Computers and Geosciences 110 (2018) 23–31

Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

Research paper

In situ visualization and data analysis for turbidity currents simulation

Jose J. Camata [a,c], Vítor Silva [b,c], Patrick Valduriez [d], Marta Mattoso [b,c], Alvaro L.G.A. Coutinho [a,c,*]

[a] High-Performance Computing Center and Department of Civil Engineering, Brazil
[b] Department of Computer Science, Brazil
[c] COPPE, Federal University of Rio de Janeiro, Brazil
[d] Inria and LIRMM, University of Montpellier, France

ARTICLE INFO

Keywords:
Turbidity currents
In situ visualization
In transit data analysis
Adaptive mesh refinement and coarsening
Parallel computing

ABSTRACT

Turbidity currents are underflows responsible for sediment deposits that generate geological formations of interest for the oil and gas industry. LibMesh-sedimentation is an application built upon the libMesh library to simulate turbidity currents. In this work, we present the integration of libMesh-sedimentation with in situ visualization and in transit data analysis tools. DfAnalyzer is a solution based on provenance data to extract and relate strategic simulation data in transit from multiple data for online queries. We integrate libMesh-sedimentation and ParaView Catalyst to perform in situ data analysis and visualization. We present a parallel performance analysis for two turbidity currents simulations showing that the overhead for both in situ visualization and in transit data analysis is negligible. We show that our tools enable monitoring the sediments appearance at runtime and steer the simulation based on the solver convergence and visual information on the sediment deposits, thus enhancing the analytical power of turbidity currents simulations.

1. Introduction

Turbidity currents are particle-laden underflows where the main driver is turbulence. According to Meiburg and Kneller (2010), turbidity currents Reynolds number in nature is of $\mathcal{O}(10^9)$. Thus particles can be carried for long distances and eventually they will settle, being responsible for sediment deposits that generate geological formations of considerable interest for the oil and gas industry. Sedimentation and erosion promoted by such particle-laden flows can mold the seabed, producing different geological structures like canyons, dunes, and ripples.

Meiburg and Radhakrishnan (Meiburg et al., 2015) review models and computational approaches for modeling gravity and turbidity currents. They vary from simple conceptual models, depth-averaged models, like shallow-water approximations, to more realistic depth-resolved models, based on the three-dimensional Navier-Stokes equations. Possible computational approaches, in this case, involve direct numerical simulation (DNS), large-eddy simulations (LES) and Reynolds averaged Navier-Stokes simulations (RANS). We use an LES finite element approach based on the residual-based variational multiscale (RBVMS) method as described in Guerra et al. (2013). However, to improve the front resolution, we extend the parallel adaptive mesh refinement/coarsening strategy used by Rossa and Coutinho (2013) for simulating three-dimensional lock-exchange configurations to the RBVMS method. Three-dimensional adaptive mesh refinement and coarsening (AMR/C) poses several challenges regarding parallel performance, but according to Burstedde et al. (2010), AMR/C is optimal for tackling large-scale problems governed by partial differential equations. Other software using AMR/C for similar problems are Fluidity-ICOM (Londonothers, 2017) and the Gerris solver (Popinet, 2017).

The standard turbidity current simulation workflow involves the following steps: (i) preprocessing and mesh generation; (ii) time stepping, saving data on disk when required, that is, velocity, pressure, sediment concentrations; and (iii) post-processing, typically visualizing the data generated by the simulation and extracting relevant information on the quantities of interest. When AMR/C is used, mesh data are also saved in step (ii). For large-scale problems, this workflow involves saving a huge amount of raw data in persistent storage.

* Corresponding author. High-Performance Computing Center and Department of Civil Engineering, Federal University of Rio de Janeiro, Brazil.
E-mail addresses: camata@nacad.ufrj.br (J.J. Camata), silva@cos.ufrj.br (V. Silva), patrick.valduriez@inria.fr (P. Valduriez), marta@cos.ufrj.br (M. Mattoso), alvaro@nacad.ufrj.br (A.L.G.A. Coutinho).

https://doi.org/10.1016/j.cageo.2017.09.013
Received 3 May 2017; Received in revised form 14 August 2017; Accepted 25 September 2017
Available online 28 September 2017

In situ visualization techniques circumvent the storage bottleneck by removing the necessity of first storing data to persistent storage before processing. A recent review discusses the advantages of these techniques (Bauer et al., 2016). Besides savings in persistent storage, we can indeed generate more detailed visualizations, since in situ techniques directly access the memory allocated by the simulation codes. Then, we can, in principle, produce pictures for every time step, which is practically impossible in the standard workflow. We can extend these ideas, using in situ visualization techniques to provide information to help control the simulations. Often, by only observing a deposition pattern, an experienced interpreter can infer that something is not going well in the simulation, deciding to stop it or change parameters, preferably at runtime, resuming the simulation. However, to do that, the visualization should be complemented with information regarding *the evolution* of quantities of interest, such as residual norms, number of linear and nonlinear iterations, often within a specific time window, not just *the current* values. To obtain this complementary information, even the experienced interpreter has difficulty in identifying the files related to the time window, opening and parsing them to obtain specific values and tracking their evolution. The present paper discusses how to integrate in situ visualization with in transit data analysis techniques in large-scale parallel three-dimensional turbidity current simulations.

The rest of this work is organized as follows. Section 2 introduces the governing equations, and the numerical formulation used to simulate turbidity currents using libMesh (Kirk et al., 2006). Section 3 describes our in transit data analysis approach using DfAnalyzer tool (Silva et al., 2017) and in situ data extraction and visualization using ParaView Catalyst (Ayachit et al., 2015). We also discuss in this section how we integrate libMesh with the data analysis tools. Section 4 provides numerical results and a parallel performance evaluation of our solution solving two turbidity current scenarios. The results show that the overhead of in situ visualization and in transit data analysis is negligible, while the added analytical power enables monitoring deposition patterns and steering simulations. The paper ends with a summary of our conclusions.

2. Turbidity currents simulation in libMesh

2.1. Governing equations

This section establishes the mathematical setting for the numerical simulation of turbidity currents within a Eulerian−Eulerian framework. The flows of interest here are mainly driven by small density differences promoted by the heterogeneous presence of sediment particles within the fluid. The double mention to Eulerian is to emphasize that suspended particles, assumed to be present in a dilute proportion in the mixing with a clear fluid, are modeled as a continuum, which motion is governed by an advection dominated transport equation. Here we adopt the simplest three-dimensional depth-resolved model, considering just one sediment granulometry. More complex models can be found in (Necker et al., 2002, 2005; Nasr-Azadani and Meiburg, 2011; Camata et al., 2012; Guerra et al., 2016).

