Directive-based, Fortran/C++ interoperable approach to GPU offloading of the high performance gyrokinetic turbulence code GENE-X

GENE-X

MAX-PLANCK-INSTITUT FÜR PLASMAPHYSIK

Jordy Trilaksono[1]*, Philipp Ulbl[1], Andreas Stegmeir[1], Frank Jenko[1,2]

[1]Max Planck Institute for Plasma Physics, Boltzmannstraße 2, 85748 Garching, Germany, [2]University of Texas at Austin, Austin, TX 78712, USA

ABSTRACT

The achievement of high plasma confinement is the key to realize commercially attractive energy production by magnetic confinement fusion (MCF) devices. Turbulence plays a significant role in maintaining the plasma confinement within MCF devices. The GENE-X code is based on an Eulerian (continuum) approach to the discretization of the five dimensional gyrokinetic equation that describes plasma turbulence. Our discretization is specialized to simulate plasma turbulence anywhere within MCF devices, from the hot plasma core to the cold wall. GENE-X is written in object-oriented modern Fortran 2008 leveraging MPI+OpenMP parallelization to facilitate large scale turbulence simulations. Here, we present our development efforts to further extend the parallelization scheme to GPUs, which is essential for scalability support towards simulations of larger, reactor-relevant fusion devices. The current implementation in GENE-X provides a proof of concept of our native Fortran/C++ interoperability approach by successfully supporting several GPU backends such as OpenACC, OpenMP offload and CUDA. We present first benchmarks of our directive-based OpenACC implementation of the most computationally expensive part of GENE-X. A significant performance increase was achieved on the GPU, compared to equivalent CPU benchmarks.

INTRODUCTION TO GENE-X

`GENE-X` [1] is a full-f gyrokinetic turbulence code written in Fortran 2008 with object-oriented design. `GENE-X` features electromagnetic field equations [2], collision models [5], and a flux-coordinate independent (FCI) [4] coordinate system. `GENE-X` shares an FCI-based mesh equilibrium and field solver library called `PARALLAX` with the Braginskii fluid `GRILLIX` [3].

`GENE-X` solves 5D arrays of distribution function consisting $RZ, \varphi, v_{\parallel}, \mu$, and species axes.

`GENE-X` uses a heterogenous parallelization:

- OpenMP for intra-node parallelization on CPUs
- MPI for inter-node parallelization across CPUs

- **CPU**: low latency, best for wide-range of tasks
- **GPU**: more cores, high throughput for more repetitive tasks

Issue: Some Fortran 2008 object-oriented features used in `GENE-X` are not yet supported by the Fortran compilers from PGI or NVIDIA SDK.

