ACROSS

HPC Big DAta ArtifiCial intelligence cross Stack PlatfoRm TOwardS ExaScale

D7.1 – Stage 1 requirements for HW/SW integration

Deliverable ID	D7.1
Deliverable Title	Stage 1 requirements for HW/SW integration
Work Package	WP7
Dissemination Level	PUBLIC
Version	2.0
Date	2021 – 08 - 31
Status	Submitted
Deliverable Leader	IT4I
Main Contributors	SINTEF, INRIA, ATOS, LINKS

The ACROSS project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955648. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Italy, France, Czech Republic, United Kingdom, Greece, Netherlands, Germany, Norway.

Published by the ACROSS Consortium

Document History

Version	Date	Author(s)	Description
0.1	2021-05-04	IT4I, SINTEF	First version, ToC
0.2	2021-05-17	IT4I	Input first info based on questionnaire
0.3	2021-06-04	SINTEF, INRIA, IT4I	Definition of work, added minor explanation
0.4	2021-06-14	LINKS	New ToC proposed by LINKS
0.5	2021-06-21	SINTEF	Integration related content
0.6	2021-06-24	IT4I	Added description of statistical modeling and expected KPIs
0.7	2021-06-25	SINTEF	Added brief introduction and scope sections, some work on 4.2.
0.8	2021-06-30	IT4I	Adding executive summary
1.1	2021-07-22	IT4I	Added section Software and Conclusion
1.2	2021-07-26	INRIA	Reworded Section 4.1.1 and added details to Damaris row of Section 5 Software Table
1.3	2021-08-10	SINTEF	Overall check of the content and small changes
1.4	2021-08-12	IT4I	Conclusion text added
1.5	2021-08-23	LINKS	Added content into 4.3
1.6	2021-08-25	LINKS, SINTEF	Removal of use case presentation in chapter 2, edited sections 4.2.1, and 4.3
1.7	2021-08-27	SINTEF, LINKS, IT4I	Final overall check
1.8	2021-08-30	LINKS	List of tables/figures and final formatting
2.0	2021-08-31	LINKS	Submitted version

Table of Contents

Glossary

Acronym	Explanation
HM	History Matching
HPC	High-Performance Computing
CPU	Central Processing Unit
GPU	Graphical Processing Unit
FPGA	Field Programmable Gate Array
MPI	Message Parsing Interface
CPR	Constrained Pressure Residual
AMG	Algebraic Multigrid Preconditioners
SA	Sensitivity Analysis
UA	Uncertainty Analysis
AD	Automatic Differentiation
KPI	Key Performance Indicator
HW	Hardware
SW	Software

List of figures and tables

Executive Summary

An important part of each technological project is to define the requirements of individual components and plan the individual steps for their integration in the final solution. This deliverable is the first of two deliverables aimed at the definition of the integration requirements in the WP7 pilot of the ACROSS project. It consists of two main parts, one is the definition of the statistical analysis results and the second one refers to the HW and SW requirements for the integration of individual components used in the WP7 pilot.

Position of the deliverable in the whole project context

This deliverable should be a roadmap for the future actions in the WP7 and it briefly describes the requirements and the action we expect to take in the following months regarding the integration of different technologies provided by the WP7 partners. It is closely related to the D2.1 which focuses on the requirements with regards to co-design with other WPs and overall ACROSS solution. This deliverable relates to the achievement of the MS1 "Awareness of project objectives and requirements".

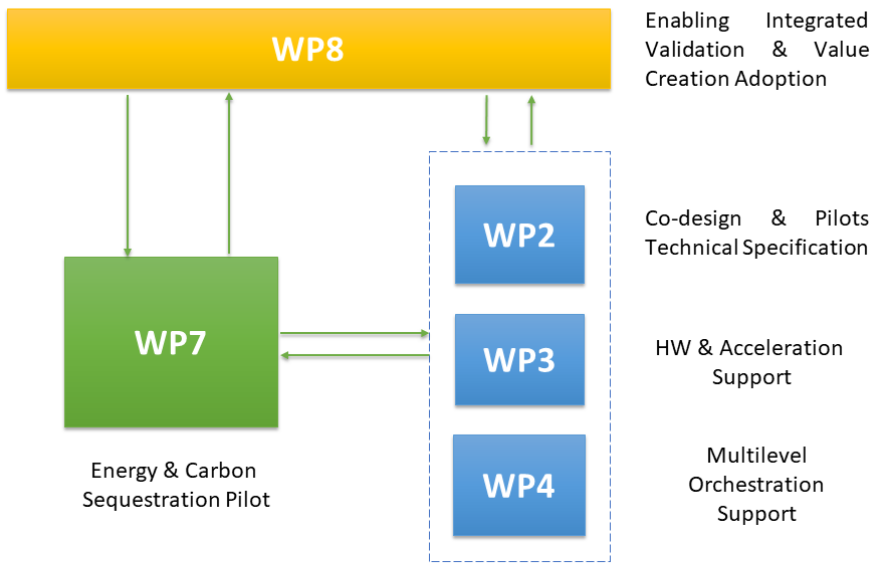
WP8
Enabling Integrated Validation & Value Creation Adoption
WP2
Co-design & Pilots Technical Specification
WP7
WP3
HW & Acceleration Support
WP4
Multilevel Orchestration Support
Energy & Carbon Sequestration Pilot

