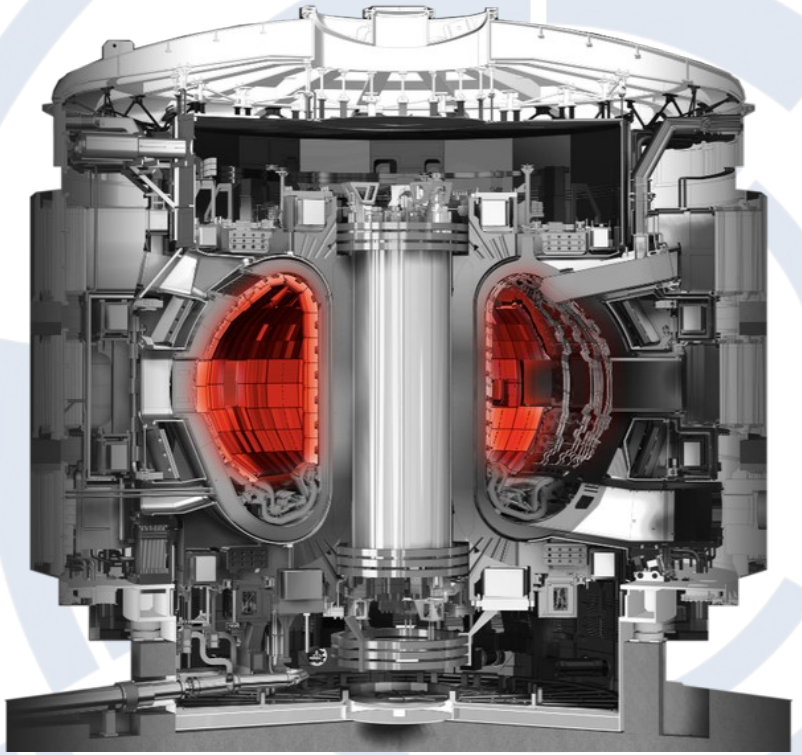
Enabling Exascale GPU Performance for Fusion Edge Modeling Codes



ERO2.0 GPU/GCS migration: results, constraints and lessons learned

A. Maidana, A. Soba, M. J. Mantsinen

Barcelona Supercomputing Center (BSC) | EUROfusion ACH-BSC

4th Annual Meeting of EUROfusion HPC ACHs

27 October 2026 | 14:35–15:00 | IPP Garching, Germany

This work has been carried out within the framework of the EUROfusion Consortium, funded by the European Union via the Euratom Research and Training Programme (Grant Agreement No 101052200 — EUROfusion). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

ERO2.0 AT A GLANCE

Fusion-edge particle transport: why the migration is challenging

PHYSICS / CPU	Particle trajectories, surface contact, collision events, re-emission and deposition
PARALLELISM	MPI ranks + OpenMP threads; dynamic load balancing; irregular particle histories
GPU/GCS EXTENSION	Geometry/proximity queries are batched and offloaded to CUDA, with iterative GPU octree search (DFS and BFS)
SCIENTIFIC CONTRACT	Preserve particle histories, final states, density, PFCS and statistics

EXECUTIVE SUMMARY

Key message

WHAT WORKS

Scientific equivalence in validated cases

WHAT REMAINS

No robust end-to-end speedup demonstrated

WHAT WE LEARNED

GPU geometry is not a GPU application

Core message: GPU kernels are necessary but not sufficient while the particle-transport step (TPS) stays on the CPU.

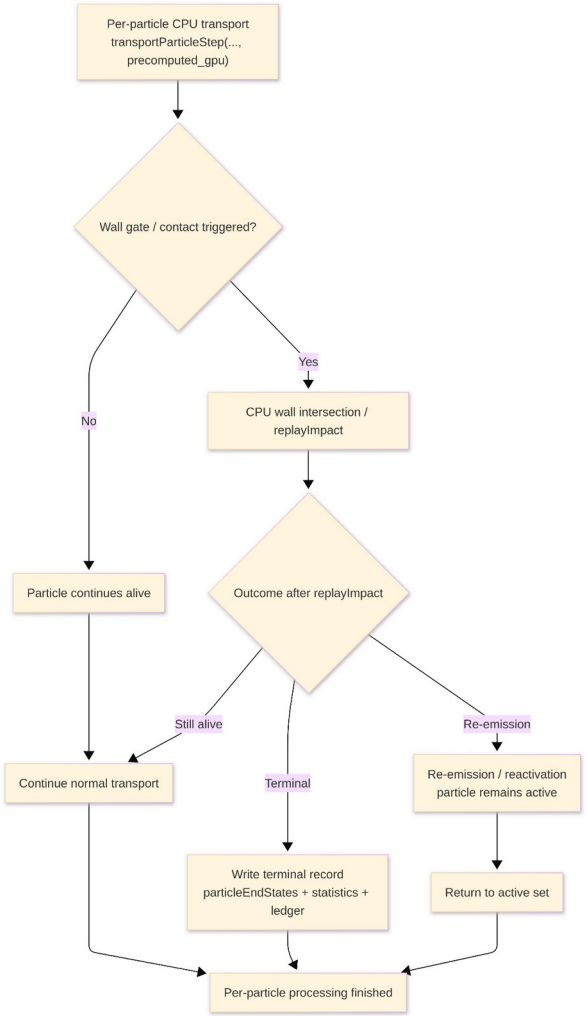
WHAT WE SET OUT TO ACHIEVE

Original objective and hypothesis

- Reduce total simulation time with GPUs, not merely accelerate isolated kernels.
- Preserve TPS physics and the consistency of scientific results.
- Retain MPI/OpenMP while offloading geometry queries to CUDA.
- Evolve a codebase originally designed for CPUs.

