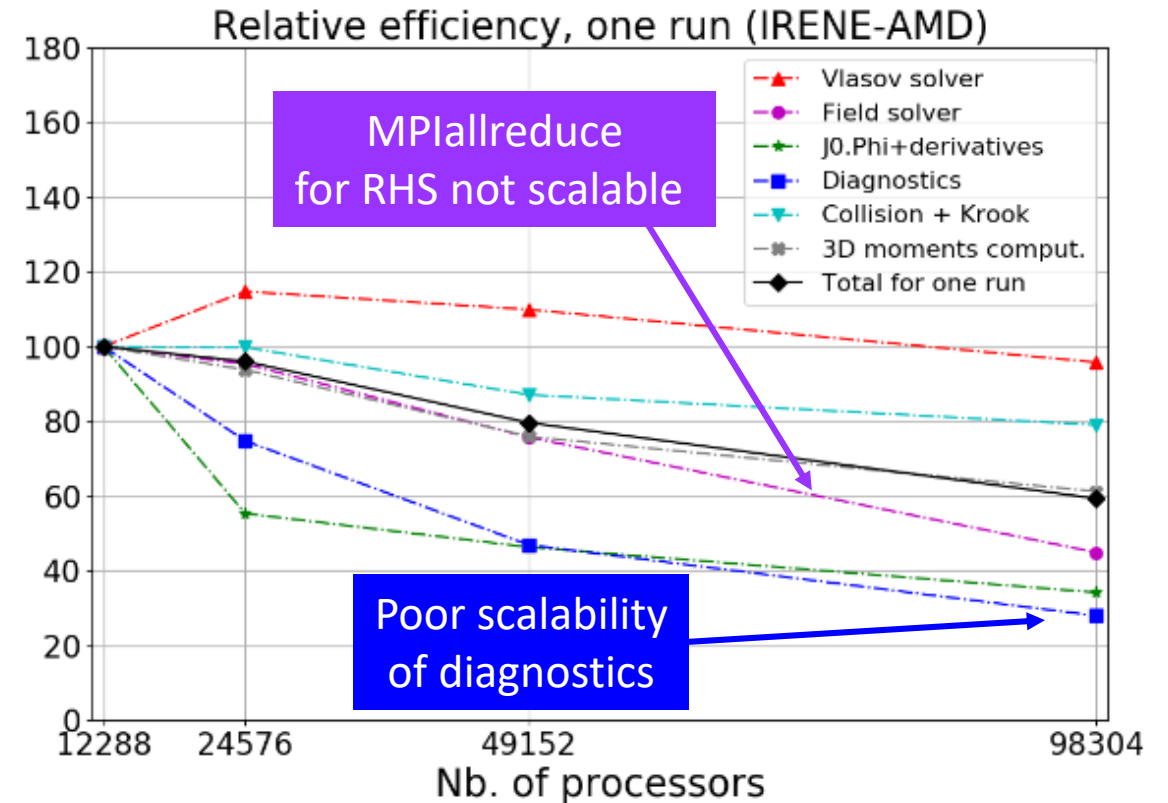
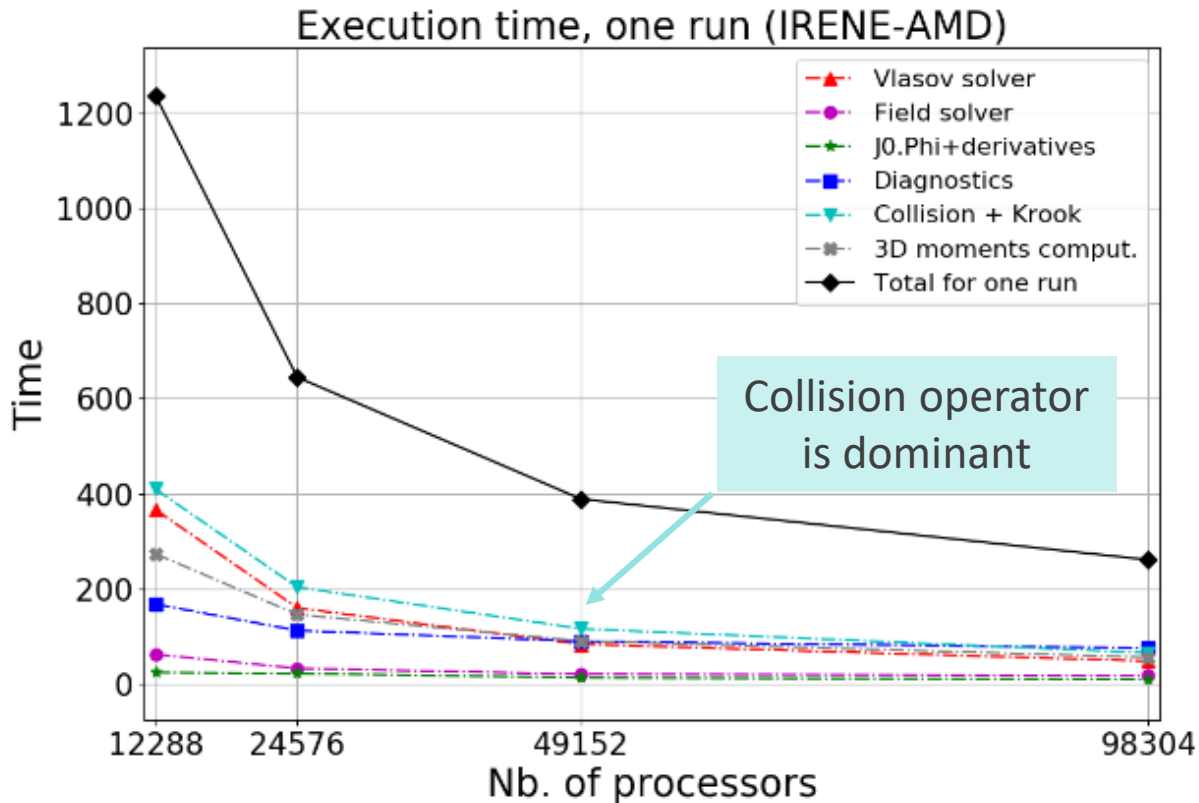
E. Bourne, Y. Munsch, Ph. Ghendrih, G. Dif-Pradalier, Y. Sarazin + Y. Güçlü (IPP Garching)

