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

RUNTIME STEERING OF PARALLEL CFD SIMULATIONS

Renan SOUZA[*], José J. CAMATA[†], Marta MATTOSO[§] AND Alvaro L. G. A. COUTINHO[‡]

[*]IBM Research
Avenida Pasteur, 138 - Urca Rio de Janeiro, RJ, 22290-240, Brazil
e-mail: rfsouza@br.ibm.com, web page: https://www.research.ibm.com/labs/brazil/

[†]Computer Science Department
Federal University of Juiz de Fora
Rua José Lourenço Kelmer, s/n, São Pedro, Juiz Fora, MG, 36036-900, Brazil
e-mail: camata@ice.ufjf.br - Web page: https://www.ufjf.br/deptocomputacao/

[§]Computer Science Department
COPPE/Federal University of of Rio de Janeiro
Av. Horácio Macedo 2030, Room H 319, Rio de Janeiro, RJ 21941-972, Brazil
e-mail: marta@cos.ufrj.br - Web page: https://www.cos.ufrj.br/index.php/en/

[‡]High-Performance Computing Center
COPPE/Federal University of Rio de Janeiro
Av. Horácio Macedo, 2030 Room I-248, Rio de Janeiro, RJ 21941-598, Brazil
e-mail: alvaro@nacad.ufrj.br - Web page: www.nacad.ufrj.br

Key words: Data Analytics, CFD methodology, Parallel CFD applications

Abstract. Fluid flow simulations in high-performance computers are still costly. They often involve the selection of many computational parameters and options. The set-up of such parameters is usually a trial-and-error process even for experienced users. In this paper, we show how to extend in-situ visualization techniques with in-transit data analysis to provide information to help steer the simulations at runtime. Often, by only observing a region of interest, an experienced analyst can infer that something is not going well, deciding to stop it or change parameters. However, to do that, visual information should be complemented with information regarding the evolution of quantities of interest and changes of parameters. We use for the simulations the `libMesh` library, which provides a platform for parallel, adaptive, multiphysics finite element computations. We discuss the integration of `libMesh` with in-situ visualization and in-transit data analysis tools. We present a parallel performance analysis showing that the overhead for both in-situ visualization and in-transit data analysis is negligible. The data analysis tools register the provenance of the simulation data for reproducibility, including the runtime changes. We show that the integration of such tools enables monitoring the quantities of interest at runtime, steer the simulation based on the solver convergence or other data and visual information, and analyze the consequences of the steering in a real fluid flow simulation.

1 INTRODUCTION

Complex high-fidelity fluid flow simulations in high performance computers are still costly. They often involve the selection of many computational parameters and options. The set-up of such parameters can be a cumbersome task, and there is no guarantee that they will lead to a successful simulation. Usually, this is a trial-and-error process, even for experienced users. Tracking at runtime some quantities of interest from output files is the regular procedure and, whenever possible, computations are halted, using checkpoint/restart procedures to resume with a new set of parameters or resubmitting the job to the queue. The typical simulation workflow involves the following steps: (i) preprocessing and mesh generation; (ii) time stepping, saving data on disk when required; and (iii) post-processing, typically visualizing the data generated by the simulation and extracting relevant information on the quantities of interest. For large-scale problems, this workflow involves saving a considerable amount of raw data in persistent storage. In-situ visualization techniques circumvent the storage bottleneck, removing the necessity of transferring data to persistent storage before visualization [3]. In this paper, we show how to extend in-situ visualization techniques with in-transit data analysis to provide information to help steer the simulations at runtime. Often, by only observing a region of interest, an experienced analyst can infer that something is not going well in the simulation, deciding to stop it or change parameters. Preferably, to optimize resource use, these actions should be at runtime. However, to do that, visual information should be complemented with information regarding the evolution of quantities of interest, residual norms, the number of linear and nonlinear iterations, often within a specific time window, not just the current values. To obtain this complementary information, often it is necessary to write specific code to identify the files related to the time window of interest, opening and parsing them to obtain specific values and tracking their evolution.

2 CASE STUDY

Of particular interest here are turbidity currents, underflows responsible for sediment deposits that generate that host possible hydrocarbon reservoirs. The mathematical model for turbidity currents involves solving coupled high Reynolds number incompressible fluid flow and transport. We use for the simulations the `libMesh` library [1], which provides a platform for parallel, adaptive, multiphysics finite element computations. `libMesh` supports adaptive mesh refinement and coarsening (AMR/C) on general unstructured meshes with a variety of error estimators.