The suspension flow is governed by the incompressible Navier–-Stokes equations considering the Boussinesq approximation

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \frac{1}{\sqrt{Gr}}\nabla^2\mathbf{u} + \mathbf{e}^g c \quad \text{in} \quad \Omega\times[0,t_f] \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \quad \text{in} \quad \Omega\times[0,t_f] \tag{2}$$

coupled with the equation for sediment transport

$$\frac{\partial c}{\partial t} + (\mathbf{u} + u_s\mathbf{e}^g)\cdot\nabla c = \frac{1}{Sc\sqrt{Gr}}\nabla^2 c \quad \text{in} \quad \Omega\times[0,t_f] \tag{3}$$

where $\mathbf{u}$, p, c, t, are respectively, non-dimensional velocity, pressure, sediment concentration and time. The dimensionless velocity u_s quantifies the settling velocity of the particles and $\mathbf{e}^g$ is the direction of gravity. Gr is the Grashof number, expressing the ratio between buoyancy and viscous effects given by

$$Gr = \left(\frac{u_b H}{\nu}\right)^2 \tag{4}$$

with ν the fluid kinematic viscosity, H a characteristic length of the flow, and u_b the buoyancy velocity. A second dimensionless number is the Schmidt number (Sc) that gives the ratio between diffusion and viscous effects,

$$Sc = \frac{\nu}{\kappa} \tag{5}$$

where κ is the diffusivity coefficient.

Essential and natural conditions for Eq. (1) are $\mathbf{u} = \mathbf{g}$ on Γ_g and $\mathbf{n}\cdot\left(-p\mathbf{I} + \frac{1}{\sqrt{Gr}}\nabla\mathbf{u}\right) = \mathbf{h}$ on Γ_h, where $\mathbf{g}$ and $\mathbf{h}$ are given functions, $\mathbf{n}$ is the unit outward normal vector of Γ_h. Γ_g and Γ_h are subsets of the domain boundary Γ. Initial conditions for velocity are chosen to respect the divergence free condition. For Eq. (3), essential conditions may be applied as $c = \bar{c}$ on Γ_{in}. We also apply no-flux boundary conditions at boundary Γ_T by imposing

$$u_s c - \mathbf{n}\cdot\left(\frac{1}{Sc\sqrt{Gr}}\nabla c\right) = 0 \qquad \text{on} \qquad \Gamma_T \tag{6}$$

This condition ensures that no particle is transported across this boundary. We also assume that particles leave the flow due to sedimentation. This is accomplished by imposing a convective boundary condition at Γ_b (typically the bottom wall).

$$\frac{\partial c}{\partial t} = \mathbf{n}\cdot(u_s\nabla c) \qquad \text{on} \qquad \Gamma_b \tag{7}$$

with $\Gamma = \Gamma_{in}\cup\Gamma_b\cup\Gamma_T$. No explicit particle resuspension mechanism, allowing particles going back to the flow after hitting the bottom, like erosion, is included. In fact, no significant amount of resuspension is expected for the flow conditions analyzed here (Necker et al., 2005). We compute the deposited particle layer thickness by integrating in time the particle flux through the bottom, that is,

$$D(\mathbf{x},t) = \int_0^t u_s\ c(\mathbf{x},t)d\tau \tag{8}$$

2.2. Weak form of the governing equations

Assuming that the test and weight functions belong to the standard discrete finite element spaces, the weak form for the incompressible Navier–Stokes, based on the Residual-Based Variational Multiscale method (RBVMS) reads

$$\begin{aligned}
&\left(\mathbf{w}^h, \frac{\partial \mathbf{u}^h}{\partial t}\right)_{\Omega^h} + \left(\mathbf{w}^h, \mathbf{u}^h - \tau_M\mathbf{r}_M, \nabla\mathbf{u}^h\right)_{\Omega^h} + \left(q^h, \nabla\cdot\mathbf{u}^h\right)_{\Omega^h} \\
&-\left(\nabla\cdot\mathbf{w}^h, p^h\right)_{\Omega^h} + \left(\nabla^s\mathbf{w}^h, \frac{1}{\sqrt{Gr}}\nabla^s\mathbf{u}^h\right)_{\Omega^h} - \left(\mathbf{w}^h, \mathbf{e}^g c^h\right)_{\Omega^h} \\
&+\left(\mathbf{u}^h\cdot\nabla\mathbf{w}^h\right)_{\Omega^h} \\
&+\left(\nabla q^h, \tau_M\mathbf{r}_M\right)_{\Omega^h} \\
&+\left(\nabla\mathbf{w}^h, \tau_C\nabla\cdot\mathbf{u}^h\right)_{\Omega^h} \\
&-\left(\nabla\mathbf{w}^h, \tau_M\mathbf{r}_M\otimes\tau_M\mathbf{r}_M\right)_{\Omega^h} = 0
\end{aligned} \tag{9}$$

while for the sediment transport equation we have,

$$\left(w^h, \frac{\partial c^h}{\partial t}\right) + \left(w^h, (\mathbf{u} + u_s \mathbf{e}^g)\cdot\nabla c^h\right)_{\Omega^h} + \left(\nabla w^h, \frac{1}{Sc\sqrt{Sc}}\nabla c^h\right)_{\Omega^h} + \sum_{e=1}^{n_{el}} \left(\tau_{SUPG}\mathbf{u}\cdot\nabla w^h, \left(\frac{\partial c^h}{\partial t} + (\mathbf{u} + u_s \mathbf{e}^g)\cdot\nabla c^h\right)\right)_{\Omega_e} + \sum_{e=1}^{n_{el}} \left(\delta(c^h)\nabla w^h, \nabla c^h\right)_{\Omega_e} = \left(w^h, h^h\right)_{\Gamma}. \quad (10)$$

The RBVMS weak formulation, Eq. (9), is based on splitting the physical variables of the problem into large scales (those explicitly captured by the numerical grid) and fine scales (sub–grid scales), that is,

$$\mathbf{u} = \mathbf{u}^h + \mathbf{u}' \quad (11)$$

$$p = p^h + p' \quad (12)$$

where the superscript h denotes the large scale component of the solution, while the superscript $\prime$ refers to the small scale complement. In Eq. (9) a simple algebraic model for the small scale part of the solution is used to close the formulation, that is,

$$\mathbf{u}' = -\tau_M \mathbf{r}_M \quad (13)$$

$$p' = -\tau_C \mathbf{r}_C \quad (14)$$

with $\mathbf{r}_M$ and $\mathbf{r}_C$ being the discrete residuals of the coarse scale equations, given by

$$\mathbf{r}_M = \frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h\cdot\nabla\mathbf{u}^h + \nabla p - \frac{1}{\sqrt{Gr}}\nabla^2\mathbf{u}^h - \mathbf{e}^g c^h \quad (15)$$

$$\mathbf{r}_C = \nabla\cdot\mathbf{u}^h \quad (16)$$

and τ_M and τ_C are scalar parameters computed as in (Akkerman et al., 2008).

For sediment transport, the stabilized SUPG finite element formulation is employed (Rossa and Coutinho, 2013). The expression for the stabilization parameter τ_{SUPG} is also given in (Rossa and Coutinho, 2013). We also added to the formulation a discontinuity-capturing operator, the last term in Eq. (10), in which the parameter δ depends nonlinearly on the concentration (Guerra et al., 2013).

2.3. libMesh library

LibMesh (Kirk et al., 2006) is an open-source library that provides a complete platform for parallel, adaptive, multiphysics finite element simulations. The library supports adaptive mesh refinement and coarsening (AMR/C) capabilities on general unstructured meshes. AMR/C can be implemented using a variety of error estimators (Ainsworth and Oden, 2000). Additionally, libMesh supports parallel distributed meshes and employs adaptive repartitioning techniques to increase scalability for AMR/C runs. Various continuous and discontinuous finite element families can be used. For a recent review of libMesh, its new variants and other related finite element libraries see Bauman and Stogner (2016).

