Is RISC-V Ready For Massively Parallel Astrophysical Code?

Nitin Shukla
Coordinator AstroPlasma Physics
HPC Application Engineer CINECA
International Workshop on RISC-V for HPC
ISC CCH Hamburg · June 2026

Collaboration: Acknowledgment Of All Academic And Industry Partners

Jenny Lynn Almerol

SISSA, Trieste · CINECA, Casalecchio di Reno

Geray S. Karademir

Universitäts-Sternwarte, Munich, Germany

Elisabetta Boella

E4 Computer Engineering SpA, Scandiano

Giacomo Madella

DEI, Università degli studi di Bologna

Federico Ficarelli

CINECA, Casalecchio di Reno

Andrea Bartollini

DEI, Università degli studi di Bologna

Emanuele Venieri

DEI, Università degli studi di Bologna

SCAN ME

Can RISC-V Run Production-scale Scientific Simulations Efficiently?

HPC is Diversifying

Modern supercomputers integrate CPUs, accelerates, specialized technologies

x86 ARM GPU RISC-V FPGA QPU

Monte Cimone: RISC-V in Practice

A real RISC-V HPC cluster at Università di Bologna — deploys and operates state-of-the-art RISC-V compute nodes.

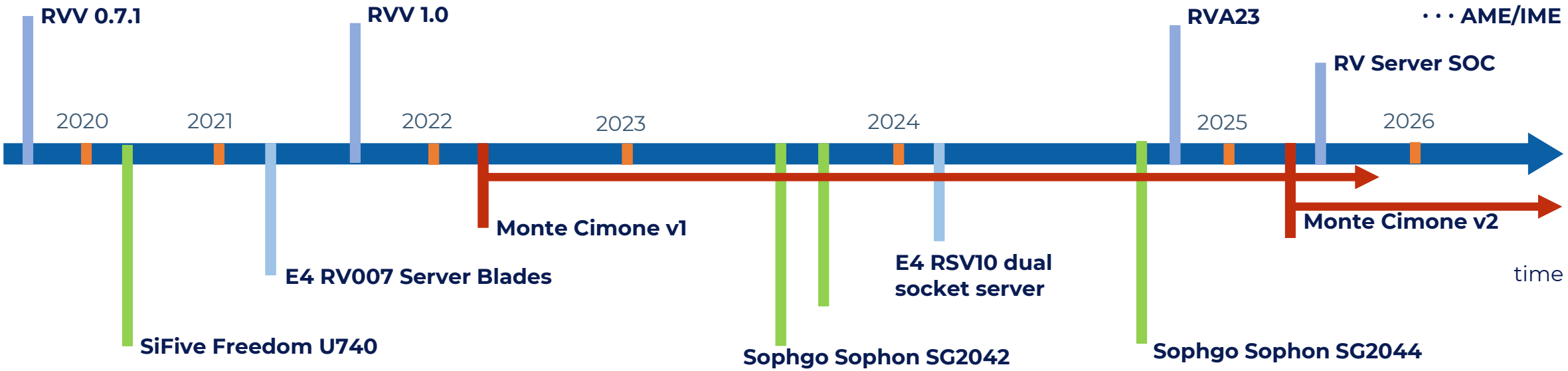
RISC-V: Open ISA, Flexibility & Independence

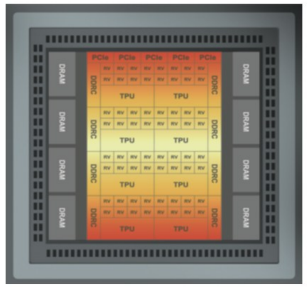
No licensing fees · No vendor lock-in · Offers flexibility to reshape future HPC systems · A growing interest in the community.

The Real Question

Can RISC-V support production-grade, massively parallel applications at scale? How well at production scale?

