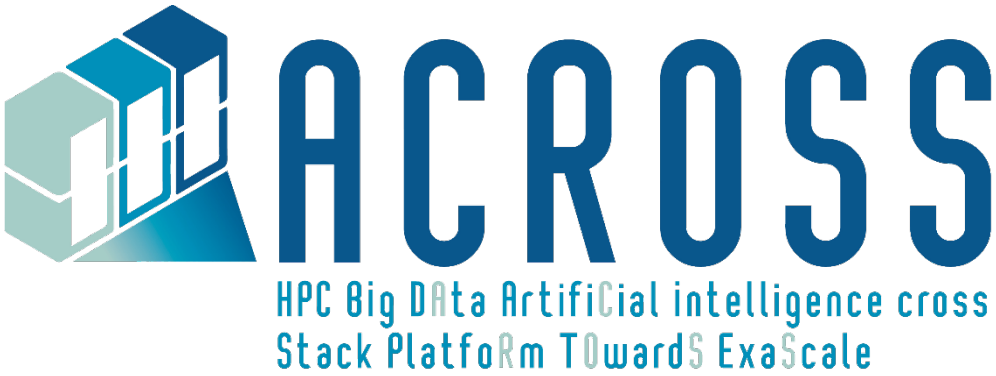

D7.4 – Stage 2 requirements for HW/SW integration

Deliverable ID	D7.4
Deliverable Title	Stage 2 requirements for HW/SW integration
Work Package	WP7
Dissemination Level	PUBLIC
Version	2.1
Date	2022 – 10 – 27
Status	FINAL
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	2022-04-25	IT4I	First version, ToC
0.2	2022-07-22	IT4I	Update section 3.1,3.2, 4.2 and added HyperQueue to section 5
0.3	2022-07-29	INRIA	Update 4.1.1 and 5
0.4	2022-09-27	SINTEF	Update sections 3 and 4
1.0	2022-09-30	IT4I	Fill in Executive Summary, Introduction and Conclusion
1.1	2022-10-21	IT4I, SINTEF, INRIA	Response to the reviewers' comments and update document accordingly.
2.0	2022-10-24	IT4I	Final formatting and check
2.1	2022-10-27	LINKS	Applying styles to the text and tables

Table of Contents

Glossary

Acronym	Explanation
HPC	High-Performance Computing
CPU	Central Processing Unit
GPU	Graphical Processing Unit
FPGA	Field Programmable Gate Array

MPI	Message Passing Interface
CPR	Constrained Pressure Residual
AMG	Algebraic Multigrid (preconditioners)
KPI	Key Performance Indicator
HW	Hardware
SW	Software
SF	StreamFlow
HQ	HyperQueue
NDA	Non-Disclosure Agreement
HDF	Hierarchical Data Format
XSD	XML Schema Definition
CWL	Common Workflow Language
HM	History Matching

List of Figures

List of Tables

Executive Summary

This is the second of two deliverables aimed at the definition of the integration requirements in the WP7 pilot of the ACROSS project. It is an update of the integration requirements collected in the first months of the project and sums up what was already integrated and how does it affect the requirements expectations. There are 5 sections: an introduction, an overview of the integration within the WP7 pilot, uncertainty, and sensitivity analysis, updated integration requirements, and finally a list of the software used in the pilot.

Position of the deliverable in the whole project context

This deliverable summarizes requirements collected in WP7 in the first half of the project and outlines the expected steps to be taken in the second half of the project. It is related to the deliverables D2.1, D7.1 and D4.1, where all these deliverables were focused on specific parts of requirements collected in the project. It is an update of the D7.1, which was describing initial requirements of WP7.

