Scalable reduced-order modeling for three-dimensional turbulent flow

Kazuto Ando[1,2,3], Rahul Bale[2,3], Akiyoshi Kuroda[1], and Makoto Tsubokura[2,3]

[1] Operations and Computer Technologies Division, RIKEN Center for Computational Science (R-CCS), Japan
[2] Complex Phenomena Unified Simulation Research Team, RIKEN Center for Computational Science (R-CCS), Japan
[3] Department of Computational Science, Graduate School of System Informatics, Kobe University, Japan

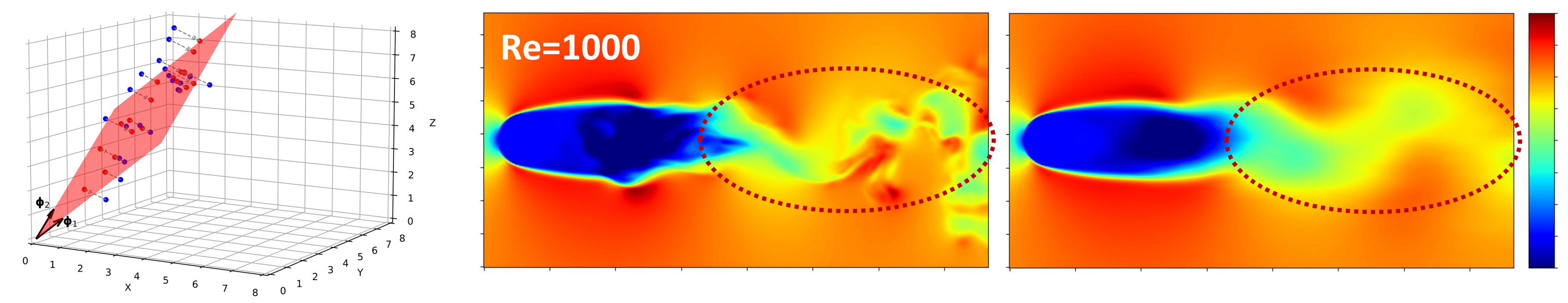
powered by FUGAKU

Reduce-order model for flow simulation

Background

- Numerical simulation, required in industrial applications, such as design optimization of automobile shapes and optimal control, must be executed repeatedly by changing the conditions, such as the model shape and in-flow velocity. The cost of such a simulation is a major obstacle for industrial users considering the feasible size of the computational system and amount of computational time.
- Reduced-order model (ROM) using POD in conjunction with Galerkin projection can reduce the calculation cost. However, it does not provide sufficient reproduction accuracy for an advection-dominant problem, that is, a case where nonlinearity appears strongly.
- To deal with such problem, a neural-network-based nonlinear dimensional reduction technique is required. Specifically, to deal with high-precision 3d data, distributed learning on massively-parallel distributed systems such as Fugaku is indispensable in terms of memory allocation and training speed.

Dimensional Reduction from 3D to 2D — Simulation result — POD reconstruction (16 variables)

