In situ data analysis and visualization in OpenFOAM with ParaView Catalyst

Simone Bnà[a], Mark Olesen[b], Andrew Bauer[c]

(a) Cineca, via Magnanelli 6/3, 40033, Casalecchio di Reno, Italy, +39-0516171938, simone.bna@cineca.it
(b) ESI Gbmh, Einsteinring 24, 85609, Munich, Germany, +49-1719710149, mark.olesen@esi-group.com
(c) Kitware Inc, Clifton Park, New York, US, andy.bauer@kitware.com

Introduction

It is well known that more and more time is spent during the simulation for I/O operations and postprocessing. I/O is recognized to be the main bottleneck to achieve the Exascale computing, which is up to 1000x faster than current Petascale systems [1]. The main cause can be identified in the disproportion of the rate of change between memory and external storage and CPUs. With a such limited I/O bandwidth and capacity, it is acknowledged that traditional 3-step workflows (pre-processing, simulation, post-processing) are not sustainable in the exascale era. One approach to overcome the data transfer bottleneck is through an in situ approach: the in situ approach moves some of the post-processing tasks in line with the simulation code [2].

ParaView Catalyst [3] is an open-source data processing and visualization library that enables in situ data analysis and visualization. Built on top of and designed to interoperate with the standard visualization toolkit VTK, Catalyst enables simulations to perform analysis, produce output data and visualize intermediate results during a running simulation concurrently [2].

OpenFOAM [4] is a well-known open-source CFD software package used by engineers and scientists from both the commercial and academic industries. The desire to apply in situ techniques to OpenFOAM using open-source softwares led to the development of a **plugin** based on the ParaView Catalyst library. Exploration of in situ techniques for OpenFOAM could have different purposes and goals:

- provide OpenFOAM users early visual feedback of their current simulation jobs from the earliest stages of the computation;
- reduce the amount of data stored as early and interactive feedback could help in tuning the frequency, resolution and format of saved data;
- computational steering connections that let the scientist to analyse the results on the fly and changing of the analysis pipelines interactively, through user feedback;

- allow for scaling of heavy postprocessing operation that would benefit of the same scaling available for the simulation;
- develop analysis pipelines using C++ or Python that are executed along side the simulation run, in the same address space;
- promote adoption of web based presentation of simulation results for sharing amongst work groups by embedding production of visual artefacts within the batch simulation jobs (ParaView Cinema).

Architectural design

The Catalyst plugin [5] is implemented in OpenFOAM as a FunctionObject named c*atalystFunctionObject*. This implies that the functionalities can be simply plugged in by adding it in the c*ontrolDict* file. This function object is called many times according to the frequency specified by the user in the catalyst dictionary.

The *catalystFunctionObject* owns a ***coprocessor*** and a list of ***inputs***. The *coprocessor* is the interface between the simulation and Catalyst and it is implemented in the c*atalystCoprocess* class. The roles of the ***coprocessor*** are:

- initialize the Catalyst library
- load/reset the Catalyst scripts (e.g. Python pipelines)
- query Catalyst to determine if any analysis is required to be run for the current timestep
- an adaptor backend that maps the OpenFOAM data structures to Catalyst, which uses a VTK data model
- finalize the Catalyst library

Inputs are instead the sources of the VTK pipelines to be processed by ParaView Catalyst. The available *inputs* in the plugin are (see Figure 2):

- ***fvMeshInput:*** an input (source) from fvMesh regions
- ***faMeshInput:*** an input (source) from faMesh regions
- ***cloudInput:*** an input (source) from clouds (lagrangian)

Since each input works with different data types, a specific adaptor backend for the coprocessor has been developed:

- ***fvMeshAdaptor***: the backend for *fvMeshInput*; the output is a multi-block dataset with one block per region. Each region block contains up to two blocks corresponding to the internal (volume) mesh (block 0) and the boundary (block 1), which are further divided into sub-blocks for each patch.