RISC-V HPC & Monte Cimone Timeline





Sophgo Sophon SG2044

- First server cpu with RVV1
- 64 cores RV64GCV
- 64M L3 cache
- LPDDR5 memory
- 40 PCIe Gen 5 lanes

Courtesy:

Prof. Andrea Bartolini

University of Bologna, DEI, ECS-Lab Italy

Monte Cimone v3: https://arxiv.org/abs/2605.22831

CINECA

Three Production Applications with Large User Bases

From the EuroHPC SPACE Centre of Excellence — selected for complementary workload profiles

iPIC3D

MEMORY-BOUND

Semi-implicit Particle-in-Cell code for collisionless plasma dynamics at kinetic scales

Key Kernels

Particle Mover
Moment Gatherer
Field Solver

Implementation: MPI · C/C++

Dominated by particle-to-grid interpolation. Non-unit-stride memory access. Implicit solver reduces MPI communication.

PLUTO

COMPUTE-BOUND

Finite-volume MHD code for computational plasma astrophysics on structured grids

Key Kernels

Riemann Solvers
Stencil Update
Time Integration

Implementation: MPI · C/C++

Intensive FP arithmetic in Riemann solvers (Roe, HLLD, HLLC). WENOZ reconstruction. MHD halo exchanges limit parallel efficiency.

OpenGadget-3

HYBRID

Full-physics N-body + SPH cosmological simulation code

Key Kernels

Gravity Tree (Oct-Tree)
SPH/MFM
PM: FFT via FFTW

Implementation: MPI · OPENMP · C/C++

Memory-bound tree traversals + compute-heavy FFTs. Hilbert curve decomposition. Hybrid MPI+OpenMP parallelization.

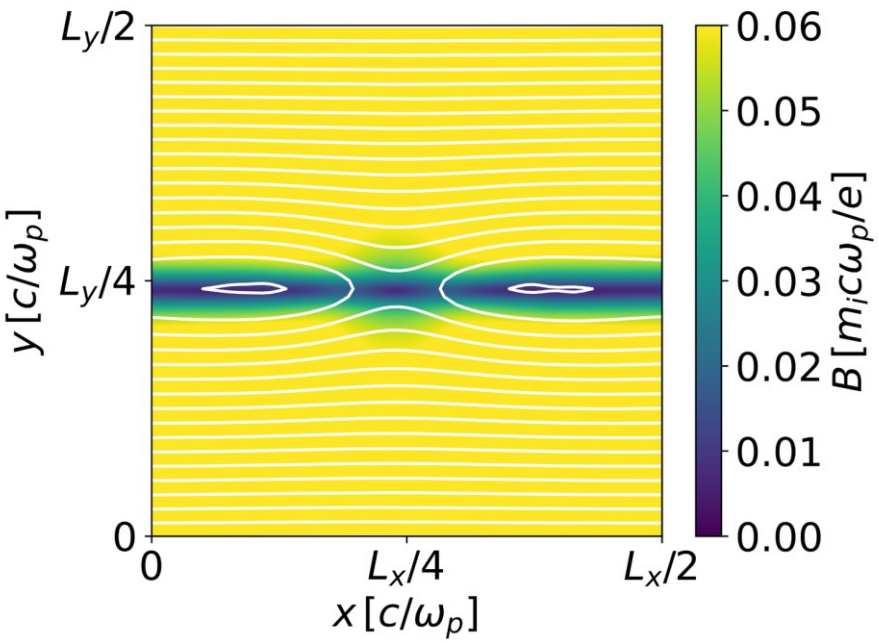
Experimental Setup

Compiler flags (all): -O3 -ffast-math -funroll-loops -ftree-vectorize + arch-specific: znver4 (AVX-512) · armv9-a/SVE 128-bit · rv64gcv -mrvv-max-lmul=m8

