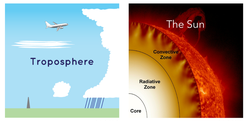
Cache-driven super-linear scaling in a Python fluid solver

Harshit Tiwari[1,2], Samar Aseeri[3], Katepalli R. Sreenivasan[1], David E. Keyes[3], Mahendra K. Verma[2]

[1] Tandon School of Engineering, New York University, USA [2] Indian Institute of Technology Kanpur, India
[3] King Abdullah University of Science and Technology, Saudi Arabia

Introduction

- Turbulence, governed by highly nonlinear Navier-Stokes equations, is one of the last unsolved problems in classical physics
- Resolving its scale cascade needs grids of billions of points → an HPC problem
- A key example: **thermal convection** – drives solar convection, Earth's atmosphere, and more



NASA NASA

DHARA: a general-purpose Python fluid solver for extreme scales

- DHARA solves a general PDE (hyperbolic, parabolic, elliptic, mixed): $\frac{\partial \mathbf{q}}{\partial t} = \mathbf{f}\left(\mathbf{q}, \nabla\mathbf{q}, \nabla^2\mathbf{q}, \ldots, t\right)$
- Structured Cartesian grids, finite difference, finite volume and spectral methods

Goal: Maximize performance, minimize development time

- Here we study one problem – **compressible convection** – where it achieves super-linear scaling.

Libraries:

NumPy (CPU), CuPy (GPU)
numba
mpi4py
h5py, matplotlib

Optimization:

Fully vectorized
Memory optimized
`numba.jit` (just in time)
`cupy.ElementwiseKernel`

DHARA Object-Oriented Architecture

- **Grid** — Grid2D, Grid3D classes (x, y, z), boundary cond, etc
- **Problem Setup** — **q**, temp arrays and functions for general problems
- **Libraries** — L-operators, Derivatives, Reconstructions, Multigrid
- **Default Problems** — Euler, incompressible & compressible NS, RBC, etc
- **Time evolution** — Runge-Kutta, IMEX, custom, etc
- **Data I/O** — In-situ saving fields and global averages, plotting, etc

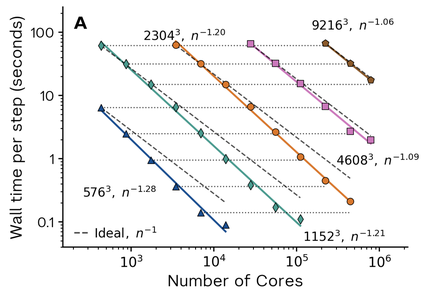
Super-linear scaling on Shaheen III (KAUST) and cache-fit effect

Shaheen III (CPUs) specifications:

HPE Cray EX, 4,608 nodes, dual-socket AMD EPYC 9654 (192 cores/node, 384 GB DDR5) – 884,736 cores total

DHARA scales up to 786,432 cores on Shaheen III with super-linear strong scaling

- Close to 90% of Shaheen used
- Doubling cores cuts step time by > 50%
- $t \propto n^{-\alpha}$: ideal linear scaling is $\alpha = 1$, ours $\alpha > 1$ (Plot **A**)
- `Float32` shows same trend, ~ **2X** faster
- Good weak scaling: step time flat as grid grows



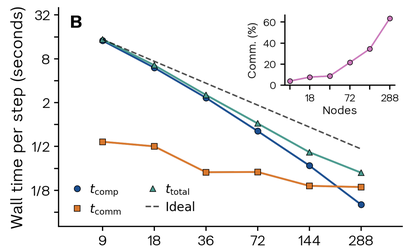
Profiling the 1152^3 grid case using TAU (Tuning and Analysis Utilities)

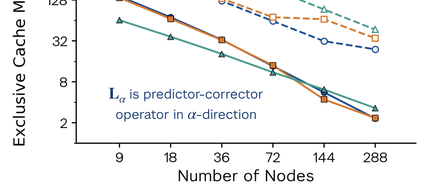
- PAPI counters: `TIME, FP_INS, TOT_CY, L1_DCM, L2_DCM`, etc
- At 144 nodes, the grid decomposition per core is $8 \times 6 \times 1152$
- t_{comp} falls faster than cores grow; t_{comm} flattens and becomes the bottleneck (Plot **B**)
- Inset: communication share grows from ~4% to ~63% (9→288 nodes)
- Per-core memory (MaxRSS) stays flat near 100 MB – rules out DRAM effect

Per-core cache on EPYC 9654:
L1 64 KB, L2 1 MB,
L3 384 MB/socket (768 MB/node)

- Subdomain shrinks from 337.5 MB to 10.5 MB – working set now fits cache
- Exclusive L1/L2 misses per $\mathbf{L}$-operator direction fall by 1–2 orders (Plot **C**)

Fewer cache misses, less DRAM stalling, faster execution



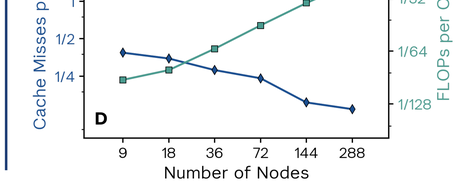


Floating-point throughput and cache misses

- L2 misses per FLOP fall 0.39→0.14 as FLOPs per cycle rise 0.011→0.038 (Plot **D**)
- Fewer misses directly boost FLOPs/cycle
- EPYC 9654 peak: 16 FLOPs/cycle; DHARA reaches only 0.038 (~0.2% of peak)

Note: Plots B–D use $\log_2$ axes; tick labels (e.g., 1/2, 1/4) denote powers of 2.