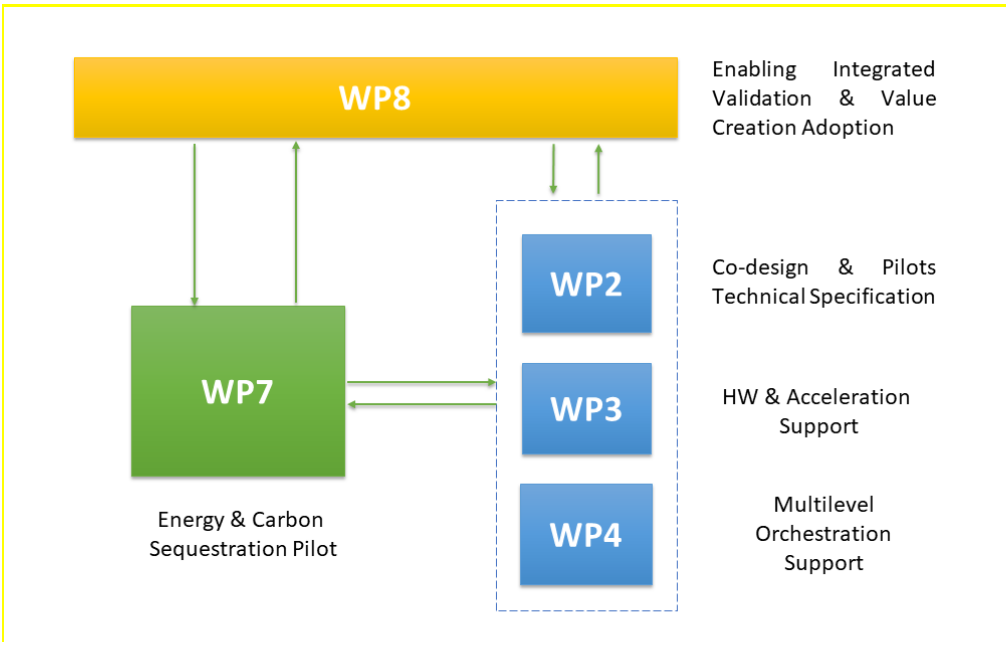

Figure 1 - Position of the WP7 in the context of the ACROSS project.

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 describes 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 second of two deliverables with the same scope. This deliverable is an updated version with revised information in the second stage. In this case, it is the overview of the work done since the D7.1 and changes in the requirements based on the discussion and integration attempts made in the first half of the ACROSS project.

1.1 Scope

This document contains the requirements for hardware and software integration work that was done and that needs to be done in the WP7 of the ACROSS project. It describes the work already done in the WP7 and the findings during the integration work as well as plan for the rest of the project. This is an update of the D7.1 which contained initial requirements plan.

1.2 Related documents

ID	Title	Reference	Version	Date
D2.1	WP2 - Summary of Pilots co-design requirements		3.0	2021-08-31
D4.1	System Requirements Analysis for Orchestrator Design		0.7	2022-02-31
D7.1	Stage 1 requirements for HW/SW integration		2.0	2021–08-31

Table 1 - Related documents

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 second part of the project will be:

- Complete and improve on the integration of Damaris into the OPM Flow program, to improve capabilities for I/O, scalability, and in-situ processing and visualization. Take advantage of this to integrate analysis and processing scripts written in Python/Dask.
- Test and improve the integration of HM and other ensemble workflows with the ACROSS platform and in particular the orchestration components.
- Further, improving the support of accelerators in OPM Flow, to take advantage of current and upcoming trends in computational hardware. In particular, integrating the support for multi-GPU setups and combination of MPI with GPUs.

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

3 Applied Methodology results

3.1 Sensitivity analysis of the carbon sequestration pilot

At the beginning of the project, we were looking into methods which are relevant for the sensitivity analysis and early problems such as unavailable derivatives for complex systems were mentioned in the D7.1.
IT4I provided partners a simulation-based uncertainty analysis example with three input parameters and one output parameter. The overall uncertainty of the model output is represented by a histogram, whereas the sensitivity of input parameters to the model output is represented by a scatterplot. The software tool can represent the uncertainty of input parameters by user given probabilistic models. For simplicity, input parameters of the example are represented by the uniform distribution. An example of sensitivity analysis presented in this deliverable will be later replaced by the complex OPM Flow model as part of the carbon sequestration use case. For this reason, SINTEF implements software codes for generation of samples for uncertainty and sensitivity analyses. The software codes for generation of samples provided by SINTEF leverage the free and open-source software tools, namely OPM Flow software and the MATLAB Reservoir

Simulation Toolbox. As a synergy with the Norwegian Carbon Capture and Storage Research Centre (NCCS), which covers the full carbon value chain from the carbon capture to carbon transport and storage and represents a well-established carbon capture research centre in Norway, SINTEF consults technical details of the carbon capture sensitivity analysis use case with NCCS, as the reservoir simulations of ACROSS represents the last step of the carbon storage cycle.