To allow for runtime steering, the first step is modeling the simulation as a workflow and identifying analysis and adaptation points [4]. Then, we add calls to in-situ visualization (`ParaView Catalyst`) and in-transit data analysis tools (`DfAnalyzer` [2] and `DfAdapter` [4]) in the simulation. `DfAnalyzer` registers the workflow's provenance data and `DfAdapter` [4] registers the provenance data of the adaptations performed by the user. In this case study, the user adapts input parameters for the numerical solvers, specified in a setup configuration file. The simulation checks, at every time step, if any modification has been made to this file. If so, redefines the parameters in case of modification. Then,

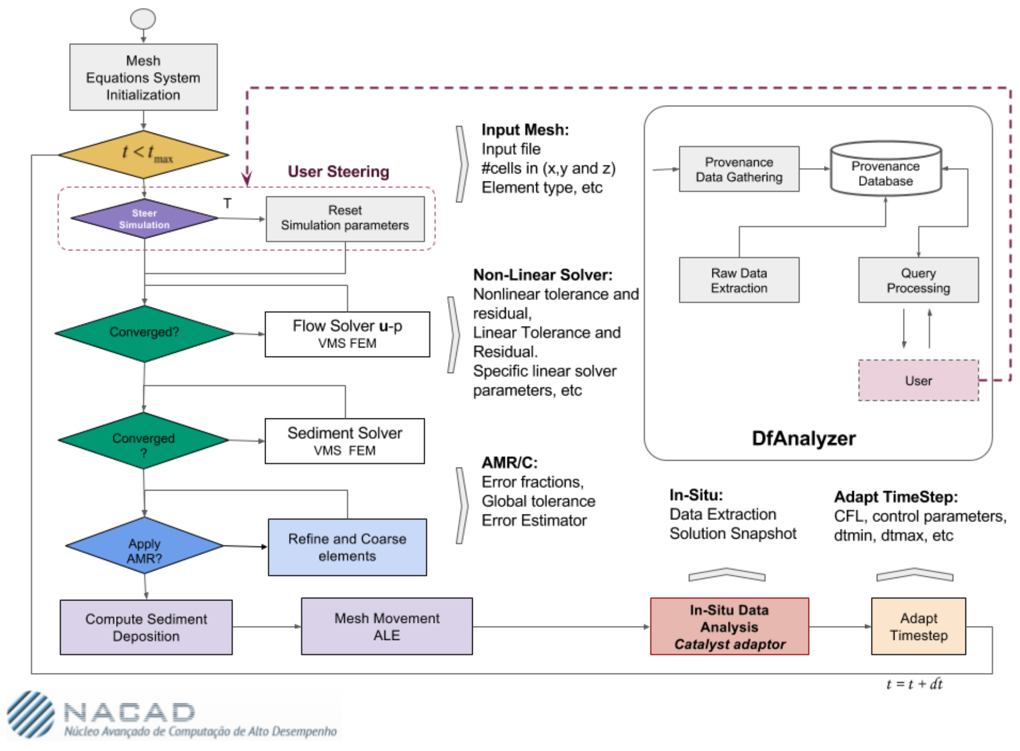

Mesh Equations System Initialization
$t < t_{max}$
User Steering
Steer Simulation
T
Reset Simulation parameters
Converged?
Flow Solver u-p VMS FEM
Converged ?
Sediment Solver VMS FEM
Apply AMR?
Refine and Coarse elements
Compute Sediment Deposition
Mesh Movement ALE
In-Situ Data Analysis Catalyst adaptor
Adapt Timestep
$t = t + dt$
Input Mesh: Input file #cells in (x,y and z) Element type, etc
Non-Linear Solver: Nonlinear tolerance and residual, Linear Tolerance and Residual. Specific linear solver parameters, etc
AMR/C: Error fractions, Global tolerance Error Estimator
Provenance Data Gathering
Provenance Database
Raw Data Extraction
Query Processing
User
DfAnalyzer
In-Situ: Data Extraction Solution Snapshot
Adapt TimeStep: CFL, control parameters, dtmin, dtmax, etc
NACAD
Núcleo Avançado de Computação de Alto Desempenho

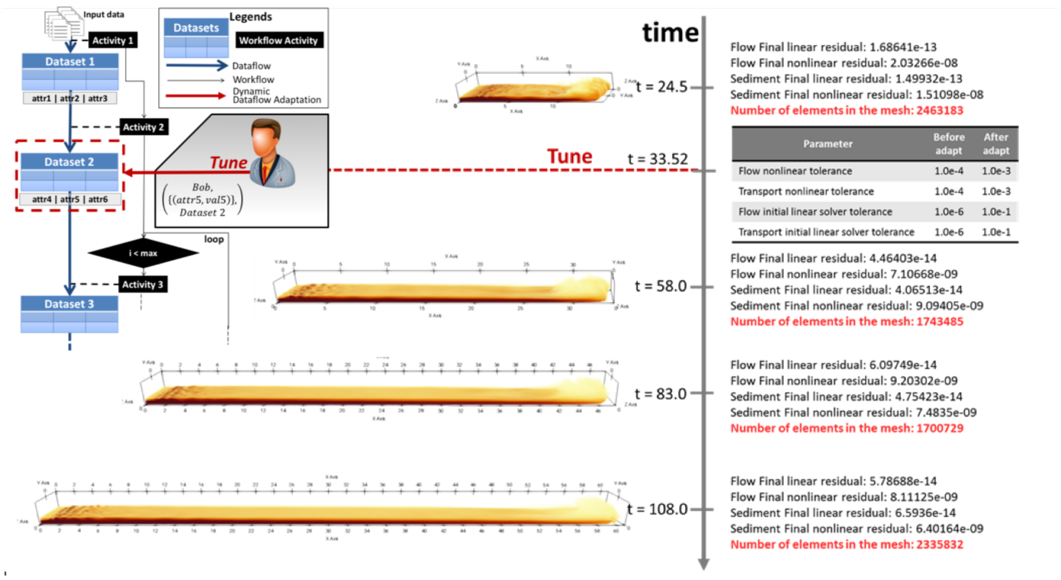
Figure 1: `libMesh` with In–Situ Visualization and In-Transit Data Analysis.

