Funded PhD subject:
Performance-portable support of complex discretizations at extreme scale in modern C++

Advisor: Thomas Padioleau ✉ thomas.padioleau@cea.fr ☎ +33 (0)1 69 08 10 37
Director: Julien Bigot ✉ julien.bigot@cea.fr ☎ +33 (0)1 69 08 01 75
Laboratory: Maison de la Simulation (CEA Saclay, 91191 Gif-sur-Yvette CEDEX)

Keywords: parallel programming models, software engineering, modern C++, partitioned global address space, performance portability, GPU, extreme-scale parallelism

Context

Extreme-scale simulation codes are typically developed in C or FORTRAN, parallelized with MPI[7] and OpenMP[16]. This approach has proven very successful for code portability during the last decades; but with the advent of GPU-based supercomputers, codes have to be ported to a new architecture for which a rewrite from scratch is often required.

Lots of efforts have been devoted recently to design programming models targeting (performance) portability between CPU and GPU including `Kokkos` [18], OpenACC, RAJA[4], or SYCL[17] for example. Some also target distributed memory parallelism like Co-array Fortran[14], HPX[11], UPC[5, 6], UPC++[3], or XMP[15]. These models rely on the same abstractions that were present in previous models: (multi-dimensional) arrays and (parallel) loops. Numerical applications then build higher-level abstractions on top of these by taking into account the specifics of the hardware they target, hence limiting performance portability. Many concepts also remain implicit in the code so as not to pay a cost at execution; leading to issues with code maintainability and adaptability.

These issues can only be solved by directly providing higher-level abstractions to the numerical simulation codes. An interesting example is offered by `numpy`, `xarray` and `dask.Array` in Python. These libraries make it possible to express numerical computations in a very natural way thanks to the abstraction of mesh and data distribution they provide and to execute it on a variety of hardware architectures by adding just a few dedicated annotations. However, this ecosystem remains far from offering performances comparable with FORTRAN + OpenMP/MPI.

Goal

The goal of this PhD thesis is to evaluate if a solution based on C++ template metaprogramming can offer high-level (zero-cost) abstractions handling a large range of data discretization at compile time. The work will take place in the framework of the `ddc`[1] library and will be evaluated on the very demanding simulation code GYSELA[2][8, 12] that leverage the largest existing super computers and manipulates multiple complex discretizations of its high-dimension data along execution. The goal is to handle seamless replacement of discretization in code (*e.g.* structured uniform mesh to unstructured), while offering the best performance from each one. Another goal is to handle parallelism at all levels: distributed-memory parallelism similarly to PGAS languages, shared-memory parallelism on both CPU and GPU, but also SIMD parallelism.

[1] `https://ddc.mdls.fr`
[2] `https://gyselax.github.io/`

Work-plan

During the first phase of the work, the candidate will study the related bibliography (see an extract bellow), focusing on parallel programming models, and especially PGAS languages to get a good understanding of the involved concepts, understand the limitations of current approaches, and seize the benefits of existing softwares. In addition to academic publications, this first phase will be used to discover existing work in `ddc` and the needs identified as part of the rewrite of the GYSELA code.

Then, the work will focus on the conception of a programming model supporting the various features identified in the previous section. This work will start by focusing on single-node CPU and GPU parallelism before moving to multi-node distributed parallelism. The proposed concepts will be tested on academic cases using the usual evaluation criteria from the literature: portability, performance, lines of code, readability, etc.

Finally, the proposed solutions will be implemented in `ddc` and applied to the development of the next-generation production code GyselaX in collaboration with the team developing the code at CEA/IRFM, Cadarache and with collaborators working on performance optimization and portability in JAEA, Tokyo, Japan. They will be tested at scale on several supercomputers including systems amongst the most powerful in the world (Fugaku[3], Joliot-Curie[4], Adastra[5], ...) but also on testbeds of new GPU accelerators in the framework of collaborations with vendors such as Intel. All results will be submitted to international peer-reviewed conferences and journals, such as SuperComputing, IPDPS, Cluster, JPDC, etc. for publication. The candidate will also be encouraged to participate in the writing of a proposal to the C++ standard comittee about the solutions found.

Skills

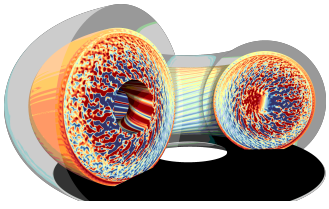
The successful candidate will master the following skills and knowledge:

- strong interest in programming models in general and parallel programming models in particular,
- proficiency and experience with modern C++ and template metaprogramming,
- motivation for team-work in an international environment.