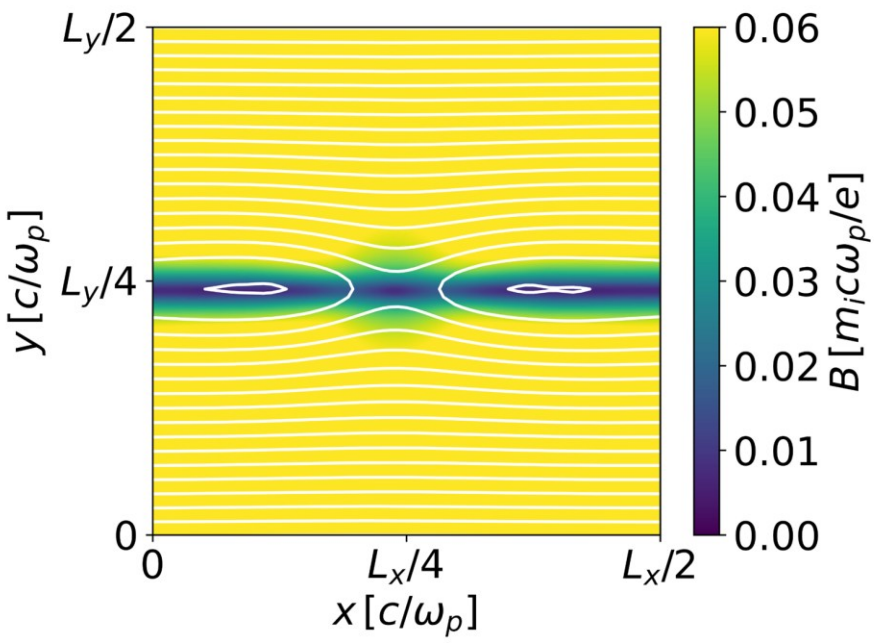
Specification	x86	ARM	RISC-V
System/Chip	AMD EPYC 9554	NVIDIA GH200 (Grace)	Sophgo SG2044
Microarchitecture	AMD Genoa	ARM Neoverse-V2	T-Head C920v2
ISA	x86	AArch64	RV64GC
Sockets	2	1	1
Total Cores	128	72	64
Peak Clock	3.76 GHz	3.47 GHz	2.50 GHz
L1-D / L1-I	32/32 KiB	64/64 KiB	64/64 KiB
L2 Cache	1 MiB/core	1 MiB/core	2 MiB/4-core
L3 Cache	32 MiB	114 MiB	64 MiB (shared)
NUMA (CPU)	2 (1/socket)	1 (CPU only)	1
SIMD/Vector	AVX-512, AVX2	SVE2 (128-bit)	RVV 1.0 (128-bit)
Memory	DDR5-4800	LPDDR5X	DDR5-4266
Compiler	GCC 12.4.0	GCC 12.4.0	GCC 14.2.0
MPI	OpenMPI	OpenMPI	OpenMPI