WHY END-TO-END GPU SPEEDUP IS HARD IN ERO2

The GPU assists geometry, but the per-particle lifecycle is still governed by CPU control flow.

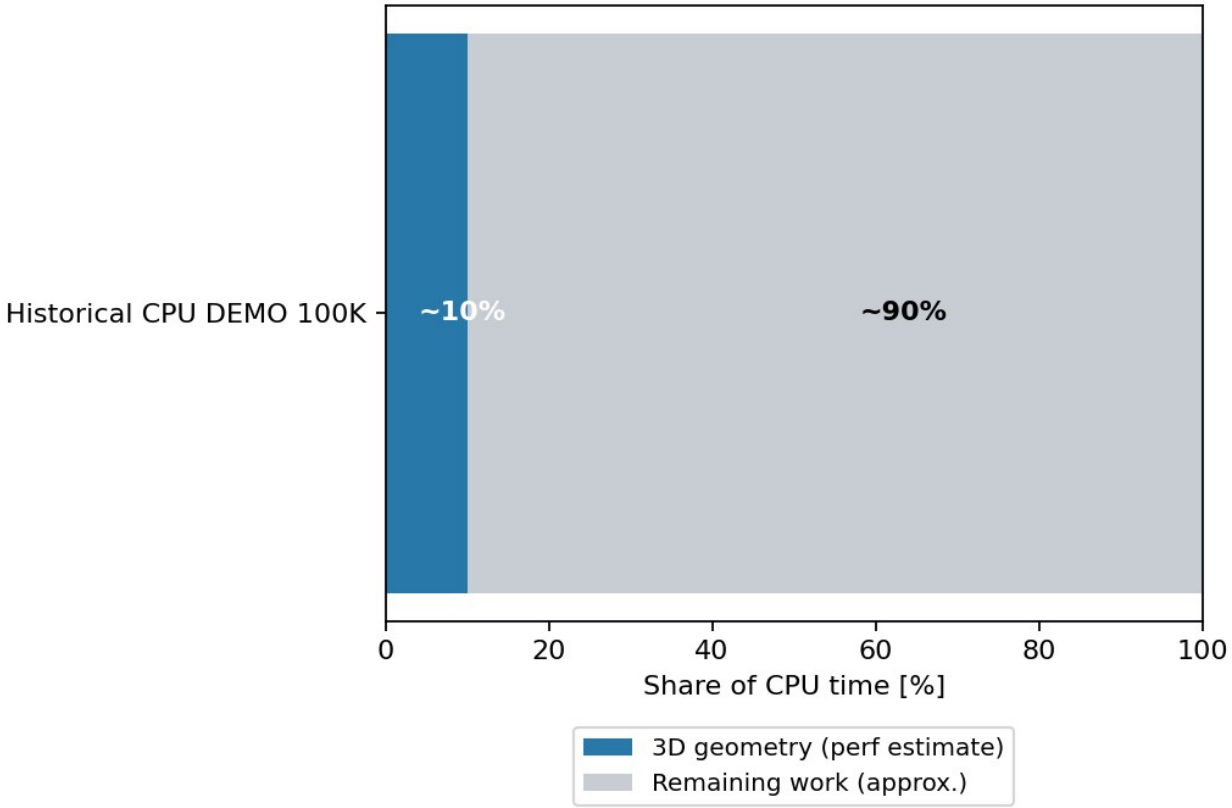


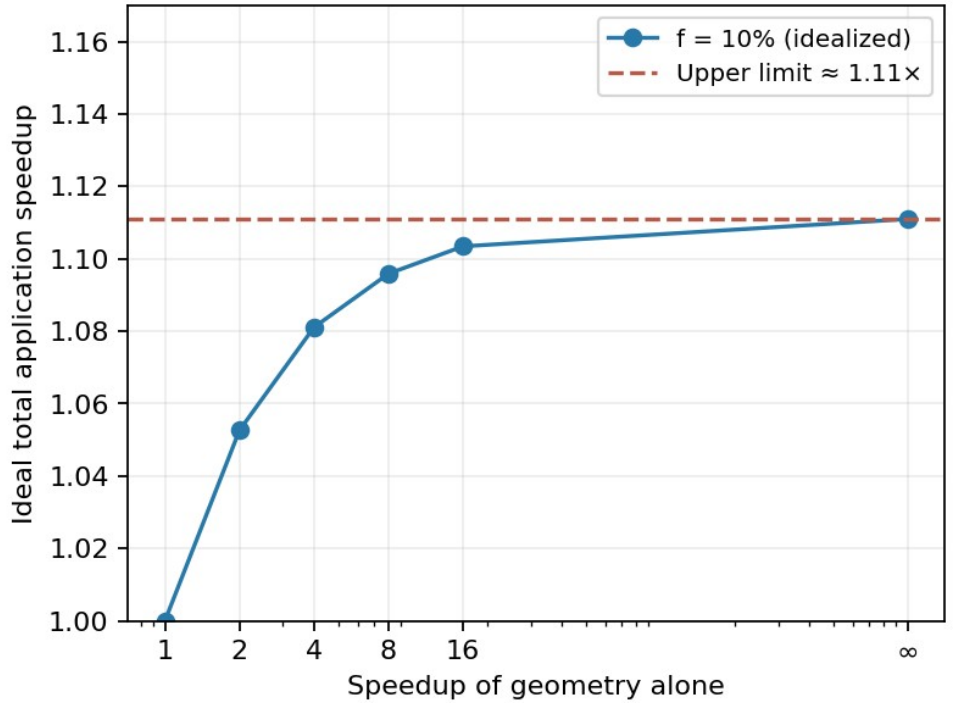
CPU: select the particles still alive
GPU: batched geometry / proximity queries
CPU: transport step (TPS) and physical outcomes
CPU: wall impacts, re-emission, bookkeeping, communication

Architectural implication