LibMesh also interfaces with several external solver packages, such as PETSc (Balay et al., 2016) and Trilinos (Heroux et al., 2005). In particular, PETSc includes a large suite of parallel linear and nonlinear equation solvers. Despite providing several tools for mesh-based solvers, applications built upon libMesh need to reimplement many common finite element kernels, including assembly loops and time integration schemes. These kernels are typically related to the physics of the problem of interest and the numerical formulation adopted.

In this work, we introduce libMesh-sedimenation, where we implement the mathematical framework presented in section 2.1. A staggered scheme advances in time the Navier-Stokes and sediment transport equations. Non-linearities are handled by the Inexact-Newton method for both systems. For time integration we adopted a predictor/multi-corrector scheme for Navier-Stokes and the Crank-Nicolson method for the sediment transport equation.

3. Data analysis tools

Data analysis typically involves sophisticated queries and visualization tools. ParaView (Fabian et al., 2011) is an open-source, multi-platform data analysis and visualization application. ParaView Catalyst is an in situ data processing visualization tool. However, enabling Catalyst on a parallel simulation code still introduces some overhead on the overall memory footprint, presents limited query processing power, and is not possible to adjust simulation parameters dynamically (i.e., user steering support).

In this section, we expand Catalyst's analytical power by extracting and inserting relevant simulation data and metadata into a Database Management System (DBMS) with more robust simulation data query support. We use a tool, DfAnalyzer, that aims at relating data from different simulation time steps and registers at runtime execution provenance, without interfering with the overall parallel performance.

3.1. DfAnalyzer: runtime data analysis tool based on dataflow management

DfAnalyzer[1] (Silva et al., 2017) is a generic tool to analyze simulation data. DfAnalyzer is based on four components: extracting selected data from the simulation; relating simulation data and metadata; ingesting data into a DBMS, and processing queries. DfAnalyzer gathers quantities of interest by instrumenting simulation codes, such as solver results (residual norms, number of linear and nonlinear iterations); extracts data that are allocated in memory using Catalyst, such as velocity, pressure and sediment concentration; gathers provenance data, describing the history of data transformations in simulation runs; and relates these simulation data from different files with provenance management support, such as nonlinear solver convergence data (i.e., residual norms), correlating them with results extracted using Catalyst (e.g., sediment concentration).

There are several advantages of this data analysis approach. First, simulation data is preserved in their format and not replicated in the DBMS. Second, data is related among different files while it is being generated, which might be cumbersome after the simulation ends, as in post-processing approaches (Oldfield et al., 2014). Third, the history of data generation is registered for further analysis or reproduction through provenance, following W3C PROV–DM (Missier et al., 2013). Fourth, efficient data management techniques from column-oriented relational DBMS (i.e., MonetDB (Boncz et al., 2008)) can be used at runtime. Consequently, DfAnalyzer provides a provenance database enriched with quantities of interest (i.e., domain data) that can be queried at runtime.

3.2. In situ data extraction and visualization using ParaView catalyst

Catalyst (Ayachit et al., 2015) leverages raw data extraction and visualization capabilities of the common post-processing platforms, VTK (Schroeder et al., 2006) and ParaView (Fabian et al., 2011). Catalyst can take advantage of the ParaView User Interface (UI) to develop Python scripts for data extraction and visualization in VTK data structures. Catalyst also provides APIs that initialize, execute, and finalize co-processing pipelines to enable in situ data processing during a simulation run (Ayachit et al., 2015). Therefore, Catalyst API provides an interface between simulation codes and Python scripts for analyzing raw data in memory. This interface, called adaptor, maps data structures from simulation codes to Catalyst VTK data model and provides an API that the

[1] https://hpcdb.github.io/armful/dfanalyzer.html.

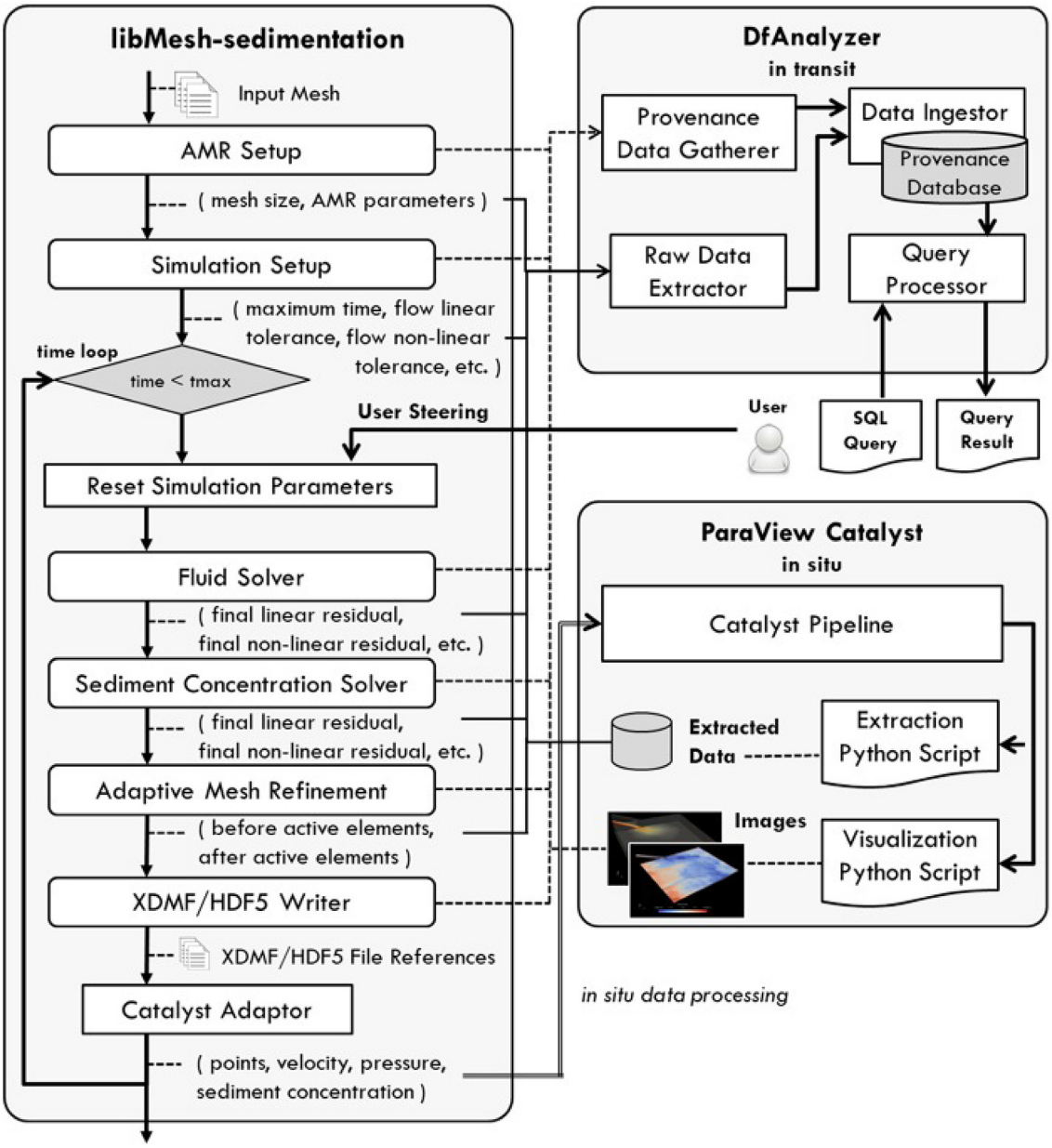

Fig. 1. LibMesh-sedimentation integration with Catalyst and DfAnalyzer.

simulation code uses to invoke Catalyst. The specification of this interface is an important step and may require a significant effort. Alternatively, a generic interface, such as SENSEI (Ayachit et al., 2016), may be used to replace the adaptor.

