The published version can be found at: https://link.springer.com/chapter/10.1007/978-3-030-90539-2_35
DOI: https://doi.org/10.1007/978-3-030-90539-2_35

Including in situ visualization and analysis in PDI *

Christian Witzler[1][0000−0002−6423−8590], J. Miguel Zavala-Aké[2], Karol Sierociński[3][0000−0002−1221−049X], and Herbert Owen[4][0000−0001−8543−303X]

[1] Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Jülich, Germany
c.witzler@fz-juelich.de
http://www.fz-juelich.de/jsc
[2] Meteorologiska institutionen, Sweden
miguelza@kth.se
[3] Poznan Supercomputing and Networking Center: Poznan, PL
ksiero@man.poznan.pl
[4] Barcelona Supercomputing Center, Barcelona, Spain
herbert.owen@bsc.es

Abstract. The goal of this work was to integrate in situ possibilities into the general-purpose code-coupling library PDI [1]. This is done using the simulation code Alya as an example. Here, an open design is taken into account to later create possibilities to extend this to other simulation codes, that are using PDI.
Here, an in transit solution was chosen to separate the simulation as much as possible from the analysis and visualization. To implement this, ADIOS2 is used for data transport. However, to prevent too strong a commitment to one tool, SENSEI is interposed between simulation and ADIOS2 as well as in the in-transit endpoint between ADIOS2 and the visualization software. This allows a user who wants a different solution to easily implement it. However, the visualization with ParaView Catalyst was chosen as default for the time being.

Keywords: in transit · SENSEI · ADIOS2 · High-performance computing (HPC).

1 Motivation

The goal of this work was to integrate in situ possibilities into the general-purpose code-coupling library PDI [1]. This is done using the simulation code Alya as an example. Here, an open design is taken into account to later create possibilities to extend this to other simulation codes, that are using PDI.

The use of in situ procedures has many advantages. It allows to see first results faster, to analyze more intermediate steps and reduce the required I/O resources because fewer or even no complete data sets have to be stored. Despite

* Supported by EoCoE-II and CoEC.

these advantages, the use of in situ methods is fraught with problems. For example, there are further dependencies when programming simulations, as well as when compiling and running the simulation. These dependencies can also change with version changes of the used in situ library, which requires additional work on the simulation code to be able to use current versions with new features. In addition, there is also a competition for resources on the individual compute nodes, as additional code must now be run for the in situ procedures. This requires CPU time, network bandwidth, and most importantly for some simulations, working memory, depending on the chosen in-situ method. Especially in the case of an in situ visualization running on the same compute nodes as the simulation, there is a risk that small changes to the visualization pipeline can dramatically increase the amount of memory required. This then has the risk of crashing the simulation and the complete progress of the simulation could be lost.

2 pdi2sensei

Here we use different solutions that are already available in the in situ area and combine them in a way to make it as easy and as error-free as possible for the end-user. For this, we first use PDI [2] (PDI data interface), which allows reducing the dependencies of the simulation to a library. PDI allows users to customize the data output of the simulation via a configuration file without recompiling the simulation. PDI enables different methods of data output to be configured, such as traditional I/O with the use of HDF5. PDI requires only a library agnostic annotation of the simulation code and then allows the use of libraries from a specification tree [1].

The further problem of changes in version jumps of in situ libraries is solved by using SENSEI [7], which provides a common interface to use different in situ libraries with one interface. This allows to support various libraries at once and permits to change the visualization backend later if there are new or different requirements for the in situ analysis or visualization.

To minimize the conflict over resources and impact on the simulation, we use the in-transit specialization of the in situ solutions. Therefore, we use different computational nodes for the actual analysis and visualization work. This reduces the impact on the simulation nodes to only the data traffic to the in-transit nodes. It also allows us to use appropriate nodes (for example use nodes with GPU for your visualization) and node counts for both visualization and simulation. For the data transport, we use ADIOS2 [8, 9], which transports the data for us. ADIOS2 also allows the use of M2N communication, so the number of simulation nodes can be different from the number of in-transit nodes, and ADIOS2 distributes the data. ADIOS2 has several transport mechanisms [9] (called engine in ADIOS2), currently, we use SST, which in our case uses a shared file system to exchange network information via a file and then transport the user data via the network. In the future, the different engines that are possible with ADIOS2 will also allow testing with new hardware such as burst buffers by

writing to the burst buffer for data transfer and retrieving this data from other nodes. This would then make the simulation code even more independent from the analysis and visualization code.

ADIOS2 then passes the data on the in-transit node to SENSEI, where it can then be flexibly transferred to various in situ solutions. Here we use the solution developed by ParaView [10] called Catalyst [11]. Using Catalyst, a predefined visualization pipeline can be used to store rendered images and pre-analyzed data. In addition, there is the possibility to use a live interactive visualization, where you can send selected data to ParaView and adjust your visualization with ongoing updates.

Here we started with the development of pdi2sensei, which makes it possible to easily transfer the data passed by PDI from the simulation to SENSEI while making the conversion to the VTK data needed by SENSEI as simple as possible. As a base configuration, this SENSEI component is configured to use ADIOS2 to send the data directly to the in-transit nodes. Here we use SENSEI again to pass the data from ADIOS2 flexibly to different in situ solutions, here Catalyst. Using SENSEI in both places allows adapting both transport and in situ solutions to new requirements. As the first simulation using this setup, we have chosen Alya, for the purpose of simulating the airflow over complex terrain.