RISC-V platform delivers correct physical results

ARM



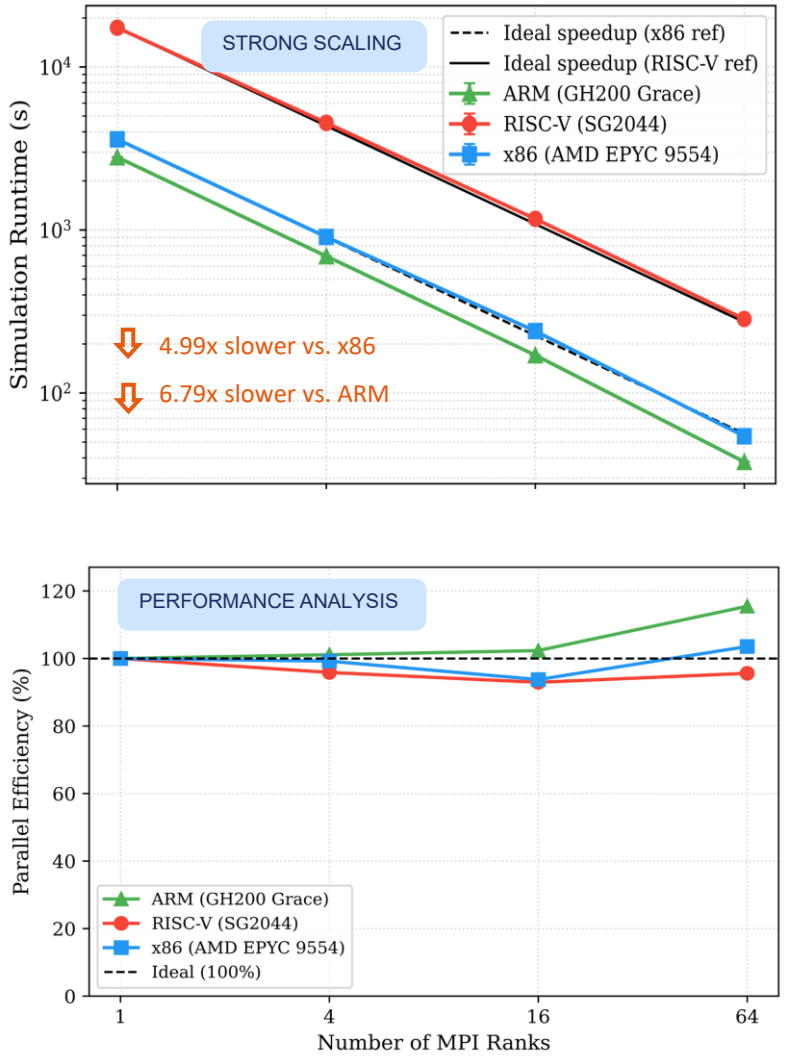
RISC-V

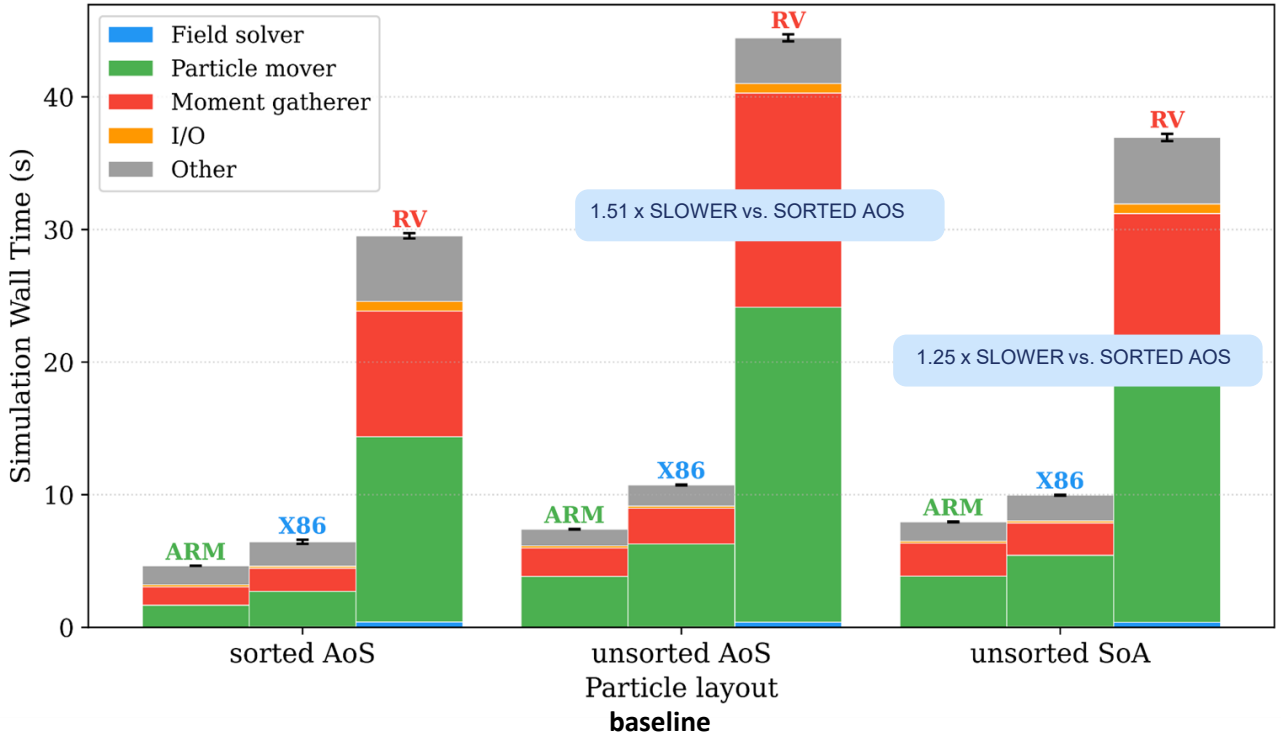


Evaluated iPIC3D employing the same setup on a 256 ×256 ×1 spatial grid with 9216 particles per cell.