3.3. *libMesh-sedimentation integration with ParaView Catalyst and DfAnalyzer*

Fig. 1 illustrates the integration among libMesh-sedimentation with Catalyst and DfAnalyzer. For in situ data analysis and visualization, we implemented a Catalyst adaptor with Initialize, CoProcess, and Finalize methods for invoking pipelines developed in Python scripts to extract simulation data and to generate visualizations. Initialize method starts Catalyst in a proper state in a simulation run. This method is invoked once per simulation run and before CoProcess method invocations. It defines analysis pipelines to be run by Catalyst. Each analysis pipeline corresponds to a Python script generated by the ParaView UI application.

CoProcess method is called for each time step that is relevant for data analysis and visualization. Therefore, the adaptor implementation maps simulation data structures to the VTK data model (from Catalyst) and uses them to run registered analysis pipelines (from Initialize method). Then, the Finalize method is responsible for releasing memory allocated during simulation run and cleaning up Catalyst state.

LibMesh uses a complex data structure and reuse of memory would only be possible by modifying the libMesh code. Thus, we have opted for a generic adaptor able to work with any libMesh version. Thus, unlike (Yi et al., 2015), we do not consider reuse of in-memory data in this data structure mapping between libMesh-sedimentation and Catalyst.

DfAnalyzer gathers high granularity provenance data from simulation stages relevant analysis, involving data from different time steps. DfAnalyzer registers simulation data from the nonlinear iterations loops for fluid and sediment solvers. Fig. 2 shows how we instrumented libMesh-sedimentation code using DfAnalyzer. We identify when a data transformation (in general, represented by the invocation of a libMesh component) initializes and finalizes to gather relevant domain data, such as the final linear residual norm for the fluid solver at a given time step.

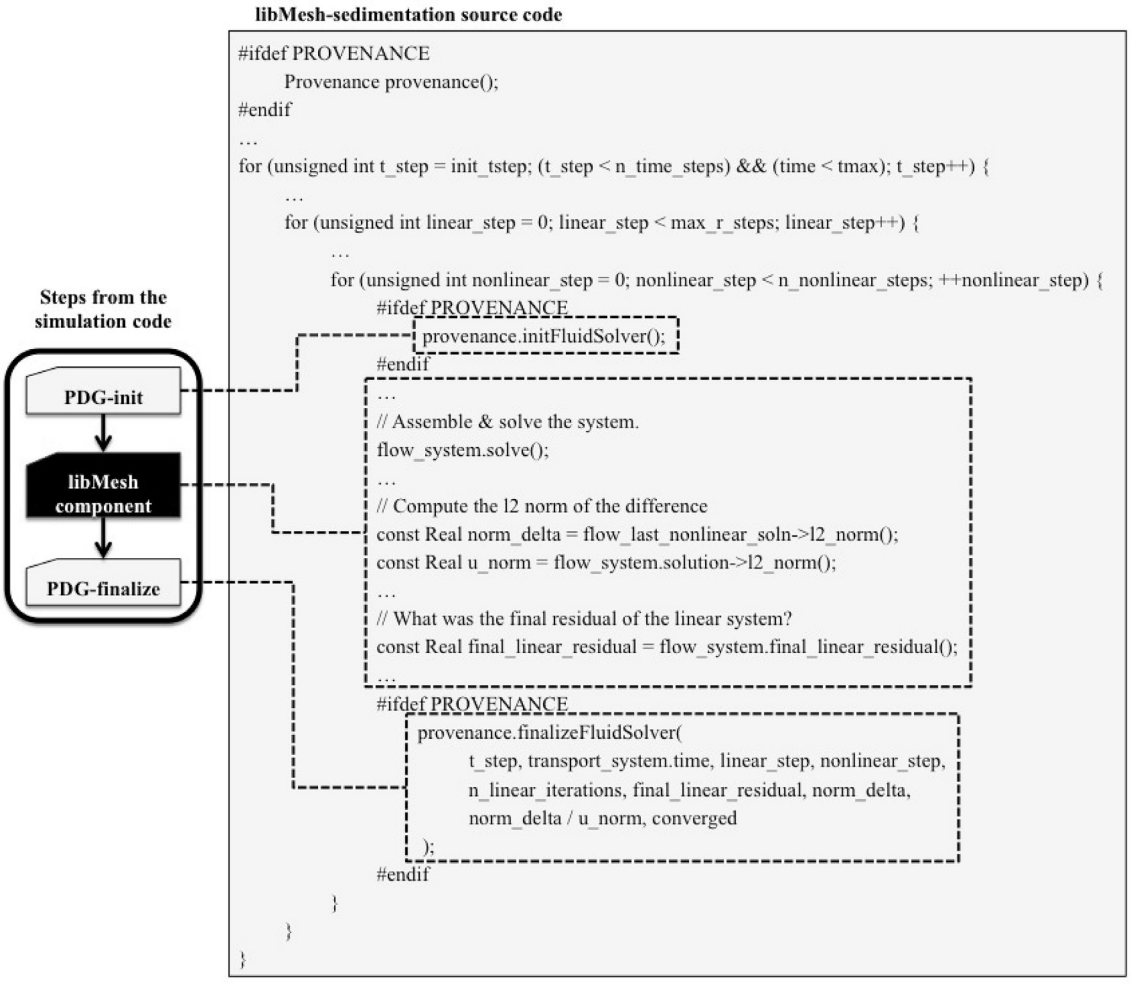

Fig. 2. LibMesh-sedimentation source code with provenance management using DfAnalyzer.

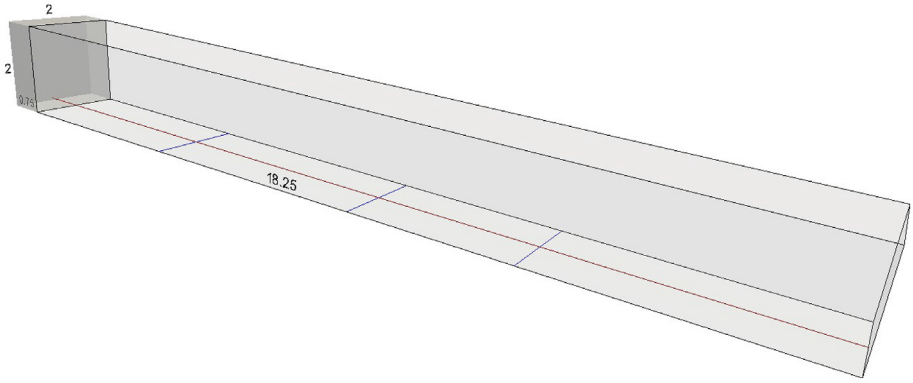

Fig. 3. Lock-exchange configuration – de Rooij and Dalziel sedimentation tank.

