Master 2 / PFE internship
(in the framework of a Franco-Japanese collaboration)
Implementation and evaluation of proposed C++ standard mechanisms for CPU/GPU parallelism

Maison de la Simulation offers a Master 2, or *projet de fin d'études* internship to evaluate proposed extensions to the C++ standard to support CPU and GPU parallelism.

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
Yuuishi Asahi ✉ y.asahi@nr.titech.ac.jp
Location: Maison de la Simulation (CEA Saclay, 91191 Gif-sur-Yvette Cedex)

Overview

Lot of efforts have been devoted recently to design programming models supporting portability between CPU and GPU for shared-memory parallelism, including HPX, Kokkos [6], OpenACC, OpenMP 4.5+, RAJA, or SYCL for example. Whether developed as libraries or compiler extensions, all of these support applications written in C++ for the sequential part, a language more and more used for the development of high-performance computing applications. The multiplication of such extensions in C++ is however challenging for inter-compatibility between codes that select distinct options for parallelism. The C++ standard committee is thus evaluating the possibility to add support for parallelism in the standard itself.

The proposed extensions to the C++ standard include

- `std::mdspan`[1] [4] and `std::mdarray`[2] for multi-dimensional arrays,
- `std::views::cartesian_product`[3] for multi-dimensional iteration,
- `std::execution::par` and `std::execution::par_unseq`[4] for parallel iteration.

While these constructs have been mostly devised with CPU support in mind, the C++ standard is abstract enough that GPU execution should be possible.

In the context of a Franco-Japanese collaboration, various solutions for CPU/GPU parallelism have been evaluated on the 2D & 4D Vlasov/Poisson toy models that act as proxy applications for the more complex Gysela 5D[5] [3] code. This work has lead to multiple publications [1, 2, 5]. A question arises now whether the proposed standard C++ extensions will be usable to support efficient parallelism in the 2D & 4D Vlasov/Poisson toy models and hence for the planned new version of Gysela 5D.

Work-plan

As a preliminary step, the candidate will familiarize oneself with the existing code-base of the 2D & 4D Vlasov/Poisson toy models. The first step will then be to use existing implementations of the proposed C++ extensions for CPU parallelism in this code-base. Once this is done, focus will switch to GPU. Some elements of the implementations will have to be modified to target these devices and

[1] http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2021/p0009r14.html
[2] https://isocpp.org/files/papers/D1684R0.html
[3] http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2021/p2374r1.html
[4] https://en.cppreference.com/w/cpp/algorithm/execution_policy_tag
[5] https://gyselax.github.io/

the candidate will contribute to the GPU port of the proposed C++ standard extensions. A design goal should be to minimize the user-visible modifications so that a single code-base can be executed on CPU or GPU. At every 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.).

Franco-Japanese collaboration

During the internship, the candidate will interact will the Japanese collaborators that take part in this project through visio-conferences and direct message systems. At the conclusion of the internship, a position should be available to continue the work. Depending on the situation of worldwide health, administrative and other constraints, in-person travel to Japan can then be possible in the framework of this collaboration.

References

[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] 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.

[4] 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.

[5] 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.

[6] 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.