- ▶ New numerical methods
 - Adding of more realistic geometry in the GYSELA-X code
 - Validation of semi-Lagrangian scheme with non-equidistant splines in VOICE (GYSELA mini-application)

 - ▶ Improvement of the parallelisation
 - Optimisation of the collision operator
 - Optimisation of I/O : implementation of PDI in GYSELA-X

 - ▶ Porting on new architecture for preparing to exascale simulations
 - First test on GPU with mini-applications (OpenACC/OpenMP or KOKKOS)
 - First feedback for GYSELA-X porting on ARM machines (FUGAKU/Japan + TGCC/France)
- Do we need to rewrite GYSELA-X ? If yes, which choice ?

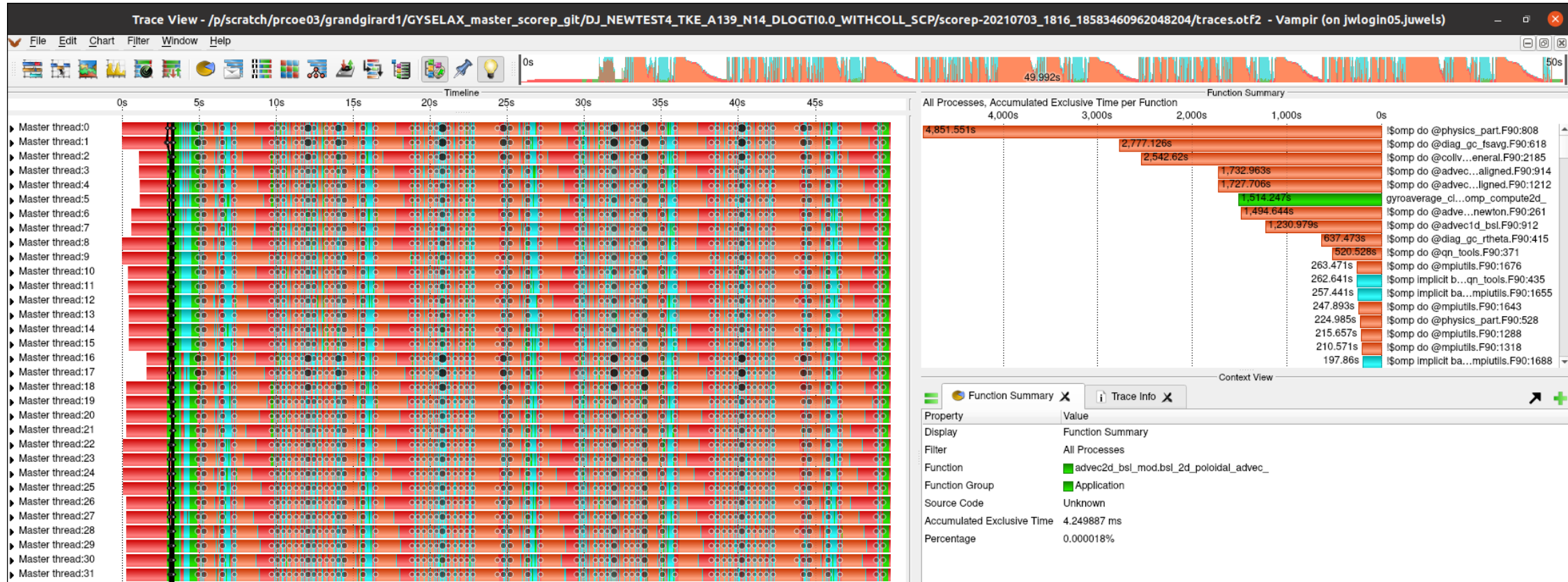
- GYSELA : MPI+OpenMP parallelism : **80% relative efficiency on 49,152 cores** and 59% on 98,304 cores



« Strong scaling » of GYSELA code on Irene-AMD partition at TGCC-France (2020)

- GYSELA : ~25% faster on IRENE-AMD than on IRENE-SKL. 2.3 times faster on IRENE-SKL than on IRENE-KNL

- ▶ Good load balancing due to eulerian nature of the semi-Lagrangian scheme (fixed grid in time)



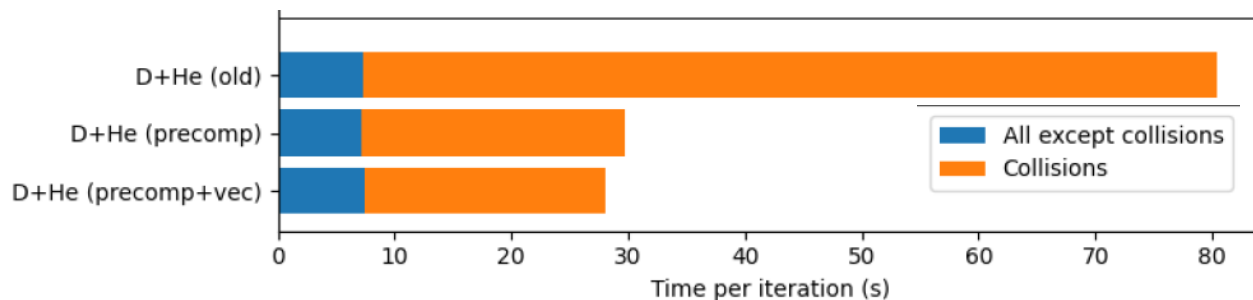
- ▶ Collision operator main drawback : Huge MPI communication to have all ($v_{//}, \mu$, all species) required data can't be avoided in the current MPI domain decomposition but can we improve the computation part ?