To avoid harming the solver performance, we instantiate MonetDB DBMS in a dedicated node, different from the nodes executing libMesh-sedimentation. As data is registered on a different computational node from where it was generated, the integration of libMesh-sedimentation with DfAnalyzer corresponds to an in transit approach.

4. Numerical, performance and data analysis results

In this section, we study two turbidity current scenarios. First, we reproduce the experimental and DNS results provided by Rooij and Dalziel (De Rooij and Dalziel, 2009) and Nasr-Azadami (Nasr-Azadani and Meiburg, 2011). The second test case simulates the sediment deposition carried by a turbidity current over a real bed bathymetry. We tested our approach on Lobo Carneiro, an SGI ICE X. The system has 252 nodes connected by an FDR InfiniBand. Each node has two 12-core Intel Xeon E5-2670v3 processor and 64 GB of memory. The total system memory is 16 terabytes. The high-performance scratch file system is powered by Lustre and has a total capacity of 500 TB.

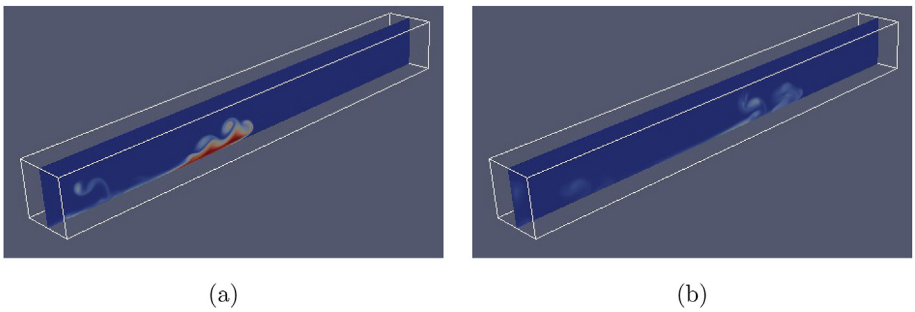

Fig. 4. Sediment concentration profiles at $t = 10$ (a) and $t = 20$ – de Rooij and Dalziel sedimentation tank.

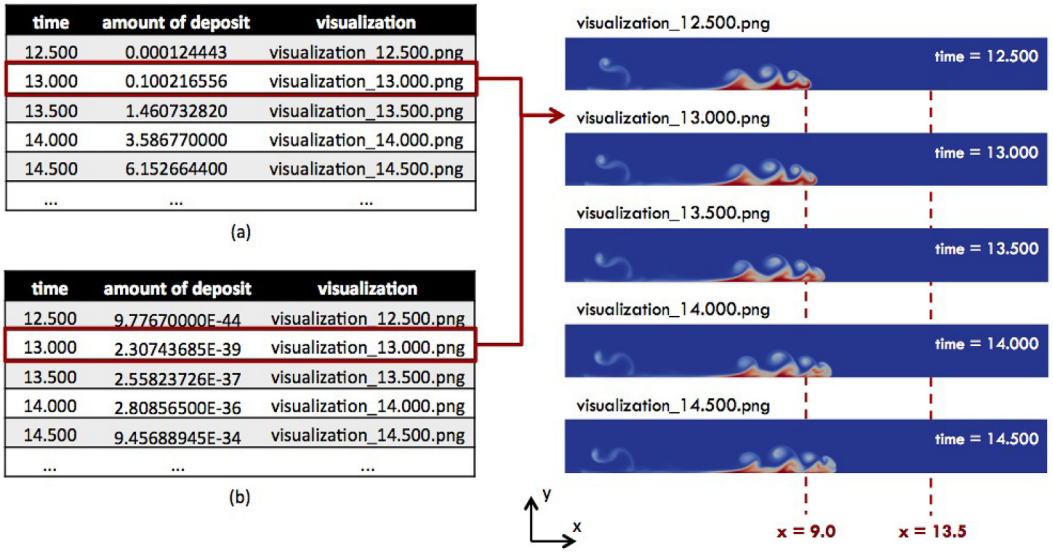

Fig. 5. De Rooij and Dalziel sedimentation tank - sediment deposition monitoring at five time instants at $x = 9.0$ (a) and $x = 13.5$ (b) combining data annotated by DfAnalyzer with Catalyst in-situ visual information.

4.1. De Rooij and Dalziel sedimentation tank

The experiments carried out by de Rooij and Dalziel (De Rooij and Dalziel, 2009) are based on a lock-exchange configuration. Fig. 3 shows the domain model used in this simulation. It is a rectangular box with dimensions $20 \times 2 \times 2$. The lock, in which the fluid initially is at rest, has a height 2 and length 0.75. In this simulation, we take $Gr = 10^6$, $Sc = 1$ and the non-dimensional settling velocity is $u_s = 0.02$. We use a hexahedral structured mesh with a 0.1 grid spacing. Two uniform refinements are applied initially, totalizing 4,608,000 hexahedra. The Kelly's error estimator (Ainsworth and Oden, 2000) is computed for both velocity and sediment concentration, and it is used to drive the adaptive mesh refinement. The Navier-Stokes and sediment concentration equations linear systems are solved in parallel by Block-Jacobi + GMRES (35) with local ILU(0) preconditioning. GMRES tolerance is set to 10^{-6}. For the nonlinear solver, the tolerance is 10^{-3} and the time step size is 0.005. XDMF/HDF5 raw data files are written every 50 time steps.

For the in situ data analysis, we created a pipeline from a Python script using ParaView client's user interface where a sequence of ParaView's filters is applied to generate a view of the sediment concentration profile on the plane $x - z$ taken at $y = 1.0$. We control how often to call the Catalyst adaptor and the raw data writer. We kept the Catalyst calling frequency identical to the raw data writer. Fig. 4 (a) and (b) shows sediment concentration snapshots at $t = 10$ and $t = 20$ generated by Catalyst.

The Catalyst adaptor is also used to extract derived information from original data, improving the data analysis. For geologists, tracking the sediment deposition along time is one of the most relevant information. A typical analysis is to verify if sediments reach a given region of interest and to map the deposit profile. To do that, we defined four monitoring lines at the bottom of the domain. These lines are discretized in 100

```
while(t < t_max) {
    (...)
    if (is_file_exist("reset.run")) {
        printf("RESETTING INPUTS");
        GetPot reset(input);
        dt        = reset("time/deltat", dt);
        tmax      = reset("time/tmax", tmax);
        (...)
        nonlinear_tolerance = reset("nonlinear_tolerance", nonlinear_tolerance);
        max_linear_iter     = reset("max_linear_iterations", max_linear_iter);
        linear_solver_tol   = reset("linear_solver_tolerance", linear_solver_tol);
        (...)
        ref_interval  = reset("amr/r_interval", ref_interval);
        r_fraction    = reset("amr/r_fraction", r_fraction);
        c_fraction    = reset("amr/c_fraction", c_fraction);
        max_h_level   = reset("amr/max_h_level", max_h_level);
        (...)
    }
    (...)
}
```

Fig. 6. LibMesh-sedimentation source code for supporting user steering – parameters amenable to be changed in runtime: time step (Δt), maximum simulation time (t_{max}), nonlinear and linear solver tolerances, AMR/C error fractions (r_{frac}, c_{frac}), and others not shown.

equally spaced points where data is extracted. Three of these lines are parallel and defined perpendicular to the x-axis (blue lines in Fig. 3). They are used to map the progress of the sediment current. The fourth line cuts the domain transversally and maps the deposition profile (red line in Fig. 3). It is defined at $y = 0$ and extends from $x = 0$ to $x = 20$.