Magnitude of the magnetic field on the simulation plane with field lines overlaid in white, during an iPIC3D simulation of the GEM reconnection

iPIC3D: Enhancing Performance via Particle Data Layout Transformation





Comparing the end-to-end time across three particle memory layouts

SoA improves cache efficiency via contiguous particle data layout

CINECA

iPIC3D: sorting particles enables vectorization on x86 and ARM, but not RISC-V

Auto-vectorization per kernel across architectures (-fopt-info-vec)

Core Finding:

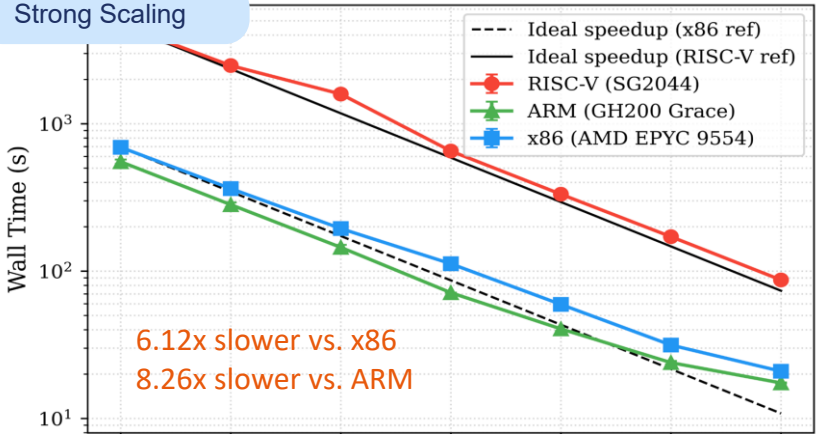
On RISC-V (GCC 14.2), the particle mover generates only 1 partial basic block vs 9 (ARM) and 3 (x86). The critical moment gathering kernel remains completely unvectorized on RISC-V, even with sorted AoS — exposing a gap in GCC RVV 1.0 auto-vectorization maturity.

Kernel	RISC-V (RVV)	x86 (AVX2)	ARM (SVE)
Particle Mover			
mover_PC (AoS)	no vectorization	no vectorization	no vectorization
mover_PC (SoA)	1 partial BB (16 B)	3 partial BBs (32 B)	9 partial BBs (16 B) ✓
Moment Gatherer (critical scatter kernel)			
sumMoments_AoS (unsorted)	no vectorization	no vectorization	no vectorization
sumMoments_vec_AoS (sorted)	no vectorization ✗	~4 partial BBs ✓	~4 partial BBs ✓
sumMoments_vec_SoA (sorted)	no vectorization ✗	~7 partial BBs ✓	~10 partial BBs ✓
Grid/Solver utilities (non-critical path)			
dot / norm / BLAS-like	RVV VL (scalable) ✓	AVX2 32B ✓	SVE scalable+16B ✓
grid stencils (grad, Lapl.)	RVV VL (scalable) ✓	AVX2 32B ✓	SVE scalable+16B ✓

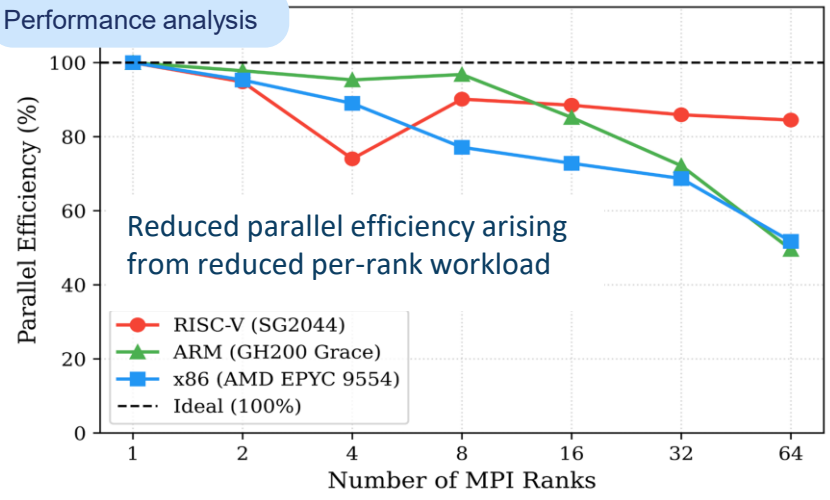
The observed performance improvement is therefore driven by increased spatial locality rather than SIMD utilization

