# Memory-Efficient CFD based on MPS: Effective One-Billion-Cell Resolution on a Single Node

Junya Onishi*
junya.onishi@riken.jp
RIKEN Center for Computational Science
Kobe, Japan

Ayato Takii*
ayato.takii@penguin.kobe-u.ac.jp
Kobe University
Kobe, Japan

Sangwon Kim
RIKEN Center for Computational Science
Kobe, Japan

Younghwa Cho
Hokkaido University
Sapporo, Japan

Makoto Tsubokura
Kobe University
Kobe, Japan

## Abstract

We investigate Matrix Product States (MPS), a tensor-network compression method, as a memory-efficient representation of flow variables. A three-dimensional incompressible Navier–Stokes solver is implemented entirely in MPS form and is applied to canonical flow problems. Results show substantial memory savings and the ability to perform a $1024^3$ simulation on a single GPU. Performance analysis revealed new bottlenecks, particularly bond-dimension growth during nonlinear operations, suggesting novel optimization strategies are needed to fully realize MPS-based CFD at extreme scales.

## CCS Concepts

• **Computing methodologies** → **Modeling and simulation**; • **Mathematics of computing** → **Numerical analysis**; • **Applied computing** → **Engineering**; • **Theory of computation** → **Quantum computation theory**.

## Keywords

Matrix Product State, Computational Fluid Dynamics, High Performance Computing, Numerical simulation, Parallel computing, Performance evaluation, Numerical analysis, Quantum-inspired algorithms, Data compression, Partial differential equations

**ACM Reference Format:**
Junya Onishi, Ayato Takii, Sangwon Kim, Younghwa Cho, and Makoto Tsubokura. 2025. Memory-Efficient CFD based on MPS: Effective One-Billion-Cell Resolution on a Single Node. In *Proceedings of The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '25).* ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

*Both authors contributed equally to this research.


*SC '25, St. Louis, MO, USA*

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

## 1 Introduction

Recently, the growth of computational power in high-performance computing (HPC) has far outpaced the growth of memory bandwidth. For applications such as computational fluid dynamics (CFD), which operate on large three-dimensional fields, this imbalance often results in memory bandwidth becoming the dominant bottleneck. While hardware innovations (e.g., 3D-stacked memory) provide partial relief, further progress requires software-level strategies that reduce memory traffic and increase arithmetic intensity (FLOP/Byte). One promising approach is to represent flow variables in compressed tensor formats, enabling both storage savings and direct in-place computation.

This work explores the application of Matrix Product States (MPS) [1, 2], a tensor-network method originally developed in quantum many-body physics, as a memory-efficient representation for CFD. Prior work has suggested that MPS can drastically reduce storage, but its performance implications remain unclear [3].

Our objective is to implement a three-dimensional incompressible Navier–Stokes solver fully based on MPS representation, evaluate its accuracy and efficiency, and clarify the computational cost structure of such an approach. Specifically, we quantify fidelity loss, identify runtime bottlenecks, compare with conventional solvers, and assess hardware performance. These investigations ultimately provide guidance on how MPS may be tuned for CFD at extreme scales.

## 2 Methodology

MPS decomposes a high-order tensor into a product of low-order tensors, obtained via repeated application of singular value decomposition (SVD). Each tensor is connected to its neighbors through bond dimensions while retaining physical dimensions originating from the simulation grid. The compression ratio can be controlled by the truncation threshold for singular values ($\varepsilon$) and the maximum bond dimension ($\chi$). This representation allows: (i) large memory savings by compressing redundant correlations in the data, and (ii) the possibility of *in-place* operations directly on the compressed data.

At the level of the MPS representation, only a limited set of native operations is supported, such as *Scale()*, *Add()*, *Mul()*, *MatVec()*, *Solve()*, and *Round()*, as shown in Fig. 1. Other kernels must be expressed as compositions of these operations. Note that many of

them increase the bond dimension of MPS, and therefore require *Round()* to reduce it back to manageable levels. For linear solvers, however, specialized methods such as the Alternating Minimal Energy (AMEn) method [2] can operate without increasing bond dimension.