- ***faMeshAdaptor***: the backend for *faMeshInput*. The output is a multi-block dataset with one block per area mesh. Each block is further divided in pieces for each processor.
- ***CloudAdaptor***: the backend for *cloudInput* for converting an OpenFOAM cloud to vtkPolyData. The output is a multi-block dataset with one block per cloud with pieces from each processor.

The *inputs* are specified by the user in the Catalyst dictionary. For each input the user can also specify its options, e.g. the fields and the regions for *fvMeshInput*, see the snippet code in Figure 1.

The *Catalyst Python scripts* are specified in the catalyst dictionary too. The Catalyst Python scripts can be generated automatically using the ParaView GUI with the Catalyst Script Generator plugin enabled [6]. An example of a rendering pipeline result for a laser melting simulation is shown in Figure 3.

```
// ParaView Catalyst function object for OpenFOAM (-*- C++ -*-)

catalyst
{
    #includeEtc "caseDicts/insitu/catalyst/catalyst.cfg"
    scripts
    (
        "<system>/scripts/slice.py"
    );
    inputs
    {
        region
        {
            // All regions
            regions     (".*");

            internal    true;
            boundary    false;

            // Selected fields (words or regex)
            fields      (T U p);
        }


        // Solid walls only
        walls
        {
            internal    false;

            regions     ( heater "(?i).*solid" );
            patches     ( "(?i).*solid_to.*" "heater.*(Air|Water)" );

            fields      (T);
        }
    }
}
```