- ▶ Mid-term objective: Decouple diagnostics computation from computation kernels → First step: Use of PDI

- ▶ With Score-p instrumentation on, in a simulation with one species, collisions ~50% of the total CPU time
 - Half of that taken by I and J (called several times on every point of the mesh at each time step)
 - Overall chain of functions $N_{a_j1} \rightarrow L \rightarrow I \rightarrow J$ very costly

type	max_buf[B]	visits	time[s]	time[%]	time/visit[us]	region
ALL	6,007,893,628	7,267,677,628	11237.66	100.0	1.55	ALL
USR	6,003,057,320	7,262,723,069	7375.43	65.6	1.02	USR
OMP	4,596,680	4,840,064	2830.14	25.2	584.73	OMP
MPI	413,576	73,197	1001.25	8.9	13678.85	MPI
COM	33,956	41,298	30.84	0.3	746.75	COM
USR	449,839,104	553,648,128	1009.56	9.0	1.82	collvparamu_general_class.collvparamu_general_j_
USR	449,839,104	553,648,128	1688.50	15.0	3.05	collvparamu_general_class.collvparamu_general_i_
USR	436,208,500	528,483,392	164.61	1.5	0.31	utils_mod.utl_is_zero_
USR	436,207,746	528,482,464	367.15	3.3	0.69	utils_mod.utl_locate_
	●	●	●			
USR	224,919,552	276,824,064	277.10	2.5	1.00	collvparamu_general_class.collvparamu_general_n_a_j1_
USR	224,919,552	276,824,064	524.05	4.7	1.89	collvparamu_general_class.collvparamu_general_l_

- ▶ Rewritten of the innermost loop to add more vectorization



- ▶ CPU time gain of 1.9 (for 1 species) and 3.6 (for two species)

- ▶ PDI offers to exchange data between the application code and various external data handlers, such as for example the file-system for I/O or another code for code-coupling.

- ▶ **New version of PDI recently implemented in GYSELA code**

- Instrumentation in the code via PDI_share, PDI_expose, PDI_reclaim

```
typedef enum { PDI_IN, PDI_OUT, PDI_INOUT } PDI_inout_t;

// A data buffer is ready (filled)
PDI_status_t PDI_share(const char *name, void *data, PDI_inout_t access);

// A buffer will be reused
PDI_status_t PDI_reclaim(const char *name);
```

- Adding of a YAML configuration file for description of all the I/O data
- HDF5 plugin choice used for diagnostics saving (~100 Gbytes)

```
plugins:
  decl_hdf5:
    - file: sp${pSpecies_id}/rprof_part_d${iter_diag:05d}.h5
      on_event: physics_part_save
```

- Restart files (several Tbytes) saved in HDF5 but can be saved in FTI or SionLib by just changing the plugin

- ▶ No extra cost for diagnostics as expected → First step before externalisation in Python

- ▶ **Gain of a factor 10 for restart file writting** still under investigation (temporary copy avoided + no checksum)

- Big advantage: I/O optimization will be treated by specialists

- ▶ PDI is a simple C/C++ API also available for Fortran and Python

```
/** Initializes PDI */
PDI_status_t PDI_init(PC_tree_t yaml_conf);

/** Finalizes PDI */
PDI_status_t PDI_finalize();
```

```
# -> for toroidal angular momentum conservation
# - Ltor : toroidal angular momentum
# - Tpol_psi : polarization term
# - Gamma_vE_r : guiding-center turbulent flux
# (due to ExB velocity)
# - Gamma_vEn0_r : n=0 component of Gamma_vE_r
# - Gamma_vD_r : guiding-center neoclassical flux
# - RSphi_vE_r : toroidal Reynolds stress due to ExB
# - RSphi_vEn0_r : n=0 component of RSphi_vE_r
# - RSphi_vD_r : neoclassical toroidal Reynolds stress
LtorGC_FSavg: { type: array, subtype: double, size: '$Nr+1' }
TpolGC_psi_FSavg: { type: array, subtype: double, size: '$Nr+1' }
GammaGC_vE_r_FSavg: { type: array, subtype: double, size: '$Nr+1' }
GammaGC_vEn0_r_FSavg: { type: array, subtype: double, size: '$Nr+1' }
GammaGC_vD_r_FSavg: { type: array, subtype: double, size: '$Nr+1' }
RSphiGC_vE_r_FSavg: { type: array, subtype: double, size: '$Nr+1' }
RSphiGC_vEn0_r_FSavg: { type: array, subtype: double, size: '$Nr+1' }
RSphiGC_vD_r_FSavg: { type: array, subtype: double, size: '$Nr+1' }
```

