ParCFD'2020
32nd International Conference on Parallel Computational Fluid Dynamics
May 11-13 2020, Nice FRANCE

# OPTIMIZING PARALLEL PERFORMANCE OF A THREE-DIMENSIONAL UNSTRUCTURED MESH CFD APPLICATION WITH BLACK-BOX SOLVERS

**CHEN JUN*[*]**

[*] Institute of Applied Physics and Computational Mathematics
High Performance Computing Center
100094 Beijing, CHINA
e-mail: chenjun@iapcm.ac.cn, web page: https://www.iapcm.ac.cn

**Key words:** Parallel performance, Unstructured mesh, Optimizing methodology, Parallel CFD application.

**Summary.** This paper provides some optimizing methods for the parallel performance of a three-dimensional unstructured mesh computational fluid dynamics application with black-box solvers. Here the black-box solvers mean that the computational kernel of the application kept as black-boxes, only some little information revealed. Our optimizing methods include extending array declaration to throw off some unnecessary computation, updating the input mode to reduce the memory requirements, and adapting the parallel data structure to find good performance for this application. The numerical results show that parallel performance had increased a lot when this optimized application with tetrahedral unstructured mesh examples run on a parallel computer.

## 1 INTRODUCTION

A lot of parallelism and optimization methods of unstructured mesh CFD applications have been developed for these years. Paper[1] introduces some unstructured mesh related issues in computational fluid dynamics, including reducing memory requirements and minimizes indirect addressing. Paper[2] used three techniques for GPU optimization of an unstructured mesh application HYDRA after studying the relationship between optimizations and resource requirements of loops, in terms of registers and shared memory. There three techniques include splitting a highly complex loop into simpler loops, a kernel specific alternative code synthesis, and configuration parameter tuning. These optimization methods strongly depend on the specific application and architecture of the parallel platform. Paper[3] provides a pre-compiler AUTO-CFD-NOW, which transforms Fortran CFD sequential program to efficient message-passing parallel program running on network of workstations. The pre-compiler applied a dependency analysis technique to parallelize self-dependent field loops that are hard to parallelize, and an optimization scheme that combines all the non-redundant synchronizations in CFD programs to further reduce the communication overhead. Paper[4] tuned the hybrid CPU+MIC code of a three-dimensional structured grid application toward better performance in intra-machine node with hundreds of CPU/MIC cores, and optimized

* This work is supported by the National Natural Science Foundation of China under contact number 61672003 and 11871109.

the communication among inter-node, inter-cores and between CPUs and MICs. Paper[5-6] provided an active library OP2 to meet the major challenges in data organization and movement in a full-scale industrial CFD application HYDRA, which performed complex simulations over highly detailed unstructured mesh geometries. OP2 is aimed to decouple the scientific specification of an application from its parallel implementation to achieve code longevity and near-optimal performance by re-targeting the back-end to different multi-core or many-core hardware. Paper[7] introduces a high-order finite volume algorithm with polynomial reconstruction on unstructured hybrid meshes used to compute compressible gas flows in domains of complex geometry, and gives a detailed description of parallel implementation of the basic operations of the algorithm on CPU+GPU platform. Paper[8] concludes the most numerical methods in CFD which requires the formation and solution of large sparse linear systems of algebraic equations. These matrices arise from the discreteness of the Navier-Stokes equations which govern compressible fluid flow. It also lists algorithms used in the formation, manipulation, and solution of sparse matrices on serial and parallel computers. Paper[9] introduces a performance tuning method for a three-dimensional unstructured grid Euler flow code from NASA based on PETSc framework. It is from a memory-centric perspective, not traditional flop-orientation to understand the code’s sequential and parallel performance. Three methods are provided to optimize parallel performance. They are data layout to enhance locality of reference, algorithmic parameters and parallel programming model. They are guided by some simple performance models developed for the sparse matrix-vector product operation. Paper[10] introduces a geometric technique to accelerate convergence to a steady state of a nonlinear CFD solver with unstructured meshes on GPU platform.

We have developed a parallel method for a serial three-dimensional unstructured mesh fluid application with black-box solvers[11]. These solvers are the main computing kernels here provides to be a library file and kept as black boxes, which only small amount of information is revealed. We take the idea of separating the data structure and the computational methods in the solvers from each other. After they two separated, the advantage is obvious that we can focus on studying the better layout of the data organization and the better optimization methods for the basic operations existed most frequently in the unstructured mesh CFD application, while almost ignores the information of the computing kernels above.

In this paper we introduce some optimizing method for the parallel unstructured mesh CFD application with black-box solvers. In Section 2, we declare array extension to throw off the unnecessary computations, update the input mode to reduce the memory requirement, and analyze some basic operations in unstructured mesh CFD applications. Section 3 gives the experiment results after optimizing the parallel performance. Section 4 concludes the paper.

## 2 OPTIMIZING THE PARLALEL PERFORMANCE

We have separated the data structure and the computing methods of the unstructured mesh CFD application from each other. There are the performance of some parts need to be optimized.