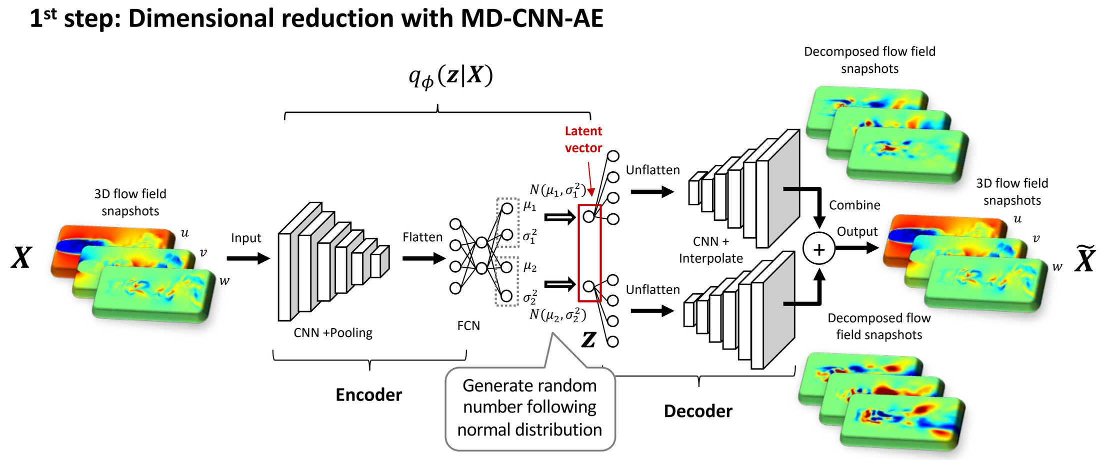


Methods

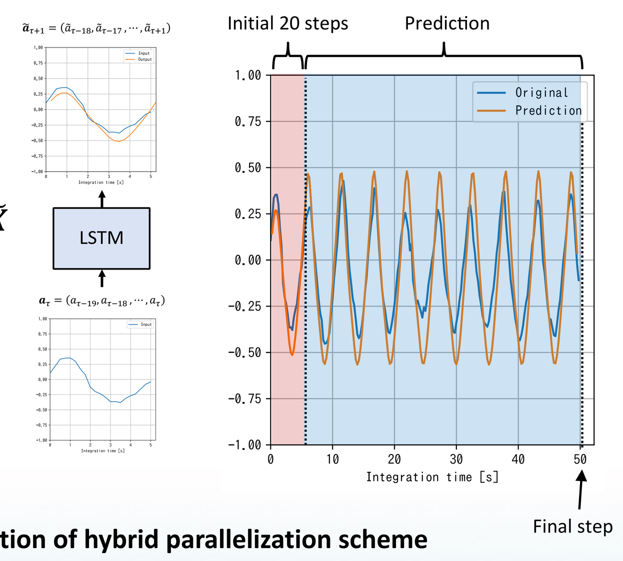
Reduced-order model using neural network

- 1st step: Reduce dimension of flow field data with autoencoder-like neural network called "MD-CNN-AE". After that, we can obtain "latent vector," which contains reduced-order variables.
- 2nd step: Predict time evolution of latent vector with neural network "LSTM".

1st step: Dimensional reduction with MD-CNN-AE



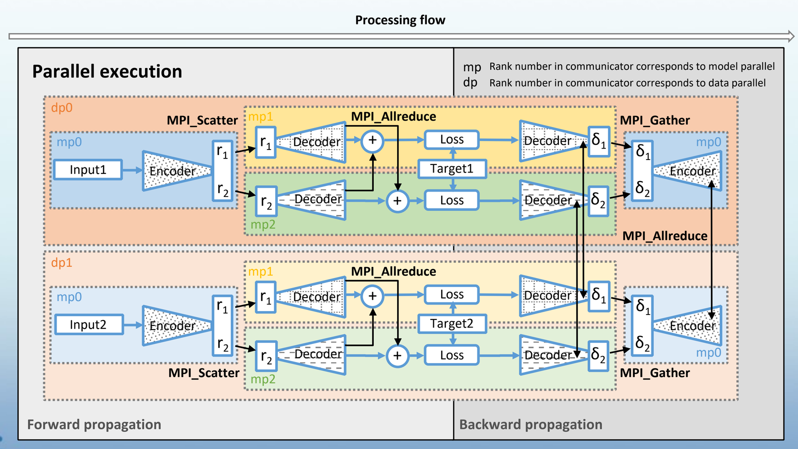
2nd step: Prediction of time evolution with LSTM



Implementation of distributed learning

- To utilize tens of thousands of computational nodes on Fugaku, we implemented a hybrid parallelization scheme.
- Domain-decompose encoder and multiple decoders and assign MPI process to each

Implementation of hybrid parallelization scheme



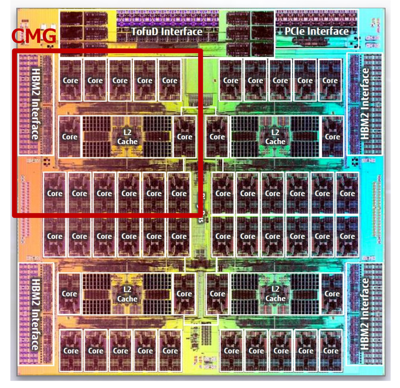
Computational performance

Single CMG computational performance

- The entire training loop, which involves I/O and communications, indicates 370.31 GFLOPS, which corresponds to 24.28% of the single-precision floating-point arithmetic peak performance. This is 1.5 TFLOPS in terms of 1 node (4 CMGs).
- The convolution kernel indicates 753 TFLOPS, which corresponds to 49.29 % of the peak performance. This is 3.0 TFLOPS in terms of 1 node (4 CMGs). This kernel calls the convolution routine in the Intel oneDNN library installed by Fujitsu and Riken in the DL4Fugaku project.
- CPU cycle counter result indicates whether the core works efficiently in each CPU cycle in the convolution routine. The light blue bar indicates the amount of time while the instructions are committed most efficiently --- that is, this kernel is highly optimized for Fujitsu A64FX CPU.