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

- ▶ New numerical methods
 - Adding of more realistic geometry in the GYSELA-X code
 - Validation of semi-Lagrangian scheme with non-equidistant splines in VOICE (GYSELA mini-application)

 - ▶ Improvement of the parallelisation
 - Optimisation of the collision operator
 - Optimisation of I/O : implementation of PDI in GYSELA-X

 - ▶ Porting on new architecture for preparing to exascale simulations
 - First test on GPU with mini-applications (OpenACC/OpenMP or KOKKOS)
 - First feedback for GYSELA-X porting on ARM machines (FUGAKU/Japan + TGCC/France)
- Do we need to rewrite GYSELA-X ? If yes, which choice ?

- ▶ Participation to the IDRIS GPU Hackathon (4 days – May 2021).
- ▶ Objective: Accelerating VOICE via GPU parallelisation
 - Testing the complexity of the non-equidistant splines on GPU
- ▶ **OpenACC parallelization** of the splines + 1D advections + RHS → **Gain factor of 5 to 8**

No parallelisation

```
Elapsed time in advec x : 40.15840
Elapsed time in advec v : 82.60031
Elapsed time in RHS : 33.78830
Elapsed time in Efield : 8.750244
Elapsed time in diags : 51.91985
Elapsed time in total : 219.0524
```

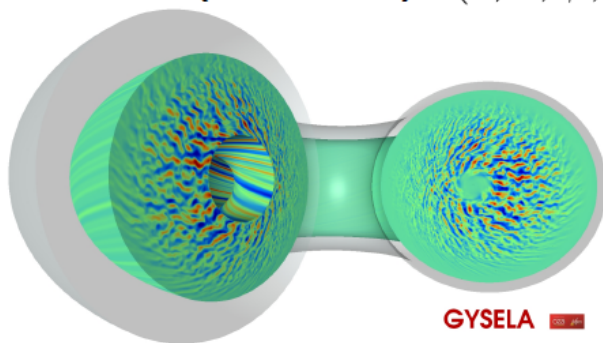
OpenACC parallelisation

```
Elapsed time in advec x : 8.795156
Elapsed time in advec v : 10.38172
Elapsed time in RHS : 0.5890590
Elapsed time in Efield : 9.755074
Elapsed time in diags : 56.86447
Elapsed time in total : 90.60061
```

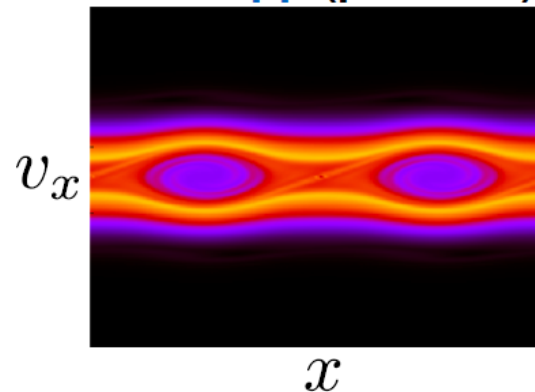
- ▶ First feedback : **OpenACC parallelisation difficult due to** presence of:
 - **derived types + pointers + LAPACK routines** (banded matrix)
- ▶ Not a big surprise: Requires lots of rewriting → This problem will be amplified for GYSELA
- ▶ **GPU porting of GYSELA planned for the next two years**

- Work initiated during Y. Asahi post-doc at IRFM (2017-2018) and pursued by Yuuichi since his return in Japan