DfAnalyzer registers deposition along time at predefined locations and the corresponding pointers to the Catalyst visualization files. Therefore, we can monitor the sediments on these lines through online queries with a negligible elapsed time (<500 milliseconds). Fig. 5 illustrates the sediment deposits at five different times for $x = 9.0$ and $x = 13.5$. As we can observe, the sediments have already reached one region of interest at $x = 9.0$ but not at $x = 13.5$. With this combined information, we may decide to stop the simulation (deposition reached the area of interest) or whether it is necessary to continue execution (deposition is still away from the area of interest). If the decision is to continue, we need to reset the maximum time interval. Fig. 6 shows a piece of libMesh-sedimentation code where we can see how simulation parameters can be changed at runtime by reading a reset file. LibMesh-sedimentation checks at the beginning of each time step whether a reset file exists. If true, the input parameters file is read superseding previous values.

Without DfAnalyzer, we would need to browse the generated files to extract and analyze raw data. Moreover, some relevant data are stored in log files. Therefore, we would have to gather those data from log files and develop programs to relate data captured from the log and raw data files. Different from this approach, our integration of libMesh-sedimentation with Catalyst and DfAnalyzer enables online query processing considering data obtained from log and raw data files, taking advantage of in situ visualization and in transit data analysis without compromising performance, as shown in Table 1.

Fig. 7 compares the deposition profile extracted from our solution with experimental and DNS results provided respectively by de Rooij and Dalziel (De Rooij and Dalziel, 2009) and Nasr-Azadami and Meiburg (Nasr-Azadani and Meiburg, 2011). Considering the limitations imposed by experimental conditions pointed out in (Necker et al., 2005), our result shows a good agreement with both experiment and DNS solutions.

Table 1 shows the elapsed time spent in the different stages of libMesh-sedimentation. In this experiment, we ran the simulation on 480 MPI processes. CPU time spent by the data analysis tools is only 6.41% of the total time, where 6.33% accounts for in situ visualization and data extraction, and the remaining 0.08% is from DfAnalyzer. Code instrumentation has shown that CPU time spent in Catalyst depends on how many filters are applied. Therefore, data extraction spent more CPU time than visualization. The most expensive stage is the nonlinear solution of fluid equations, taking almost 33% of the total time, followed by mesh refinement and coarsening (AMR/C) with 20.70%. XDMF/HFD5 raw data writer took 0.92% of the total time.

Although in situ visualization and extraction have higher CPU times than the XDMF/HDF5 raw data writer, this could be offset by disk bandwidth constraints as well as limited disk capacity. Table 2 shows the storage usage for the PNG images, data extracted, provenance data and XDMF/HDF5 raw data files. As we can see, raw data files consume 10,000 MB of disk space. The space used to store the provenance database takes only 2.96% of XDMF/HDF5's storage space. Visualization files use 2.48% of this space and data extracted files consume only 0.02% of XDMF/HDF5 raw data files.

4.2. Real Bathymetry Tank

In this second experiment, all computations were performed taking into consideration the physical scenario described in Fig. 8. The color map in Fig. 8 represents a real bathymetry where deeper regions are colored in dark blue. In this scenario, a diluted mixture of sediments is continuously injected into a channel that releases sediments into the tank at an angle of 45°. Note that the channel is located near a deeper region of the domain. This computational configuration is divided in two main

Table 1
Elapsed time for different stages on libMesh-Sedimentation – de Rooij and Dalziel sedimentation tank.

Time Contribution	CPU Time (in secs)	Cost/ Call	%Cost
Flow Solver	16,203.71	0.87	32.67%
AMR/C	10,268.32	17.11	20.70%
Sediment Solver	2797.36	0.15	5.64%
XDMF/HDF5 Writer	453.96	4.93	0.92%
In Situ Visualization + Data Extraction	3137.24	33.73	6.33%
Provenance (DfAnalyzer)	38.47	0.01	0.08%
Others (libMesh)	16,598.00	–	33.67%
Total	62,171.00		

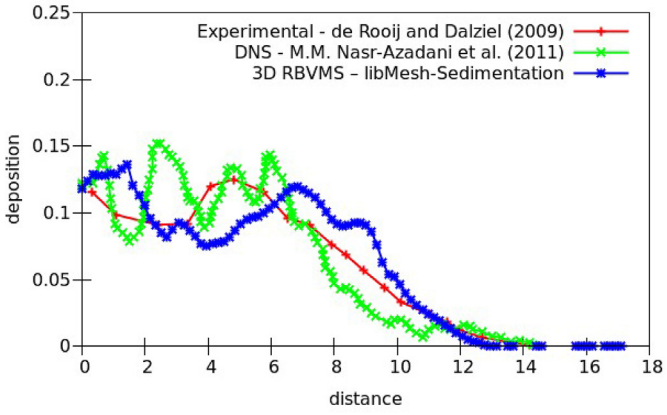

Fig. 7. In Situ Catalyst data extraction - deposition plotted over line filter – de Rooij and Dalziel sedimentation tank.

Table 2
Storage requirements - de Rooij and Dalziel sedimentation tank.

	Storage (MB)	files
In Situ Visualization	248.00	62
Extracted data	2.48	248
Provenance	296.01	75,696
XDMF/HDF5 Raw Data	10,000.00	44,254

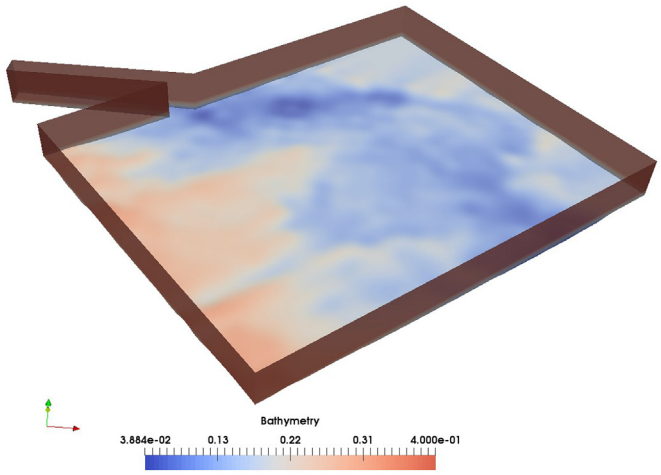

Fig. 8. Real bathymetry tank: physical scenario.

areas: the channel with length $L_c = 2.5$, width $W_c = 0.2$ and height $H_c = 1.5$, and the tank with length $L_T = 14$, width $W_c = 12$ and height $H_c = 1.5$. Spatial discretization employs a fixed unstructured mesh with 7,674,812 tetrahedra and 1,418,934 nodes.