The GPU accelerates a subset of an irregular, CPU-controlled particle lifecycle. Launch, batch and synchronization costs may offset kernel gains.

MEASURED CPU FRACTION → AMDAHL CEILING

Geometry was only ~10% of CPU time in the historical DEMO 100K profile





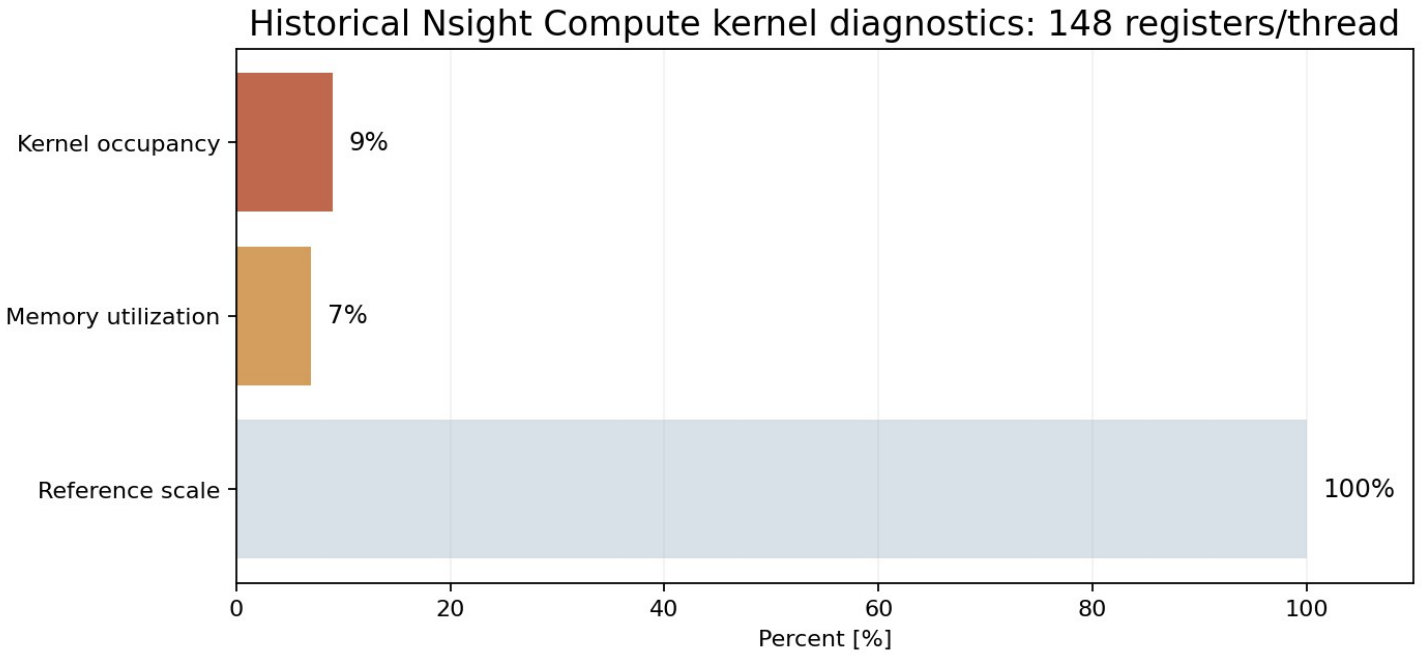
Amdahl: speedup = 1 / [(1 − f) + f / s]; with f = 0.10, even infinitely fast geometry gives at most 1 / 0.90 ≈ 1.11×.