GYSELA (3D torus) $(r, \theta, \phi, v_{\parallel}, \mu)$



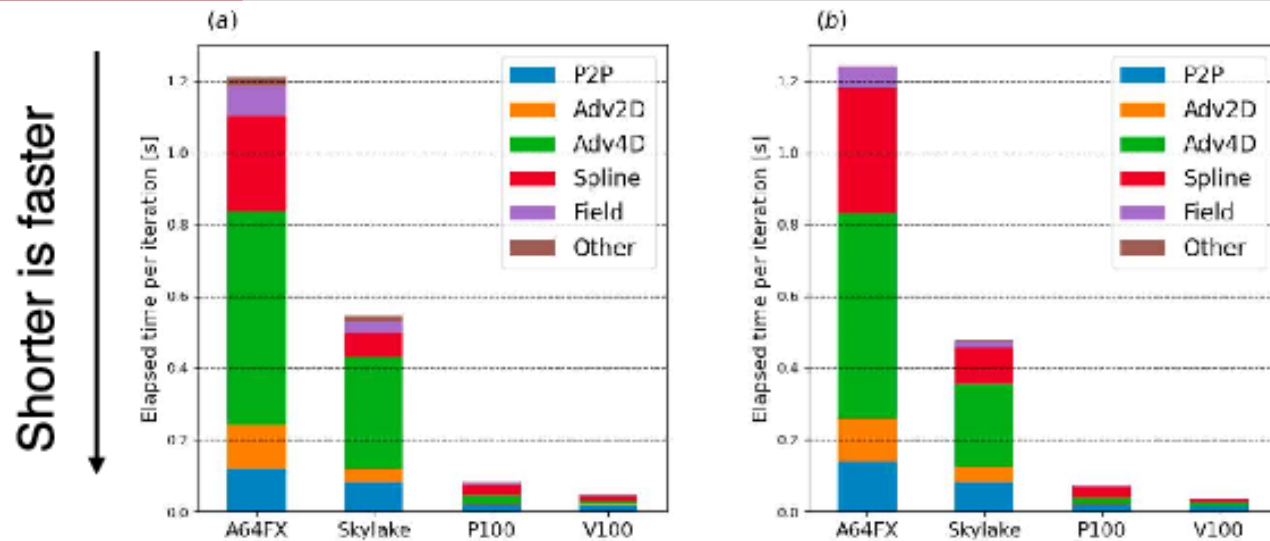
Mini-app (periodic) (x, y, v_x, v_y)



	GYSELA	Mini-app	Mini-app MPI
System	5D Vlasov + 3D Poisson	4D Vlasov + 2D Poisson	4D Vlasov + 2D Poisson
Geometry	Realistic tokamak geometry	Periodic boundary conditions	Periodic boundary conditions
Scheme	Semi-Lagrangian (Spline) + Operator splitting	Semi-Lagrangian (Lagrange) + Operator splitting	Semi-Lagrangian (Spline) without Operator splitting
MPI	Yes	No	Yes
X	OpenMP	OpenACC/OpenMP/Kokkos	OpenACC/Kokkos
Language	Fortran 90	C++	C++
Lines of	More than 50k	About 5k	About 8k

<https://github.com/yasahi-hpc/vlp4d>

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



Kernel	A64FX	Skylake	P100	V100
P2P (Kokkos)	0.117 [s]	0.0812 [s]	0.0189 [s]	0.0179 [s]
Adv2D (Kokkos)	0.125 [s]	0.0379 [s]	0.00275 [s]	0.00179 [s]
Adv4D (Kokkos)	0.592 [s]	0.311 [s]	0.0225 [s]	0.00730 [s]
Spline (Kokkos)	0.269 [s]	0.0673 [s]	0.0277 [s]	0.0163 [s]
Field (Kokkos)	0.080 [s]	0.0307 [s]	0.00729 [s]	0.00267 [s]
Total (Kokkos)	1.211 [s]	0.545 [s]	0.0805 [s]	0.0468 [s]
Kernel	A64FX	Skylake	P100	V100
P2P (OpenMP/OpenACC)	0.140 [s]	0.0820 [s]	0.0191 [s]	0.0140 [s]
Adv2D (OpenMP/OpenACC)	0.118 [s]	0.0425 [s]	0.00350 [s]	0.00137 [s]
Adv4D (OpenMP/OpenACC)	0.573 [s]	0.231 [s]	0.0174 [s]	0.00925 [s]
Spline (OpenMP/OpenACC)	0.348 [s]	0.101 [s]	0.0261 [s]	0.00982 [s]
Field (OpenMP/OpenACC)	0.0594 [s]	0.0155 [s]	0.00470 [s]	0.00260 [s]
Total (OpenMP/OpenACC)	1.241 [s]	0.476 [s]	0.0729 [s]	0.0378 [s]