A monodisperse mixture is considered with dimensionless settling velocity $u_s = 5.6651 \times 10^{-03}$, $Gr = 10^6$ and $Sc = 1.0$. The time step is fixed as $t = 5 \times 10^{-3}$ and the analysis is performed over a total of 100 units of dimensionless time. Nonlinear and linear solver parameters are the same of the previous experiment. Non-penetration and non-slip velocity boundary conditions are set on the channel walls and the tank wall in contact with the channel. In the other walls, free slip conditions are applied. Sediment loss is allowed only through the tank bottom. The sediments are injected into the channel with a velocity magnitude of 0.1.

In this test, we have setup the Catalyst script to provide visual information of the deposition map during the simulation run. Catalyst is called with a frequency five times greater than the raw data writer. Fig. 9(a) shows the final snapshot taken at $t = 100$. Fig. 9(b) illustrates a post-processing visualization generated by ParaView. As in the previous test, the Catalyst adaptor was used to extract four lines in four regions of interest.

In this experiment, DfAnalyzer is explored also as a tool to assist the simulation robustness. The main idea is to help in the detection of possible misbehavior of nonlinear and linear solvers. Fig. 10 displays a time window showing the nonlinear residual norms in five consecutive instants. By querying DfAnalyzer, we can detect possible convergence issues and decide to steer the simulation by resetting some solver parameters or even recovering the latest correct checkpoint solution without interrupting the simulation run. Again, steering is done for these parameters as shown in Fig. 6.

Table 3 presents the performance results for the different simulation stages on 480 cores. Once more analysis tools added a small overhead to overall simulation costs (< 2% of total time). Storage requirements are smaller than those of the XDMF/HDF5 raw data files, as reported by Table 4. Analysis data uses less than 3% of the space needed to store the XDMF/FDF5 files. Moreover, we can increase the write interval of these files and possibly reducing the data stored in persistent file systems.

Table 3
Elapsed time (in secs) for different stages on libMesh-Sedimentation – Real bathymetry tank.

Time Contribution	CPU Time (in secs)	%Cost
Flow Solver	75,523.49	50.71%
Sediment Solver	28,000.50	19.58%
XDMF/HDF5 Writer	421.23	0.29%
In Situ Visualization + Data Extraction	2175.16	1.52%
Provenance (DfAnalyzer)	451.70	0.32%
Total	143,029.00	100%

Table 4
Storage requirements - Real bathymetry tank.

	Storage (GB)	% Raw Data
XDMF/HDF5 Raw Data	23.44	–
Provenance	0.38	1.60
In Situ Visualization	0.28	1.21

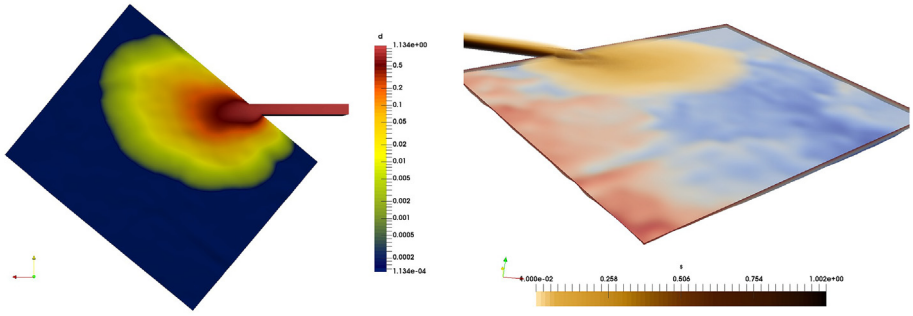

Fig. 9. Real Bathymetry Tank: (a) deposition map generated by Catalyst at $t = 100$, (b) post visualization of sediment concentration at $t = 100$ using ParaView filters.

time	nonlinear iteration	nonlinear residual norm	converged
40.000	0	0.046785018	FALSE
40.000	1	0.001376149	FALSE
40.000	2	0.000115880	TRUE
40.005	0	0.046767830	FALSE
40.005	1	0.001375507	FALSE
40.005	2	0.000115810	TRUE
40.010	0	0.046750684	FALSE
40.010	1	0.001374865	FALSE
40.010	2	0.000115741	TRUE
40.015	0	0.046733580	FALSE
40.015	1	0.001374222	FALSE
40.015	2	0.000115672	TRUE
40.020	0	0.046716517	FALSE
40.020	1	0.001373580	FALSE
40.020	2	0.000115602	TRUE

time	nonlinear iteration	linear iterations	final linear residual norm
40.000	0	11	3.8135E-05
40.000	1	100	1.1200E-07
40.000	2	100	1.3000E-08
40.005	0	11	3.8118E-05
40.005	1	100	1.1200E-07
40.005	2	100	1.3000E-08
40.010	0	11	3.8102E-05
40.010	1	100	1.1200E-07
40.010	2	100	1.3000E-08
40.015	0	11	3.8086E-05
40.015	1	100	1.1100E-07
40.015	2	100	1.3000E-08
40.020	0	11	3.8069E-05
40.020	1	100	1.1100E-07

Fig. 10. Numerical analysis to detect possible misbehavior of nonlinear and linear solvers.

5. Conclusions

In-situ visualization tools when coupled to simulations solvers reduce storage space and improve data analytics while the simulation is running. In this work, we coupled Catalyst to libMesh-sedimentation, a solver based on the libMesh library, that provides support for parallel adaptive mesh refinement and coarsening. However, despite reducing the number of files, steering the simulation was still limited to the difficulties in finding the adequate visualization files and relating them to the main quantities of interest registered in XDMF/HDF5 files. Our approach is to couple DfAnalyzer, a runtime data steering tool, to the libMesh-sedimentation/Catalyst solver. DfAnalyzer registers quantities of interest related to in-situ visualization files and their time histories. We demonstrated performance efficiency combined with sophisticated simulation steering at runtime running two turbidity current simulations. The results on a 480 core analysis evidenced high performance with negligible time for both in-situ visualization and data analytics.

Acknowledgements

The research has received funding from CNPq, FAPERJ and Inria (SciDISC projects), the European Commission (HPC4E H2020 project) and the Brazilian Ministry of Science, Technology, Innovation and Communications through Rede Nacional de Pesquisa (RNP) grant agreement n° 689772. It has been performed (for P. Valduriez) in the context of the Computational Biology Institute. Computer time on Lobo Carneiro was provided by the HPC Center at COPPE/Federal University of Rio de Janeiro.

References

Ainsworth, M., Oden, J.T., 2000. A Posteriori Error Estimation in Finite Element Analysis. Wiley.

Akkerman, I., Bazilevs, Y., Calo, V.M., Hughes, T.J.R., Hulshoff, S., 2008. The role of continuity in residual-based variational multiscale modeling of turbulence. Comput. Mech. 41 (3), 371–378. https://doi.org/10.1007/s00466-007-0193-7.

Ayachit, U., Bauer, A., Geveci, B., O'Leary, P., Moreland, K., Fabian, N., Mauldin, J., 2015. Paraview catalyst: enabling in situ data analysis and visualization. In: Proceedings of the First Workshop on in Situ Infrastructures for Enabling Extreme-scale Analysis and Visualization, ISAV2015. ACM, New York, NY, USA, pp. 25–29. https://doi.org/10.1145/2828612.2828624. http://doi.acm.org/10.1145/2828612.2828624.