This is a workload-specific, historical estimate, not a measured C1 speedup.

Source: ERO2.0 on JURECA: G6/A3 close-out report, 22 Oct 2026; perf CPU profile; model is idealized

NSIGHT COMPUTE: REGISTER PRESSURE

Why an earlier GPU kernel left the GPU mostly idle



148 registers per GPU thread

~9% kernel occupancy

~7% measured memory utilization

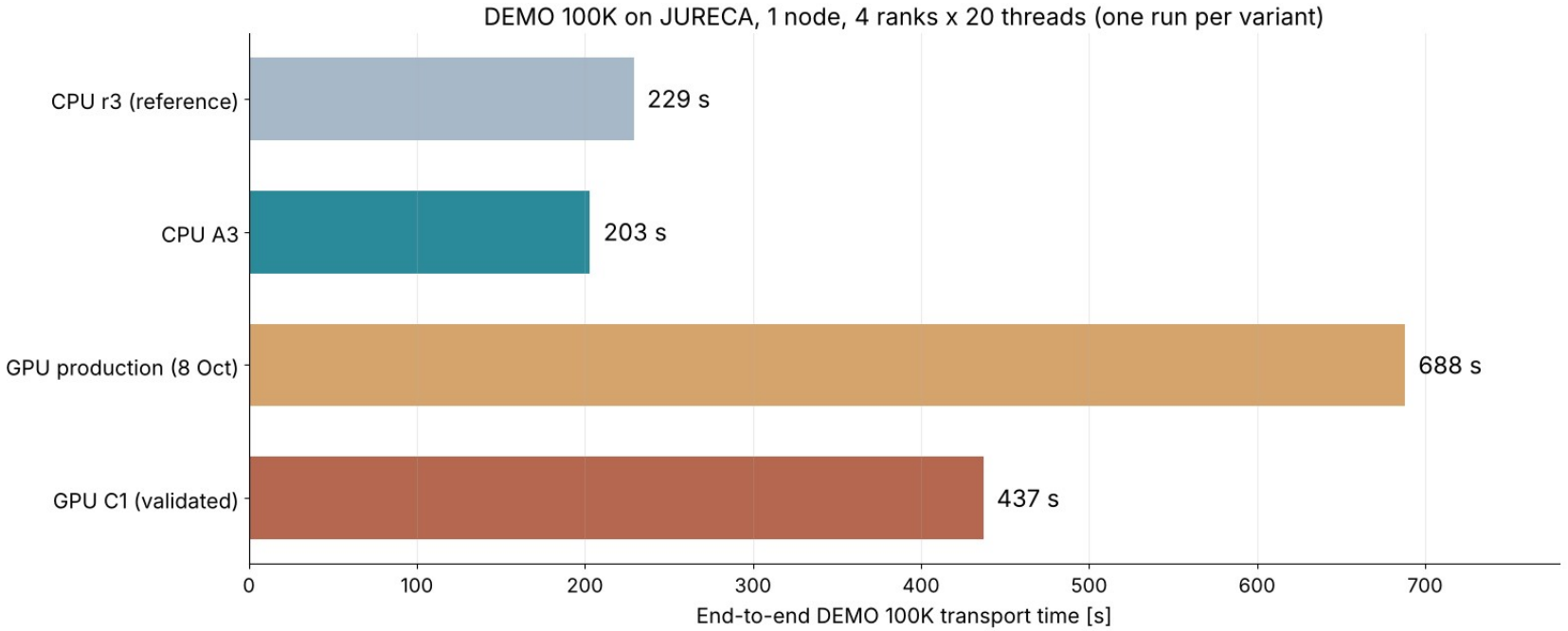
Fix: cap at 64 registers, ~4× more work in flight.

Diagnosis of an earlier kernel, not of final C1.

Source: ERO2.0 on JURECA: G6/A3 close-out report, 22 Oct 2026; ncu diagnostic

OPTIMIZATION GAINS ≠ CPU–GPU SPEEDUP

GPU improved within its own path, while CPU still remained faster



GPU path: 688 → 437 s (C1)