while the simulation runs, the provenance data captured by these two tools are stored in a provenance database where other data, like quantities of interest and references to raw data files, are also stored in an integrated way so that users can perform data analysis at runtime. Figure 1 shows the integration of the `libMesh`-based simulation with in-situ visualization and in-transit data analysis tools.

Performance analysis for a real case of turbidity currents simulations shows that the overhead for both in-situ visualization and in-transit data analysis is negligible. Our tools enable monitoring the sediment appearance at runtime and steer the simulation based on the solver convergence and visual information on the sediment deposits. When users steer the simulation, our tools enable further data analysis on how a particular action (*e.g.*, a fine-tune of a simulation parameter), influenced the simulation. Thus enhancing the analytical power of turbidity currents simulations, as shown in Figure 2.

3 CONCLUSIONS

In this paper, we presented a solution that integrates CFD simulation code with in-situ visualization and in-transit data analysis tools to allow for runtime steering of parallel CFD simulations. We discussed how the simulation, implemented on top of `libMesh`, is integrated with `Para View Catalyst`, for in-situ visualization, and with `DfAnalyzer` and `DfAdapter`, for in-transit data analysis. These tools work as visualization libraries in the



Flow Final linear residual: 1.68641e-13
Flow Final nonlinear residual: 2.03266e-08
Sediment Final linear residual: 1.49932e-13
Sediment Final nonlinear residual: 1.51098e-08
Number of elements in the mesh: 2463183

Parameter	Before adapt	After adapt
Flow nonlinear tolerance	1.0e-4	1.0e-3
Transport nonlinear tolerance	1.0e-4	1.0e-3
Flow initial linear solver tolerance	1.0e-6	1.0e-1
Transport initial linear solver tolerance	1.0e-6	1.0e-1

Flow Final linear residual: 4.46403e-14
Flow Final nonlinear residual: 7.10668e-09
Sediment Final linear residual: 4.06513e-14
Sediment Final nonlinear residual: 9.09405e-09
Number of elements in the mesh: 1743485

Flow Final linear residual: 6.09749e-14
Flow Final nonlinear residual: 9.20302e-09
Sediment Final linear residual: 4.75423e-14
Sediment Final nonlinear residual: 7.4835e-09
Number of elements in the mesh: 1700729

Flow Final linear residual: 5.78688e-14
Flow Final nonlinear residual: 8.11125e-09
Sediment Final linear residual: 6.5936e-14
Sediment Final nonlinear residual: 6.40164e-09
Number of elements in the mesh: 2335832

Figure 2: Runtime steering of the maximum tolerance for the inexact Newton method reduced the execution time by 10 days (37%), with overhead $< 1\%$ and allowed job to finish successfully. The red lines show how the mesh was adapted based on fine tunings. Job executed on 480 cores.

sense that strategic calls are inserted within the simulation code. Together, `DfAnalyzer` and `DfAdapter` capture quantities of interest, data extracted from raw files, and provenance of the runtime adaptations performed by a user while steering the simulation. These captured data are related and stored in a provenance database available for runtime data analysis. Using a real case study of a turbidity current simulation, we can verify that our solution enabled the correlation of which parameters contributed to significant reductions of simulation time. Users can then steer again based on the analysis of the impact of each previous action. Without our tools, the runtime steering could be error-prone and compromise the reliability of the results. We also observed that the added overhead for provenance and steering accounted for less than 1% of total simulation time.

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): 237-254 (2006).

[2] Vítor Silva et al. DfAnalyzer: runtime dataflow analysis of scientific applications using provenance. Proceedings of the VLDB Endowment 11.12: 2082–2085 (2018).

[3] J. J. Camata, et al, In situ visualization and data analysis for turbidity currents simulation, Comput. Geosci., 110:23–31 (2018).

[4] R. Souza et al, Keeping track of user steering actions in dynamic workflows. Future Generation Computer Systems, 99: 624–643 (2019).