Toward Scalable Voxelization of Meshes with High Growth Rate

Markéta Hrabánková
IT4Innovations
VŠB - Technical University of Ostrava
Ostrava, Czech Republic
marketa.hrabankova@vsb.cz

Ondřej Meca
IT4Innovations
VŠB - Technical University of Ostrava
Ostrava, Czech Republic
ondrej.meca@vsb.cz

Tomáš Brzobohatý
IT4Innovations
VŠB - Technical University of Ostrava
Ostrava, Czech Republic
tomas.brzobohaty@vsb.cz

Lubomír Říha
IT4Innovations
VŠB - Technical University of Ostrava
Ostrava, Czech Republic
lubomir.riha@vsb.cz

Milan Jaroš
IT4Innovations
VŠB - Technical University of Ostrava
Ostrava, Czech Republic
milan.jaros@vsb.cz

Petr Strakoš
IT4Innovations
VŠB - Technical University of Ostrava
Ostrava, Czech Republic
petr.strakos@vsb.cz

***Abstract*—The poster presents a scalable approach that converts the results of large-scale Computational Fluid Dynamics (CFD) simulations into a volumetric representation used by volume rendering-based visualization. Even if this functionality is provided by common post-processing tools, its efficient parallelization requires an appropriate load-balancing. Unfortunately, load-balancing according to the number of cells does not scale for unstructured meshes with high growth rate that is common in CFD. In the poster, we show that with an appropriate redistribution of data among available resources it is possible to perform the operation in just several seconds with significantly improved scalability.**

***Index Terms*—CFD, voxelization, parallelization**

I. Introduction

Many engineers use numerical methods for the simulation of physical phenomena such as Computational Fluid Dynamics (CFD). The simulation process is computationally demanding as high precision simulations require very detailed and therefore large models. It is therefore common to use HPC infrastructure. The ever-increasing precision enables to simulate more complex problems in more detail, but it also leads to a large amount of resulting data that needs to be post-processed and visualized. Utilization of HPC infrastructures can significantly speed up the whole visualization process.

Typically, the process of visualization is composed of applying several filters (computation of streamlines, iso-surfaces, etc.). For efficient utilization of parallel resources, it is desirable to perform as much filters as possible in parallel. However, not all filters can be easily parallelized.

In this poster, we focus on the filter for voxelization (the process of re-sampling of the original data to a structured regular grid of voxels), which is utilized for volume rendering. Its efficient parallelization is challenging due to non-trivial load-balancing for unstructured meshes with high growth rate that is common in CFD simulations. For such meshes, a straightforward distribution according to the number of cells can lead to domains with highly different volumes. It causes inefficient utilization of available resources, low performance, and from the user's point of view it slows down the whole visualization process and examination of the results.

In this poster, we present our newly developed approach that produces a distribution with both (i) a balanced number of cells per process as well as (ii) a balanced volume per process. We show that with this approach, it is possible to redistribute cells among thousand of MPI processes and perform the voxelization in several seconds.

II. Visualization Workflow

To present our approach, we use results computed by OpenFOAM [1] that is publicly available tool for CFD simulation. Then, the proposed high-level visualization workflow of these results is the following: (i) read results by our tool from files in parallel; (ii) redistribute data to improve load balance for voxelization; (iii) compute the voxelization; (iv) store the voxelized data in volumetric database, e.g., in an OpenVDB [2] file (one OpenVDB file per simulation timestep). Finally, OpenVDB data can be imported into Blender [3] for cinematic-style visualization [4] using volume rendering as implemented in Blender Cycles path-tracing renderer [5].

The performance of our approach is compared to implementation in the VTK library and Paraview [6]. ParaView is recommended by OpenFOAM developers for visualization of OpenFOAM results. We start the Paraview in MPI-parallel client-server mode using *pvserver* and *pvbatch* to process data in parallel. The pipeline is as follows: (i) load OpenFoam results in parallel using its parallel format; (ii) use the *Clip* filter to select only a box around interesting part of the input mesh **or** keep the mesh unchanged. (iii) voxelize the data using *ResampleToImage* filter; (iv) visualize the data in Paraview using volume rendering **or** save the voxelized data into OpenVDB file (supported in Paraview 5.10 and newer).

III. Voxelization

Volume rendering is one of the techniques for scientific data visualization which exists for both unstructured and structured data [7]. Algorithms for volume rendering of unstructured data are generally more time demanding and therefore not applicable for interactive visualization of large datasets. Rendering of structured data is faster; however, it requires voxelization (resampling to a regular grid). Once the regular grid is computed, it can be visualized or stored to output database such as an OpenVDB database that can be loaded and interactively rendered by visualization tools such as Blender.