PLUTO: RISC-V Scales Efficiently to Higher Core Counts

Strong Scaling



Performance analysis



NP	RISC-V (s)	ARM (s)	Slowdown vs ARM	x86 (s)	Slowdown vs x86
1	4711.5 ± 7.0	553.0 ± 63.3	**8.52×**	**692.7 ± 0.4**	**6.80×**
2	2484.9 ± 5.0	282.7 ± 32.3	**8.79×**	**363.4 ± 0.4**	**6.84×**
4	1591.3 ± 6.7	145.0 ± 19.4	**10.97×**	**194.6 ± 0.4**	**8.18×**
8	653.3 ± 2.6	71.4 ± 0.2	**9.15×**	**112.3 ± 0.4**	**5.82×**
16	332.7 ± 0.8	40.6 ± 4.4	**8.20×**	**59.5 ± 0.1**	**5.60×**
32	171.4 ± 0.4	23.9 ± 2.1	**7.16×**	**31.5 ± 0.0**	**5.44×**
64	87.1 ± 0.2	17.4 ± 0.5	**5.00×**	**20.9 ± 0.1**	**4.16×**

Table · Grid: 256^3 · 10 steps · RISC-V slowdown vs ARM and x86

This behavior is mainly due to reduced single-core performance, since the PLUTO MHD solver depends on floating-point-intensive kernels sensitive to hardware throughput and data-level parallelism

CINECA

PLUTO: Performance Gap Is Both Software And Hardware

Kernel	x86 (AVX-512)	ARM (SVE-128)	RISC-V (RVV)	Δ dp
Riemann solvers — innermost hot path				
`roe.cpp` — main flux loop	64 B : 8 dp	no vectorization	8 B : 1 dp : scalar	8×↓
`hlld.cpp` — HLLD solver body	64 B partial block	no vectorization	no vectorization	—
`hllc.cpp` — solver body	64 B partial block	no vectorization	no vectorization	—
`hll.cpp` + `tvdlf.cpp` + `rhs.cpp`	no vectorization	no vectorization	no vectorization	—
Reconstruction				
`array_reconstruct.cpp`	64 B : 8 dp	16 B : 2 dp	128 B : 16 dp	2×↑
`mp5_states.cpp`	64 B : 8 dp : 2 sites	no vectorization	16 B : 2 dp : 1 site	4×↓
`mhd.hpp` — PrimEigenvectors loop	no vectorization	16 B : 2 dp	128 B : 16 dp	—
`mhd.hpp` — inlined bb partials	64 B : 8 dp	16 B : 2 dp	16 B : 2 dp	4×↓
Stencil update / time integration				
`update_stage.cpp`	64 B : 8 dp	16 B : 2 dp	128 B : 16 dp	2×↑
`rk_step.cpp`	64 / 32 / 16 B	16 B : 2 dp	128 B : 16 dp	2×↑
Constrained transport				
`ct_emf_average.cpp`	64 / 32 / 16 B	16 / 8 B	128 / 16 B	∼
`ct_update.cpp`	64 B : 8 dp	16 B : 2 dp	128 B : 16 dp	2×↑

