Julia tools for HPC

For 20 years now I have been working as a research engineer in scientific computing on the implementation of numerical methods in plasma physics. My first codes were written in Fortran and followed the evolution of the language to try to improve the user experience and facilitate the integration of new collaborators in our projects. In 2012, with the numerical analysis team of IRMA in Strasbourg, we launched the SeLaLib project whose objective was to propose different implementations to solve the Vlasov equation with semi-Lagrangian methods. This project was then extended by adding Particle-Mesh methods here at IPP. This library was not fully HPC but care was taken to optimise it so that SeLaLib could test numerical methods in a reasonable time. A lot of work has been done in SeLaLib but unfortunately we have been confronted with several problems: the difficulty to integrate new contributors, Fortran is less and less taught in France and newcomers prefer Python because it offers a better productivity. This is why they started at IPP the Pyccel project. Another problem is that Fortran is not well suited for software projects that will run on new architectures with GPUs. This led to the GEMPIC-AMRex project. In 2019 I presented some examples of the Julia language which can be a good compromise between the productivity of Python, the performance of Fortran and an adaptability to GPU devices. I have been trying for a few years to offer these tests in the Julia Vlasov GitHub organization. In this talk I will show you some examples of Julia in an HPC oriented context with also some interesting packages like Gridap.jl, Trixi.jl, PencilArrays.jl, PartitionedArrays.jl, DistributedSparseGrids.jl, ParallelStencil.jl... that can considerably speed up the time for computations but also the development time. The first feedback we have with young researchers is extremely positive as it is an attractive language with an easy to learn syntax.