In Situ Data Steering on Sedimentation Simulation with Provenance Data

Vítor Silva, José Camata, Daniel de Oliveira, Alvaro L G A Coutinho, Patrick Valduriez, Marta Mattoso

In Situ Data Steering on Sedimentation Simulation with Provenance Data

Vítor Silva
Computer Science - COPPE
Federal University of Rio de Janeiro
silva@cos.ufrj.br

Daniel de Oliveira
Institute of Computing
Fluminense Federal University
danielcmo@ic.uff.br

Patrick Valduriez
Inria and LIRMM
patrick.valduriez@inria.fr

José Camata
Civil Engineering - COPPE
Federal University of Rio de Janeiro
camata@nacad.ufrj.br

Alvaro L.G.A. Coutinho
Civil Engineering - COPPE
Federal University of Rio de Janeiro
alvaro@nacad.ufrj.br

Marta Mattoso
Computer Science - COPPE
Federal University of Rio de Janeiro
marta@cos.ufrj.br

***Abstract*—Parallel adaptive mesh refinement and coarsening (AMR) are optimal strategies for tackling large-scale simulations. libMesh is an open-source finite-element library that supports parallel AMR and is used in multiphysics applications. In complex simulation runs, users have to track quantities of interest (residuals, errors estimates, etc.) to control as much as possible the execution. However, this tracking is typically done only after the simulation ends. This paper presents DfAnalyzer, a solution based on provenance data to extract and relate strategic simulation data for online queries. We integrate DfAnalyzer to libMesh and ParaView Catalyst, so that queries on quantities of interest are enhanced by in situ visualization.**

Keywords— libMesh; in situ data analysis; raw data analysis; dataflow management; provenance.

I. OVERVIEW

Parallel adaptive mesh refinement and coarsening (AMR) are optimal strategies for tackling large-scale sedimentation simulations. libMesh [1] is an open-source finite-element library that supports parallel AMR and has been used in several multiphysics applications. As an application built upon the libMesh library, libMesh-sedimentation[1] simulates turbidity currents typically found in geological processes. The sediment transported due to fluid motion is described by a Eulerian framework in which the mathematical models result from the incompressible Navier-Stokes equation (fluid) combined with an advection-dominated transport equation (sediment concentration). libMesh-sedimentation employs a variational multiscale finite element method [2] where a staggered approach is used to evolve in time the coupled fluid-sediment equations.

In complex simulation runs, users need *in situ* data steering to track quantities of interest such as residuals, errors estimates, etc. to control as much as possible the execution. Often users need to analyze raw data from multiple related files generated in different solver steps at runtime. To help on this data steering, in a previous work [3], we defined a dataflow provenance aware database schema, PROV-Df, as an extension of W3C PROV [4], to represent selected domain raw data captured by a Scientific Workflow Management System (SWMS) with provenance data [5]. In this SWMS approach, the simulation has to be modeled as a workflow, and we use the Chiron SWMS [6] to control the execution while extracting domain data from the files generated by the simulation. Chiron is a non-intrusive solution with respect to the simulation code. However, when using a SWMS to manage the parallel execution of the workflow, it often conflicts with the simulation solver scheduling and data parallelism. Most SWMS have provenance data support, but do not manage workflow activities that invoke highly parallel codes and, consequently, they may jeopardize performance. Therefore, we propose online provenance and dataflow queries for solver libraries or other workflows without using a SWMS.

Different from the SWMS approach, we present ARMFUL[2], a component-based architecture that enables the Analysis of Raw Data from Multiple files. ARMFUL allows for raw data analysis by extracting raw data from files and relating them through dataflow provenance data. Furthermore, ARMFUL data analysis support can be coupled to a simulation code, library, script or even a SWMS. In this paper we instantiate ARMFUL as DfAnalyzer, which is coupled to the libMesh-sedimentation solver to extract and relate strategic simulation data for online queries. The DfAnalyzer approach keeps all the parallel execution control at the simulation solver.