Our CFD solver is implemented based entirely on these MPS operations. All field variables are stored in MPS form from initialization through time integration. Several mappings (lexicographic, Z-order) are tested with varying spatial locality. Field variables are decoded back into full three-dimensional arrays only for visualization.

For consistency, we adopt the same finite-volume discretization and time-stepping scheme as a conventional solver. Specifically, the incompressible three-dimensional Navier–Stokes equations are discretized using the finite volume method on orthogonal structured grids with cell-centered variables. A fractional-step method is employed for time integration to enforce incompressibility and couple pressure and velocity. There may exist MPS-specific algorithms, but exploring them is left for future work.

| | Operation | Result rank | Complexity |
|---|---|---|---|
| **1. Scale** | $z = x \cdot c$ | $r(z) = r(x)$ | $O(dr(x))$ |
| **2. Add** | $z = x + y$ | $r(z) \le r(x) + r(y)$ | $O\left(nd\left(r(x) + r(y)\right)^2\right)$ |
| **3. Mul** | $z = x \odot y$ | $r(z) \le r(x)r(y)$ | $O(ndr^3(x)r^3(y))$ |
| **4. MatVec** | $z = Ax$ | $r(z) \le r(A)r(x)$ | $O(ndr^3(A)r^3(x))$ |
| **5. Solve** | Solve $Ax = y$ | $r(x)$ | $O(ndr^3(x)r(A))$ |
| **6. Round** | $z = \mathrm{round}(x, \varepsilon)$ | $r(z) \le r(x)$ | $O(dr^3(x))$ |

**Figure 1: Operations and their resulting bond dimensions and complexities in MPS.**

## 3 Results and Discussion

We first investigated the effect of the compression parameters ($\varepsilon$ and $\chi$) on physical fidelity, using a turbulent channel flow simulation as a benchmark. Larger $\varepsilon$ produced higher compression ratios but suppressed fine-scale vortex structures, while smaller $\varepsilon$ preserved accuracy at the cost of rapid bond-dimension growth. This revealed a clear trade-off between memory savings and fidelity, underscoring the importance of adaptive truncation strategies.

To demonstrate scalability, we carried out a simulation with $1024^3$ grid points on a single GPU, an effective billion-cell resolution that would normally exceed device memory capacity, as shown in Fig. 2. The MPS representation made this possible by reducing the memory footprint of flow variables. Note, however, the Alternating Minimal Energy (AMEn) [2] solver for the pressure equation still executed on the CPU, and the cost of CPU–GPU data transfers was non-negligible. Implementing AMEn directly on the GPU is therefore a key priority for future work and remains ongoing.

Next, we analyzed the execution performance by decomposing the entire solver into computational kernels. Unlike conventional CFD solvers, where the Poisson equation dominates runtime, our solver was primarily limited by the evaluation of convection terms. This was traced to bond-dimension growth in element-wise multiplication (*Mul()*) and repeated truncation (*Round()*). This suggests that the performance characteristics of MPS-based CFD solvers differ significantly from conventional CFD codes, necessitating new optimization strategies.