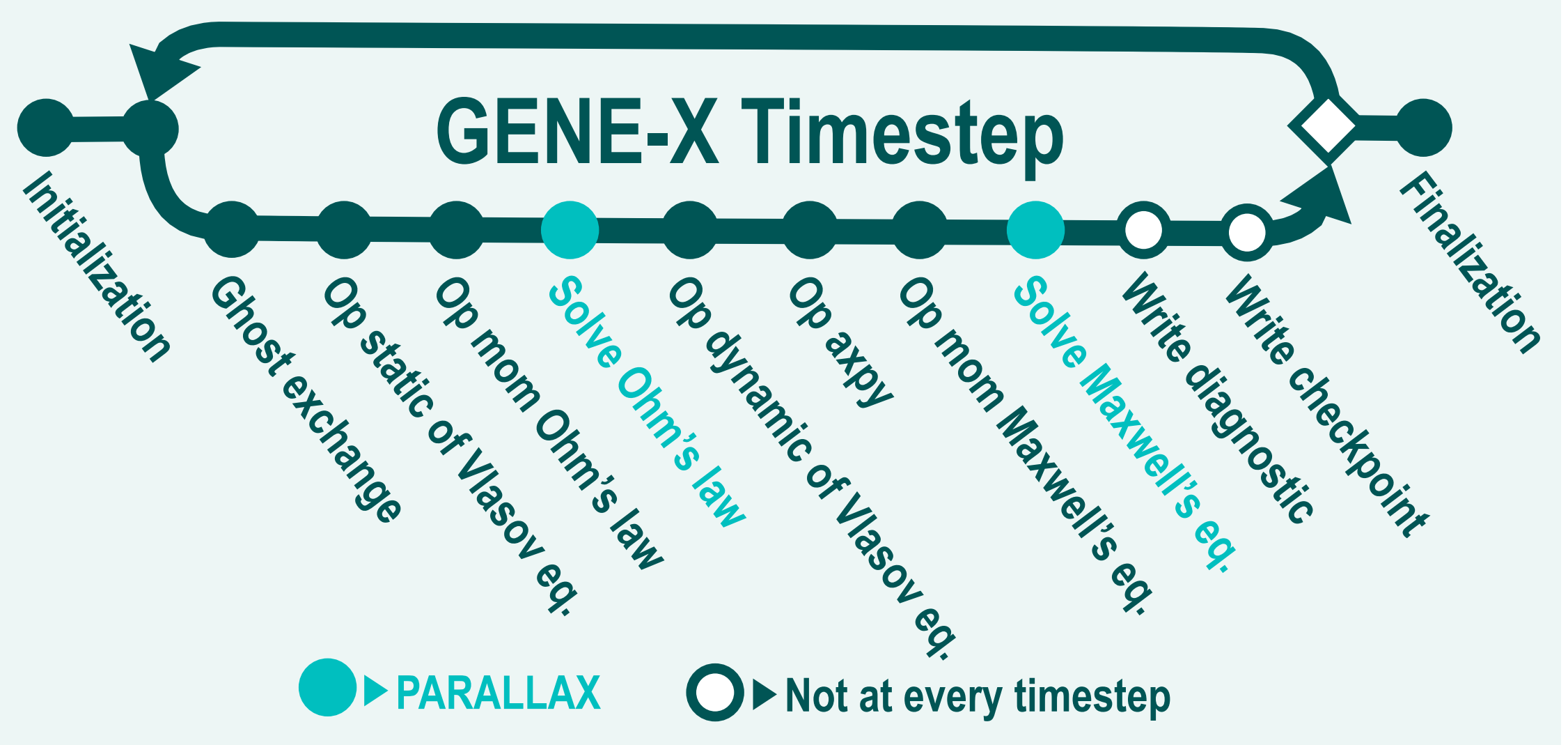

Figure 1: Overview of `GENE-X` timestep.

Operators (abbr. as **op**) in GENE-X are classes containing the compute region of specific algorithms based on strategy design pattern.

The following arithmetic operations in `GENE-X` are used for the preliminary GPU implementations:

1. `y[i] = c`
2. `y[i] = x[i]`
3. `y[i] = y[i] + a*x[i]`
4. `y[i] = a1*x1[i] + a2*x2[i]`

The following static and dynamic operators are the right-hand-side (RHS) terms of the **gyrokinetic Vlasov equation** used in `GENE-X` which are used for preliminary benchmarks:

5. **op_static**

$$
\begin{aligned}
&= - v_{\parallel}\frac{\boldsymbol{B}^*}{B_{\parallel}^*}\cdot\nabla f_\alpha && \text{(Parallel } \boldsymbol{B}^* \text{ advection)}\\
&\quad - \frac{c}{q_\alpha B_{\parallel}^*}\boldsymbol{b}\times(\mu\nabla B + q_\alpha\nabla\phi_1)\cdot\nabla f_\alpha && \text{(Cross-field advection)}\\
&\quad + \frac{\boldsymbol{B}^*}{m_\alpha B_{\parallel}^*}\cdot(\mu\nabla B + q_\alpha\nabla\phi_1)\frac{\partial f_\alpha}{\partial v_{\parallel}} && \text{(Velocity-space advection)}
\end{aligned}
$$

6. **op_dynamic**

$$
= \frac{q_\alpha}{m_\alpha c}\frac{\partial A_{1,\parallel}}{\partial t}\frac{\partial f_\alpha}{\partial v_{\parallel}} \qquad \text{(Electromagnetic flutter)}
$$

Additionally, the following operators calculate the moments of the distribution function and contain MPI communicators which are used for multi-GPU testing:

7. **op_mom_ohms_law** for solving the Ohm's law.
8. **op_mom_maxwell_eq** for solving the quasi-neutrality eq.