~1.57× faster GPU

CPU path: 229 → 203 s

~11.4% less time

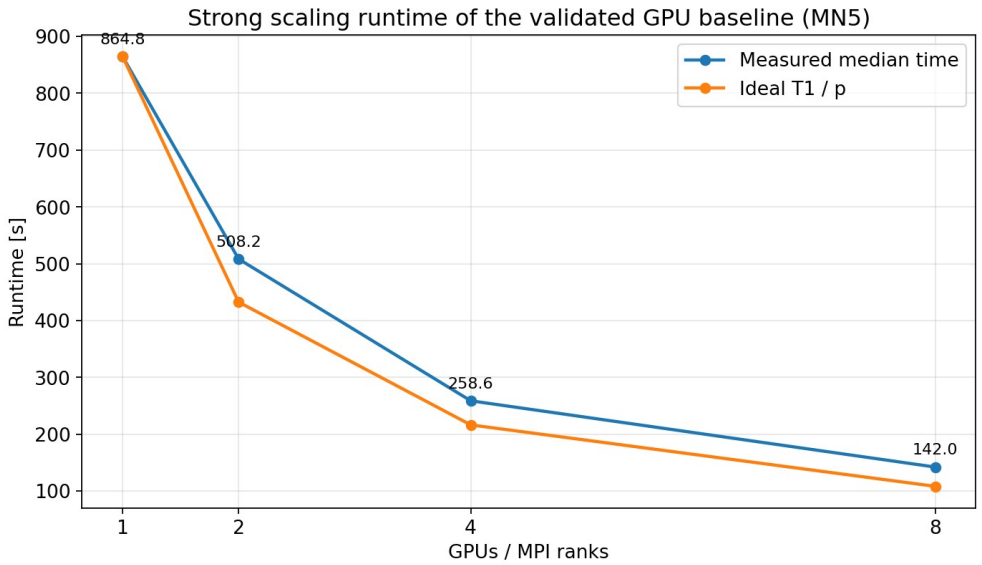
Best GPU / best CPU:

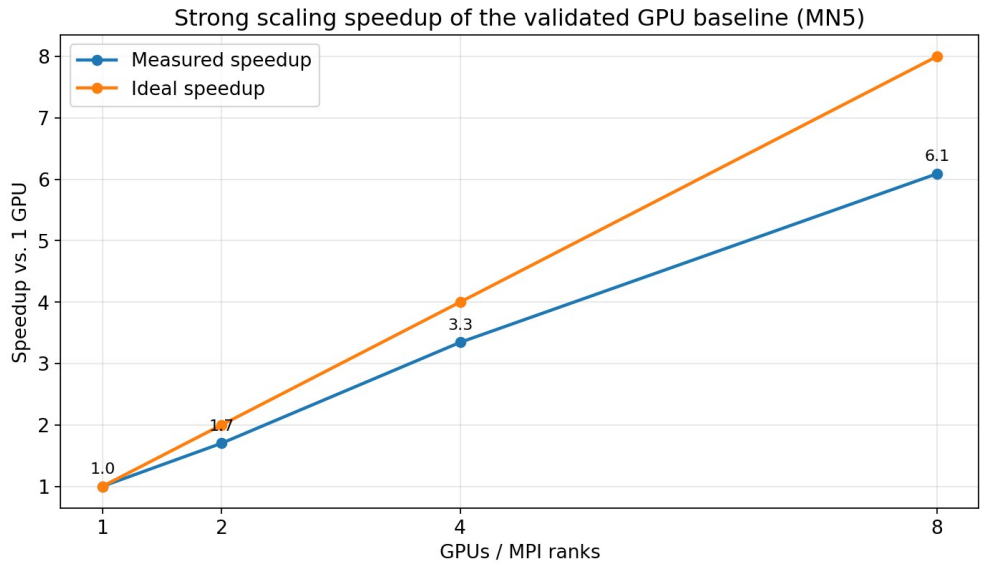
437 / 203 ≈ 2.15× slower

Source: ERO2.0 on JURECA: G6/A3 close-out report, 22 Oct 2026; full uninstrumented 100K runs; C1 DEMO job 15732600

GPU BASELINE: STRONG SCALING

100,000 particles total | 1, 2, 4, 8 H100 GPUs | MareNostrum 5





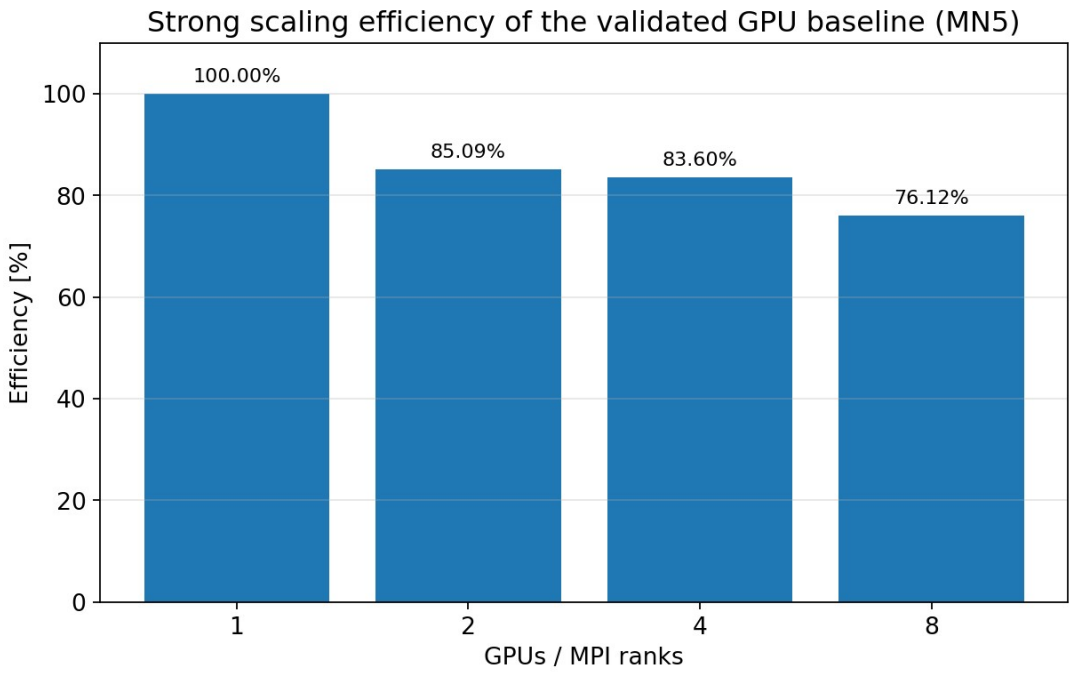
GPUs	Measured time	Ideal time (T1 / p)	Measured speedup	Ideal speedup	Efficiency
1	**~865 s**	865 s	**1.0×**	1×	100%
2	**~508 s**	432 s	**1.7×**	2×	85%
4	**~259 s**	216 s	**3.3×**	4×	84%
8	**~142 s**	**108 s**	**6.1×**	**8×**	**76%**

