A Deployment of HPC Algorithm into Pre/Post-Processing for Industrial CFD on K-computer

Keiji Onishi
RIKEN
7-1-26, Minatojima-minami-machi, Chuo-ku
Kobe, Hyogo 650-0047, Japan
keiji.onishi@riken.jp

Niclas Jansson
KTH Royal Institute of Technology
SE-100 44
Stockholm, Sweden
njansson@csc.kth.se

Rahul Bale
RIKEN
7-1-26, Minatojima-minami-machi, Chuo-ku
Kobe, Hyogo 650-0047, Japan
rahul.bale@riken.jp

Wei-Hsiang Wang
RIKEN
7-1-26, Minatojima-minami-machi, Chuo-ku
Kobe, Hyogo 650-0047, Japan
wei-hsiang.wang@riken.jp

Chung-Gang Li
Kobe University
1-1 Rokkodai-cho, Nada-ku
Kobe, Hyogo 657-8501, Japan
RIKEN
Kobe, Japan
cgli@aquamarine.kobe-u.ac.jp

Makoto Tsubokura
Kobe University
1-1 Rokkodai-cho, Nada-ku
Kobe, Hyogo 657-8501, Japan
RIKEN
Kobe, Japan
tsubo@tiger.kobe-u.ac.jp

ABSTRACT

Pre- and post-processing is still a major problem in industrial computational fluid dynamics (CFD). With the rapid development of computers, physical solvers are getting faster, while pre- remains slow because it's mainly a serial process. A methodology using MPI+OpenMP hybrid parallelization has been proposed to eliminate the manual work required during pre-processing for correcting the surface imperfections of CAD data. Compared to the rapidly increasing amount of data in recent years, the speed-up of visualization is insufficient. We address this limitation of post- by adapting the in-situ visualization to parallelize the post-processing using libsim (Visit) library. The performance of pre-/post- processing is investigated in this work and we show that the pre- processing time has been reduced from several days in the conventional framework to order of minutes. The post-processing time has been reduced seconds order per frame, and approx. 30% increase of computational time was observed in vehicle aerodynamics case.

CCS CONCEPTS

• **Computing methodologies** → **Massively parallel and high-performance simulations**; *Physical simulation*; • **Theory of computation** → *Computational geometry*; • **Applied computing** → *Computer-aided design*;

KEYWORDS

Parallel pre-processing, Dirty CAD data, In-situ visualization, Computational Fluid Dynamics, K-computer

SC'17, Denver, Colorado USA

DOI:

ACM Reference format:
Keiji Onishi, Niclas Jansson, Rahul Bale, Wei-Hsiang Wang, Chung-Gang Li, and Makoto Tsubokura. 2017. A Deployment of HPC Algorithm into Pre/Post-Processing for Industrial CFD on K-computer. In *Proceedings of The International Conference for High Performance Computing, Networking, Storage, and Analysis, Denver, Colorado USA, Nov 2017 (SC'17),* 3 pages.
DOI:

1 INTRODUCTION

Pre-/post- processing is still a major problem in industrial CFD analysis. With the rapid development of computers, physical solvers are getting faster, while pre- still remains slow because it is mainly a serial process and it is less likely of benefit from HPC technology. For example, Furukawa, et. al showed an example of breakdown of turn-around time in automobile aerodynamics, where pre- accounts for 80% of the total cost [1]. In the field of aeronautics, because pre- is required at every iteration in the design optimization cycle, processing speed up is hindered due to this bottleneck no matter how fast a solver can be made [8]. For post- as well, speedup of visualization is insufficient compared to the rapidly increasing amount of data. For example, in typical industrial CFD analysis, it is reported that the total amount of CFD data is increasing by a factor of 4 year by year [7]. Now, the data size is approaching Peta-Byte (PB) order. Meanwhile, the effective disk access speed of commodity-based products is about 50 to 100 MB/s. Post- will still be a bottleneck until the high-speed parallel file access, visualization and, perhaps, network systems become cheap enough to be available at client site. We develop a method that enables pre-/post- parallelization using HPC techniques to harness the power of modern supercomputers for the CFD based industrial design cycle.