Ayachit, U., Whitlock, B., Wolf, M., Loring, B., Geveci, B., Lonie, D., Bethel, E.W., 2016. The sensei generic in situ interface. In: 2016 Second Workshop on in Situ Infrastructures for Enabling Extreme-scale Analysis and Visualization (ISAV), pp. 40–44. https://doi.org/10.1109/ISAV.2016.013.

Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Dalcin, L., Eijkhout, V., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Rupp, K., Smith, B.F., Zampini, S., Zhang, H., Zhang, H., 2016. PETSc Users Manual, Tech. Rep. ANL-95/11-revision 3.7. Argonne National Laboratory. http://www.mcs.anl.gov/petsc.

Bauer, A.C., Abbasi, H., Ahrens, J., Childs, H., Geveci, B., Klasky, S., Moreland, K., O'Leary, P., Vishwanath, V., Whitlock, B., Bethel, E.W., 2016. In situ methods, infrastructures, and applications on high performance computing platforms. Comput. Graph. Forum 35 (3), 577–597. https://doi.org/10.1111/cgf.12930.

Bauman, P.T., Stogner, R.H., 2016. GRINS: a multiphysics framework based on the libmesh finite element library. SIAM J. Sci. Comput. 38 (5), 78–100.

Boncz, P.A., Kersten, M.L., Manegold, S., 2008. Breaking the memory wall in monetdb. Commun. ACM 51 (12), 77–85. https://doi.org/10.1145/1409360.1409380.

Burstedde, C., Ghattas, O., Gurnis, M., Isaac, T., Stadler, G., Warburton, T., Wilcox, L., 2010. Extreme-scale amr. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC '10. IEEE Computer Society, Washington, DC, USA, pp. 1–12. https://doi.org/10.1109/SC.2010.25.

Camata, J., Elias, R., Coutinho, A., 2012. FEM simulation of coupled flow and bed morphodynamic interactions due to sediment transport phenomena. J. Comput. Sci. Technol. 7 (2), 3–4. https://doi.org/10.1299/jcst.7.306.

De Rooij, F., Dalziel, S.B., 2009. Time- and Space-resolved Measurements of Deposition under Turbidity Currents. Blackwell Publishing Ltd, pp. 207–215. https://doi.org/10.1002/9781444304275.ch15.

Fabian, N., Moreland, K., Thompson, D., Bauer, A.C., Marion, P., Gevecik, B., Rasquin, M., Jansen, K.E., 2011. The paraview coprocessing library: a scalable, general purpose in situ visualization library. In: IEEE Symposium on Large Data Analysis and Visualization, pp. 89–96. https://doi.org/10.1109/LDAV.2011.6092322.

Guerra, G.M., Zio, S., Camata, J.J., Rochinha, F.A., Elias, R.N., Paraizo, P.L., Coutinho, A.L., 2013. Numerical simulation of particle-laden flows by the residual-based variational multiscale method. Int. J. Numer. Methods Fluids 73 (8), 729–749. https://doi.org/10.1002/fld.3820.

Guerra, G.M., Zio, S., Camata, J.J., Dias, J., Elias, R.N., Mattoso, M., Paraizo, P.L.B., Coutinho, A.L.G.A., Rochinha, F.A., 2016. Uncertainty quantification in numerical simulation of particle-laden flows. Comput. Geosci. 20 (1), 265–281. https://doi.org/10.1007/s10596-016-9563-6.

Heroux, M.A., Phipps, E.T., Salinger, A.G., Thornquist, H.K., Tuminaro, R.S., Willenbring, J.M., Williams, A., Stanley, K.S., Bartlett, R.A., Howle, V.E., Hoekstra, R.J., Hu, J.J., Kolda, T.G., Lehoucq, R.B., Long, K.R., Pawlowski, R.P., 2005. An overview of the Trilinos project. ACM Trans. Math. Softw. 31 (3), 397–423. https://doi.org/10.1145/1089014.1089021.

Kirk, B.S., Peterson, J.W., Stogner, R.H., Carey, G.F., 2006. libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations. Eng. Comput. 22 (3–4), 237–254.

London, I.C., others, 2017. Fluidity: an Open-source Computational Fluid Dynamics Code with Adaptive Unstructured Mesh Capabilities. http://http://fluidityproject.github.io/.

Meiburg, E., Kneller, B., 2010. Turbidity currents and their deposits. Annu. Rev. Fluid Mech. 42 (1), 135–156. https://doi.org/10.1146/annurev-fluid-121108-145618.

Meiburg, E., Radhakrishnan, S., Nasr-Azadani, M., Gravity, Modeling, Currents, Turbidity, 2015. Computational approaches and challenges. Appl. Mech. Rev. 67 (4), 040802 https://doi.org/10.1115/1.4031040.

Missier, P., Belhajjame, K., Cheney, J., 2013. The w3c prov family of specifications for modelling provenance metadata. In: 16th International Conference on Extending Database Technology, EDBT'13. ACM, New York, NY, USA, pp. 773–776. https://doi.org/10.1145/2452376.2452478.

Nasr-Azadani, M.M., Meiburg, E., 2011. TURBINS: an immersed boundary, Navier–Stokes code for the simulation of gravity and turbidity currents interacting with complex topographies. Comput. Fluids 45 (1), 14–28. https://doi.org/10.1016/j.compfluid.2010.11.023.

Necker, F., Härtel, C., Kleiser, L., Meiburg, E., 2002. High-resolution simulations of particle-driven gravity currents. Int. J. Multiph. Flow 28 (2), 279–300. https://doi.org/10.1016/S0301-9322(01)00065-9.

Necker, F., Härtel, C., Kleiser, L., Meiburg, E., 2005. Mixing and dissipation in particle-driven gravity currents. J. Fluid Mech. 545, 339–372.

Oldfield, R.A., Moreland, K., Fabian, N., Rogers, D., 2014. Evaluation of methods to integrate analysis into a large-scale shock physics code. In: Proceedings of the 28th ACM International Conference on Supercomputing, ICS '14. ACM, New York, NY, USA, pp. 83–92. https://doi.org/10.1145/2597652.2597668.

Popinet, S., 2017. Gerris Flow Solver. http://gfs.sourceforge.net.

Rossa, A.L., Coutinho, A.L., 2013. Parallel adaptive simulation of gravity currents on the lock-exchange problem. Comput. Fluids 88, 782–794. https://doi.org/10.1016/j.compfluid.2013.06.008.

Schroeder, W., Martin, K., Lorensen, B., 2006. Visualization Toolkit: an Object-oriented Approach to 3D Graphics, fourth ed. Kitware.

Silva, V., Leite, J., Camata, J.J., de Oliveira, D., Coutinho, A.L., Valduriez, P., Mattoso, M., 2017. Raw data queries during data-intensive parallel workflow execution. Future Gener. Comput. Syst. https://doi.org/10.1016/j.future.2017.01.016.

Yi, H., Rasquin, M., Fang, J., Bolotnov, I.A., 2015. In-situ visualization and computational steering for large-scale simulation of turbulent flows in complex geometries. In: Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014, pp. 567–572. https://doi.org/10.1109/BigData.2014.7004275.