Master 2 / PFE internship
(with option for a PhD continuation)

Design of a *modern C++* library for performance-portable mesh handling on GPU.

Maison de la Simulation offers a Master 2, or *projet de fin d'études* internship to design the GPU version of a library leveraging the latest features of C++ to handle high-dimensional meshes.

Contacts:	Julien Bigot	🖂 julien.bigot@cea.fr	☎ +33 (0)1 69 08 01 75
	Thomas Padioleau	🖂 thomas.padioleau@cea.fr	☎ +33 (0)1 69 08 10 37

Location: Maison de la Simulation (CEA Saclay, 91191 Gif-sur-Yvette CEDEX)

Overview

`ddc` is a C++ header-only **d**omain **dec**omposition library developed at Maison de la Simulation. `ddc` aims to support intensive computation over high-dimensional meshes. It provides multi-dimensional containers (based on `std::mdspan` [2] proposed for C++23), iterators over geometric elements and type-safe array indexing. By giving a mesh semantic to algorithms, `ddc` eases working on fields mapped on sub-meshes, and switching between meshes; it supports writing generic mesh-independant algorithms, or optimized mesh-specific algorithms.

`ddc` design is a work in progress whose first prototype targets CPU. The goal of the internship is to evaluate the design changes required to support portability to GPU. The proposed design should support seamless feature and **performance portability** between CPU and GPU. To reach that goal, `std::mdspan` might be replaced by Kokkos [3] equivalents and iteration algorithms will have to be designed that work on both CPU and GPU.

Work-plan

As a preliminary step, the candidate will acquire expertise on the existing `ddc` library. An example code, solving the heat equation, as well as the `Voice++` code, solving the Vlasov-Poisson in 2D, will be provided to demonstrate the use of the library. In a second step the candidate will identify and implement the changes required in `ddc` to port the heat equation on GPU for both uniform and non-uniform Cartesian meshes. Finally, the selected porting strategy will be applied to the `Voice++` code. At each step, the candidate will have access to parallel machines and super-computers amongst the most powerful in the world to validate the developments.

Skills

The successful candidate will master the following skills and knowledge:
- team-work in an international environment,
- unix environment (shell, git, etc.),
- software engineering and library design,
- proficiency in "*modern*" `C++`.

In addition, the following will be considered a plus:
- knowledge and experience with GPU computing,

- HPC and parallel libraries such as OpenMP and MPI,
- experience with the C++ ecosystem and tooling (CMake, unit-testing libraries, etc.).

Context

The 5D non-linear gyrokinetic code GYSELA 5D[1] [1] has been developed at CEA/IRFM over the last 20 years to study turbulence in Tokamak plasmas. The kinetic model couples the six dimensional Vlasov or Fokker-Planck equation for the distribution of particles, with Maxwell's equations for the electromagnetic fields. Since turbulent fluctuations develop at much lower frequencies than the cyclotron motion, this can be reduced to the 5D *gyrokinetic* model (in addition to the time and species dimensions). Even with this dimensionality reduction however, gyrokinetic codes requires state-of-the-art high performance computing resources and GYSELA 5D regularly use between 8 192 and 65 536 cores in parallel for multiple weeks simulations.

While the GYSELA 5D code currently answers the needs of physicists very well, two changes will have to be be handled in the coming years. First, the regular toroidal mesh of GYSELA 5D will have to be replaced with more complex meshes in order to handle the geometries of real Tokamaks. Second, the code will have to be ported to GPU to keep using the most powerful super-computers in the world. Many assumptions are made in the existing code that make these changes very complex, and a rewrite of the code from scratch is planned. This rewrite will provide the opportunity to rewrite the existing Fortran code in C++.

The Kokkos library offers multi-dimensional arrays for C++ and supports GPU. Its direct use with no additional abstraction would however lead to a design similar to that of the existing GYSELA 5D code where the absence of an explicit mesh concept makes it very difficult to replace. The choice has been made to instead implement a **zero-cost abstraction library** that explicitly implements the concept of mesh and supports arrays mapping values on meshes: `ddc`. The design and implementation of `ddc` is lead by Maison de la Simulation and a first prototype is available for CPU, built on top of the proposed `std::mdspan` C++ extension.

PhD continuation

A PhD subject is also available in the continuation of this internship subject. The goals of the PhD will be to build on `ddc` to propose a PGAS-style programming model with performance portability over large scale distributed-memory parallel super-computers including GPU and multi-GPU nodes. The full PhD subject is available at https://work.julien-bigot.fr/.

References

[1] V. Grandgirard, J. Abiteboul, and J. et al. Bigot. A 5D gyrokinetic full- f global semi-Lagrangian code for flux-driven ion turbulence simulations. *Computer Physics Communications*, 207:35–68, October 2016.

[2] David S. Hollman, Bryce Adelstein Lelbach, H. Carter Edwards, Mark Hoemmen, Daniel Sunderland, and Christian R. Trott. mdspan in c++: A case study in the integration of performance portable features into international language standards. In *2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, pages 60–70, 2019.

[3] Christian R. Trott, Damien Lebrun-Grandié, Daniel Arndt, Jan Ciesko, Vinh Dang, Nathan Ellingwood, Rahulkumar Gayatri, Evan Harvey, Daisy S. Hollman, Dan Ibanez, Nevin Liber, Jonathan Madsen, Jeff Miles, David Poliakoff, Amy Powell, Sivasankaran Rajamanickam, Mikael Simberg, Dan Sunderland, Bruno Turcksin, and Jeremiah Wilke. Kokkos 3: Programming model extensions for the exascale era. *IEEE Transactions on Parallel and Distributed Systems*, 33(4):805–817, 2022.

[1] https://gyselax.github.io/