CMG (Core Memory Group)

- 12 computational cores + 1 HBM2 memory
- Single node has 4 CMGs

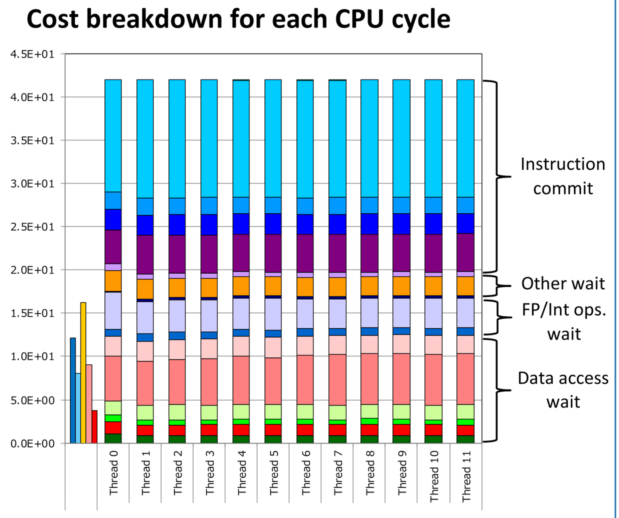


Computational performance (Running at 2.0GHz)

	Entire training loop	Convolution kernel
FP arithmetic performance	370.31 GFLOPS (24.28%ᵃ) → 1.5TFLOPS/node	753.93 GFLOPS (49.29%ᵃ) → 3.0TFLOPS/node
Memory throughput	22.97 GB/sec. (8.97%ᵃ)	23.92 GB/sec. (9.34%ᵃ)
L1D cache miss ratio	3.14% (79.51%ᵇ)	1.45% (77.72%ᵇ)
L2 cache miss ratio	0.55% (15.00%ᵇ)	0.20% (11.90%ᵇ)