Figure 1 - Position of the WP7 in the context of the ACROSS project.

Description of the deliverable

The deliverable is composed of two main parts. The first one describes our results of the first discussions about the statistical analysis that could be made on the ensemble models of the reservoir simulations. The second part is describing the requirements and actions needed for the successful integration of the software and hardware components of the WP7. The main discussed topics are OPM Flow and Damaris and how hardware accelerators, such as GPUs and FPGAs, could impact reservoir simulation performance, and preliminary requirements for the orchestration integration.

1 Introduction

This is the first of two deliverables with the same scope. This deliverable is for the first stage of the project, and deliverable D7.4 will be an updated version with revised information in the second stage. In this case, it is the overview of the preliminary requirements based on the discussion and integration attempts made in the first months of the ACROSS project.

1.1 Scope

This document describes the requirements for hardware and software integration work necessary for the WP7 pilot use cases. It concentrates on the specific tasks that should be done in WP7, requirements that must be satisfied by other WPs, such as memory or CPU requirements, are instead given in D2.1.

1.2 Related documents

ID	Title	Reference	Version	Date
D2.1	WP2 - Summary of Pilots co-design requirements		3.0	2021-08-31

2 Pilot integration overview

The WP7 pilot has two use cases, both using the reservoir simulator program OPM Flow: seismic cube and carbon sequestration use cases. The pilot and its use cases are described in D2.1, see section 2.3 and section 5 of that document for details.

The main integration efforts in the first part of the project will be:

- Integrating Damaris into the OPM Flow program, to improve capabilities for I/O, scalability, and in-situ processing and visualization.
- Integrating the current workflow, as well as the future workflows to be created in WP7 Task 7.3, with the ACROSS orchestration and resource broker facilities. This requires changes and improvements to be made to the ERT program.
- Improving the support of accelerators in OPM Flow, to take advantage of current and upcoming trends in computational hardware.

These efforts are described in detail in section 4. Section 3 describes how the efforts will be evaluated.

3 Applied Methodology definition

3.1 Statistical analysis of integration results

Uncertainty of the model response, shortly “uncertainty analysis” (UA) aims to quantify the variability of the model output due to the variability of the model inputs D7.1 [1]. The quantification is usually performed by the summary of the model output data by exploratory data analysis.
The uncertainty analysis can be summarized by the following steps:

1. Define uncertainty of input variables (for example, by setting the minimum and maximum value of the input parameter) and the probabilistic model for sampling, for example, the uniform distribution U(0,1), or lognormal distribution.
2. Generate a random sample of the model inputs.
3. Evaluate the model at each of the random input points, each represents a different combination of input parameters.
4. The output of the uncertainty analysis is the unconditional distribution p(y), which represents the overall uncertainty of the model output.

In the ACROSS project, we will assume the OPM Flow reservoir simulator as the model for uncertainty analysis. At the moment of writing, the most sensible approach seems to be using various distributions for properties like permeability to create model ensembles that represent the present uncertainty. It is clear that Step 3 of the uncertainty analysis represents the most time-consuming part of UA, as the model should be evaluated many times. This may pose a problem for the planned large-scale execution as they will require an enormous amount of computational resources. We will look into possibilities to quantify the uncertainty of the results based on the smaller number of executed model variations.