In the same way, users plug visualization components to their simulation code, in DfAnalyzer, we provide for raw data extraction components to be plugged in strategic places of the simulation code, *e.g.* registering time step control data, solver convergence, AMR behavior, and data dependencies between solver steps. DfAnalyzer can also take advantage of *in situ* data managed by ParaView Catalyst [7]. In this case, DfAnalyzer communicates with ParaView to extract data from Catalyst and relate it to its provenance database, benefiting from data that is already in memory. The resulting provenance database of DfAnalyzer complements the data coverage supported by Catalyst queries. While Catalyst supports query processing on a specific dataset (*e.g.*, XDMF file addressing HDF5 files), DfAnalyzer allows for users to monitor the file flow generation during the simulation execution and to relate data elements from different solver steps (dataflow management). Fig. 1 shows an overview of DfAnalyzer, where the query results, obtained at any time during the simulation, allow for *in situ* data steering. Queries can show, for example, the appearance

[1] https://bitbucket.org/nacadhpc4e/libmesh-sedimentation

[2] https://hpcdb.github.io/armful

of sediments for a specific time step or solver convergence difficulties, which helps to change set-up parameters such as tolerances, at runtime.

DfAnalyzer has two main components, named as Provenance Gatherer (PG) and Query Processor (QP), to manage dataflow at physical (*i.e.*, file flow) and logical (*i.e.*, data flow) levels [3]. PG component captures provenance and domain-specific data from application code, generating a JSON file with the extracted data and their dependencies. Then, all captured provenance and domain-specific data are loaded into the database that follows PROV-Df schema and is managed by the column-store Database Management System MonetDB [8]. QP helps users to submit SQL queries to the provenance database and to show results.

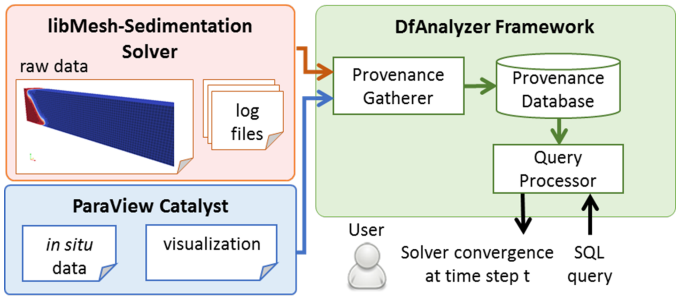

Fig. 1. Overview of our solution with DfAnalyzer.

To complement *in situ* provenance database analysis with visual information at runtime, ParaView Catalyst is embedded into libMesh-sedimentation. Catalyst can access libMesh data structures, extracting additional data information from solution fields and performing live visualization. Catalyst can provide profiles of the deposited sediments that are the relevant quantities of interest for geologists. Those profiles are generated through filters applied on the original data and are defined in Python scripts at the beginning of the simulation. Comparing this solution to the *ad-hoc* programs for raw data analysis, our approach only needs to export Python scripts using analyses already developed in ParaView UI, without modifying the libMesh-sedimentation code. Then, Catalyst will perform the parallel raw data analysis and visualization based on those scripts.

II. Experimental Evaluation

We ran DfAnalyzer integrated to libMesh-sedimentation and ParaView Catalyst using 1,040 cores in Stampede cluster[3], at Texas Advanced Computing Center (TACC). This proof-of-concept experiment executes the sedimentation solver with adaptive mesh refinement and consumes an initial input mesh with 480×80×80 hexahedra. The adaptive mesh refinement is applied every 10 time steps. The simulation took 137.75 minutes running 200 time steps. Catalyst is also invoked every 10 time steps. Considering the total elapsed time, 136.80 minutes correspond to the solver elapsed time (99.31% of the total elapsed time), 0.60 minute to the *in situ* data analysis (0.44%), and 0.35 minute to the provenance data capture (0.25%) – and manipulates approximately 3.17 GB of data – 5.7 MB corresponds to the provenance database, 3.15 GB to raw data files, and 15.0 MB to log files. Even in real-life larger runs, where number of files and their sizes are much larger, execution time overhead of DfAnalyzer will remain negligible compared to the solver time.

