TTK se prépare à intégrer MPI
TTK is Getting MPI-Ready

Eve Le Guillou, Michael Will, Pierre Guillou, Jonas Lukasczyk, Pierre Fortin, Christoph Garth, Julien Tierny

Fig. 1: Results of a real-life use case combining several algorithms implemented in TTK on a dataset (the Turbulent Channel Flow dataset) which was too large (120 billion vertices) to be handled by TTK prior to this work. The dataset is a three-dimensional regular grid with two scalar fields, the pressure of the fluid and its gradient magnitude. The spheres correspond to the pressure critical points and the tubes are the integral lines starting at saddle points. Figure (a) shows all of the produced geometry, while (b) and (c) show parts of the output zoomed in. These images were produced on a quarter of the total dataset due to offscreen rendering related issues.

English Abstract—This system paper documents the technical foundations for the extension of the Topology ToolKit (TTK) to distributed-memory parallelism with the Message Passing Interface (MPI). While several recent papers introduced topology-based approaches for distributed-memory environments, these were reporting experiments obtained with tailored, mono-algorithm implementations. In contrast, we describe in this paper a versatile approach (supporting both triangulated domains and regular grids) for the support of topological analysis pipelines, i.e., a sequence of topological algorithms interacting together, possibly on distinct numbers of processes. While developing this extension, we faced several algorithmic and software engineering challenges, which we document in this paper. Specifically, we describe an MPI extension of TTK’s data structure for triangulation representation and traversal, a central component to the global performance and generality of TTK’s topological implementations. We also introduce an intermediate interface between TTK and MPI, both at the global pipeline level, and at the fine-grain algorithmic level. We provide a taxonomy for the distributed-memory topological algorithms supported by TTK, depending on their communication needs and provide examples of hybrid MPI-thread parallelizations. Detailed performance analyses show that parallel efficiencies range from 20% to 80% (depending on the algorithms), and that the MPI-specific preconditioning introduced by our framework induces a negligible computation time overhead. We illustrate the new distributed-memory capabilities of TTK with an example of advanced analysis pipeline, combining multiple algorithms, run on the largest publicly available dataset we have found (120 billion vertices) on a standard cluster with 64 nodes (for a total of 1536 cores). Finally, we provide a roadmap for the completion of TTK’s MPI extension, along with generic recommendations for each algorithm communication category.

1 INTRODUCTION

Modern datasets are constantly growing in size, due to the continuous improvements of acquisition technologies and computational systems. This growth induces finer levels of details, in turn inducing more complex geometrical structures in the data. To apprehend this complexity, advanced techniques are required for the concise encoding of the core patterns in the data, to facilitate analysis and visualization.

Topological Data Analysis (TDA) serves this
purpose. It is based on robust, multi-scale algorithms, which capture a variety of structural features. However, with the above data size increase, it becomes frequent in the applications that the size of a single dataset exceeds the memory capacity of a single computer, hence requiring the combined memories of distributed systems.

The Topology ToolKit (TTK) [6] is an open-source library which implements a substantial collection of algorithms [1] for topological data analysis and visualization. While most of its algorithms support shared-memory parallelism using multiple threads with OpenMP [5], TTK did not support, up to now, distributed-memory parallelism and thus, was restricted to datasets of limited size, fitting in the memory of a single computer.

2 Contributions

This system paper addresses this issue by documenting the technical foundations which are required for the extension of TTK to distributed-memory parallelism using multiple processes with the Message Passing Interface (MPI) [4], hence enabling the design of topological pipelines for the analysis of large-scale datasets on supercomputers. Specifically, after formalizing our conceptual model for the distributed representation of the input and output data, we present the extension of TTK’s internal triangulation data-structure (a central component of its performance and versatility) to the distributed setting. We also document an interface between TTK and MPI enabling the consistent combination of multiple topological algorithms within a single, distributed pipeline.

Unlike previous work, this paper does not focus on the distributed computation of a specific topological object (such as merge trees or persistence diagrams). Instead, it documents the necessary building blocks for the extension to the distributed setting of a diverse collection of topological algorithms such as TTK. To evaluate the efficiency of our extension, we document several examples, extending to the distributed setting a selection of topological algorithms. We also provide a taxonomy of TTK’s topological algorithms, depending on their communication needs and provide examples of hybrid MPI+thread parallelizations for each category, with detailed performance analyses. We illustrate the new distributed capabilities of TTK with an example of advanced analysis pipeline, combining multiple algorithms, run on a dataset of 120 billion vertices distributed on 64 nodes of 24 cores each. Finally, we provide a roadmap for the completion of TTK’s MPI extension, with generic recommendations for each algorithm communication category. This work has been integrated in the main source code of TTK and is available in open-source.

3 Conclusion

In this paper, we presented a software framework for the support of topological analysis pipelines in a distributed-memory model. Specifically, we instantiated our framework with the MPI model, within the Topology ToolKit (TTK). The next step consists in adding distributed-memory support to all of TTK’s topological algorithms. For more complex topological algorithms (such as Discrete Morse Sandwich, Topological Simplification, Contour tree or Rips complex computations), the port may be much more complicated than the algorithms described in this paper. For each of these algorithms, their distributed-memory parallelization may be a substantial research problem, on which we will focus in future work.

Acknowledgments

This work is partially supported by the European Commission grant ERC-2019-COG “TORI” (ref. 863464, https://erc-tori.github.io/). The code of this paper [3] is integrated in TTK (https://topology-tool-kit.github.io/).

References