Scalable reduced-order modeling for three-dimensional turbulent flow

KAZUTO ANDO, RIKEN Center for Computational Science (R-CCS), Japan and Kobe University, Japan
RAHUL BALE, RIKEN Center for Computational Science (R-CCS), Japan and Kobe University, Japan
AKIYOSHI KURODA, RIKEN Center for Computational Science (R-CCS), Japan
MAKOTO TSUBOKURA, RIKEN Center for Computational Science (R-CCS), Japan and Kobe University, Japan

A neural network-based reduced order modeling method for three-dimensional turbulent flow simulation is proposed in this study. This method was implemented as the scalable distributed learning framework on Fugaku. Our modeling method can be divided into two steps of training different networks. First, a dimensional reduction method was applied to three-dimensional flow field data using a convolutional-autoencoder-like neural network. Then, the time evolution of reduced-order variables was predicted using long short-term memory neural networks. Consequently, it was demonstrated that the time evolution of the turbulent three-dimensional flow (e.g., $Re = 2.8 \times 10^6$) could be simulated at a significantly lower cost (approximately four orders of magnitude) without a major loss in accuracy. Using the single core memory group (CMG), our implementation shows 370 GFLOPS (24.28% of the peak performance) for the entire training loop and 753 GFLOPS (24.28% of the peak performance) for the convolution kernel, respectively. Our distributed learning implementation utilized a hybrid parallelization scheme that scales up to 25,250 computational nodes (1,212,000 cores). Thus it shows 72.9 % of weak scaling performance and 7.8 PFLOPS for the entire training loop. On the other hand, it shows 100.8 % of weak scaling performance and 113 PFLOPS for the convolution kernel.

Additional Key Words and Phrases: Reduced-Order Model, Turbulence, Three-dimensional flow, Distributed machine learning, Convolutional Autoencoder (CAE), Long Short-Term Memory networks (LSTMs), supercomputer Fugaku

ACM Reference Format:
Kazuto Ando, Rahul Bale, Akiyoshi Kuroda, and Makoto Tsubokura. 2018. Scalable reduced-order modeling for three-dimensional turbulent flow. In . ACM, New York, NY, USA, 3 pages. https://doi.org/XXXXXXX.XXXXXXX

1 REDUCE-ORDER MODELING FOR FLOW SIMULATION

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 the amount of computational time.

Reduced-order model (ROM) using POD in conjunction with Galerkin projection is the major method for dimensional reduction to 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 kind of problem, a neural-network-based nonlinear dimensional reduction technique is attracting attention. Murata et al. proposed the mode-decomposing convolutional neural network autoencoder (MD-CNN-AE) [3]. In this study, this network was extended to a three-dimensional flow field. In recent years, there have also been examples of using a variational autoencoder (VAE) [2], where the network is trained such that the elements of the latent vector follow a normally distributed random variable. To deal with high-precision 3d data, distributed learning is indispensable in terms of memory allocation and training speed. In this study, a scalable method for massively parallel distributed systems such as Fugaku [4, 6] is implemented.

SC23, November 12–17, 2023, Colorado, Denver
© 2018 Association for Computing Machinery.
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in , https://doi.org/XXXXXXX.XXXXXXX.

2 METHODS

2.1 Reduced-order model using neural network

2.1.1 1st step: Dimensional reduction. Reduce the dimension of flow field data with an autoencoder-like neural network called "MD-CNN-AE". Figure 1 shows a schematic of the network structure. The first half of the network, the encoder, inputs the high-precision flow field snapshots and gradually reduces the dimensions as the data passes through the layers, and it outputs the vector containing the reduced variables, named "latent vector".

The second half of the network, the decoder, branches for each mode for decomposition. Each branched network inputs each element of the latent vector. Then, the data dimension is expanded as it passes through the layers. Each branched network outputs a decomposed flow field for each mode. Finally, these decomposed flow fields are combined, and the flow field that reproduces the original flow field is output. These networks are trained to minimize the errors between the original flow field $\mathbf{x}(t)$ and reconstructed flow field $\sum_{j=1}^{r} \mathcal{F}_{dec,j}([\mathcal{F}_{enc}(\mathbf{x}(\mathbf{t}))]_j)$. This optimization problem is formulated as

$$\{\boldsymbol{\phi}_j\}_{j=1}^{r} = \underset{\{\tilde{\boldsymbol{\phi}}_j\}_{j=1}^{r}}{\arg\min} \int_{t_{min}}^{t_{max}} \|(\mathbf{x}(t) - \sum_{j=1}^{r} \mathcal{F}_{dec,j}([\mathcal{F}_{enc}(\mathbf{x}(t))]_j)\|^2 dt, \tag{1}$$

where $\mathcal{F}_{enc}$ is the encoder; $\mathcal{F}_{dec,j}$ is the decoder corresponding to the j-th mode, and $\{\boldsymbol{\phi}_j\}_{j=1}^{r}$ is the trained network weights.