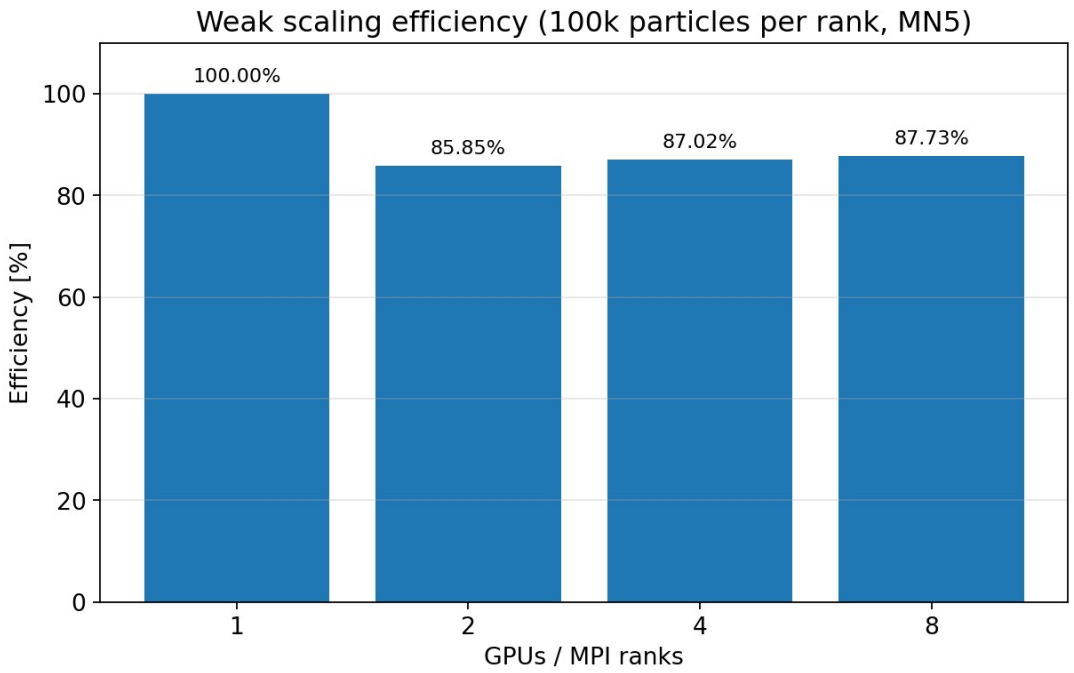
8 GPUs: 142 s median, 6.1× faster than 1 GPU (76% efficiency). This is GPU-to-GPU scaling, not GPU-versus-CPU acceleration.

MN5 P1 R5.2 · 3 retained replicas per point · total mpirun wall time

GPU BASELINE: PARALLEL EFFICIENCY

Strong: 100,000 particles total | Weak: 100,000 particles per MPI rank





At 8 GPUs: 76% strong efficiency vs 88% weak efficiency. Multi-node operation was stable once runtime environment propagation was fixed.