3 Example

Alya is a parallel multi-physics/multi-scale simulation code developed at the Barcelona Super-computing Center to run efficiently on high-performance computing environments. It can solve a wide range of problems, including solid mechanics, compressible and incompressible flow, flows with interfaces using the level set method, combustion, and thermal problems. The case presented in this work is an incompressible flow simulation using wall-modeled Large Eddy Simulation. The convective term is discretized using a recently proposed Galerkin finite element (FEM) scheme, which conserves linear and angular momentum, and kinetic energy at the discrete level described in [13]. Neither upwinding nor any equivalent momentum stabilization is employed. To use equal-order elements for both velocity and pressure, numerical dissipation is introduced only for pressure stabilization via a fractional step scheme. The set of equations is integrated in time using a third-order Runge-Kutta explicit scheme. Due to the high Reynolds number of Atmospheric Boundary Layer flows, wall modeling is required at the ground. A novel wall modeling finite element implementation is used [14] in this work.

The Bolund experiment is a classical benchmark for microscale atmospheric flow models over complex terrain [15–17]. It has been the basis for a unique blind comparison of flow models. Despite its relatively small size, its shape induces complex 3D flow. The wind comes from the sea, and thus the inflow profile is relatively simple to impose. The flow collides against a 10m height cliff, and a complex recirculation is formed at the region of interest located at the top of the cliff. For the results presented in this work, an unstructured grid with 31M

elements has been used. The number of Elements can be scaled up by Alya as well, by passing on a subdivision command in the configuration file, splitting each cell into eight cells [18]. As this can be done multiple times the simulation's cell numbers can be increased by powers of eight, increasing the resolution of the simulation data.

4 Conclusion

To be able to use this in such a way some steps were necessary. First of all, PDI must be integrated to be able to transfer the data to PDI. Here it was very helpful that there is already a previous possibility to do in situ visualization in Alya, so the access to all needed data was already possible in one place in the code. However, the previous solution is not as flexible as the new solution, which is why this update was made. Next, a simple way had to be created in pdi2sensei to start the previously described setup with in-transit as easy as possible and with as little setup as necessary as a base configuration. Then, functions were implemented in pdi2sensei that allows for the user of pdi2sensei to connect data provided by the simulation through PDI to the data expected by SENSEI. This allows using the setup with few lines of code. You only have to pass your data to PDI in the simulation code and specify in the configuration file of PDI which values you want to pass to pdi2sensei.

Acknowledgment

This work was supported by the European Council under the Horizon 2020 Project Energy oriented Center of Excellence for computing applications - EoCoE II, grant agreement No 824158 and the Center of Excellence for combustion - CoEC, 952181.

References

1. PDI concepts, `https://pdi.julien-bigot.fr/master/Concepts.html`. Last accessed 18 Jun 2021
2. PDI documentation, `https://pdi.julien-bigot.fr/master/`. Last accessed 18 Jun 2021
3. Corentin Roussel, Kai Keller, Mohamed Gaalich, Leonardo Bautista Gomez, Julien Bigot. PDI, an approach to decouple I/O concerns from high-performance simulation codes. 2017.
4. YAML specification `https://yaml.org/spec/`. Last accessed 18 Jun 2021
5. HDF5 specification `https://support.hdfgroup.org/HDF5/doc/H5.format.html`. Last accessed 18 Jun 2021
6. pybind11 repository `https://github.com/pybind/pybind11`. Last accessed 18 Jun 2021
7. Sensei homepage, `https://sensei-insitu.org/`. Last accessed 24 Apr 2021
8. ADIOS2 documentation, `https://adios2.readthedocs.io/en/latest/`. Last accessed 24 Apr 2021

9. William F. Godoy, Norbert Podhorszki, Ruonan Wang, Chuck Atkins, Greg Eisenhauer, Junmin Gu, Philip Davis, Jong Choi, Kai Germaschewski, Kevin Huck, et al.: ADIOS 2: The Adaptable Input Output System. A framework for high-performance data management. SoftwareX, vol. 12, 100561 (2020). doi: 10.1016/j.softx.2020.100561
10. ParaView homepage, Kiteware, `https://www.paraview.org/`. Last accessed 24 Apr 2021
11. Catalyst homepage, Kiteware, `https://www.paraview.org/in-situ/`. Last accessed 24 Apr 2021
12. Usage introduction to PDI, `https://pdi.julien-bigot.fr/1.2/First_steps.html`. Last accessed 18 Jun 2021
13. O. Lehmkuhl, G. Houzeaux, H. Owen, G. Chryso- kentis, and I. Rodriguez, "A low-dissipation finite element scheme for scale resolving simulations of turbulent flows," Journal of Computational Physics, vol. 390, pp. 51 – 65, 2019.
14. H. Owen, G. Chrysokentis, M. Avila, D. Mira, G. Houzeaux, R. Borrell, J. C. Cajas, and O. Lehmkuhl, "Wall-modeled large-eddy simulation in a finite element framework," International Journal for Numerical Methods in Fluids, 2019.
15. J. Berg, J. Mann, A. Bechmann, M. Courtney and H. Jorgensen, The Bolund experiment, part I: flow over a steep, three-dimensional hill. *Bound Layer Meteorol*, Vol. **141(2)**, pp. 219—243, 2011.
16. A. Bechmann, N. Sørensen, J. Berg, J. Mann and P.E. Rethore, The Bolund experiment, part II: flow over a steep, three-dimensional hill. *Bound Layer Meteorol*, Vol. **141(2)**, pp. 245-–271, 2011.
17. DTU Wind Energy, `https://www.bolund.vindenergi.dtu.dk/the_bolund_experiment`. Last accessed 24 Apr 2021
18. G. Houzeaux, R. de la Cruz, H. Owen, M. Vázquez, Parallel uniform mesh multiplication applied to a Navier–Stokes solver, Computers & Fluids 80, pp. 142-151, 2013.