Figs. 2 and 3 present two analytical queries that users are able to run using DfAnalyzer's provenance database. The query in Fig. 2 monitors the appearance of sediments in the domain bottom layer for a specific time step. Fig. 3 shows how users can monitor the progress of their solver at runtime. For example, in Fig. 3 the query returns solver data in a specific time step that allows for users to steer their simulation based, for instance, on the solver convergence. When solver convergence or AMR difficulties are detected, users can benefit from provenance support to trace simulation data from previous non-linear iterations and time steps or meshes, since our provenance database represents the dataflow while being generated by the sedimentation simulation execution and acting in runtime.

```
SELECT time_step, x, y, z, d
FROM horizontal_line_extraction_1
WHERE simulation_id = 1 AND d <> 0;
```

Fig. 2. Monitoring query: analysis of a horizontal line in the mesh associated to the simulation identifier 1, when the deposited sediment (d) in the bottom layer is different from zero.

```
SELECT f.final_linear_residual,
       f.final_non_linear_residual,
       s.final_linear_residual, s.final_non_linear_residual
FROM solver_simulation_fluid as f,
     solver_simulation_sediments as s
WHERE f.converged = true  AND s.converged = true
    AND f.simulation_id = s.simulation_id
    AND f.time_step = s.time_step AND f.simulation_id = 1;
```

Fig. 3. Query for debugging and user steering: analysis of the algorithm output parameters after the convergence of the simulation solver in the fluid and sediments loops in a specific simulation.

Acknowledgments

This work was partially funded by CNPq, CAPES, FAPERJ and Inria (MUSIC project) and the EU H2020 Programme jointly with MCTI/RNP-Brazil (HPC4E project). Computer time on Stampede is provided by TACC at UT-Austin.

References

[1] B.S. Kirk, J.W. Peterson, R.H. Stogner, and G.F. Carey. libMesh : a C++ library for parallel adaptive mesh refinement/coarsening simulations. *Engineering with Computers*, 22(3–4):237–254, 2006.

[2] G.M. Guerra, S. Zio, J.J. Camata, J. Dias, R.N. Elias, M. Mattoso, P.L. B. Paraizo, A.L. G. A. Coutinho, and F.A. Rochinha. Uncertainty quantification in numerical simulation of particle-laden flows. *Computational Geosciences*, 20(1):265–281, 2016.

[3] V. Silva, D. de Oliveira, P. Valduriez, and M. Mattoso. Analyzing related raw data files through dataflows. *CCPE*, 28(8):2528–2545, 2016.

[4] P. Groth and L. Moreau. W3C PROV - An Overview of the PROV Family of Documents. Available at: https://www.w3.org/TR/prov-overview, 2013.

[5] S.B. Davidson and J. Freire. Provenance and scientific workflows: challenges and opportunities. *ACM SIGMOD*, 1345–1350, 2008.

[6] E. Ogasawara, J. Dias, V. Silva, F. Chirigati, D. Oliveira, F. Porto, P. Valduriez, and M. Mattoso. Chiron: A Parallel Engine for Algebraic Scientific Workflows. *CCPE*, 25(16):2327–2341, 2013.

[7] U. Ayachit, A. Bauer, B. Geveci, P. O'Leary, K. Moreland, N. Fabian, and J. Mauldin. ParaView Catalyst: Enabling In Situ Data Analysis and Visualization. *In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, 25–29, 2015.

[8] P.A. Boncz, M.L. Kersten, and S. Manegold. Breaking the memory wall in MonetDB. *Communications of the ACM*, 51(12):77, 2008.

[3] https://portal.tacc.utexas.edu/user-guides/stampede