2 NUMERICAL METHOD

2.1 Pre-processing Method

The CAD geometry is discretized into cell-oriented values based on a hierarchical Cartesian grid (BCM [3]). BCM consists of two level of discretizations. First, the computational space is divided

by 'Cubes' which become finer as they approach the geometry wall, then each Cube is subdivided by 'Cells', typically 16x16x16 in three dimensions. The finest calculation unit in this system is a 'Cell'. Since each Cube is composed of exactly the same number of cells, it is advantageous for parallelization. An arbitrary boundary representation is used with an immersed boundary method (IBM [2, 6]). And the dummy-cell technique based on degenerated cell information [4, 5] is adapted to enable physical solver handling 'dirty' CAD without necessitating any clean-up. This handling process is equivalent to the pre- processing of other CFD solvers. The process is parallelized with MPI+OpenMP. The data exchange between halo-region of each Cube is conducted with multi-thread calculation and communication overlapping. In addition, all of the file reading/writing process is parallelized with MPI-IO, which is also important in achieving faster pre- processing.

2.2 Post-processing Method

The libsim (VisIt) library was used for in-situ visualization which requires OSMesa (OpenGL on CPU) and VTK library. The data access is tightly coupled with physical solver which has meta data access pipeline providing direct or copy access to the data on memory. The meta data is decomposed based on the BCM grid decomposition which is done on a Cube basis using Z-ordering or Parmetis library. The compilation is done on K-computer using Fujitsu cross-compiler with parallel option. The libsim library can be called under the batch or interactive render mode.

3 RESULTS AND DISCUSSION

Fig.1 shows the parallel performance of pre-processing. In the execution on a single node (1 MPI rank), increasing the number of OpenMP threads shows almost ideal speed up. Execution with the 91 million cell, which is typically used in the design process, shows reasonable speed up to 4,096 cores. For 4,096 core case, the pre processing can be completed in less than 10 seconds, which typically requires several days of human labor. In the case of execution with a large case using 26.8 billion cells, although the rate of speedup slows down, pre processing can still be completed in about 5 minutes using the 26 thousand cores. Next, file write performance by MPI-IO was examined on K-computer. To write 251 GB data required 24.5 seconds, and 58.4 seconds to write 618 GB of data. In both cases a write speed of 10 GB/s or more was achieved. Although the write speed is not close to design I/O bandwidth of the system, we believe that a write speed of 10 GB/s in real-world simulation conditions is more than satisfactory.

Fig.2 shows the parallel performance of in-situ rendering for two cases: 573K cells and 91M cells. In both the cases the rate of speed up slows down as the degree of parallelism increases. This could be due to the fact that the image rendering by the OSMesa may not be fully parallelized. The rendering time per frame is only about 10 seconds, thus it can be said that the speed is sufficient for practical use. For the 91M cells case, the in-situ visualization involved rendering the full 3D vehicle geometry, which is superimposed on a 2D slice of the flow field. The in-situ renderer is invoked once every 100 timestep of the flow solver. For a simulation that involved 290,000 timesteps, the total calculation time was 75,743 seconds. In the total time, 21,782 seconds were spent in generating 2,901 frames through the

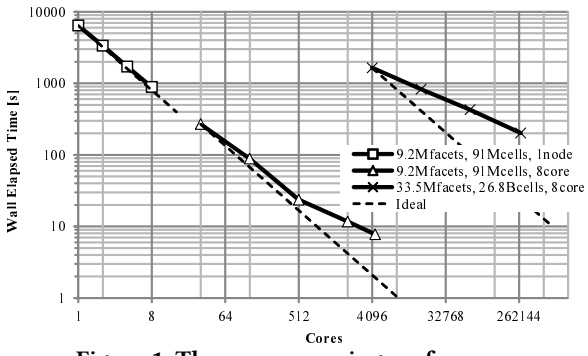

Figure 1: The pre-processing performance.