ᵃ Ratio of peak performance
ᵇ Demand rate

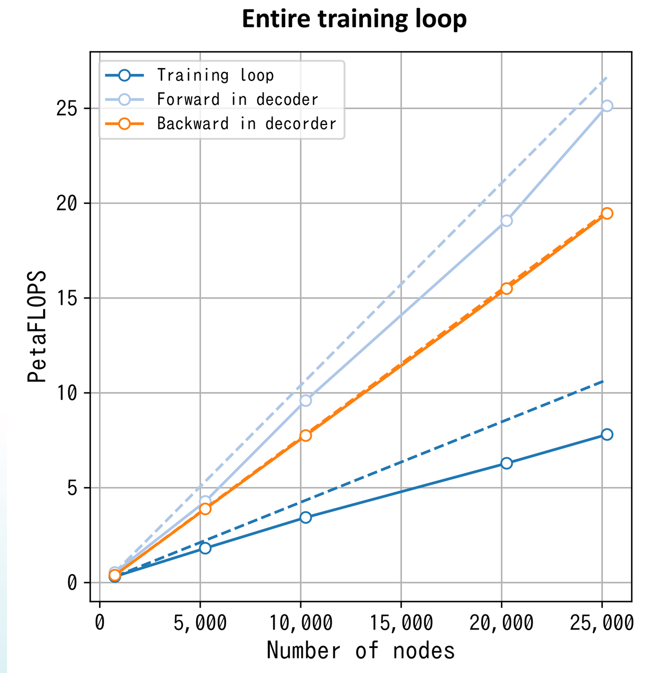
Cost breakdown for each CPU cycle



Multi-node computational performance

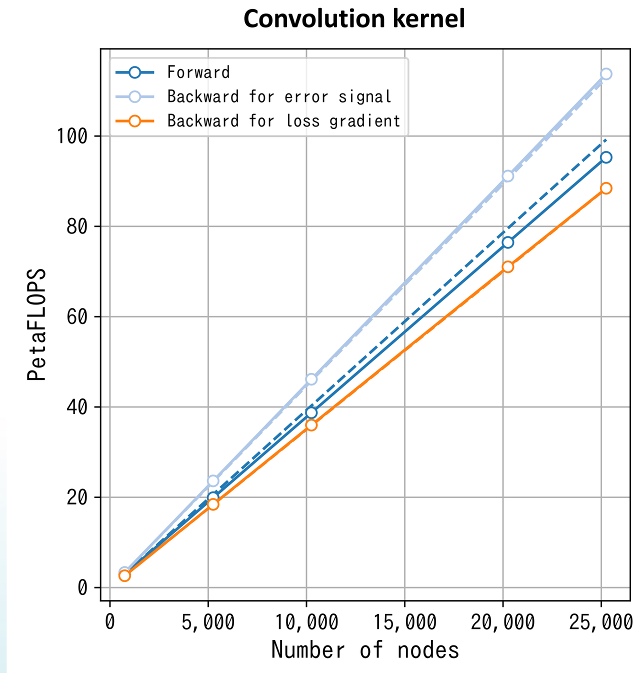
- The single-precision floating-point arithmetic performance of the entire learning procedure is 7.8 PFLOPS with 25,250 nodes (1,212,000 cores). The weak scaling performance is 72.9% (relative to 750 node).
- The forward propagation routine's performance, and the back propagation routine's performance indicates 25.1PFLOPS and 19.4 PFLOPS, respectively.
- The convolution routines show almost perfect scaling and achieve around 100 PFLOPS.

Entire training loop



Modes	Nodes	PFLOPS	Scaling
2	750	0.31	100.0%
20	5250	1.81	81.5%
40	10,250	3.43	79.0%
80	20,250	6.28	73.3%
100	25,250	7.80	72.9%

Convolution kernel



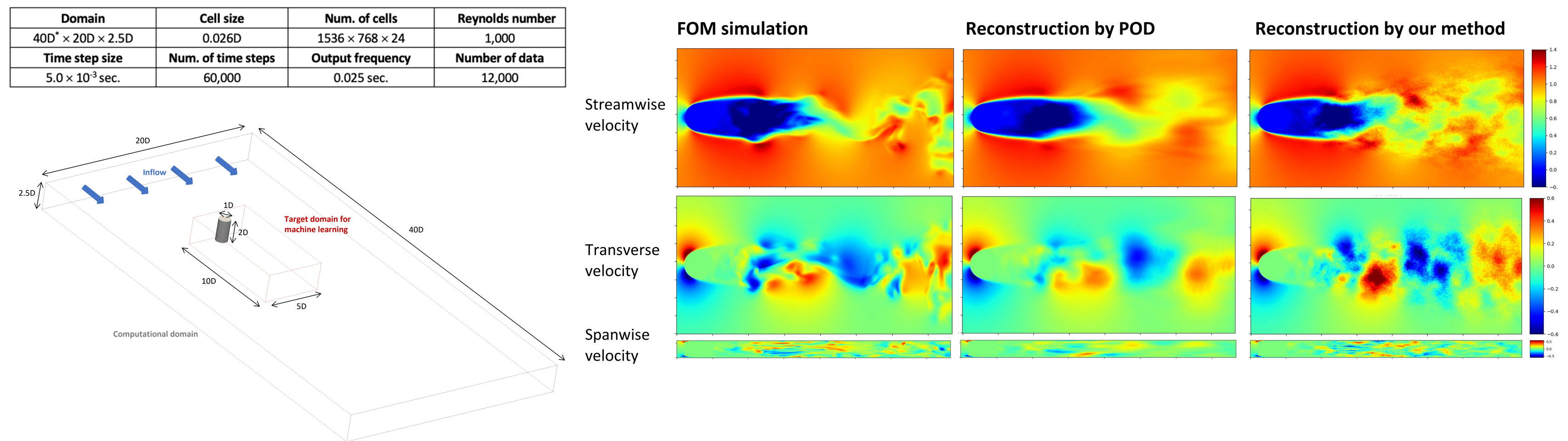
Modes	Nodes	PFLOPS	Scaling
2	750	3.35 PFLOPS	100.0%
20	5,250	23.62 PFLOPS	100.7%
40	10,250	46.14 PFLOPS	100.7%
80	20,250	91.14 PFLOPS	100.7%
100	25,250	113.75 PFLOPS	100.8%

Reproduction of turbulent flow field by ROM simulation

Application 1: Three-dimensional cylinder flow (Re=1000)

- The flow field reconstructed by POD after reducing into 128 variables does not contain small flow field structures contained in FOM simulation result.
- However, using the same number of variables, our method (MD-CNNAE + LSTM) reproduces the complex vortex structures close to those created with the FOM simulation result, especially in spanwise velocity.