Figure 1: example of a catalyst dictionary for *fvMeshInput*

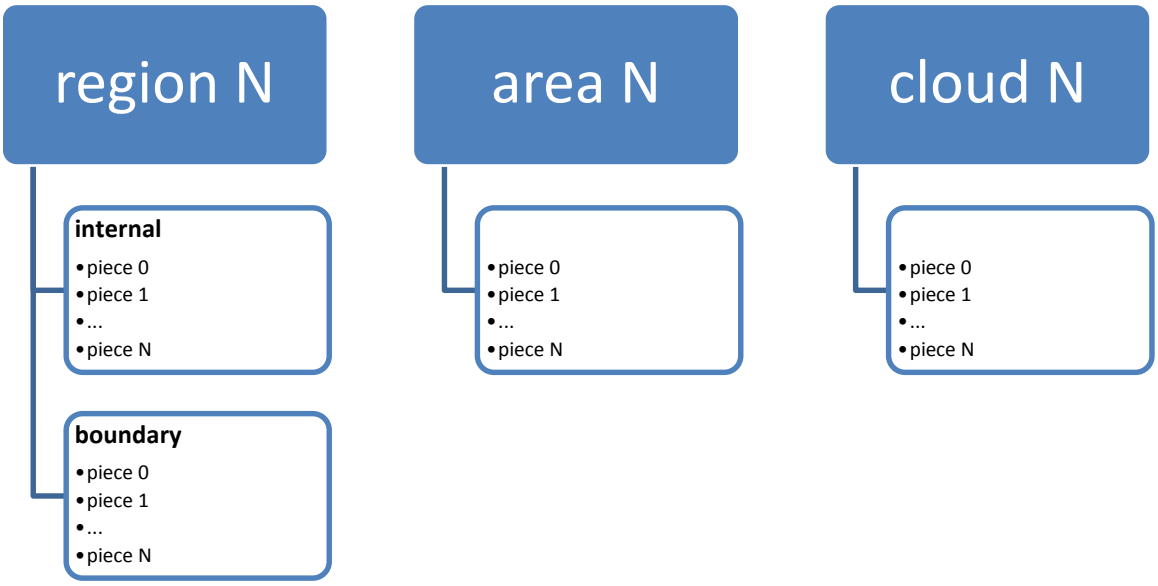

Figure 2: Structure of the inputs available in the Catalyst plugin

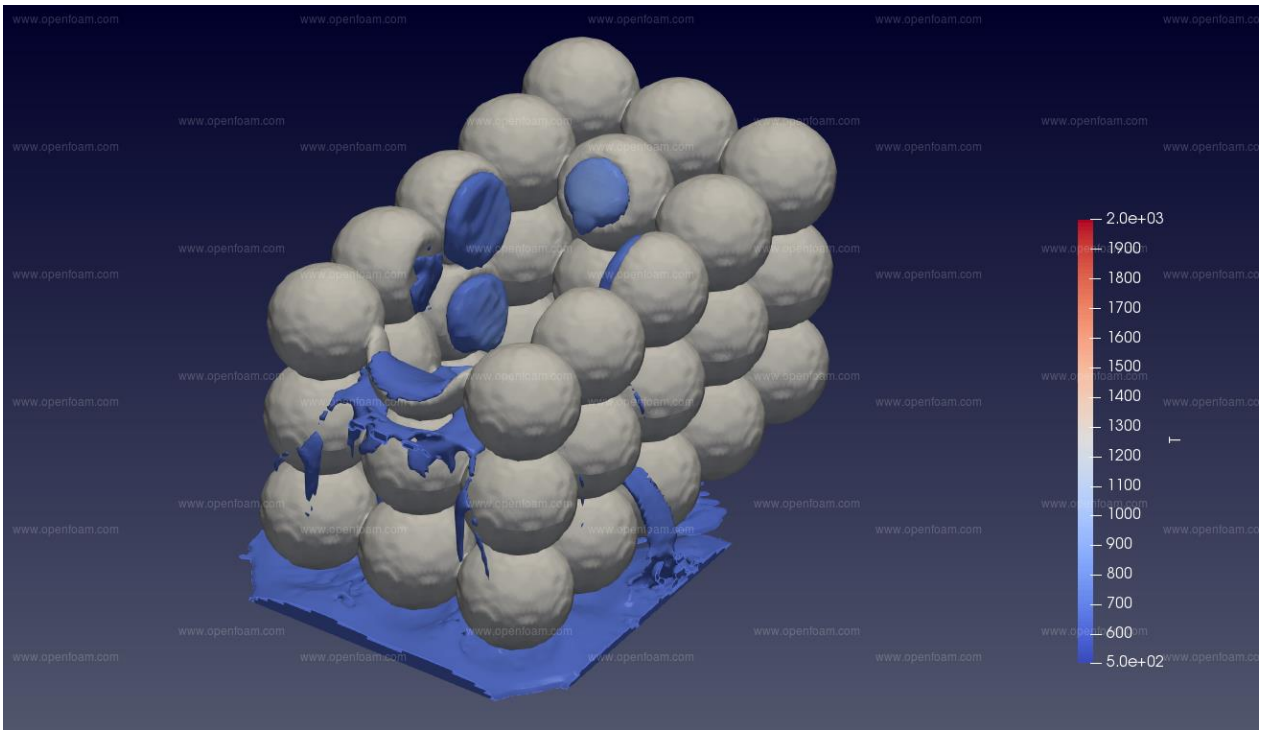

Figure 3: Laser melting rendering generated using the Catalyst plugin

References

[1] Klasky, S., et al.: In Situ Data Processing for Extreme Scale Computing. In: Proc. Conf. Scientific Discovery through Advanced Computing Program (SciDAC '11), (2011)

[2] Ayachit, U., et al.: 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, pp. 25–29, ACM (2015)

[3] Kitware: ParaView (2002–2016). http://www.paraview.org

[4] Esi-CFD: OpenFOAM (2018). https://www.openfoam.com/

[5] Esi-CFD - Cineca: Catalyst (2018). https://develop.openfoam.com/Community/catalyst

[6] Kitware: ParaView Catalyst User's Guide. https://www.paraview.org/files/catalyst/docs/ParaViewCatalystUsersGuide_v2.pdf