3.2 Sensitivity analysis

While uncertainty analysis aims to quantify uncertainty in the model output, sensitivity analysis (SA) investigates the dependency of the model output from various sources of uncertainty in the model inputs [2]. According to Becker [3], sensitivity analysis is the concept of finding how sensitive the model output is to each input or set of inputs. Becker claims that often the majority of output uncertainty is caused by only a small set of input parameters: so it is of interest to focus efforts to reduce input uncertainty on these parameters.
The typical approach to the uncertainty and sensitivity analysis in reservoir modelling is history matching. A simple description of history matching would be the selection of the model, or ensemble of models, most likely to represent the real reservoir. For the history matching method, a great number of similar models will be evaluated. In the ACROSS project, we will leverage the results of these models and look for alternative methods of uncertainty and sensitivity analysis that could leverage the computed data, consequently increasing the reliability of the models. At WP7 telcos we discussed the possibility of testing a probabilistic approach to the uncertainty quantification and chaos expansion for the sensitivity analysis.
Sensitivities of some output with respect to the inputs can be approximated using an ensemble of cases, or calculated from the partial derivatives (adjoints), if available. For most complex simulation codes, such derivatives are not available. OPM Flow is written using automatic differentiation (AD) which makes it feasible to implement calculation of adjoints in the simulator.

3.3 KPIs definition of integration results

Part of the T7.1 is the definition of the KPIs of the integration results. As we are at beginning of the project and still facing a lot of uncertainties about the integration process, it is difficult to define concrete KPIs.
Despite this fact, we have identified possible candidates for the KPIs related to the integration as the following:

- Ability to perform in-situ analysis on simulation output data.
- Scaling to 1000 processes with reasonable efficiency.
- Ability to successfully run history matching with no human intervention.

As the integration plan will progress in the coming months, these KPIs will be assessed and confirmed in the deliverable D7.4, or more suitable ones will be proposed.

4 Integration requirements for the use cases

4.1 Software and framework integration requirements

4.1.1 Damaris integration

The OPM Flow program is currently limited in its scalability due to all output operations being performed on a single process. After each time step, the necessary data are collected on a single process, and then written to disk in a serial output format. This must be improved to extend the scalability of OPM Flow towards higher process numbers and larger simulation cases.
The serial output format and processing also limit the ability to create scalable workflows, combining OPM Flow for reservoir simulation with other tools for analysis, visualization or other processing. The ability to do efficient, parallel and scalable processing of simulation output will enable new workflows to be created, that extract more insight and knowledge from simulation runs. Doing this in-situ will reduce the turnaround time, as the analyst or engineer will not have to wait for a long simulation to finish before starting to investigate or analyse the results.
We have chosen to integrate OPM Flow with the Damaris middleware [4], [5] to provide these capabilities. The following were the main drivers in that decision:

- Damaris is tested and mature enough to be considered a low-risk integration. It has proven capabilities in terms of scalability and has been integrated into multiple other software packages over time.
- The license of Damaris is compatible with that of OPM Flow.
- Damaris provides both parallel I/O and in-situ visualization processing capabilities, eliminating the need to integrate two different systems or packages to get both.
- Damaris is written in standard C++, as is OPM Flow, and the interfaces already created for Damaris are already appropriate for use with OPM Flow (other than additions outlined below).

The following modifications to Damaris are required for its integration with OPM Flow, the ACROSS platform and other software:

1. It must become possible to control Damaris programmatically, rather than only through an XML file. Damaris is designed to accept entries in the XML file (as '*parameter*' types) and then the host software can programmatically modify the entries. The parameters are then used as variables in other defined elements of the XML file, such as *layouts* and *variables*. Changing a parameter value within the host program (programmatically), will subsequently update the sizes in the layouts and variables that use

the parameters in their definitions. This permits a lot of flexibility for defining the sizes and shapes of arrays. The need to dynamically add or remove a specific field variable (e.g. pressure, velocity, etc.) should not be required as the XML file can specify all fields and then only ones that need to be written (i.e. as determined by the Eclipse input stack) can be written. Damaris is designed to take the I/O pressure off the main computation, this pressure is caused by large arrays that are typically present in a simulation. Small I/O tasks can be handled by the original I/O methods implemented in OPM Flow. The coordination of variables available to Damaris and the OPM Flow software must be carried out so that names and used types match. The lack of type safety is a specific limiting factor in the use of the XML file, and an automated method for checking the validity of these inputs will be investigated. Programmatic generation of the Damaris inputs may be a valid option to remove this issue.