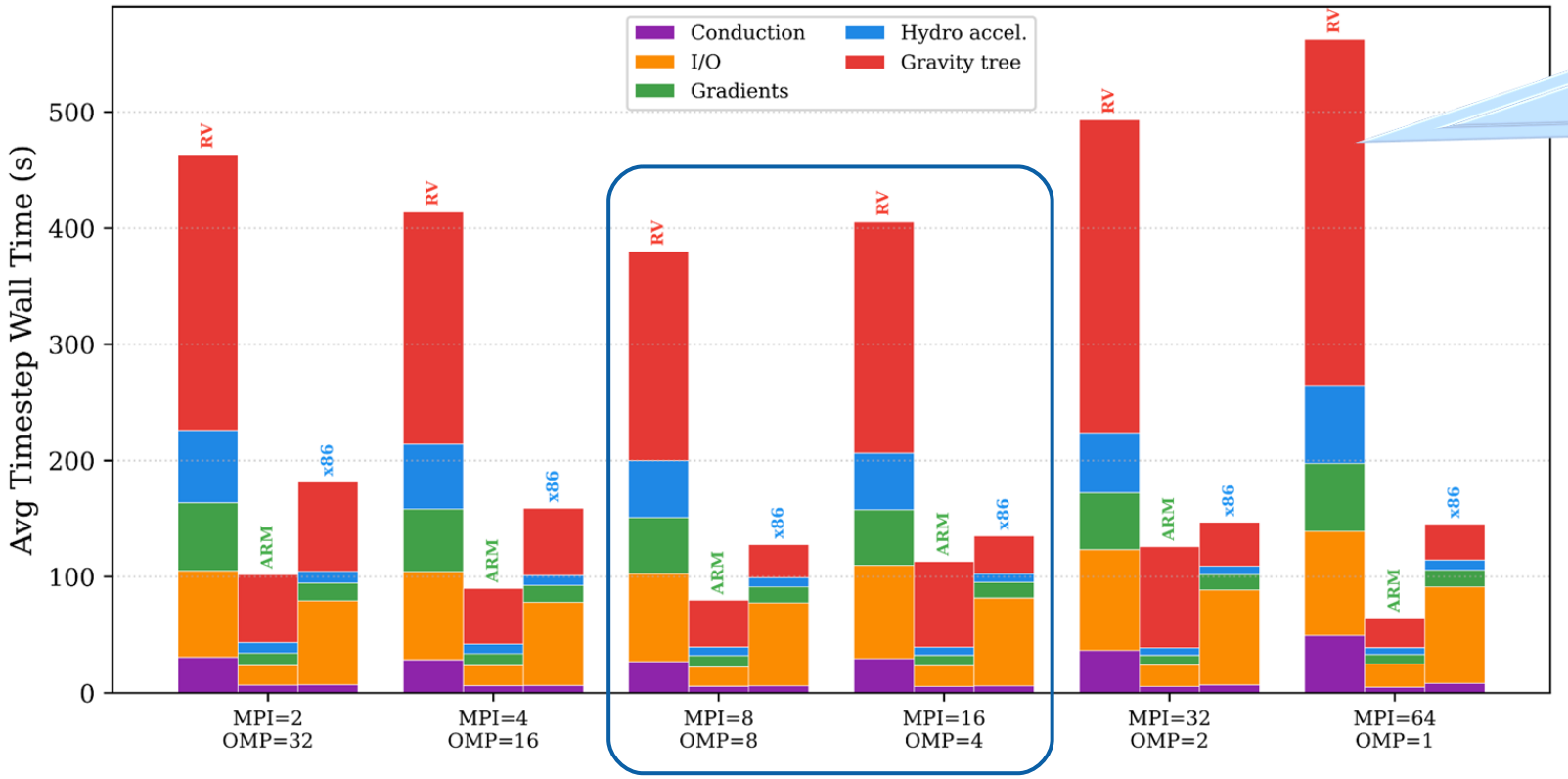
The performance difference is primarily driven by hardware limitations and weaker vectorization compiler support on RISC-V, where 89% of auto-vectorized loops are guarded by runtime aliasing checks (vs. 72% on x86), leading to more frequent scalar fallbacks and lower effective vector utilization

Opengadget3: Gravity Tree is particularly resource-intensive on RISC-V

The gravity tree and hydrodynamic acceleration phases are memory-bound and involve irregular, indirect pointer- chasing traversals over particle data

Note

To optimize locality, ranks were bound to contiguous cores aligned with shared hardware domains.



Fastest configurations!

The gravity tree is the dominant kernel, accounting for 43-50% of total wall time for MPI ≥ 4 and up to 89% in the single-rank configuration.