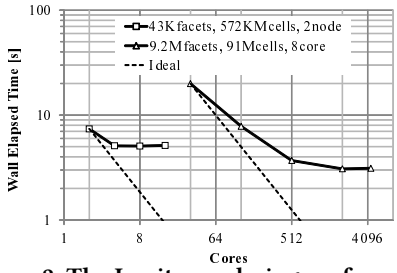

Figure 2: The In-situ rendering performance.

in-situ visualization. This corresponds to about 30% of the total time. The data size of the 3D flow field per snapshot is approximately 2.1 GB, and the image data is at most about 200 KB. In order to generate same snapshots using conventional post processing based visualization, it would be necessary to store about 6.1 TB of data. Furthermore, conventional process is typically serial, so the time taken to generate images would be of the order of tens of hours in addition to the simulation time.

Lastly, it is better to present the accuracy and the performance of physical solver because the pre-method has a deep impact on the solver itself. As the typical usage of a conventional CFD method used in industry, the Raynolds averaged analysis with 5,000 timesteps, 60M elements, 512 cores are used for same vehicle geometry mentioned above. The error of delta drag prediction between 2 aerodynamic configurations was 12% obtained by the conventional method with 8 hours calculations on recently designed Intel Xeon based cluster, compared with 8% of this method.

4 CONCLUSION

Pre-processing, which typically requires several days of human labor, has been reduced to the order of minutes. For a data size of 2GB/frame, the post-processing time has been reduced to the order of several seconds. This results in an approx. 30% increase in computational time for a vehicle aerodynamics case. We believe these are indispensable technologies towards the EXA FLOPS CFD.

ACKNOWLEDGMENTS

This research was supported by MEXT as "Priority Issue on Post-K computer" (Development of innovative design and production processes) and used computational resources of the K computer provided by the RIKEN Advanced Institute for Computational Science.

REFERENCES

[1] N Furukawa, H Shiozawa, R Koyama, Y Ishihara, and H Aoki. 2004. Application of CFD Aerodynamic Computation Methods to New Vehicle Development Based on a Model that Reproduces a Engine Bay and Floor. *JSAE Paper* 20045513 (2004).

[2] R Ghias, R Mittal, and H Dong. 2007. A sharp interface immersed boundary method for compressible viscous flows. *J. Comput. Phys.* 225, 1 (jul 2007), 528–553. https://doi.org/10.1016/j.jcp.2006.12.007

[3] K. Nakahashi. 2003. Building-Cube Method for Flow Problems with Broadband Characteristic Length. In *Computational Fluid Dynamics 2002*. Springer Berlin Heidelberg, Berlin, Heidelberg, 77–81. https://doi.org/10.1007/978-3-642-59334-5{_}7

[4] K. Onishi, S. Obayashi, K. Nakahashi, and M. Tsubokura. 2013. Use of the Immersed Boundary Method within the Building Cube Method and its Application to Real Vehicle CAD Data. In *21st AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, Sandiego, CA. https://doi.org/10.2514/6.2013-2713

[5] K. Onishi, M. Tsubokura, S. Obayashi, and K. Nakahashi. 2014. Vehicle Aerodynamics Simulation for the Next Generation on the K Computer: Part 2 Use of Dirty CAD Data with Modified Cartesian Grid Approach. *SAE Int. J. Passeng. Cars - Mech. Syst.* 7, 2 (april 2014), 528–537. https://doi.org/10.4271/2014-01-0580

[6] C.S. Peskin. 1972. Flow patterns around heart valves: A numerical method. *J. Comput. Phys.* 10, 2 (1972), 252–271. https://doi.org/10.1016/0021-9991(72)90065-4

[7] N. Sogo. 2016. Introduction of high-performance computing consulting service. (Oct. 2016). Retrieved April 1, 2017 from http://www.vinas.com/en/ugm2016/program.html

[8] M. Todd. 2009. CSE 2009: Preprocessing for Industrial CFD: More Important Than You Might Think. (June 2009). Retrieved April 1, 2017 from http://www.siam.org/news/news.php?id=1598