Memory-Efficient CFD based on MPS

Effective One-Billion-Cell Resolution on a Single Node

Junya Onishi[1], Ayato Takii[1,2], Sangwon Kim[1], Younghwa Cho[3], Makoto Tsubokura[1,2]

[1]RIKEN Center for Computational Science, [2]Kobe University, [3]Hokkaido University

Our contributions:
(i) a fully MPS-based 3D incompressible CFD solver,
(ii) a single-GPU large-scale demonstration enabled by compression,
(iii) a performance analysis identifying new bottlenecks, along with practical tuning guidelines for CPUs and GPUs.

Introduction

- Modern HPC systems face a widening gap between computational power and memory bandwidth. For CFD, this imbalance means that large-scale 3D simulations are often limited by memory transfer rather than floating-point operations.
- Matrix Product States (MPS) [1,2], developed in quantum physics, provide a compact tensor-network representation that compresses high-dimensional data while enabling direct operations on the compressed form. This approach promises reduced memory usage but its performance impact in CFD remains uncertain [3].
- This work develops a 3D incompressible Navier–Stokes solver based on MPS representation. We evaluate accuracy, performance, and memory savings relative to conventional methods, analyze computational cost distribution to identify bottlenecks, and explore the trade-off between compression and flow-structure fidelity. Furthermore, we demonstrate the feasibility of billion-cell simulations on a single node.

Matrix Product State (MPS)

- Originally introduced in quantum many-body physics as an efficient representation of quantum states
- Key idea: To decompose a high-order tensor into a product of low-order tensors, by using Singular Value Decomposition (SVD)
  - Low-order tensors have physical dimensions and bond dimensions
  - Compression ratio is controlled by:
    - Singular value threshold: ε
    - Maximum bond dimension: χ
  - Example: 8th-order tensor decomposed into 3rd-order tensors
    - With 2 bond dimensions and 1 physical dimension

$$x(i_1,i_2,i_3,i_4,i_5,i_6,i_7,i_8) = \sum_{\alpha_1,\ldots,\alpha_7} A^{i_1}_{\alpha_0\alpha_1} A^{i_2}_{\alpha_1\alpha_2} A^{i_3}_{\alpha_2\alpha_3} A^{i_4}_{\alpha_3\alpha_4} A^{i_5}_{\alpha_4\alpha_5} A^{i_6}_{\alpha_5\alpha_6} A^{i_7}_{\alpha_6\alpha_7} A^{i_8}_{\alpha_7\alpha_8}$$

(8th-order tensor; Bond dimensions; Physical dimensions; 3rd-order tensors)

- Advantage: Compute directly in the compressed (MPS) form
  - Most CFD operations are supported
  - Bond dimension grows during most operations
  - *Round()* suppresses this bond dimension growth
    - With its effect directly controlled by ε and χ
  - Bond-dimension-stable linear solvers are available (e.g., [2])

	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	$z = \mathrm{round}(x, \varepsilon)$	$r(z) \le r(x)$	$O(dr^3(x))$

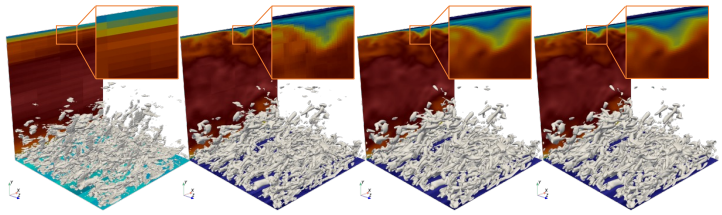
x,y,z: MPS, A: MPO, c: Scalar

r(x): Rank (bond dimension) of x, n: Size of physical dimension, d: Order of the original tensor

Effect of Truncation Threshold

- We evaluate the impact of the truncation threshold (ε) on the accuracy of the MPS representation, using a turbulent channel flow simulation as a benchmark case. A reference field is compared with MPS-compressed fields at different values of ε.
- The figure shows distinct differences in flow structures as ε is varied:
  - A larger ε yields stronger compression but significant loss of fine-scale structures.
  - A smaller ε preserves turbulent structures at small scales, but at the cost of a larger effective bond dimension.
- These results illustrate the trade-off between compression efficiency and fidelity in capturing flow structures.

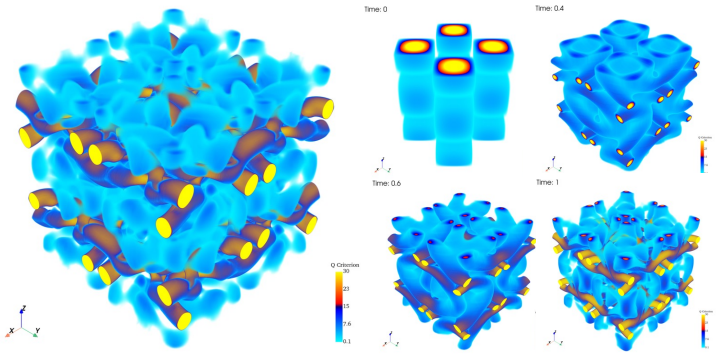
ε	Compression Ratio	Relative L_2Error
10^{-1}	3.1×10^{5}	6.8×10^{-2}
10^{-2}	1.1×10^{2}	6.6×10^{-3}
10^{-3}	1.3×10	4.3×10^{-4}



$\varepsilon = 10^{-1}$ $\varepsilon = 10^{-2}$ $\varepsilon = 10^{-3}$ Reference field (256^3 resolution)

Large-Scale Demonstration

- 3D CFD simulation of the Taylor–Green vortex at a resolution of 1024×1024×1024, corresponding to approximately 1 billion cells, on a single NVIDIA GeForce RTX GPU with 16 GB of memory
- Thanks to MPS compression, the effective array storage was reduced by a factor of about 10^5 compared to the raw data size.
- This extreme compression enabled the simulation of a problem size far beyond the nominal memory capacity of the device, while still maintaining fidelity in the resolved flow structures.