Moving data costs more than computing



Numerical method

- DHARA solves the fully compressible Navier-Stokes equations in a Cartesian box:

$$\partial_t \rho + \partial_i(\rho u_i) = 0$$
$$\partial_t(\rho u_i) + \partial_j(\rho u_i u_j + p\delta_{ij} - \tau_{ij}) = -\rho g\, \delta_{i3}$$
$$\partial_t E + \partial_i\left(u_i(E+p) - u_j\tau_{ji} - \Phi_i\right) = 0$$

- MacCormack-TVD (total variation diminishing) finite-difference scheme
- Collocated grid, second-order accurate in space and time
- Strang splitting factors each step into a symmetric sequence of 1D sweeps:

$$\mathbf{q}^{n+1} = \mathbf{L}_x\mathbf{L}_y\mathbf{L}_z\mathbf{L}_z\mathbf{L}_y\mathbf{L}_x\,\mathbf{q}^n,$$

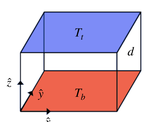
where $\mathbf{q} = [\rho, \rho u_x, \rho u_y, \rho u_z, E]^T$ and $\mathbf{L}_\alpha$ operators (predictor-corrector)

$$\mathbf{q}^p = \mathbf{q}^n - \tfrac{\delta t}{2}\Delta_- \mathbf{F}^n_\alpha + \tfrac{\delta t}{2}\mathbf{S}^n_\alpha \;\;;\;\; \mathbf{q}^c = \mathbf{q}^n - \tfrac{\delta t}{2}\Delta_+ \mathbf{F}^p_\alpha + \tfrac{\delta t}{2}\mathbf{S}^p_\alpha \;\;;\;\; \mathbf{L}_\alpha\mathbf{q}^n = \tfrac{1}{2}(\mathbf{q}^p + \mathbf{q}^c).$$

- TVD term is added to preserve monotonicity

Parallelization:

Pencil (2D) on CPUs – avoids too-thin subdomains at scale
Slab (1D) on GPUs – fewer ranks, simpler communication
Non-blocking MPI (`Isend/Irecv`) overlaps halo exchange with interior computation



Convection in a box


Slab decomposition (GPUs)

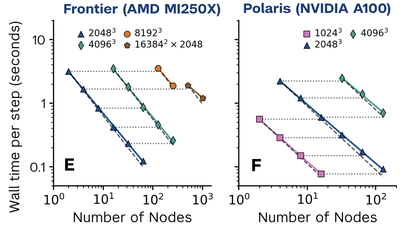

Pencil decomposition (CPUs)

GPU portability and benchmarks

- Same code with minimal change works on GPUs
- Supports both NVIDIA and AMD GPUs
- **200X** speedup on A100 compared to single-core AMD EPYC 7543

DHARA scales up to 8,192 GPUs on Frontier (Plot **E**) **and 512 GPUs on Polaris** (Plot **F**)

- `cupy.ElementwiseKernel` and `cupy.RawKernel` optimize the computationally heavy kernels



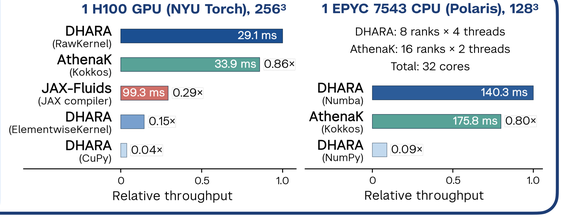
Compressible Euler benchmark

WENO5-Z · HLLC · SSPRK3

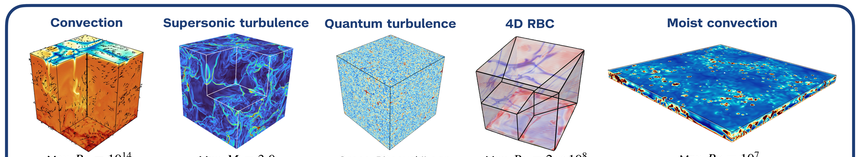
All solvers using same numerical schemes

Metric: wall time per step

Throughput normalized to DHARA



APPLICATIONS



Conclusion

- DHARA reaches super-linear performance on Shaheen III, scaling ~ **800,000** CPU cores
- The same codebase ports to exascale GPU systems like Frontier and Polaris with only minimal changes
- Cache locality, not additional parallelism, is what actually drives this super-linear speedup on CPUs
- **Next:** Pushing DHARA further for more problems at extreme parameters, including moist turbulent convection

References

1. **H. Tiwari**, L. Sharma, and M. K. Verma, *Int. J. Heat Mass Transfer* 242, 126821 (2025).
2. **H. Tiwari**, L. Sharma, and M. K. Verma, *Proc. Natl. Acad. Sci. U.S.A.* 122 (44) e2513474122 (2025).
3. A. Pandey, **H. Tiwari**, K. R. Sreenivasan, *J. Fluid Mech* 1042, R3 (2026).
4. **H. Tiwari**, D. Singh, M. K. Verma, and R. Ranjan, *Phys. Rev. E* (Under Review).

Acknowledgements

We thank Lekha Sharma, Dhananjay Singh, Shashwat Bhattacharya, Shashwat Nirgudkar, and Rajesh Ranjan for their valuable discussions. We acknowledge computational resources from King Abdullah University of Science and Technology (KAUST), NYU, Oak Ridge Leadership Computing Facility (OLCF), Argonne Leadership Computing Facility (ALCF), and the Kotak School of Sustainability (KSS) at IIT Kanpur.