2.1.2 2nd step: Time evolution prediction. Next, another neural network, "LSTM", is harnessed for predicting the time evolution of latent vector elements. LSTMs take the 20-time steps of the latent vectors to step τ and predict the unknown latent vector of step τ+1. Once the LSTM is trained, using only the initial 20-time steps of the latent vectors, it is possible to predict the following time steps up to the last step. Finally, the trained decoder network can decode the time series of latent vectors to the flow field snapshots.

2.2 Implementation of distributed learning

To utilize tens of thousands of computational nodes on Fugaku, a distributed parallel machine-learning environment was implemented.

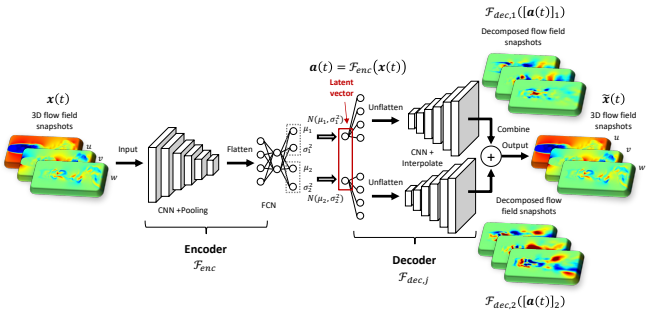

Fig. 1. Network structure for dimensional reduction.

The hybrid-parallel scheme, the combination of data parallelism that distributes the training data and model parallelism that distributes parts of the entire network structure, was enhanced (Figure 2).

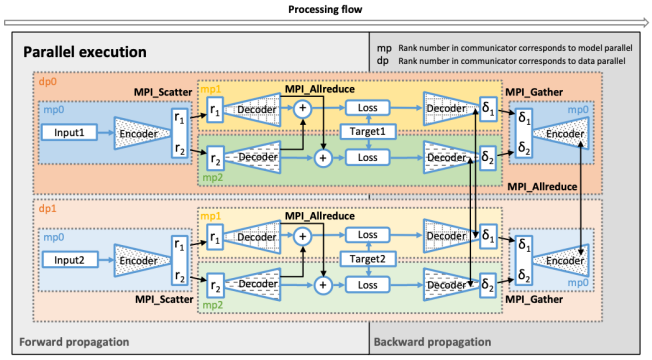

Fig. 2. Parallelization scheme.

3 COMPUTATIONAL PERFORMANCE

3.1 Single CMG computational performance

The single-precision floating-point arithmetic with one core memory group (CMG) is shown below. The entire training loop, which involves communications, indicates 370.31 GFLOPS (24.28% of the peak performance). This corresponds to 1.5 TFLOPS in terms of 1 node (4 CMGs).

The convolution kernel indicates 753 TFLOPS (49.29 % of the peak performance). This corresponds to 3.0 TFLOPS in terms of 1 node. This kernel calls the convolution routine in the Intel oneDNN library installed by Fujitsu and Riken [1] in the DL4Fugaku project [5].

3.2 Multi-node computational performance

Figure 3(a) show the results of the weak scaling test — that is, the number of nodes increased while maintaining the computational cost per node. In this case, the number of modes was increased; thus, the number of branches in the encoder increased as the number of nodes increased. The single-precision floating-point arithmetic performance of the entire training procedure is 7.8 PFLOPS with 25,250 nodes (1,212,000 cores). The weak scaling performance is 72.9% (relative to 750 nodes). The forward propagation routine's performance and the back propagation routine's performance indicate 25.1 PFLOPS and 19.4 PFLOPS, respectively. Besides, the convolution routines show almost perfect scaling and achieve around 100 PFLOPS (Figure 3(b)).

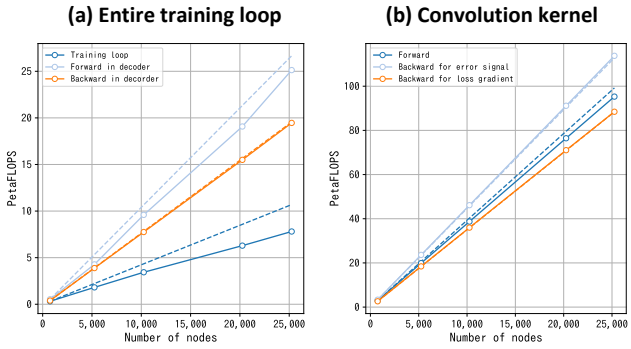

Fig. 3. Weak scaling performance of (a) the entire training loop and (b) the convolution routines. The solid line corresponds to measured values, and the dashed line corresponds to the ideal scaling of that.