CINECA

Opengadget3: memory subsystem and low throughput lead to longer times

Why the gravity tree and SPH are hard for RISC-V

Memory Subsystem Bottlenecks

Lower Bandwidth

DDR5-4266 vs DDR5-4800 on AMD. Fewer memory channels → lower aggregate throughput.

Early Bandwidth Saturation

Memory bandwidth saturated at relatively low thread count. Beyond this point, adding threads adds contention without throughput gain.

Reduced Memory-Level Parallelism

Fewer in-flight memory requests, less aggressive prefetching, smaller reorder buffer vs AMD Zen 4 / ARM Neoverse-V2.

L2 Contention in Tree Walk

Working set of gravity tree walk exceeds 2 MiB L2 cluster at high thread count → L3/DRAM fallback with non-sequential access.

C920v2 RVV: VLEN=128 bit → 2 dp/instruction vs AVX-512: 8 dp/instruction → 4× throughput gap on vectorizable kernels

Vectorization Landscape

Kernel	RISC-V	Note
Gravity tree walk	✗ no vec	Data-dependent branches (BarnesHut criterion) + indirect pointers prevent SIMD on all platforms
SPH force inner loop	✗ no vec	Irregular neighbor indexing P[ngblist[i]] — gather-like AoS access, algorithmic not compiler limitation
MFM gradient loops	16 B / 0.25 dp	Compiler multi-versions with runtime aliasing checks — low effective utilization
Tree build (aux)	16 B BB	Only basic-block level; x86 achieves 64 B (4× higher throughput)
SPH hydro aux	16 B BB	vs 64 B on x86 — 4× throughput gap
FFT (FFTW3)	RVV (lib)	Handled by library; correctness confirmed

Summary

Portability Achieved

- End-to-end evaluation of three production astrophysical codes (iPIC3D, PLUTO, OpenGadget3) on RISC-V.
- Numerical correctness verified across all platforms. Zero algorithmic changes required.

Clear Performance Gap

- RISC-V is 3–9× slower than x86 and 5–9× slower than ARM, depending on workload.
- Memory-bound and irregular kernels are most severely affected.

Root Causes Identified

- 128-bit vector units vs 512-bit
- Lower clock frequency
- Immature GCC RVV 1.0 auto-vectorization — scalar fallbacks in critical kernels
- Lower memory bandwidth & reduced MLP.

Optimization Delivers Gains

- Sorted AoS layout gives 1.51× speedup on iPIC3D via cache locality alone.
- Optimal MPI=8, OMP=8 for OpenGadget3 aligns with L2 cache cluster topology. Memory locality improvements are the most effective lever.

RISC-V Can Support Scientific Workloads

Functionally correct MPI communication, meaningful parallel speedup, and production-grade code portability — RISC-V is on the path to HPC. The gap is addressable.

Future Directions

- Kernel-level optimization study on representative workloads
- Improve vectorization (explicit + compiler auto-vectorization)
- Enhance data layout and memory locality
- Reduce irregular memory access patterns
- Explore architectural needs:
  - higher memory bandwidth (e.g., HBM)
  - improved cache hierarchy
  - wider vector units
- Strengthen RISC-V compiler ecosystem for HPC workloads

Related Work & Context

Brown & Jamieson (2025)

SG2042/SG2044 HPC performance characterisation — initial RaJaPerfSuite evaluation. Baseline architecture benchmarking on RISC-V.

Diehl et al. (2023, 2024)

HPX + Kokkos + OctoTiger on RISC-V — promising scalability, some cases outperforming optimized x86. Confirmed growing RISC-V maturity for mini-apps.

Almerol et al. (2025)

N-body simulations on Tenstorrent Wormhole RISC-V AI chip — performance exceeding CPU implementations in specific cases.

Bartolini et al. (2022)

Monte Cimone v1 cluster — first-generation RISC-V HPC cluster evaluation at Univ. Bologna.

Venieri et al. (2026)

Monte Cimone v2 — updated cluster evaluation and optimization. Deployment environment for this work.

Shukla et al. (2025–26)

EuroHPC SPACE CoE: exascale astrophysics codes. This work extends SPACE results to RISC-V evaluation.

Grazie

n.shukla@cineca.it