SOFTWARE ENGINEERING OF THE NATIVE FORTRAN/C++ INTERFACE

An auxiliary C++ layer dedicated to the compute region for GPU offloading can be interoperable with the main Fortran layer. The benefits of having such auxiliary C++ layer are:

- Well established Fortran compilers can still be used
- Additionally GPU-optimized C++ compilers can be used
- More flexibility towards various machines
- Creating GPU kernels on the C++ layer is possible
- Non-invasive refactoring to the main Fortran layer
- Maintain the modular concept and design

This multi-language approach is referred here as **Fortran/C++ hybrid** model.

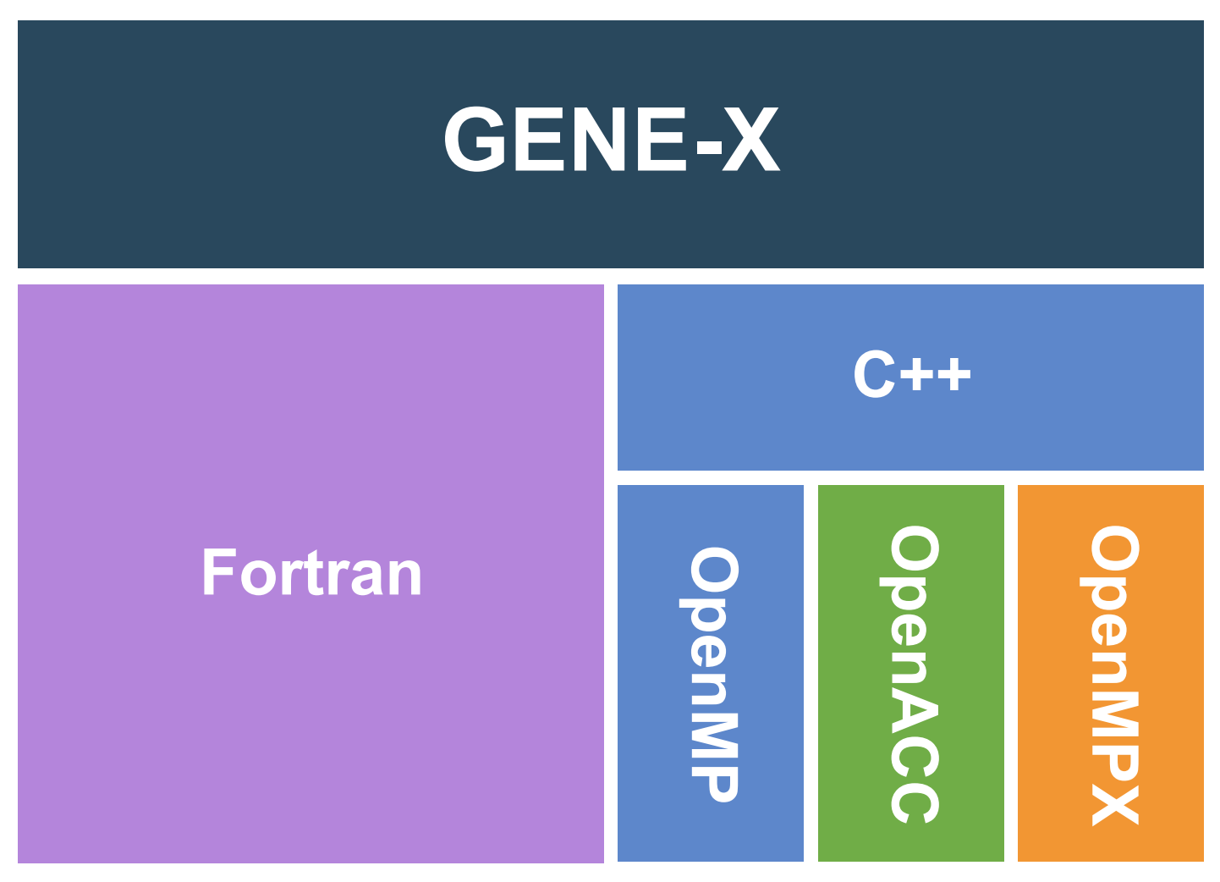

Figure 2: GENE-X composition by language and GPU backends.

