Powering Plasma-Physics with RISC-V vector extension: the case of Vlasiator

Gerard Oliva[1*], Pablo Vizcaino[1], Urs Ganse[2], Marta Garcia-Gasulla[1] and Filippo Mantovani[1]

[1]Barcelona Supercomputing Center (Spain)
[2]University of Helsinki (Finland)

Abstract

In this talk we will present the performance study and optimization of the Vlasiator plasma physics simulation on a RISC-V vector CPU. We aimed to improve computational efficiency by increasing vector length (VL) and refining memory access strategies. Initial adjustments expanded the loop size from 8 to 64 elements, enhancing performance but still far from the accelerator's maximum potential of 256-element VL. We examined the challenges of adapting the GPU-optimized `vlasiator_gpu` *branch for CPUs, where vectorization is constrained by the* `Vectorclass`*'s 64-element limit. Our benchmarking showed that unit-strided loads outperform indexed loads, providing a 2.5× performance gain for larger cell sizes. Additional optimizations, including loop fusion and array reference passing, led to a 2.14× speedup. This study offers key insights into optimizing scientific simulations on RISC-V, with an emphasis on maintaining portability.*

Context

European Centers of Excellence are European projects that bring together scientific communities with shared research interests. One such example is Plasma-PEPSC, a Center of Excellence focused on plasma physics. The project aims to advance five plasma physics codes (Vlasiator, PIConGPU, BIT1, GENE, and GENE-X) and optimize their performance for the deployment on EuroHPC's exascale and pre-exascale computing clusters. As part of this effort, a co-design task is being conducted with European architectures.

EPAC-VEC

The Barcelona Supercomputing Center (BSC) leads the testing and studying of plasma physics codes on the RISC-V vector processor EPAC-VEC, developed within the European Processor Initiative (EPI). EPAC-VEC is a RISC-V processor featuring an in-order scalar core developed by Semidynamics and a vector processing unit developed by BSC capable of processing up to 256 double-precision data elements per instruction. The processor implements several extensions, including RVV0.7.1, and was fabricated using GlobalFoundries' 22nm GF22FDX technology, with tapeout completed in early 2023 and bring-up finalized in October 2023. EPAC-VEC supports booting fully-fledged Linux distributions. Within the EPI project, a compiler based on LLVM is being developed, offering support for C/C++/Fortran with intrinsics and pragma omp simd. Significant effort is being dedicated to enhancing the compiler's auto-vectorization capabilities.

*Corresponding author: `gerard.oliva@bsc.es`

Vlasiator

The Vlasiator code, part of the Plasma-PEPSC CoE, is a cutting-edge simulation tool for studying plasma physics in space and astrophysical environments. Developed by researchers, it uses a hybrid kinetic approach, blending particle-in-cell (PIC) and magneto-hydrodynamic (MHD) methods to model plasma with exceptional accuracy. Unlike traditional MHD, Vlasiator resolves ion velocity distribution functions, capturing kinetic effects like wave-particle interactions, non-thermal particle acceleration, and turbulence. This makes it ideal for studying Earth's magnetosphere, solar wind, and other space plasma systems. Highly parallelized, it leverages modern HPC architectures to meet the computational demands of kinetic simulations. Vlasiator has advanced understanding of space weather, magnetic reconnection, and shock waves, offering insights crucial for scientific research and applications like satellite operations. Its open-source nature promotes collaboration and innovation in plasma physics, driving new methodologies and expanding the field.

Contribution to the summit

Our presentation will highlight the co-design work conducted on Vlasiator, following the software development vehicle methodology outlined in [1] and leveraging the RISC-V architecture. The process involves three key steps. First, the Vlasiator code is compiled and executed on commercial RISC-V platforms, specifically using a compute cluster with nodes based on SiFive Unmatched boards. Second, the code is com-

piled using the LLVM compiler developed within the European Processor Initiative (EPI), which supports auto-vectorization, and is then emulated using various emulator types. Finally, the code is executed and its performance is analyzed on the prototype RISC-V processor developed within the EPI framework. This structured approach ensures a comprehensive evaluation of Vlasiator's performance across different RISC-V environments.

The efforts of the study have been focusing on vectorising the Vlasov acceleration and translation solvers. Here we report a summary of the work and results achieved that may be included in the talk:

Vector Length Increase

- The application has a low vector length (VL), limiting performance. In the main branch of Vlasiator, *VECL* is capped at 8 doubles or 16 floats.
- The Cudasiator branch, designed for GPUs, increases *VECL* up to 64 doubles but is still under development and less stable than the main branch.
- To adapt the mini-app, hard-coded VL limits were removed, a new constructor was added for arbitrary vector sizes, and *WID* was increased from 4 to 8.
- Tests with *VECL* values of 8, 16, 32, and 64 (keeping *WID* at 8) showed that VL scales with *VECL*, while vector instruction count remains constant.
- FPGA tests confirmed that increasing *VECL* improves efficiency, with the scalar version slowing down while the vectorized version maintains performance.

Performance Analysis and Optimisation

- Profiling revealed that "compute filtered face values derivatives" function of the code consumed the most execution time after previous optimizations.
- The function had a low proportion of vector instructions (Vector Mix) at only 3%, and a high Vector Memory Mix (49.02%), indicating inefficient vectorization and excessive memory operations.
- Applying loop fusion improved performance by reducing scalar loop control instructions, increasing the Vector Mix to 8% and lowering execution cycles.
- Further analysis showed that scalar copies of arrays were introduced by the compiler, which were eliminated by passing arrays by reference, increasing the Vector Mix to 19%.
- The final profiling confirmed that "compute filtered face values derivatives" was no longer the dominant runtime bottleneck, achieving a 2.14× speedup.

Further optimizing the Vector Length

- Previous work increased loop size from 8 to 64 elements, improving vector length (VL), but still far from the accelerator's 256-element capability.
- The `vlasiator_gpu` branch offers more parallelism but is optimized for GPU execution, limiting vectorization on CPUs.
- The innermost loops rely on the `Vectorclass`, restricted to 64 elements per cell, preventing straightforward VL expansion.
- Grouping multiple cells into a single vector would increase VL but requires major data layout restructuring due to sparse memory distribution.
- We can circumvent this restructuring by using indexed loads, allowing to fetch multiple non-contiguous cells into a vector. However, benchmark results show poor performance beyond a cell size of 16.
- Without restructuring, the best approach remains unit-strided loads, as they outperform indexed loads by 2.5× for large cell sizes.
- Placing groups of cells in contiguous memory could enable larger VL without indexed loads but would complicate Vlasiator's data layout.

Conclusions

The presentation will provide insights of the study of a complex code of Plasma physics running on a RISC-V vector CPU. This study shows the viability of a thorough performance analysis of a complex scientific code on a prototype platform. A similar study has been conducted on a CFD code and published in [2] The talk will show how powerful analysis tools and methodology allow to leverage the RISC-V vector architecture of EPI maintaining the code portable.

References

[1] Filippo Mantovani et al. "Software Development Vehicles to enable extended and early co-design: a RISC-V and HPC case of study". In: *International Conference on High Performance Computing*. Springer. 2023, pp. 526–537.

[2] Marc Blancafort et al. "Exploiting long vectors with a CFD code: a co-design show case". In: *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE. 2024, pp. 453–464.