► Directive based approach : mixed OpenACC/OpenMP

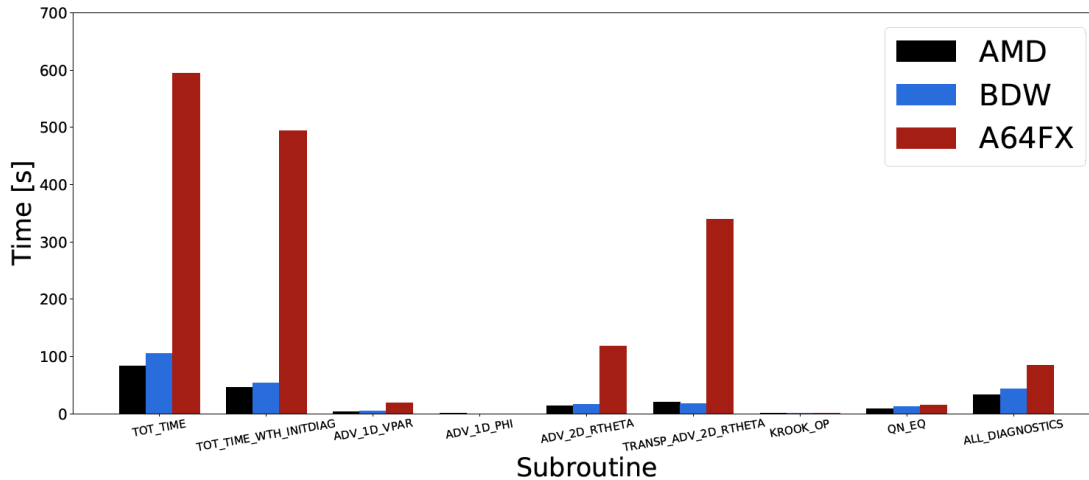
- Mixed OpenACC/OpenMP **achieves high performance** (marginal on A64FX)
- **Suitable for porting a large legacy** code (e.g more than 50k LoC)
- Introducing OpenACC View improves the readability
- SIMD optimizations (like strip-minig) are critical on CPUs

[Y. Asahi et al, to be submitted to P3HPC 2021]

► Higher level abstraction: Kokkos

- **Kokkos can achieve good performance portability** except for A64FX
- Appropriate choice of an execution policy seems critical for CPUs
- Layout and SIMD tunings improve the performance when cache matters

- ▶ Comparison between A64FX (FUGAKU/Japan machine) and AMD (TGCC/France machine)



- ▶ Total time 7 times bigger on A64FX
- ▶ First tests on the ARM partition recently installed at TGCC/France (06-2021) show the same behaviour

- ▶ First analysis with Riken team show problem of compiler optimization:
 - Most of the code is not vectorized due to pointer variables
 - The main loop of the routine advec2d is affected by inline-expansion (A function that is not inline expanded becomes a function call. Fujitsu compiler seems to not vectorize do loops that contains function calls)
- ▶ Investigation will be pursued via CEA/Riken collaboration + TGCC High Level Support Team
- ▶ But currently **do not see how to overcome the problem without a strong refactoring of the code**

Riken/Japan : M. Sato, M. Tsuji, T. Odsuya + Y. Asahi (JAEA/Japan)

- ▶ More realistic geometry has been implemented in GYSELA-X
 - First tests with GAM with FKE successful

- ▶ Generalized non-equidistant splines successfully implemented in VOICE
 - Next step : Implementation in GYSELA-X → not trivial because impact almost all the code

- ▶ PDI successfully implemented in GYSELA-X
 - Restart files can be saved in HDF5 or other formats (as FTI or SionLib, ...) to facilitate future I/O tests on exascale machines
 - Next step: Pursue the decoupling of I/O from the computing kernels → coupling with Python + DASK array to perform diagnostic computations

- ▶ First porting on new architecture (GPU and ARM) for preparing the code to ITER exascale future simulations
 - In both cases seem to require lots of refactoring to get good performance
 - As the implementation of the non-equidistant mesh to treat strong gradient in the SOL will need strong modification of the code

→ maybe good time to think to a rewriting of GYSELA-X code : Fortran + OpenACC or OpenMP, C++/Kokkos ?