Figure 3(b) shows the result of the weak scaling test for convolution routines. The blue line shows the forward propagation routine's performance, and the light blue and orange line shows the back propagation routine's performance. All of the routines show almost perfect scaling and achieve around 100 PFLOPS.

4 REPRODUCTION OF TURBULENT FLOW FIELD BY ROM SIMULATION

4.1 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 full-order model (FOM) simulation results. However, using the same number of variables, our method reproduces the complex vortex structures close to those created with the FOM simulation result, especially in spanwise velocity.

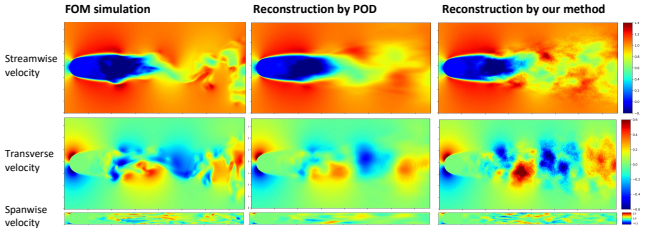

Fig. 4. ROM simulation result for cylinder flow ($Re = 1000$).

4.2 Application 2: Three-dimensional turbulent flow around vehicle ($Re = 2.8 \times 10^6$)

The large-scale vortex, which determines the aerodynamic performance of the vehicle body, can be successfully reproduced with reconstruction after reducing 128 variables.

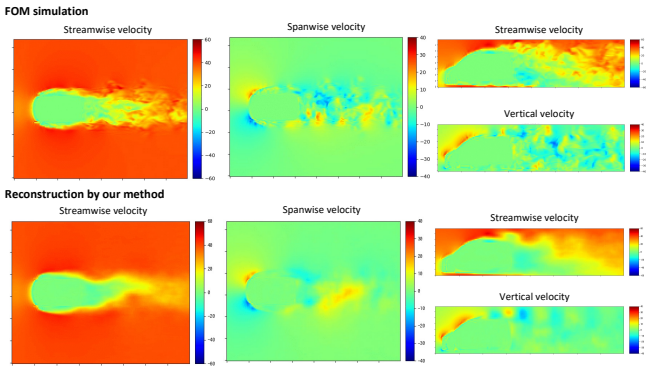

Fig. 5. ROM simulation result for flow around vehicle body ($Re = 2.8 \times 10^6$).

4.3 How much is computational cost reduction?

Compared to the FOM, if you decompose into 20 modes, the number of operations is reduced by five orders of magnitude. Even if you use 400 modes, the number of operations is estimated to reduce by 3 and 4 orders of magnitude.

5 CONCLUSION

A neural network-based reduced order modeling method for three-dimensional turbulent flow simulation was implemented as the scalable distributed learning framework on Fugaku. It was demonstrated that the time evolution of the turbulent three-dimensional flow field could be simulated at a significantly lower cost (approximately three orders of magnitude) without a major loss in accuracy. Using the single CMG, our implementation shows 370 GFLOPS (24.28% of the peak performance) for the entire training loop and 753 GFLOPS (24.28% of the peak performance) for the convolution kernel, respectively. Our distributed learning implementation scales up to 25,250 computational nodes (1,212,000 cores). Thus it shows 72.9 % of weak scaling performance and 7.8 PFLOPS for the entire training loop. On the other hand, it shows 100.8 % of weak scaling performance and 113 PFLOPS for the convolution kernel.

REFERENCES

[1] Fujitsu. 2021. Deep Neural Network Library for AArch64. https://github.com/fujitsu/dnnl_aarch64.
[2] Diederik P Kingma and Max Welling. 2013. Auto-Encoding Variational Bayes. (Dec. 2013). arXiv:1312.6114v10 [stat.ML]
[3] Takaaki Murata, Kai Fukami, and Koji Fukagata. 2020. Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. *Journal of Fluid Mechanics* 882 (2020), A13. https://doi.org/10.1017/jfm.2019.822
[4] R-CCS. 2021. About Fugaku. https://www.r-ccs.riken.jp/en/fugaku/about/. (Accessed on 10/07/2021).
[5] Kento Sato. 2020. DL4Fugaku: AI frameworks on Fugaku. 11th JLESC Workshop.
[6] Toshio Yoshida. 2018. Fujitsu high performance CPU for the Post-K Computer. In *Hot Chips*, Vol. 30.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009