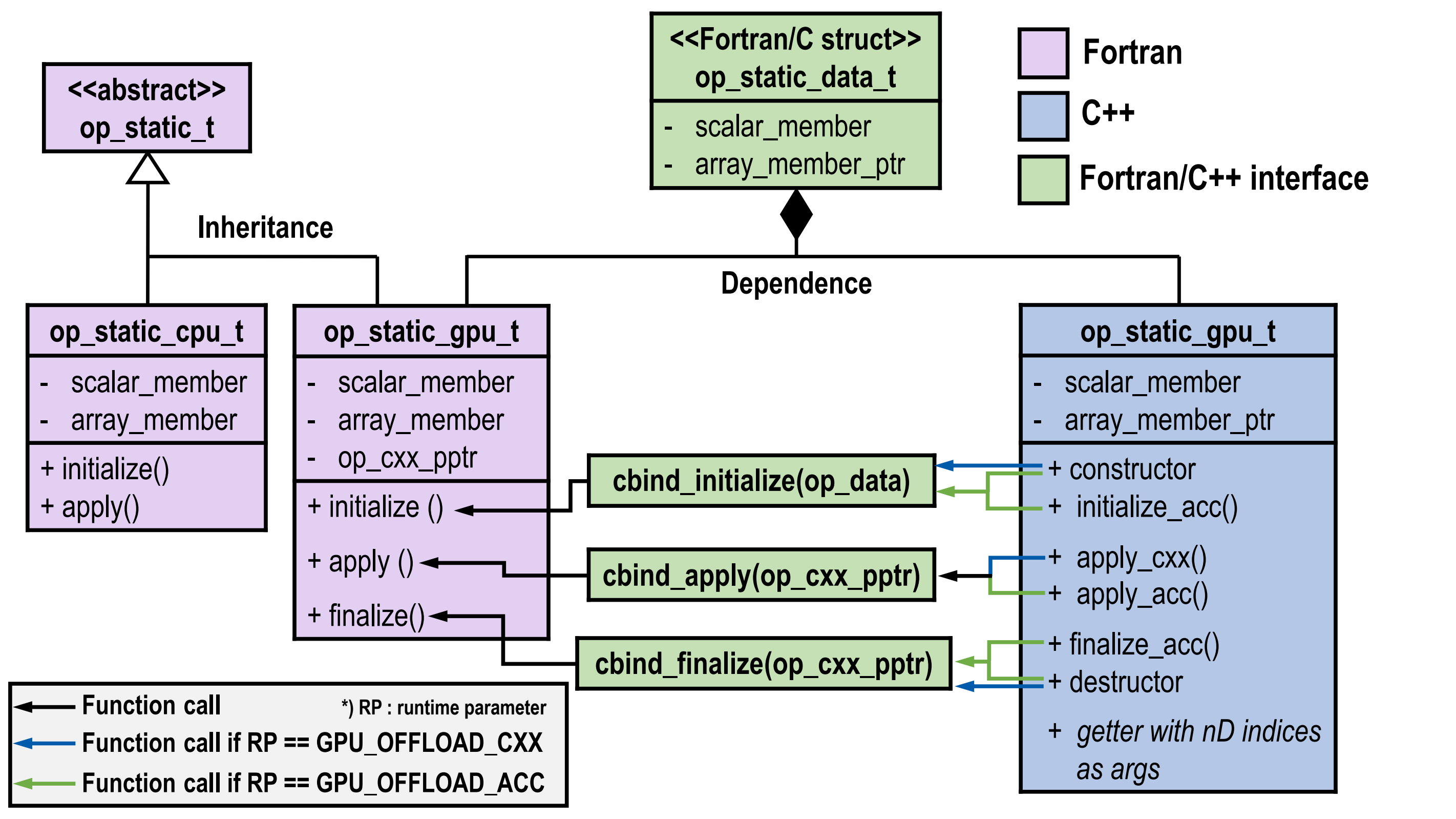

Figure 3: Simplified UML diagram containing the static operator classes on Fortran and C++ layers and their interoperability.