2. It should become possible to build Damaris with as few dependencies as possible, by making optional any dependencies that are only required for features that will not be used in ACROSS. Currently, the hard runtime dependencies of Damaris are MPI, XSD, xerces-c and Boost (libraries: thread, log, date_time, program_options, filesystem, system). All other library dependencies are optional (HDF5, ParaView, VisIt, cppunit) and configurable through the Damaris CMake build system.
3. It should be possible for Damaris to select a specific subset of ranks within a node to be used as dedicated cores instead of relying on external tools (e.g. omplace) to select and pin its processes to cores. It should be possible for Damaris to select a specific node/set of nodes to use in 'dedicated node' mode. This capability should be aligned with the ACROSS WP4 tasks of defining workflows.

4.1.2 OPM Flow integration

The following modifications to OPM Flow are required for its integration with the ACROSS platform and other software:

1. OPM Flow must become able to run in an MPI setting without assuming that it is running with the MPI_COMM_WORLD (global) communicator, rather it must be able to run on a subset of the total MPI ranks used for the process, by using a variable communicator.
2. OPM Flow must add Damaris as an optional build dependency, along with any Damaris dependencies that cannot be avoided.
3. When compiled with Damaris support, OPM Flow must modify its startup procedures to accept a string pointing to the Damaris XML configuration file. The configuration file will allow a user to optionally use Damaris in dedicated core(s) or dedicated node modes (or neither) of asynchronous I/O and processing. This will be changed to instead control Damaris programmatically when that feature becomes available in Damaris.
4. OPM Flow must add HDF5 as an optional build dependency.
5. A preliminary parallel output format must be agreed on. The output format that is most widely used within the industry is the Eclipse binary output format, which is serial in nature and not well suited for large-scale parallel I/O. Work is underway with stakeholders to define a new output format. Long-term a permanent solution must be agreed on, which also can get buy-in from commercial vendors. RESQML is one possibility that should be explored.
6. Improve support for pausing and resuming simulations, to support enhancing in-situ visualization with simulation control capabilities.
7. (Seismic cube use case only) Add dynamic coarsening/refinement support including load balancing to OPM Flow, by using already existing features of Dune grids such as ALUGrid, or an entirely new grid based on octree structures.

4.1.3 Other frameworks integration

(Carbon sequestration use case only) The ERT software may need to improve its coupling with SLURM or to integrate with other orchestrators. It may be necessary to define, adapt or improve an Application Programming Interface (API) for such couplings that enable ERT to ignore orchestration details, yet perform the job of updating ensembles as designed. ERT is a Python program, any interface defined should be possible to use from Python easily. The interface (orchestrator, intra-job scheduler or other) should then be used from ERT to avoid ERT making orchestration decisions.

4.2 Hardware integration requirements

4.2.1 Preliminary evaluation of communication between CPU and accelerators

OPM Flow by default does not use GPUs, thus is only run on CPUs. However, experimental code exists for running the linear solver part on GPUs, for both CUDA (using CuSparse) and OpenCL. In a normal simulation run on CPUs, the linear solver part takes from 50% to 80% of the total time, depending on the complexity of fluid models and to some extent the number of MPI processes used. The experimental GPU code has been

shown to give a modest speed increase, completing the linear solve phase in half time compared to a single CPU thread on a small test case (Norne). Clearly, this is far from fully exploiting the GPU capacity.
The current code only implements the ILU0 preconditioner. This is a fairly strong preconditioner, and it is in general not very easy to make massively parallel, as there is an inherent sequencing in the algorithm. However, weaker preconditioners usually fail to work well for reservoir simulation due to the strongly heterogeneous and discontinuous coefficients. The best preconditioner for larger cases tends to be the Constrained Pressure Residual (CPR) preconditioner, a two-stage preconditioner that first approximates the pressure using an Algebraic Multigrid (AMG) solve or cycle, and then applies ILU0 on the full system. The problem of making a massive parallel preconditioner is even harder for CPR.
Some experiments have been conducted before the start of ACROSS to accelerate the ILU0-BiCGStab section of OPM Flow with FPGA accelerators and the code has already been released on *Github*. So far these experiments have not yet yielded significant performance improvements, but we do not rule out the possibility that future developments will change this. Benchmarking the performances of the actual FPGA implementation shows non-optimal memory usage, with access patterns not able to fully exploit the available HBM bandwidth. Among the major challenges are performance portability between FPGAs (the experiments have been targeting Xilinx Alveo accelerator cards, using RTL code) as well as between accelerator types, and memory management on the FPGAs.