MN5 P1 R5.2 · 3 retained replicas per point · total mpirun wall time

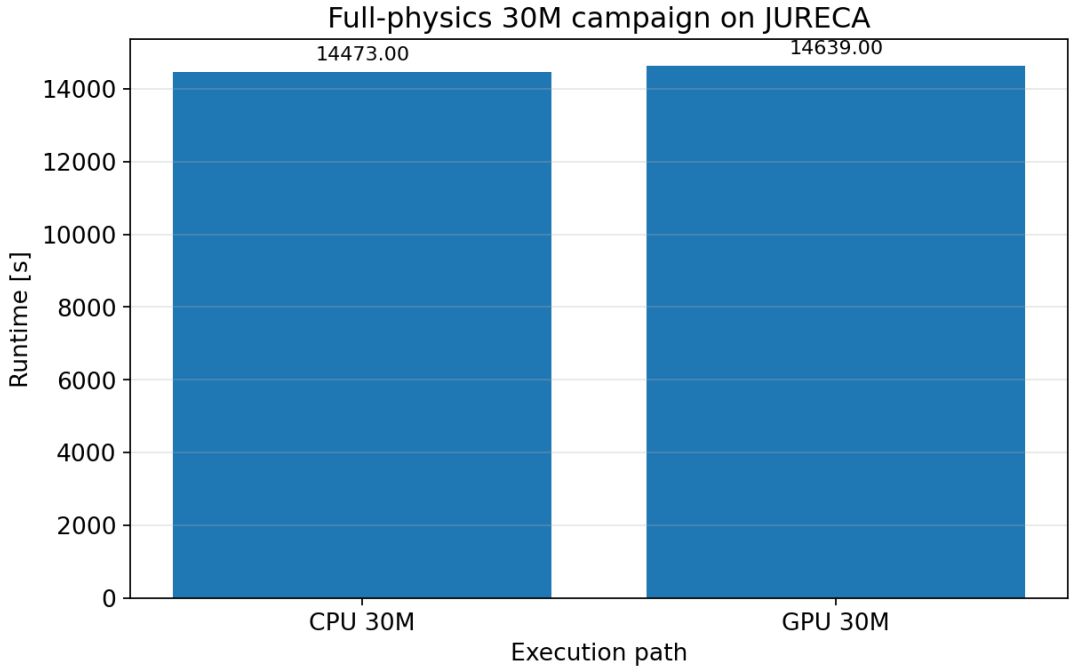
VALIDATE BEFORE OPTIMIZING

The hidden cost of equivalence

- Surface selection and geometry precision influence discrete events.
- A small discrepancy can change trajectories, re-emission, and deposition.
- We need STARTS, ENDS, density, PFCS, and statistics comparisons.
- Performance diagnosis must distinguish physics, tooling, and execution errors.

FULL-PHYSICS REALITY AND SCIENTIFIC VALIDATION

What the project did achieve: scientific equivalence and a controlled, validated GPU/GCS integration.



Historical JURECA 30M comparison
CPU 14,473 s | hybrid GPU 14,639 s
Observed end-to-end speedup: 0.989×

Important: this is a DIFFERENT campaign/configuration from the MN5 P1 R5.2 GPU-scaling results.

Validation check	C1 CPU/GPU vs master CPU	C1 GPU vs C1 CPU / accepted GPU
CASE1	PASS	PASS
CASE2	PASS	PASS
CASE3	PASS	PASS
PFCS CASE3	PASS	PASS
DEMO starts	PASS	PASS
DEMO density	PASS	PASS
DEMO ends	PASS	PASS
DEMO statistics	PASS	PASS

WHAT MADE PROGRESS DIFFICULT

Development constraints and context

- Legacy code with tight coupling among physics, geometry, and MPI.
- Combined MPI + OpenMP + CUDA concurrency and hard-to-reproduce failures.
- Performance instrumentation and scientific validation infrastructure built during the project.
- Valid scope constraint: do not migrate TPS physics to GPU without explicit approval.

WHAT WE WOULD DO DIFFERENTLY

Methodological lessons

1	**Measure first**	Break down TPS time and measure the geometry fraction.
2	**Test minimum cost**	End-to-end CPU ↔ GPU benchmarks with realistic batches.
3	**Set milestones**	Early speedup gates and stop criteria.
4	**Instrument from the start**	Counters, traces, reproducible validators from day one.

RESULTS AND LIMITS, WITHOUT SPIN

Project assessment

POSITIVE OUTCOMES

- Real GPU execution path integrated
- Iterative GPU octree traversal: DFS and BFS (DFS kept for speed)
- CPU/GPU builds available
- Reproducible scientific validation
- GitLab delivery branch published

OUTSTANDING OBJECTIVES

- Useful end-to-end speedup
- C1 scalability measurements
- Dominant bottleneck profile
- TPS redesign feasibility

BEFORE INVESTING FURTHER

Possible next steps

A · End-to-end profiling — TPS, geometry, transfers, queues and synchronization.

B · Performance model — Measured Amdahl fraction and overhead vs. batch size.

C · Architectural decision — Evaluate TPS redesign or limit scope if returns are insufficient.

Decision: a bounded profiling phase and a clear go/no-go gate.

THE RIGHT QUESTION

Conclusion

Running geometry on a GPU is not enough.
We must accelerate the entire critical path.

C1 provides a scientifically validated foundation.
Performance requires profiling and potentially TPS redesign.