The analysis of input parameters for the ACROSS carbon sequestration sensitivity analysis use case is inspired by the state of the art of the ENOS project[1], in which six uncertain input parameters have been considered (porosity, permeability, permeability anisotropy, regional hydraulic gradient, relative permeability and capillary pressure). Authors in the ENOS project assumed 1000 numerical reservoir flow simulations.

Finally, the following structure of output parameters were discussed for the ACROSS carbon sequestration sensitivity analysis use case: amount of injected CO_2, amount of free-flowing CO_2, amount of leaked CO_2 from the structure, capillary trapped CO_2 and dissolved CO_2. The parallel scaling of simulations is of interest for the ACROSS project and it is the subject of the future research cooperation.

The draft version of input parameters for the ACROSS carbon sequestration sensitivity analysis use case is summarized in Table 2. All parameters will be represented by variation of data in an ensemble of simulation input decks.

Parameter	Uncertainty type	Minimum	Maximum
Temperature gradient	Parametric/model	30 degrees C	40 degrees C
permeability of sand layers	Parametric/model	1100 mD	5000 mD
permeability of shale between sand layers	Parametric/model	0.00075 mD	0.0015 mD
permeability of feeder chimneys connecting sand layers	Parametric/model	1100 mD	5000 mD
porosities of sand, shale, and feeders	Parametric/model	0.27	0.4
topography of top surface	Parametric/model	-10 m	+10 m

Table 2 - Draft version of input parameters for the ACROSS carbon sequestration sensitivity analysis use case.

The above (Table 2) sensitivity analysis will be performed using an ensemble of cases. An alternative method not in the Monte Carlo family, is to obtain directly from the simulator the sensitivities of the quantities we seek. These partial derivatives are possible to get, using the automatic differentiation framework upon which OPM Flow is built. However, this will require some modification to the simulator, as this feature is only experimental at this point. We therefore do not foresee using this possibility in ACROSS, at least not in the near term (2022).

3.2 KPIs definition of integration results

The KPIs that were chosen initially were:

- 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.

We think they are appropriate still, although the precise meaning of the second should be adjusted. Since we seek to exploit a mix of distributed (MPI) and shared-memory (OpenMP) parallelism, we define the target instead as (changes in bold):

- Scaling to 1000 **cores** with reasonable efficiency, **with a mixture of OpenMP threads and MPI**.

[1] http://www.enos-project.eu/media/21783/d2_2_uncertainty_vf.pdf

This target applies to scaling a single simulation case run across the spatial domain (i.e., domain decomposition) with MPI and within each subdomain exploiting thread parallelism. We note that for an ensemble, the individual case runs are independent and therefore can scale arbitrarily well up to the ensemble size.
At this point, we scale well to 3 nodes on the Karolina system (384 cores). The target for Karolina, in particular, is therefore to scale well to 8 nodes on that system.

4 Integration requirements for the use cases update

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 [1], [2] 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.
- The interface of Damaris is a pure C interface, which is simple to integrate, and offers the functionality that is appropriate for use with OPM Flow (other than additions outlined below). Extending Damaris plugins if required will make use of C++ classes, as the Damaris core implementation is written in C++.

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) the parameters are then used as variables in other defined elements of the XML file, such as <*layouts*> and <*variables*>. Then the host software can programmatically modify the <*paramater*> entries and any element dependent on the parameter is then updated i.e., 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 the 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 of 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 has been found to be a valid option to remove this issue. A new class has been integrated into the Damaris library that allows a regular expression-based substitution of XML content in a templated Damaris XML file to be created at runtime and the substituted XML file then used to configure Damaris. This functionality was released in Damaris v1.6.0

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, Python, 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. The work on this specific aspect will be investigated more in the second half of the project, with the need to focus on the use of hwloc to place ranks.