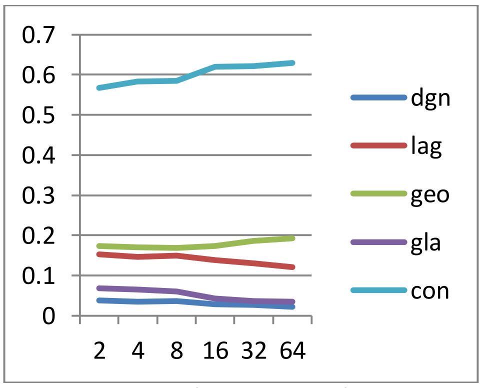
Usually, the data structure of a physical field is set to be a compressed one dimensional array in the case of three-dimensional unstructured mesh CFD applications. If there are extra memory defined clearly in serial code, which can be used for distinguishing the internal computing domain from the physical boundary and virtual boundary needed in parallel version, then the serial code with the compressed array can be transplanted easily to the parallel version. Here it is not the case. So we extend the size of the compressed array when declaration to get the extra memory for the purpose above. We just introduce a new item *max_size* for each size of the physical variables, which declared as *size*. Now these two items can specify the limits of the internal physical domain and the both of the two boundaries. Thus we can get rid of some unnecessary computation costs with the old method.

In the last version, all the processes read the same copy of all the input, including the topology of the entire grid. It has a very large memory assumption in the three dimensional unstructured mesh application, which has a great influence on the performance of the parallel version. So we analyze the dependence relationship between these data structures, and updating the original input to be a parallel input mode.

We also analyze some basic operations in many unstructured mesh CFD applications. Due to the indirect access pattern by using the data structure in these codes, those are the irregular applications. Since we have separate the parallel data structure from the numerical computing methods, we focus on finding good data structure to provide good performance for these basic operations and get some initial results.

## 3 NUMERICAL EXPERIENCES

The nodes of the parallel computer we used are connected with Intel Ommi-Path network. Each node is equipped with 64GB RAM and two 12-core 2.5GHz Intel E5-2680 processors. The model has a tetrahedral unstructured mesh with one million grid cells.



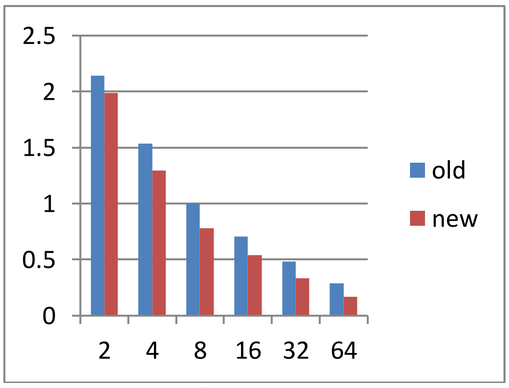


Fig. 1 (left) The ratio of each kernel; (right) Execution time of "con": new vs. old.

The left of Figure 1 gives the ratio between the execution time of each kernel and the total when running the parallel code with different numbers of processor cores. There are five computing kernels. Kernel "con" takes up 60% of the total execution time and becomes the most time-consuming. The right of Figure1 compares the average execution time (in seconds) per iteration of kernel "con" in the old version and in the new when running the codes with different cores, where the parallel code is optimized by using the above methods. Also the memory requirements of the new version are decreased sharply. When running on 64 processor cores, each process occupies only 5% memory capacity of the previous version.

## 4 CONCLUSIONS

- Some methods to optimize the parallel performance of a three-dimensional unstructured mesh CFD code are provided here, including extending the array size when declaration to get rid of unnecessary computations, and updating input mode to decrease memory assumption.
- Some common basic operations used in those unstructured mesh CFD codes are analyzed here. Since the data structure is separated form the computing methods, we focus on finding better ways for data organization and access sequence to fit for the high performance of these basic operations, which is also our next work.

## REFERENCES

[1] Dimitri J. Mavriplis, "*Unstructured mesh related issues in computational fluid dynamics (CFD) –based analysis and design*", 11th International meshing roundtable, September 15-18, 2002. Ithmca, New York, USA.

[2] Bertollic C., Betts A., Loriant N., et al., "*Compiler optimizations for industrial unstructured mesh CFD applications on GPU*", International workshop on languages and compilers for parallel computing, Springer Berlin Heidelberg, 2013.

[3] Xiao Li, Zhang Xiaodong, Kuang Zhengqian, et al., Auto-CFD-NOW: a pre-compiler for effectively parallelizing CFD applications on networks of workstations, Journal of Supercomputing, 38(2), 189-217(2006).

[4] Wang Yongxian, Zhang Lilin, Liu Wei, et al., Performance optimizations for scalable CFD applications on hybrid CPU+MIC heterogeneous computing system with millions of cores, Computers & Fluids, 2018: s00457930:8301038.

[5] Mudalige G.R., Giles M.B, Reguly I, et al., OP2: An active library framework for solving unstructured mesh-based applications on multi-core and many-core architectures, Innovative parallel computing(Inpar), 2012, IEEE, 2012.

[6] Carlo Bertolli, Adam Betts, Giham Mudalige, et al., Design and performance of the OP2 library for unstructured mesh application, Lecture Notes in Computer Science, 7155:191-200, 2011.

[7] Bogdanov P.B, Gorobets A.V, Sukov S.A, Adaptation and optimization of basic operations for an unstructured mesh CFD algorithm for computation on massively parallel accelerators, Computational Mathematics and Mathematical Physics, 53(8): 1183-1194, 2013.

[8] Timothy J. Barth, Parallel CFD algorithms on unstructured meshes, 1995.11.

[9] Gropp W.D, Kaushik D.K, Keyes D.E, et al., Performance modeling and tuning of an unstructured mesh CFD application, ACM/IEEE conference of Supercomputing, 2000.

[10] Emelyanov V, Karpenko A, Volkov K, Development and acceleration of unstructured mesh-based CFD solver, Progress in Flight physics, 2017.

[11] Chen Jun, Parallelization of an unstructured mesh fluid application with black-box solvers, 31th International conference on parallel computational fluid dynamics, Turkey, 2019.