An important part of voxelization algorithms is to find a cell that contains a particular point of the uniform grid. Structures like octrees or kd-trees can be used to locate a correct cell for a given point. In our case, a reversed approach was used. We compute a bounding box for each cell of the unstructured grid and test all voxels from the box whether they lie in the cell or not by the point-in-polyhedron algorithm [8]. The output is stored in the sparse structure that can be easily stored to the OpenVDB database.

IV. Efficient Data Redistribution

The proposed voxelization method with data redistribution is implemented in our open-source tool MESIO [9] originally designed for parallel loading and storing unstructured meshes. The added value of the MESIO library is extremely fast re-balancing of unstructured meshes and simulation results independently to input database format or order of cells within the database. The re-balancing is based on Hilbert's Space-Filling Curve (SFC) and parallel histogram sorting. Usually, decomposed unstructured meshes are balanced w.r.t. the number of cells on each MPI process. Nevertheless, the main benefit of SFC w.r.t. voxelization is straightforward balancing according to the domain volume just by splitting the SFC into equal intervals (in the case of a *box* geometry without holes, otherwise the histogram sorting must be used).

In the poster, we show that neither decompositions with balanced number of cells nor decomposition with balanced volume provides good load balancing. Therefore we propose a hybrid approach which assign more SFC intervals to each MPI processes. We combine intervals with large cells (large volume) and intervals with small cells (small volume) in order to balance both volume and the number of cells per MPI process. As shown in the poster, the voxelization with this balancing scheme significantly reduces load imbalance which results in better scalability and allows to convert meshes with millions cells to volumetric representation with billion voxels in several seconds.

V. Results

The poster presents performance results measured on the Karolina machine CPU partition at IT4Innovations National Supercomputing Center. The partition consists of 720 compute nodes (2 x AMD 7H12, 64 cores, 2.6 GHz; 256 GB DDR4 3200MT/s) interconnected by the Fat Tree network with InfiniBand HDR100. During the experiments, we ran 128 MPI per node.

The use case was prepared in ANSYS and computed by OpenFOAM. The model has 30 millions of nodes and 120 millions of cells. The results were stored into 1024 directories in the OpenFOAM parallel format. Unstructured mesh database size with results was 41 GB.

Graphs in the poster compare the scalability of the voxelization filter in ParaView (*ResampleToImage*) and the approach implemented in our MESIO library. As can be seen in the poster, for meshes with a high growth rate, voxelization stops scaling early. In the case of ParaView, only 16 processes can be utilized effectively. The hybrid domain decomposition provided by the MESIO library scales well up to 1024 processes.

VI. Conclusion

The poster presents an approach for load-balancing of parallel algorithms for distributed voxelization of unstructured meshes with high growth rate that is common in CFD simulation. The approach is implemented in our MESIO library. Its scalability is compared with ParaView. The results show that with the appropriate load-balancing it is possible to implement highly scalable voxelization.

Acknowledgement

This work was supported by the Doctoral grant competition VSB - Technical University of Ostrava, reg. no. CZ.02.2.69/0.0/0.0/19_073/0016945 within the Operational Programme Research, Development and Education, under project DGS/TEAM/2020-008 "Development of a tool for scientific data processing and visualization in VR with multi-user support" and by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90140).

References

[1] "OpenFOAM," 2022. [Online]. Available: https://www.openfoam.com/

[2] K. Museth, "Vdb: High-resolution sparse volumes with dynamic topology," *ACM Trans. Graph.*, vol. 32, no. 3, jul 2013. [Online]. Available: https://doi.org/10.1145/2487228.2487235

[3] "Blender," 2022. [Online]. Available: https://www.blender.org/

[4] K. Borkiewicz, A. J. Christensen, R. Wyatt, and E. T. Wright, "Introduction to cinematic scientific visualization," in *ACM SIGGRAPH 2020 Courses*, ser. SIGGRAPH '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3388769.3407502

[5] K. Museth, "Nanovdb: A gpu-friendly and portable vdb data structure for real-time rendering and simulation," in *ACM SIGGRAPH 2021 Talks*, ser. SIGGRAPH '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3450623.3464653

[6] "ParaView," 2021. [Online]. Available: https://www.paraview.org/

[7] A. Maximo, A. Varshney, and R. Farias, "Improved algorithms for volume rendering and mesh processing," Ph.D. dissertation, 07 2010.

[8] J. Lane, B. Magedson, and M. Rarick, "An efficient point in polyhedron algorithm," *Computer Vision, Graphics, and Image Processing*, vol. 26, no. 1, pp. 118–125, 1984. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0734189X84901336

[9] O. Meca, L. Říha, B. Jansík, and T. Brzobohatý, "Toward highly parallel loading of unstructured meshes," *Advances in Engineering Software*, vol. 166, p. 103100, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965997822000138