Resulting from work in T7.3, The Damaris library (repository tag v1.7.0) has the ability to perform in-situ analysis on simulation output data using the Python based Dask library. Further details of this integration can be found on the Damaris Wiki site https://gitlab.inria.fr/Damaris/damaris/-/wikis/Damaris-Python-Support. The development of Python integration in Damaris has required the development of a Python module that wraps the Damaris API, named damaris4py. This module is dependent on the MPI4Py library as MPI4Py allows exposure of MPI sub-communicators such as those used by Damaris. Some example workflows have been generated to show the functionality of this integration, using both standard shell script-based use of cluster resource allocators and the CWL StreamFlow runtime.

4.1.2 OPM Flow integration

The following modifications to OPM Flow were identified in D7.1 as required for its integration with the ACROSS platform and other software. For each item, we repeat its description and note the current status.

1. OPM Flow must become able to run in a 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.
   Status: Complete.
2. OPM Flow must add Damaris as an optional build dependency, along with any Damaris dependencies that cannot be avoided.
   Status: Complete.
3. When compiled with Damaris support, OPM Flow must modify its start up 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.
   Status: Completed through programmatic control (not requiring user to supply external XML file).
4. OPM Flow has an indirect dependency on HDF5 library through the the Damaris library.
   Status: Complete, done by using the Damaris Cmake configuration.
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.
   Status: Partially complete. A preliminary format has been decided. HDF5 will be used with a keyword-data mapping similar to the Eclipse binary output, using the same keywords for the data as in the Eclipse format. A long-term permanent solution has not been reached yet. Such a long-term solution should also include a vizschema description to easily integrate with ParaView.
6. Improve support for pausing and resuming simulations, to support enhancing in-situ visualization with simulation control capabilities.
   Status: Not complete.

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.
   Status: Partially complete. ALUGrid has been integrated into OPM Flow, but the use of dynamic coarsening and refinement supported by that grid has not yet been integrated.

The following new modification has been identified as required for integration during the first half of the ACROSS project:

- Ensure that OPM Flow can exploit MPI and GPUs in combination, including nodes with multi-GPU setups.

4.1.3 Other frameworks integration

The original requirement was: (Carbon sequestration use case only). The ERT software (see Table 3) may need to improve its coupling with SLURM workload manager 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.
Status: This has been completed by adding the capability of ERT to communicate with HyperQueue, which in turn provides access to both the queuing systems relevant for ACROS (SLURM and PBS).

4.2 Hardware integration requirements

4.2.1 Preliminary evaluation of communication between CPU and accelerators

Experiments have been carried out to identify whether the use of mixed precision (single instead of double precision) could provide additional performance improvements on the GPU. Results were not promising as the improvement in performance was amortized by more iterations in the linear solver to account for lower precision and higher numeric errors.

At the beginning of the ACROSS project, there was no multi-GPU support in OPM Flow. Due to no improvement in single/mixed precision computations, it was decided to address scalability using multiple GPUs instead. This work is currently under way and benchmarked for a single multi-GPU node. The selection of GPUs is envisioned via MPI, so it integrates seamlessly into the existing OPM Flow and HPC environment. Multi-GPU support is initially considered only for the linear solver of OPM Flow.

During the investigation phase of potential development directions for the OPM Flow application, the FPGA acceleration technology was taken into consideration. Indeed, several investigative and evaluation actions of this technology for this specific application, have been undertaken by ATOS and other partners. These actions can be summarized as follows:

- FBLAS: Streaming Linear Algebra on FPGA: A porting of the BLAS numerical library (including BiCG solvers for small size matrix) for Intel FPGA platform.
- Evaluation of the Linear Solver of OpenCL/OneAPI Library on Arria 10.
- Pre-condition Conjugate Gradient on AI Engine: Solving the PCG problem on Xilinx AI Engine Array Architecture.

These actions were for the most part carried out also with support from with Intel Corp. Italia on one hand, and Xilinx Corp. on the other. NDAs do not allow the report of intermediate results.