4.2.2 Accelerators for improving scalability and performance

In order to exploit the potential of hardware accelerators, we will do the following:

1. Perform analysis of the performance profile of the current GPU implementation to identify bottlenecks and potential for improvement.
2. Test the existing GPU code on larger cases to assess if that yields better performance.
3. Investigate using 32-bit floats rather than 64-bit doubles in some parts of the linear solver calculations.
4. Investigate other preconditioners that may better exploit the GPU, yet still, provide reasonable linear iteration counts. Development is underway on an SPAI preconditioner by a group outside the ACROSS project, this and other variants will be tested. Also, the ILU0 preconditioner can be applied in a more parallel and less accurate way, this can also be exploited to optimize total runtime.
5. Investigate the effort needed to implement an advanced CPR + AMG preconditioner on the GPU, as well as the possible performance benefits.
6. Investigate the effort needed to put the rest of the code (equations, properties, assembly) on the GPU.
7. Investigate the potential of using multiple GPUs and combining MPI and GPUs, as well as the effort needed to implement this.
8. (Seismic cube use case only) The data will have a simpler structure in this case and can be organized in an octree. Investigate exploiting the octree structure for better GPU acceleration.
9. Ensure that other stakeholders, in particular those who currently work on or have supported the GPU experiments, are included in the decision processes regarding GPU acceleration.
10. Investigate the use of FPGA accelerators, in particular tracking the development of programming interfaces and flexibility, with a view towards replacing parts of OPM Flow with kernels running on FPGAs.

4.3 Scheduling/workflows management tools for improving WP7 pilot use cases

The advanced workflow management system, which will be developed in the context of ACROSS and composed of a combination of the workflow engine YORC and a newly developed resource broker, will allow the reduction of the execution time of the WP7 workflows by leveraging workflow aware scheduling, automated jobs execution and submission, and efficient resource usage. This entails reducing queue time overhead of multi-step workflows and optimal allocation of resources given the information available about the upcoming workload. Besides that, orchestration of workflows spanning multiple resources (e.g. HPC and cloud) can be performed.
In detail, the new orchestration approach will allow assigning only the amount of resources actually needed by each step of the workflow dynamically, even if the requirements change between two workflow steps. In this context, along with developing the baseline interface between ERT and scheduling technologies such as SLURM or PBS, the ACROSS orchestration technology will be leveraged by the ERT tool in order to improve the orchestration strategy of the ensemble components on the available resources. Moreover, YSTIA will allow for high-level preparation and execution of the workflow instances and status monitoring, while the advanced resource broker will loosen up the traditional boundary between nodes, allowing for a more flexible and fine-grained partitioning of resources. This translates into queueing time reduction and resources occupation/energy efficiency, especially useful during the ensemble update iteration phase.

4.4 Expected future requirements

The hardware accelerators available, as well as the technologies used to access or program them, are being developed rapidly. While it is important for the OPM Flow software, as open-source software with a broad user base, to maintain the ability to run on generic hardware, we cannot ignore the trend towards heterogeneous systems. While the above indicates our immediate plans for this, in particular with regards to GPUs, they are predicated on the current situation, where no technologies are available to provide true “write once, run anywhere” performance. If this situation improves, requirements for accelerators may need to be revised.
Concerning orchestration requirements, the co-design effort in the context of WP2 and WP4 will proceed as the advanced workflow management system is developed and the requirements will be updated accordingly. The new assessment of these requirements will be made in deliverable D7.4.

5 Software

In this section, we present a comprehensive table with a description of all the software currently planned to be used in WP7.

Software	Description	Role in WP7	License
OPM Flow	OPM Flow is a reservoir simulator for solving subsurface porous media flow problems. It is targeted towards carbon sequestration and petroleum-related scenarios, and it is implemented using a flexible automatic differentiation (AD) approach to allow for easy extension with new fluid models. Existing fluid models include black-oil, polymer, solvent, and CO2 capabilities.	OPM Flow will be used to simulate the individual runs of the two primary use case scenarios addressed in the WP. It will be used to perform both to simulate single cases and to do the forward evaluation part of history matching on ensembles.	GNU General Public License, version 3 or later (GPLv3+)
Damaris	Damaris is a middleware for asynchronous I/O, visualization and analytics. It is targeted towards large-scale MPI based simulations that iteratively compute simulation results and then output data for post-processing at each iteration. Damaris enables the I/O to occur concurrently with the next iteration of the simulation and uses dedicated resources to carry out its processing. Dedicated resources (i.e. CPU cores) can be distributed per node or partitioned as separate dedicated nodes. Besides output of simulation data to disk, the dedicated resources can be used for other processing such as in-situ visualization or analytics, either of which is easily integrated with the Damaris, which has out-of-the-box rendering capability with Paraview Catalyst and VisIt visualization packages and the ability to integrate plug-in analytics functions written by the user.	Damaris is to be integrated with OPM Flow for improved I/O efficiency and allow the extension of OPM Flow with in-situ visualization and online/real-time analytics.	GNU Lesser General Public License (LGPL)