Code Performance

- Cost analysis: New bottlenecks emerge in MPS
  - Bond-dimension growth increases compute and memory traffic, especially in *MatVec()/Mul()*
  - Additional overhead results from the repeated application of *Round()*

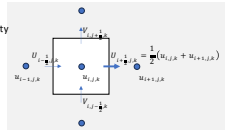
Subroutine	Elapsed time [sec]	[%]
Update Uc	1.74	25.9
Update Vc	1.56	23.2
Update Wc	1.55	23.1
Calc Uf, Vf, Wf	0.246	3.67
Update P	0.652	9.72
Correct	0.953	14.2
Total	6.70	100

Runtime breakdown of MPS-based CFD solver (typical case)

```
for n=1, Nsteps:
  UpdateUc()
  UpdateVc()
  UpdateWc()
  CalcFaceVel()
  UpdateP()
  Correct()
end for
```

CFD code

Calculation of cell-face velocity



```
CalcFaceVel():
  uef[i,j] = (u[i,j] + u[i+1,j])/2
  wf[i,j] = (u[i,j] + u[i-1,j])/2
  unf[i,j] = (u[i,j] + u[i,j+1])/2
  usf[i,j] = (u[i,j] + u[i,j-1])/2
```

Converntional version

```
CalcFaceVel():
  uef = u.matvec(Ae, u)
  uef.round(eps,chi)
  wf = u.matvec(Aw, u)
  wf.round(eps,chi)
  unf = u.matvec(An, u)
  unf.round(eps,chi)
  usf = u.matvec(As, u)
  usf.round(eps,chi)
```

MPS version

- Roofline analysis for *MatVec()* operation: Kernel is memory-bound
  - *MatVec()* is a collection of per-site 2x2 dense matvecs (6 FLOPS/site)
  - Performance is bound by contiguous stores to the output matrix
  - Arithmetic intensity (AI): 0.19 FLOP/Byte (Fugaku, 1 CMG, measured)

$$C(l)^{i}_{\gamma_i\gamma_{i+1}} = \sum_{j=1,2} A(l)^{i,j}_{\alpha_i\alpha_{i+1}} B(l)^{j}_{\beta_i\beta_{i+1}} \quad (i = 1,2)$$

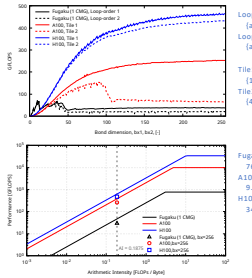
$$\gamma_i = \alpha_i\beta_i, \gamma_{i+1} = \alpha_{i+1}\beta_{i+1}$$

Bond dimension relation

```
for b2 in [0..bx2):
for b1 in [0..bx1):
for a2 in [0..ax2):
for a1 in [0..ax1):
    a11 = A[a1,0,a2]
    a12 = A[a1,1,a2]
    a21 = A[a1,1,0,a2]
    a22 = A[a1,1,1,a2]
    b11 = B[b1,0,b2]
    b21 = B[b1,1,b2]
    c1 = a1 + ax1*b1
    c2 = a2 + ax2*b2
    C[c1,0,c2] = a11*b1j + a12*b2j
    C[c1,1,c2] = a21*b1j + a22*b2j
```

Loop length
- ax1 and ax2 are fixed to 15 (Assumed to be cached)
- bx1 and bx2 are variable (From 1 to several hundred)

Mathematical expression and corresponding computational kernel for MatVec operation at local sites in MPS



Conclusion

- MPS-based CFD enables substantial memory savings while preserving acceptable accuracy, making billion-cell resolution feasible on a single node. For performance improvement, it is required
  - To accelerate *Round()*, and
  - To fuse compression into *MatVec()/Mul()*, eliminating extra traffic
- This framework has the potential to extend to broader applications, including combustion simulation, while also enabling advances in computational capabilities such as real-time simulation and energy-efficient computing.

References

[1] Dolgov, S. V., B. N. Khoromskij, I. V. Oseledets, and D. V. Savostyanov. 2014. "Computation of Extreme Eigenvalues in Higher Dimensions Using Block Tensor Train Format." *Computer Physics Communications* 185 (4): 1207–16.

[2] Dolgov, Sergey V., and Dmitry V. Savostyanov. 2014. "Alternating Minimal Energy Methods for Linear Systems in Higher Dimensions." *SIAM Journal on Scientific Computing: A Publication of the Society for Industrial and Applied Mathematics* 36 (5): A2248–71.

[3] Gourianov, Nikita, 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): 30–37.