However, these experiences also demonstrated that these standard technologies on FPGA are not yet suitable to improve the performance of large problems like that of WP7 (sparse 10^6x10^6 matrix) and therefore, significant effort would be required to achieve significant improvements with this approach. This consideration of resource limitation leads us to refocus on the GPU technology which seems to be the one adopted by other commercial products on the same topic.

4.2.2 Accelerators for improving scalability and performance

The following list of actions were identified in D7.1, we repeat the description and add information on current status:

1. Perform analysis of the performance profile of the current GPU implementation to identify bottlenecks and potential for improvement.
   Status: Complete. Main conclusion is that other preconditioning methods than ILU0 are necessary to exploit the accelerators well, see item 4. Also *see Figure 2 for some results from this work; note that the problem size needs to be fairly large for the GPU to have an advantage, and that the advantage is limited for ILU0.*
2. Test the existing GPU code on larger cases to assess if that yields better performance.
   Status: Complete. Indeed, larger cases get better acceleration, but not sufficient to change the conclusion to item 1.
3. Investigate using 32-bit floats rather than 64-bit doubles in some parts of the linear solver calculations.
   Status: Complete. While this yielded faster application of the preconditioner, attaining the required convergence for solution of the nonlinear systems required more iterations. There was therefore no significant net gain identified.
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.
   Status: Partially complete. SPAI is in testing, and the ISAI preconditioner has been identified as a promising method.
5. Investigate the effort needed to implement an advanced CPR + AMG preconditioner on the GPU, as well as the possible performance benefits.
   Status: Not done in ACROSS, but an experimental implementation by another group has become available.
6. Investigate the effort needed to put the rest of the code (equations, properties, assembly) on the GPU.
   Status: Work is ongoing. It is not likely that such an effort will be completed within the ACROSS timeframe, but initial work may take place.
7. Investigate the potential of using multiple GPUs and combining MPI and GPUs, as well as the effort needed to implement this.
   Status: Complete. An initial implementation has been created, and work is ongoing to improve and optimize it.
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.
   Status: Not started.
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.
   Status: This has been accomplished through video meetings and open discussions on GitHub.
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.
    Status: Complete. We have decided to not consider FPGAs for acceleration for the time being.