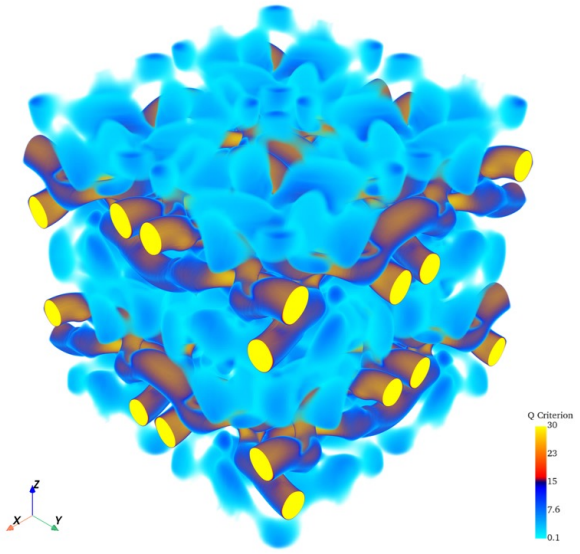


**Figure 2: Equivalent to a one-billion-cell simulation of the Taylor–Green vortex on a single GPU using MPS.**

Furthermore, we performed a roofline analysis for the key *MatVec()* kernel. From the code analysis, we view *MatVec()* in MPS as a collection of per-site $2 \times 2$ dense matvecs with 6 FLOPs each. Under the steady-state assumption that the $2 \times 2$ block is tiny and strongly reused from cache, the arithmetic intensity is evaluated as $6/32 = 0.1875$ FLOP/Byte for double precision. We then measured the performance and memory throughput of *MatVec()* on Fugaku (1 CMG), confirming that the kernel is memory-bound and that its realized arithmetic intensity is close to the theoretical limit ($\approx$ 0.19 FLOP/Byte). Note that this required optimizing the loop order to preserve unit-stride stores. We also measured execution on NVIDIA A100 and H100 GPUs. For GPUs, the kernel is written in Kokkos using `MDRangePolicy<Kokkos::Rank<4> >`. We tuned the tile so that the last index matches the contiguous output dimension (unit-stride/coalesced stores). Performance on these systems is summarized in Fig. 3. We confirmed that performance was likewise memory-bound in all the systems.

## 4 Conclusion

We have demonstrated that MPS-based CFD solvers can achieve substantial memory savings while maintaining acceptable accuracy, enabling simulations at unprecedented grid resolutions on limited hardware. However, performance analysis revealed new bottlenecks unique to MPS, particularly bond-dimension growth in nonlinear operations and associated truncation costs. Addressing these challenges requires algorithmic innovations such as faster *Round()* procedures and fusion of compression into *MatVec()* and *Mul()* operations.

Despite these challenges, MPS offers a promising new direction for large-scale CFD, extending the feasible range of applications and enabling new computational capabilities, such as real-time

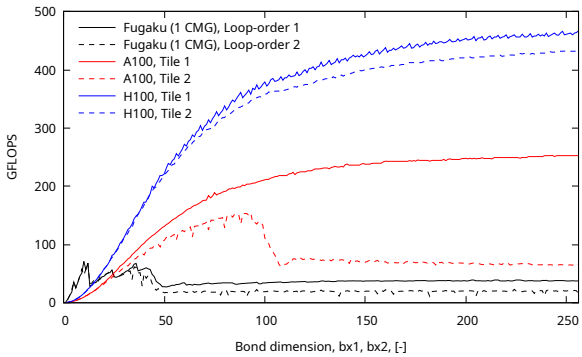


**Figure 3: Measured performance of the *MatVec()* kernel on Fugaku, A100, and H100 systems.**

simulation and energy-efficient computing on next-generation HPC systems.

## References

[1] S V Dolgov, B N Khoromskij, I V Oseledets, and D V Savostyanov. 2014. Computation of extreme eigenvalues in higher dimensions using block tensor train format. *Comput. Phys. Commun.* 185, 4 (April 2014), 1207–1216.

[2] Sergey V Dolgov and Dmitry V Savostyanov. 2014. Alternating minimal energy methods for linear systems in higher dimensions. *SIAM J. Sci. Comput.* 36, 5 (Jan. 2014), A2248–A2271.

[3] Nikita Gourianov, Michael Lubasch, Sergey Dolgov, Quincy Y van den Berg, Hessam Babaee, Peyman Givi, Martin Kiffner, and Dieter Jaksch. 2022. A quantum-inspired approach to exploit turbulence structures. *Nature Computational Science* 2, 1 (Jan. 2022), 30–37.