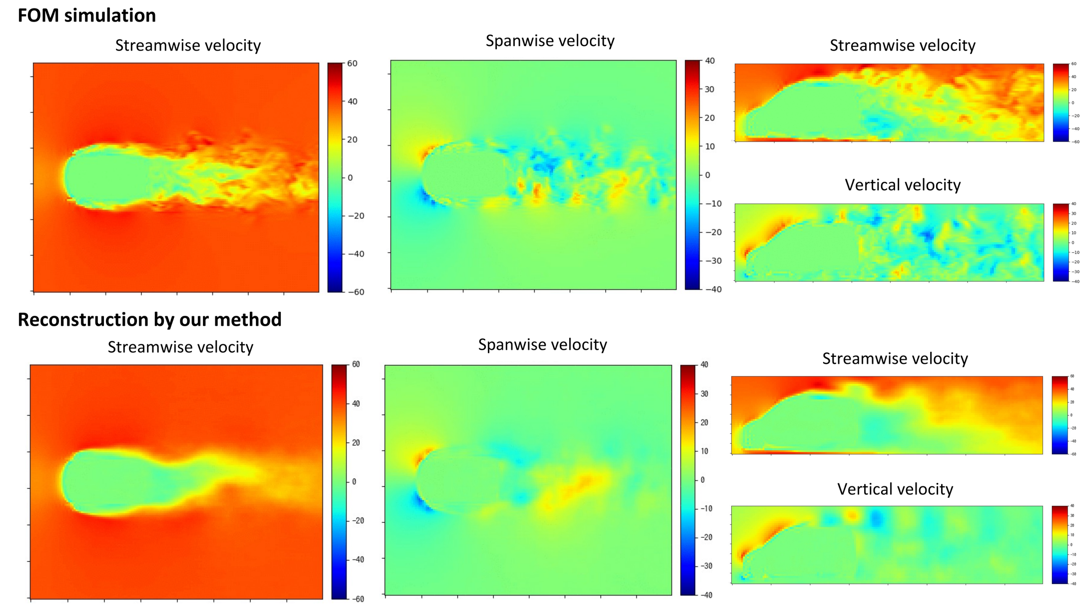
Domain	Cell size	Num. of cells	Reynolds number
$40D^* \times 20D \times 2.5D$	0.026D	$1536 \times 768 \times 24$	1,000
Time step size	**Num. of time steps**	**Output frequency**	**Number of data**
5.0×10^{-3} sec.	60,000	0.025 sec.	12,000



Application 2: Three-dimensional turbulent flow around vehicle (Re=2.8×10^6)

- Due to the not sufficient number of decomposing modes, small vortex structures cannot be reproduced in the reconstruction. However, the vortex scale which determines the aerodynamic performance of the vehicle body can be successfully reproduced with reconstruction after reducing 128 variables.

Solver type	Incompressible flow solver
Domain size	20m×10m×5m
Number of cubes	1,800
Number of cells per cube	8×8×8 = 512
Total number of cells	1,800×512 = 921,600
Minimum cell size	19.5 mm
Time step size	1.0×10^{-4} sec
Integration time	120 sec (1,200,000 steps)
Reynolds number	2.8×10^6
Time integration	Crank-Nicolson method
Pressure Poisson	Red-Black SOR
Viscous term	2nd order central difference
Convection term	QUICK



How much is computational cost reduction?

ROM reduces the number of floating point operations by 5 orders of magnitude relative to FOM

	Num. of cells / Num. of modes	CPU	Num. of CPUs / Num. of CMGs	Total num. of cores	Execution time (/ 1 time-step)	FP operations* (/ 1 time-step)
FOM	28,311,552 cells	Intel Xeon Gold 6148(2.4GHz)	32 CPU	384 cores	1.74E+00 sec.	8.55E+04 Gflop
ROM	2 modes	Fujitsu A64FX (2.0GHz)	1 CMG	12 cores	5.72E-04 sec.	4.39E-01 Gflop
	20 modes				7.37E-04 sec.	5.66E-01 Gflop
	516 modes				5.28E-03 sec.†	4.06E+00 Gflop†

Conclusion

- We implemented neural network-based reduced order modeling method for three-dimensional turbulent flow simulation using distributed learning on Fugaku.
- Time evolution of turbulent three-dimensional flow could be simulated at significantly lower cost (approximately four orders of magnitude) without major loss in accuracy.
- Using single CMG, entire training loop indicates 24.28%, and convolution kernel shows 49.29% of the peak performance.
- Our hybrid parallelization implementation scales up to 25,250 computational nodes (1,212,000 cores) in the distributed training.