GPU OFFLOADING STATUS

The arithmetic operators are experimented on to show the compatibility of the C++ layer to various GPU offload backends, i.e. OpenACC, OpenMP offload and CUDA.

The directive-based backend, i.e. OpenACC, is prioritized due to the multidimensional array of the distribution function.

The following are the compiler combinations used on MPCDF machines (Cobra and Raven):

- **Fortran**: GNU compiler (gfortran)
- **C++**: NVIDIA HPC SDK compiler (nvc++)

↘ Proof of concept with built-in arithmetic operators

Operators	C++ base (OMP)	GPU offloading backend		
		ACC	OMPX	CUDA
Arithmetic op (4)	🟩	🟩	🟩	🟩

Fortran compiler: GNU
C++ compiler: NVIDIA
CUDA compiler: NVIDIA

Tested on GPU partitions of raven and cobra

↘ Infrastuctures

Infrasturcture	C++	ACC	OMPX
Mesh	🟩	🟩	🟩
Comm. handler	🟩	⬛	⬛

↘ Top-level priority to offload to GPU

Operators* of RHS gyrokinetic Vlasov eq.	C++	ACC	OMPX
Dynamic operator	🟩	🟩	🟩
Static operator	🟩	🟩	🟩

↘ Multi-GPU milestone

Op with MPI comm.	MPI+OMP	MPI+ACC
op_mom_ohms_law	🟩	⬛
op_mom_maxwell_eq	🟩	⬛

OMP : OpenMP
ACC : OpenACC
OMPX : OMP offload

🟩 Complete
⬛ Ongoing

Figure 4: GPU offloading status of `GENE-X`. Top: The status of various GPU backend supports of the Arithmetic operators, the porting status of various operators and infrastructures.

CONCLUSION AND OUTLOOK

`GENE-X` build configuration now supports Fortran/C++ hybrid model with mainly OpenACC and OpenMP offload as GPU backend. The directive-based approach is chosen due to maintainability factor and good affinity with the numerics of GENE-X. Here, the preliminary performance monitoring of the static and dynamic operators of gyrokinetic Vlasov equation are presented and showing promising speedups. Next milestones are:

- Multi-GPU implementation
- Latency analysis
- Kernel optimization

REFERENCES

[1] D. Michels et al. In: *Comp. Phys. Commun.* 264 (2021), p. 107986. DOI: `10.1016/j.cpc.2021.107986`.

[2] Dominik Michels et al. In: *Phys. of Plasmas.* 29.3 (2022), p. 032307. DOI: `10.1063/5.0082413`.

[3] A. Stegmeir et al. In: *Phys. of Plasmas.* 26 (2016), p. 052517. DOI: `10.1063/1.5089864`.

[4] A. Stegmeir et al. In: *Comp. Phys. Commun.* 198 (2016), pp. 139–153. DOI: `10.1016/j.cpc.2015.09.016`.

[5] P. Ulbl, D. Michels, and F. Jenko. In: *Contrib. Plasma Phys.* (2021), e202100180. DOI: `10.1002/ctpp.202100180`.

PRELIMINARY BENCHMARK RESULTS

The preliminary benchmark of the static and dynamic operators is done in one GPU node of MPCDF machine called Raven with the following specifications:

MPCDF Cobra GPU partition: 1 node, 1 out of 2 GPUs

- 20 CPU cores: 1 Intel Xeon Gold Skylake 6148
- 1 GPU: NVIDIA V100, 32GB HBM2

Distribution function: 105,012,224 points in total

Number of points in $RZ = 205{,}102$
Number of points in $\phi = 16$
Number of points in $v_{\parallel} = 4$
Number of points in $\mu = 4$
Number of species $= 2$