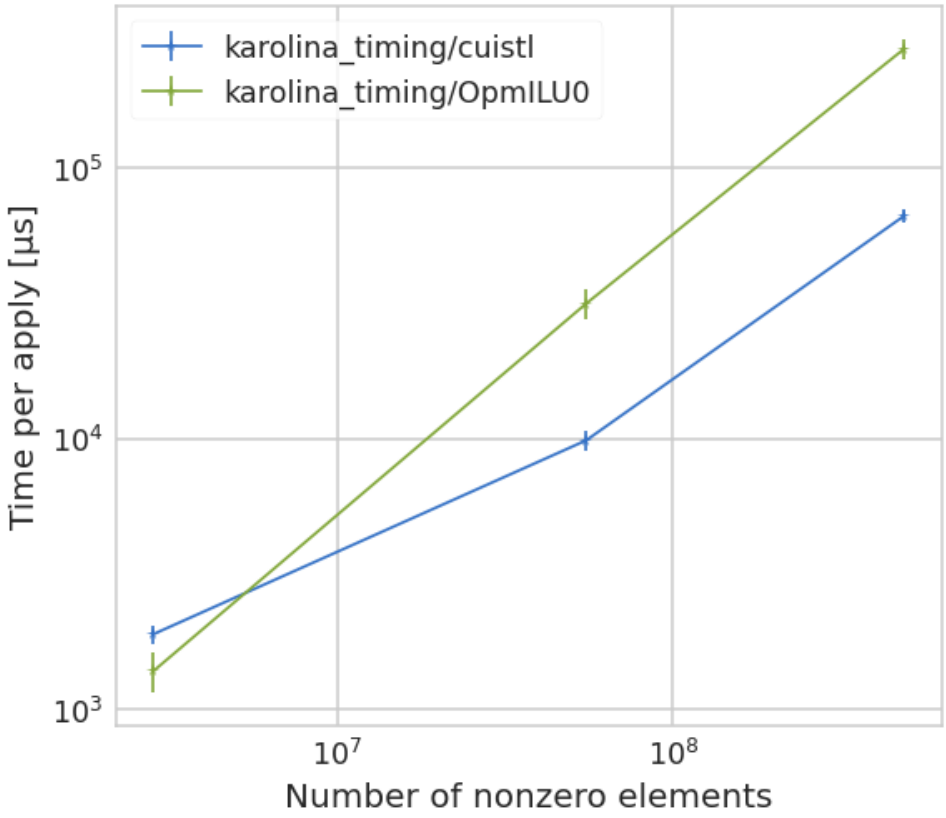

Figure 2 - Time for a single application of the block ILU0 preconditioner at a few problem sizes, single-thread CPU versus one GPU. Shows that the ILU0 method has limited GPU scaling potential.

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

Considering the architecture presented in D4.1 and the specific features exposed by Yorc Orchestrator[2] and StreamFlow[3] (SF) in relation to the Pilot workflows requirements, the tool of choice for the workflow execution will be StreamFlow, while Yorc Orchestrator will be used to support cloud deployment of relevant components. SF will be used together with the WARP resource broker and the HyperQueue (HQ) scheduler in order to provide efficient execution of the WP7 workflows, particular focus will be posed to the deterministic execution of workflows (in contrast to the current state-of-the-art queuing systems, which do not provide any guarantee on the execution time for the typical operational setting), workflow-aware scheduling, and fine-grained resource allocation.

The main idea is to be enable the OPM user to reserve the necessary resources for the workflow execution in advance, so that the workflow steps can be executed deterministically without waiting in the queue between each step, leading to a faster time-to-solution. The ERT tool is also being integrated with HyperQueue, providing several advantages over the current implementation: 1) SLURM and PBS are not currently fully supported by ERT, HQ can act as a proxy for both, reducing the development effort and enabling portability; 2) HQ can support more flexible and fine-grained resource usage (i.e., assigning less than a full node to a job), allowing for better exploitation of available resources.

Finally, a working example of the integration between Damaris + DASK and StreamFlow has been provided to demonstrate interoperability, providing benefits to anything uses Damaris to enable in-situ/in-transit I/O.

[2] https://github.com/ystia/yorc

[3] https://streamflow.di.unito.it

4.4 Expected future requirements

In D7.1 we stated that there were no technologies available to provide true "write once, run anywhere" performance. We now consider that SYCL[4] may be a technology that, if adopted for OPM Flow, may provide a path towards this goal. We will continue evaluating SYCL and similar open standards with this in mind.

5 Software

In this section, we present a comprehensive table with a description of all the software currently 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 CO_2 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 (via Dask).	GNU Lesser General Public License (LGPLv3+)
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	History matching workflows for the CO_2 storage use case will be run using ERT.	GNU General Public License, version 3 (GPLv3)

[4] https://www.khronos.org/sycl/

	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.		
HyperQueue	HyperQueue allows to build a computation plan consisting of a large number of tasks and then execute it transparently over a system like SLURM/PBS. It dynamically groups jobs into SLURM/PBS jobs and distributes them to fully utilize allocated nodes. It removes the need to manually aggregate computational tasks into SLURM/PBS jobs.	HQ is used as a new ERT queuing system.	MIT
Dask	Dask is a Python based general purpose distributed analytics and computational framework	Dask is integrated with Damaris to allow simulation data to be passed through	BSD 3-Clause

Table 3 - Software of WP7

6 Conclusions

In this deliverable, we presented mainly the updates in the requirements collection and integration work in the WP7 software stack and HW such as accelerators. In section 2 the use cases of the WP7 were presented and current status of their integration. Then, in section 3 we described work on the uncertainty analysis and how it will be extended in the later stages of the project. Section 4 described the work done on the integration of different SW components such as OPM Flow and Damaris integration, linear solver GPU benchmarking and multi-GPU evaluation, integration of ERT with HyperQueue, and work with other accelerators such as FPGA. Finally, in section 5 is an overview of the software used in the WP7.

References

[1] 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.

[2] 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.