In addition, the following will be considered a plus:

- knowledge and experience with GPU, parallel and high-performance computing,
- interest for applied mathematics and numerical simulation,
- experience with software engineering and library design.

Existing software

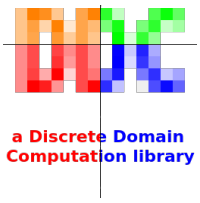


The 5D non-linear **gy**rokinetic, **se**mi-**La**grangian code **GYSELA** has been developed at CEA/IRFM for the last 20 years to study turbulence in Tokamak plasmas. The kinetic model couples the six dimensional Vlasov or Fokker-Planck equation for the distribution of particles, with Maxwell's equations for the electromagnetic field. 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 regularly use between 8 192 and 65 536 cores in parallel for multiple weeks simulations.

While GYSELA answers current needs of physicists very well, two changes will have to be be handled in the coming years. First, the regular toroidal mesh of GYSELA will have to be replaced with more complex meshes in order to handle the geometries of real Tokamaks. Second, with the arrival of Exascale, computational nodes become more and more heterogeneous often featuring GPUs. The code will have to be ported to these new compute architecture to keep using the most powerful

[3]Fugaku, top 1 supercomputer in Nov 2021 https://www.top500.org/system/179807/
[4]Joliot-Curie french supercomputer https://www.top500.org/system/179700/
[5]Adastra, upcoming supercomputer in France https://www.cines.fr/en/12600/

super-computers in the world. Many assumptions are made in the existing code that make these changes very complex, and a rewrite from scratch of the existing Fortran code into C++ is planned.



`ddc` is a C++ template meta-programming **d**discrete **d**omain **c**omputation library developed at Maison de la Simulation. `ddc` supports intensive computation over high-dimensional meshes. It provides multi-dimensional containers (based on `std::mdspan` [9] proposed for inclusion in the C++ standard), and supports 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.

While similar to (and based on top of) other C++ template meta-programming libraries such as `Kokkos` or the standard library `std::mdspan` and the standard parallel algorithms, `ddc` offers a higher-level of abstraction and supports typed indices, preventing error when manipulating many different discrete dimensions. `ddc` is currently in the prototype phase and support for parallelism in the library is a work in progress.

Laboratory and collaborations

Maison de la Simulation (MdlS – `http://www.maisondelasimulation.fr/`) located on the Plateau de Saclay, is a joint laboratory of the alternative energies and atomic energy commission (CEA), the national center for scientific research (CNRS), Paris-Saclay University, and Versailles Saint-Quentin-en-Yvelines University. The laboratory groups activities around high performance computing (HPC): research in computer science and applied mathematics, engineering, and development for high-performance simulation applications. Of specific interest at MdlS are software engineering aspects of HPC, especially regarding separation of concerns, performance optimization and performance portability.

The **Institute for Magnetic Fusion Research** (IRFM – `http://irfm.cea.fr/`) located at the CEA research centre of Cadarache, is one of the 15 institutes that make up the fundamental research division in the alternative energies and atomic energy commission (CEA). For almost 60 years, its responsibility has been to carry out research on thermonuclear magnetically-confined fusion at the CEA in association with the Euratom fusion programme. In Tokamaks, matter is in a hot plasma state in interaction with a strong magnetic field, and controlling nuclear fusion requires to understand the mechanisms that govern heat confinement. Ion temperature gradient instabilities is an important mechanism whose understanding requires numerical simulation, this objective lead to the development of the GYSELA code.

Japan atomic energy agency (JAEA – `https://www.jaea.go.jp/english/`) is Japan top nuclear research agency. Amongst many others, it hosts researchers interested in the development of performance-portable simulation codes. IRFM and Maison de la Simulation have strong collaborations ongoing regarding these aspects for the design of the new-generation GYSELA code.

Select bibliography

[1] Yuuichi Asahi, Guillaume Latu, Julien Bigot, Shinya Maeyama, Virginie Grandgirard, and Yasuhiro Idomura. Overlapping communications in gyrokinetic codes on accelerator-based platforms. *Concurrency and Computation: Practice and Experience*, 32(5):e5551, 2020. e5551 cpe.5551.

[2] Yuuichi Asahi, Guillaume Latu, Virginie Grandgirard, and Julien Bigot. Performance portable implementation of a kinetic plasma simulation mini-app. In Sandra Wienke and Sridutt Bhalachandra, editors, *Accelerator Programming Using Directives*, pages 117–139, Cham, 2020. Springer International Publishing.