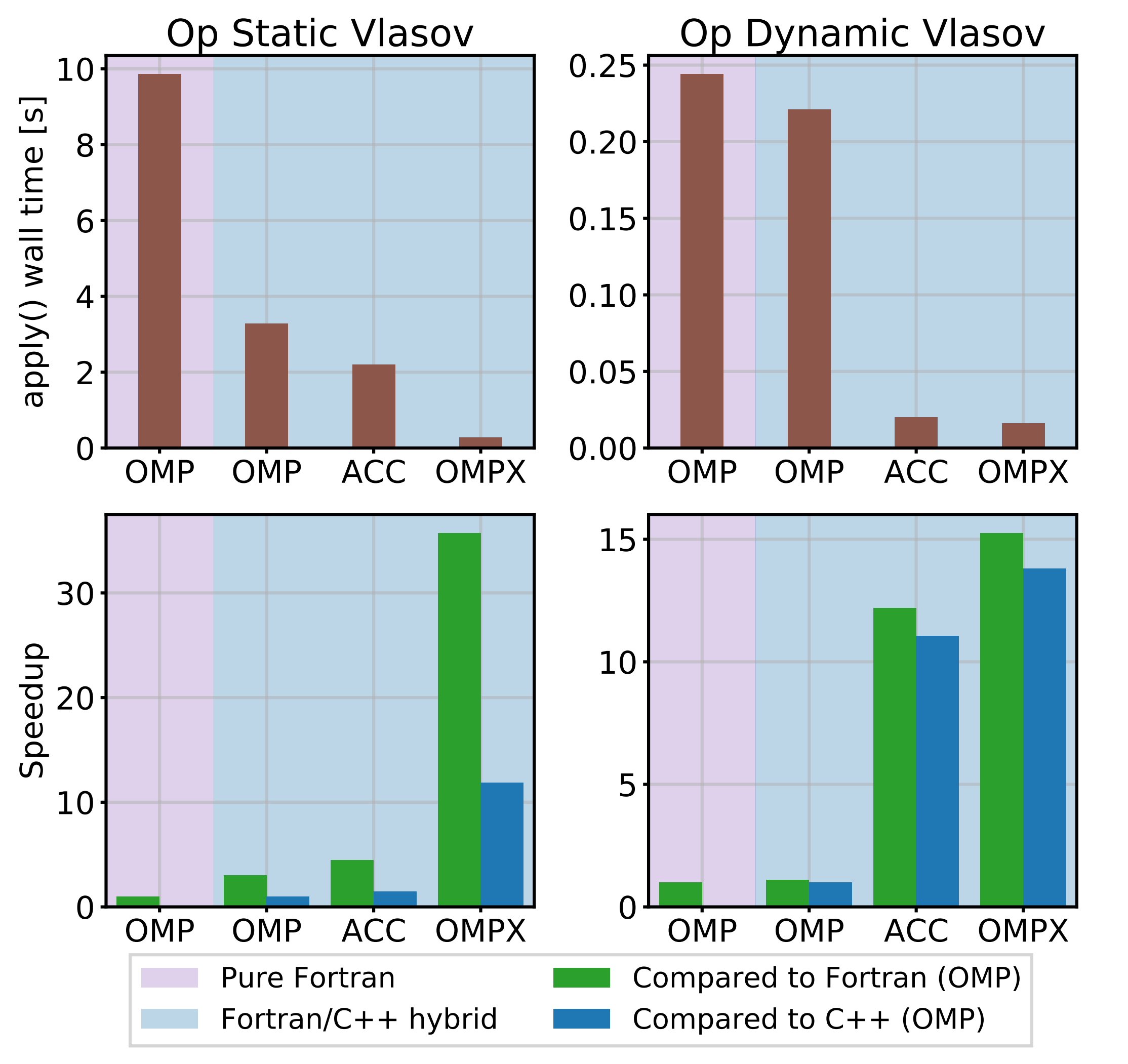

Figure 5: Top: Preliminary wall time measurements of the apply procedure of the static (left) and dynamic (right) operators. Bottom: Speedups compared to Fortran and C++ wall time.

Performance difference between OpenACC and OpenMP offload is yet to be investigated. OpenMP kernels on C++ indicates possible rooms of improvement to OpenMP parallelism in GENE-X Fortran kernels

Problem size (in points) in various MCF devices

- TCV $\sim 10^{10}$
- AUG $\sim 10^{11}$
- ITER $\sim 10^{13}$

HEPP

Corresponding author: *jordy.trilaksono@ipp.mpg.de;
PASC 2023, Davos, Switzerland,
June 26-28, 2023

EUROfusion

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.