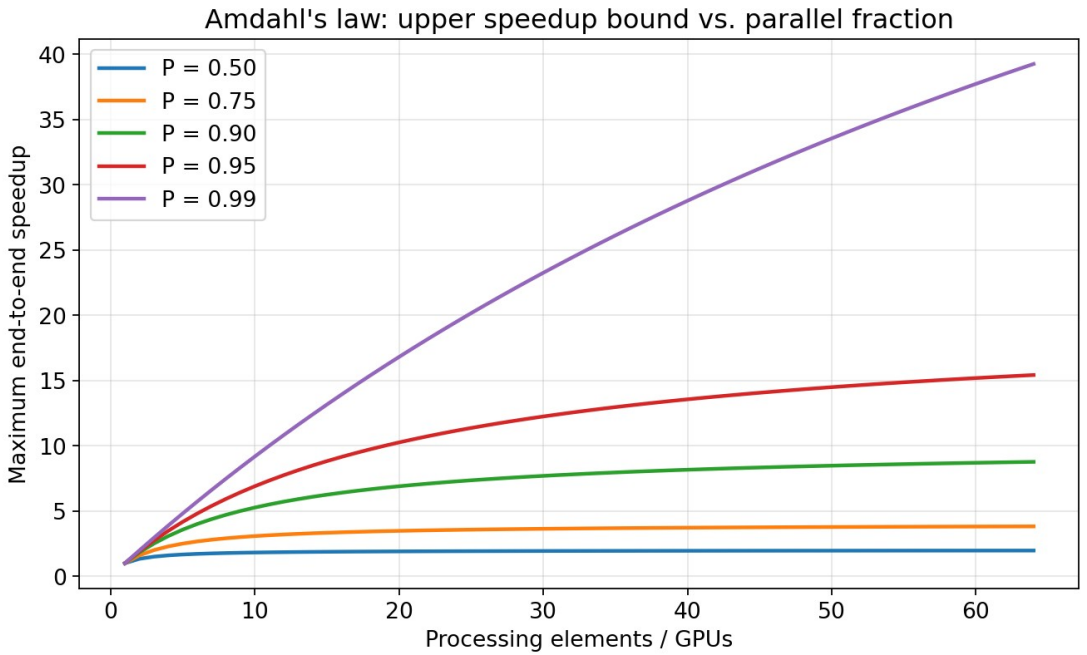
APPENDIX · TRACEABILITY

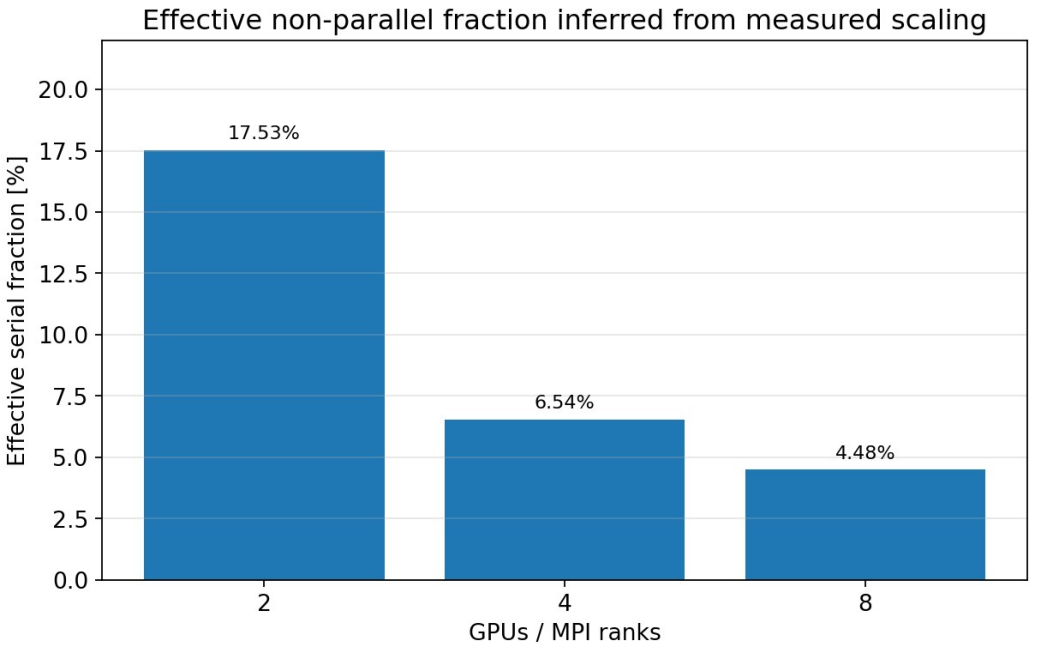
Evidence and scope

- Reference master: f99f0ee552ff21f16340ffb89fbc70f6c53152c7
- Validated C1: 2070bb60c1d96037d1999a3da4dc8016276aaa6d
- GitLab: delivery/ero2026_gpu_gcs_c1_validated
- C1 DEMO job 15732600: 437 s internal, STARTS/ENDS/density/statistics PASS.
- JURECA 30M (other setup): CPU 14,473 s; hybrid GPU 14,639 s.

BACKUP · AMDAHL LIMITS: MODEL VS MEASUREMENT

End-to-end speedup is bounded by the fraction of the workflow that is actually accelerated.





Amdahl curves: illustrative scenarios, NOT a measured TPS fraction.
Effective serial-equivalent fractions inferred from GPU strong scaling: 17.6% (2), 6.5% (4), 4.5% (8); these include overheads and are not a CPU geometry-time profile.