[3] John Bachan, Scott B. Baden, Steven Hofmeyr, Mathias Jacquelin, Amir Kamil, Dan Bonachea, Paul H. Hargrove, and Hadia Ahmed. Upc++: A high-performance communication framework

for asynchronous computation. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 963–973, 2019.

[4] David A. Beckingsale, Jason Burmark, Rich Hornung, Holger Jones, William Killian, Adam J. Kunen, Olga Pearce, Peter Robinson, Brian S. Ryujin, and Thomas RW Scogland. Raja: Portable performance for large-scale scientific applications. In *2019 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*, pages 71–81, 2019.

[5] Li Chen, Lei Liu, Shenglin Tang, Lei Huang, Zheng Jing, Shixiong Xu, Dingfei Zhang, and Baojiang Shou. Unified Parallel C for GPU Clusters: Language Extensions and Compiler Implementation. In Keith Cooper, John Mellor-Crummey, and Vivek Sarkar, editors, *Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science, pages 151–165, Berlin, Heidelberg, 2011. Springer.

[6] UPC Consortium, Dan Bonachea, and Gary Funck. Upc language and library specifications, version 1.3. 11 2013.

[7] Message Passing Interface Forum. *MPI: A Message-passing Interface Standard, Version 3.1 ; June 4, 2015.* High-Performance Computing Center Stuttgart, University of Stuttgart, 2015.

[8] 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.

[9] 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.

[10] Dana Jacobsen, Julien Thibault, and Inanc Senocak. An MPI-CUDA Implementation for Massively Parallel Incompressible Flow Computations on Multi-GPU Clusters. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics. _eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2010-522.

[11] Hartmut Kaiser, Patrick Diehl, Adrian S. Lemoine, Bryce Adelstein Lelbach, Parsa Amini, Agustín Berge, John Biddiscombe, Steven R. Brandt, Nikunj Gupta, Thomas Heller, Kevin Huck, Zahra Khatami, Alireza Kheirkhahan, Auriane Reverdell, Shahrzad Shirzad, Mikael Simberg, Bibek Wagle, Weile Wei, and Tianyi Zhang. Hpx - the c++ standard library for parallelism and concurrency. *Journal of Open Source Software*, 5(53):2352, 2020.

[12] Guillaume Latu, Yuuichi Asahi, Julien Bigot, Tamas Feher, and Virginie Grandgirard. Scaling and optimizing the gysela code on a cluster of many-core processors. In *2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 466–473, 2018.

[13] Jinpil Lee, Minh Tuan Tran, Tetsuya Odajima, Taisuke Boku, and Mitsuhisa Sato. An Extension of XcalableMP PGAS Lanaguage for Multi-node GPU Clusters. In Michael Alexander, Pasqua D'Ambra, Adam Belloum, George Bosilca, Mario Cannataro, Marco Danelutto, Beniamino Di Martino, Michael Gerndt, Emmanuel Jeannot, Raymond Namyst, Jean Roman, Stephen L. Scott, Jesper Larsson Traff, Geoffroy Vallée, and Josef Weidendorfer, editors, *Euro-Par 2011: Parallel Processing Workshops*, Lecture Notes in Computer Science, pages 429–439, Berlin, Heidelberg, 2012. Springer.

[14] John Mellor-Crummey, Laksono Adhianto, William N. Scherer, and Guohua Jin. A new vision for coarray fortran. In *Proceedings of the Third Conference on Partitioned Global Address Space Programing Models*, PGAS '09, New York, NY, USA, 2009. Association for Computing Machinery.

[15] Masahiro Nakao, Jinpil Lee, Taisuke Boku, and Mitsuhisa Sato. Productivity and performance of global-view programming with xcalablemp pgas language. In *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pages 402–409, 2012.

[16] OpenMP Architecture Review Board. *OpenMP Application Programming Interface Specification Version 5.2*. Independently published, November 2021.

[17] Ruyman Reyes and Victor Lomüller. Sycl: Single-source c++ accelerator programming. In *Parallel Computing: On the Road to Exascale*, pages 673–682. IOS Press, 2016.

[18] Christian R. Trott, Damien Lebrun-Grandie, 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.

[19] Lei Zhou and Karl Fuerlinger. DART-CUDA: A PGAS Runtime System for Multi-GPU Systems. In *2015 14th International Symposium on Parallel and Distributed Computing*, pages 110–119, June 2015. ISSN: 2379-5352.