# POD-Based Parallel Compression for Visualizing Large-Scale Dataset

Chongke Bi Kenji Ono

Advanced Institute for Computational Science, RIKEN

## Abstract

Visualizing and analyzing large-scale dataset is an important task for scientific research in various fields. However, the visualization process is time-consuming, which is quite inconvenient for researchers and engineers to analyze the time-varying dataset. In this poster,we proposed an approach to generate a small-scale dataset from the original large-scale one. The key idea is to divide the large-scale dataset into small groups firstly, and then compress the dataset in each group by using proper orthogonal decomposition (POD) method in parallel. This process is recursively carried on until the obtained dataset cannot be compressed any more. Here, the parallel computing greatly decreases the computational cost of the eigen resolving problem in the POD algorithm. Furthermore the compressed dataset can be easily restored linearly.

## 1 Introduction

The most important thing for a new compression method is evaluation. Here the following four technical issues will be used to evaluate the proposed approach: 1) *compression ratio* (Eq. (1)), 2) *errors of the compressed datasets* (Eq. (2)), 3) *computational cost*, and 4) *efficiency of restoration*.

$$c = size_{compressed} / size_{original} \times 100\%. \quad (1)$$

$$error = \sum |v_{original} - v_{compressed}| / \sum v_{original} \times 100\%. \quad (2)$$

## 2 Parallel-POD Compression and Restoration

The basic idea of the proposed POD-based parallel algorithm is to divide the large-scale datasets into small groups first, and then the POD-based compression algorithm [1] is carried on in each small group in parallel. After that the obtained POD-basements should be collected, and be divided into new small groups. These POD-basedments will be compressed in parallel agarin. This process will be recursively repeated until the datasets cannot be further compressed. Now the final POD-basements and all the coefficients (very small size) are the final compressed datasets.

Figure 1 shows the detail of the proposed approach. The original dataset in the *layer 0* was divided into 3 small groups firstly. POD algorithm is used to compressed the datasets of the 3 small groups in parallel. However, if the compression process is stopped here, the compression ratio is $4/11$, which is still high. Therefore the POD-basements in the *layer 1* are compressed again as shown in the second line of Figure 1. This process will be repeated until the POD-basements cannot be compressed any more. Finally, the compressed POD-basements in *layer 2* are obtained.

Restoring the compressed datasets efficiently is as important as compression process. A linear restoration method will be introduced. As shown in Figure 1, the red arrows show the process of

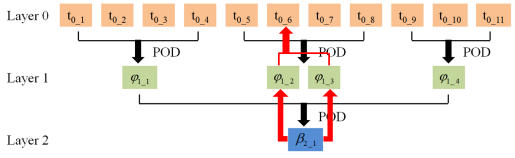


Figure 1: The flow chart of POD-based parallel compression.

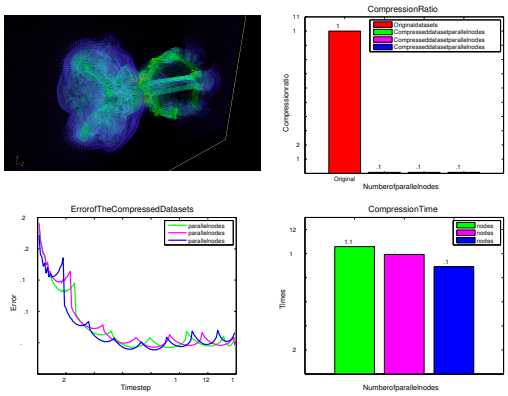


Figure 2: The result of compressing the large-scale time-varying dataset of a flow simulation in the air jet mixer of a machinery.

restoring the dataset in the 6th timesteps $t_{0\_6}$. In the restoring process, it is unnecessary to calculate the POD-basements in the middle layers. The original datasets can be linearly restored using the POD-basements in the deepest layer and the related coefficients.

## 3 Results and Discussions

Our prototype system is implemented on the supercomputer of *FX10* in the University of Tokyo. FX10 has 4800 nodes, 6 dimensional network, Each node has 16 cores, 32GB memory, 1.848GHz, and 14.78GFlops. The source code has been written in C++.

Figure 2 shows the result to compress a large-scale dataset ($500 \times 200 \times 125 \times 139$ $[x \times y \times z \times timesteps]$) obtained from a flow simulation in the air jet mixer of a machinery, which is rendered by the magnitude of velocity (Figure 2(a)). The colors of red, green, magenta, and blue represent original dataset, the compression results with 6 nodes, 7 nodes, and 8 nodes in parallel, respectively.

Figure 2(b) is the compression ratio calculated by using Eq. (1). The result shows that our approach can successfully compress such kind of large-scale dataset in a quite low compression ratio. Meanwhile, the precision of the compressed dataset can also be preserved in such low compression ratio, as shown in Figure 2(c). The errors of the compressed dataset are calculated by using Eq. (2). In Figure 2(c), the number of parabola shape is the same with the numbers of nodes. These parabola shapes can be used to prove the correctness of our algorithm. This is because the mean velocity for the neighbour time steps is nearly equal to that of the middle time step in a small size group. Furthermore, the computational cost is also investigated in temporal space, as shown in Figure 2(d). It becomes smaller as the numbers of parallel nodes are increased. This also described one of the merits of our parallel algorithm.

## Acknowledgement

Part of the results is obtained by early access to the K computer at the RIKEN Advanced Institute for Computational Science.

## References

[1] K. Taira. Proper orthogonal decomposition in fluid flow analysis: 1. introduction. *Nagare-Journal of Japan Society of Fluid Mechanics*, 30:263–271, 2011.