Accelerating the visualization of spatio-temporal simulations with non-evolving meshes

ALEJANDRO RIBÉS, EDF Lab Paris-Saclay, France
ANTHONY GEAY, EDF Lab Paris-Saclay, France
MATHIEU WESTPHAL, Kitware Europe, France

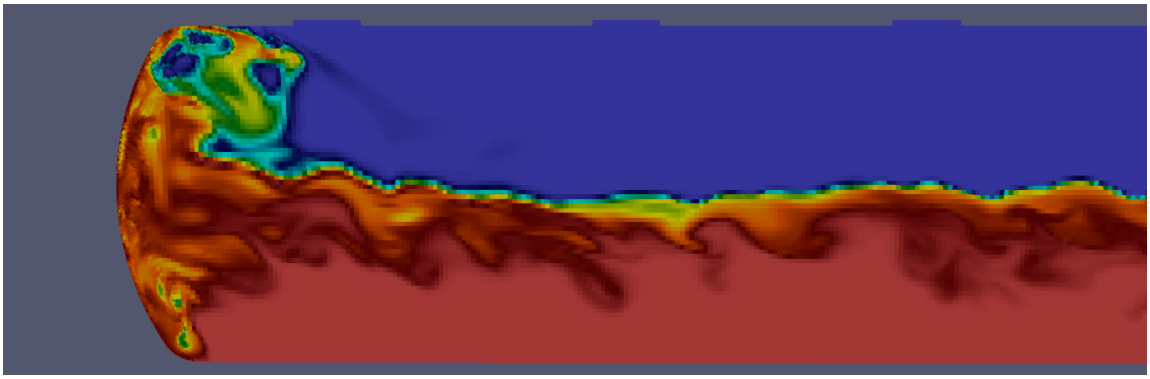

Fig. 1. A screenshot from a temporal dataset, used in our tests, with a non-evolving mesh, from the CFD solver *Code Saturne*

Numerical simulations, such as finite elements, deal with both space and time discretizations of fields. For the discretization of space, a mesh is normally used. Concerning the temporal evolution of the spatial discretization, in the general case, a different mesh can be used per time step. However, it is often the case that the mesh is non-evolving, meaning that it stays the same for all time steps. This case is of great importance in industrial practice. In this work, we show that taking into account the non-evolving nature of a simulation leads to a strong acceleration of the visualization. We demonstrate this point by the use of the ParaView open-source software. We implement this idea and compare the results with the baseline ParaView. We report strong accelerations, speedups of the order of hundreds, in our results.

Additional Key Words and Phrases: high-performance visualization, ParaView, non-evolving mesh

ACM Reference Format:
Alejandro Ribés, Anthony GEAY, and Mathieu Westphal. 2021. Accelerating the visualization of spatio-temporal simulations with non-evolving meshes. 1, 1 (October 2021), 3 pages. https://doi.org/XXXXXX

Authors' addresses: Alejandro Ribés, EDF Lab Paris-Saclay, Boulevard Gaspard Monge, Palaiseau, France, alejandro.ribes@edf.fr; Anthony GEAY, anthony.geay@edf.fr, EDF Lab Paris-Saclay, Boulevard Gaspard Monge, Palaiseau, France, 91120; Mathieu Westphal, mathieu.westphal@kitware.com, Kitware Europe, Lyon, France.

Table 1. Results for the synthetic dataset

	SurfaceFilter	SliceWithPlane
Baseline	25.5137s	17.9882s
Accelerated	0.8198s	0.0674s
Speedup	**31.2**	**266.65**

Table 2. Results for the CFD Code Saturne dataset

	SurfaceFilter	SliceWithPlane
Baseline	7.4548s	2.4890s
Accelerated	0.0693s	0.0032s
Speedup	**107.5**	**777.8**

1 INTRODUCTION

Numerical simulations, such as finite elements, deal with both space and time discretizations of fields. Fields are continuous physical quantities (represented by a number or a tensor) that have values for each point in space and time. For the spatial discretization of fields, a mesh is typically used as geometrical support. Concerning the temporal evolution of the spatial discretization, in the general case, a different mesh is used per time step. However, it is often the case that the mesh is non-evolving, meaning that it stays the same for all time steps. This case is of great importance in industrial practice, in our experience, most industrial simulations belong to this category. Indeed, mesh time-dependent simulations are currently only used for advanced simulation projects.

In this work, we show that taking into account the non-evolving geometrical nature of a simulation leads to a strong acceleration of the visualization operations. We demonstrate this point by use of the ParaView open-source software (www.paraview.org) [1]. We implement this idea, by use of several plugins, and compare the results with the baseline current implementation of ParaView. A static mesh optimized (SMO) implementation of a ParaView filter typically uses a cache to store the geometrical information. It will then only compute operations on field data and attach it to a shallow copy of the cached mesh. A static filter can detect if its cache becomes invalid. In that case, the cache will be recomputed and the general architecture of ParaView does not lose its generality.

2 RESULTS

We present results on **two datasets** using non-evolving meshes. The first dataset contains 16M cells. It is synthetic and has been generated by scripting for test purposes. The second dataset has been obtained by executing the CFD solver *Code Saturne* (www.code-saturne.org) [2], which generated an Ensight Gold Binary output file containing approximately 13M cells and 34M nodes. Figure 1 shows a slice of this dataset.

We chose **two ParaView filters** for benchmarking these datasets: *SurfaceFilter*, which extracts the polygons forming the outer surface of the input dataset; *SliceWithPlane*, which extracts the portion of the input dataset that lies along a specified plane. We remark that *SurfaceFilter* is called by ParaView each time a 3D rendering of an object surface is performed, thus being probably the most CPU time-consuming filter involved in the visualization of fields.

Tables 1 and 2 show the results for both datasets and filters. The presented timings are averaged values of three executions of the code. We observe that speedups for the *SurfaceFilter* are very significant (**31.2** and **107.5**). These speepups are even better, of the order of hundreds, for *SliceWithPlane* (**266.65** and **777.8**).

3 CONCLUSION

We have demonstrated that taking into account the non-evolving nature of a mesh, in a spatio-temporal simulation, leads to strong accelerations on the visualization operations. Thanks to this development the dataset shown in Figure 1 was animated interactively by ParaView on a PC (OS: Linux; CPU: Intel i9-9900K (16) @ 5.000GHz; GPU: NVIDIA GeForce RTX 2080 Ti; RAM: 32 Gb; Disk: SSD), instead of using the ParaView server mode on the GAIA cluster (which contains 64 graphical nodes with NVIDIA V100 GPUs) or the in-situ capabilities of *Code Saturne* [3].

REFERENCES

[1] James Ahrens, Berk Geveci, and Charles Law. 2005. Paraview: An end-user tool for large data visualization. *The visualization handbook* 717, 8 (2005).

[2] Frédéric Archambeau, Namane Méchitoua, and Marc Sakiz. 2004. Code Saturne: A finite volume code for the computation of turbulent incompressible flows-Industrial applications. *International Journal on Finite Volumes* (2004).

[3] Alejandro Ribés, Benjamin Lorendeau, Julien Jomier, and Yvan Fournier. 2015. In-situ visualization in computational fluid dynamics using open-source tools: integration of catalyst into Code_Saturne. In *Topological and Statistical Methods for Complex Data*. Springer, 21–37.