Melissa [6]	Mellisa is an in-transit data analytics platform used for sensitivity analysis. It is designed to remove the burden of saving and post-processing large numbers of simulation outputs to form real-time updated statistical variances of simulation field data. It does this by sending data via the network in a coordinated way to a distributed parallel server. Melissa is fault-tolerant and elastic thus supporting efficient use of computational and storage resources.	Integration of the Melissa client routines with Damaris server processes (which are subsequently used in OPM Flow) may be an efficient way to compute sensitivity statistics of large ensembles of OPM Flow runs, removing the need for intermediate files.	BSD 3-Clause License
ERT	The Ensemble based Reservoir Tool (ERT) is a tool for updating models (history matching) using Ensemble Kalman Filter or Ensemble Smoother methods. Starting from an initial ensemble that captures the important variation and uncertainty of the scenario, in each iteration of the history matching process the tool creates an updated ensemble of cases. It requires a reservoir simulator (OPM Flow or the commercial Eclipse simulator) to run each individual ensemble case.	History matching workflows for the CO2 storage use case will be run using ERT.	GNU General Public License, version 3 (GPLv3)

Table 1 - Software of WP7

6 Conclusions

In this deliverable, we were concerned mainly with the requirements of the WP7 software stack and the main ideas behind the statistical analysis of the outputs of models. In section 2 the use cases integration effort of WP7 was presented. Next, the uncertainty and sensitivity analyses were explained in section 3 as they should be used later in the project to assess the results of the use cases. Section 4 was concerned with the requirements on the successful deployment of main tools OPM Flow with Damaris in the ACROSS project. Other frameworks that will be used were also briefly mentioned and their relation to the ACROSS project in terms of WP7. Additionally, the hardware requirements were considered with a dominant focus on the possibility to extend the current tools to leverage the GPU accelerators. In section 5 an overview of the software which will be used, or is being considered to be tested is presented.

References

[1] *Thermal Hydraulics Aspects of Liquid Metal Cooled Nuclear Reactors*. Elsevier, 2019. doi: 10.1016/C2016-0-01216-0.

[2] E. Pisoni, D. Albrecht, T. A. Mara, R. Rosati, S. Tarantola, and P. Thunis, "Application of uncertainty and sensitivity analysis to the air quality SHERPA modelling tool," *Atmos. Environ.*, vol. 183, pp. 84–93, Jun. 2018, doi: 10.1016/j.atmosenv.2018.04.006.

[3] W. E. Becker, "Uncertainty Propagation Through Large Nonlinear Models," 2011, doi: 10.13140/RG.2.1.4721.1921.

[4] M. Dorier, G. Antoniu, F. Cappello, M. Snir, and L. Orf, "Damaris: How to Efficiently Leverage Multicore Parallelism to Achieve Scalable, Jitter-free I/O," in *2012 IEEE International Conference on Cluster Computing*, Beijing, China, Sep. 2012, pp. 155–163. doi: 10.1109/CLUSTER.2012.26.

[5] M. Dorier, R. Sisneros, T. Peterka, G. Antoniu, and D. Semeraro, "Damaris/Viz: A nonintrusive, adaptable and user-friendly in situ visualization framework," in *2013 IEEE Symposium on Large-Scale Data Analysis and Visualization (LDAV)*, Atlanta, GA, USA, Oct. 2013, pp. 67–75. doi: 10.1109/LDAV.2013.6675160.

[6] T. Terraz, A. Ribes, Y. Fournier, B. Iooss, and B. Raffin, "Melissa: large scale in transit sensitivity analysis avoiding intermediate files," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Denver Colorado, Nov. 2017, pp. 1–14. doi: 10.1145/3126908.3126922.