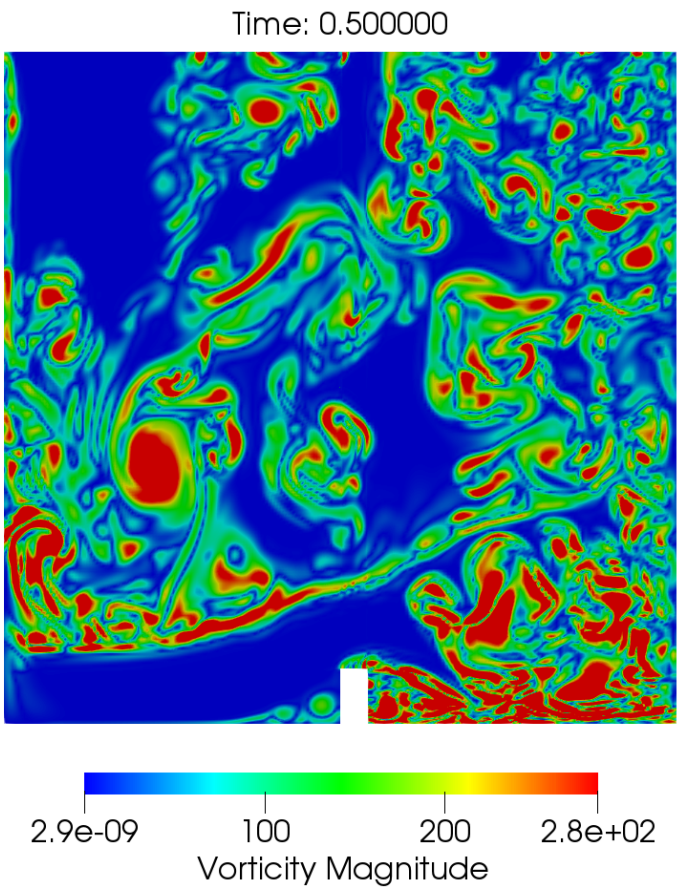
Ever seen a postprocessing result, before completion of the run? You've come just to the right place.

Instant visualisation of CFD data with OpenFOAM

Atul Singh



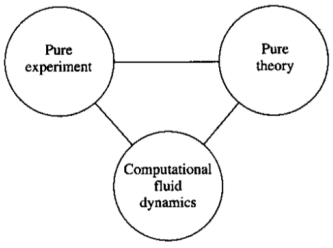
CFD simulations are both memory as well as communication intensive. The situation is even grimmer when supercomputers are involved as they anyways do more communication between processors. For such a task, In-Situ visualisation offers a huge incentive as it saves a lot of memory space, by generating the visualisation while the simulation is running. This project aims to combine these In-Situ capabilities of Paraview with CFD capabilities of OpenFOAM, which is also an opensource software. Hence empowering a wide range of researchers i.e from students to companies in their CFD research.

The image seen above is not a Vincent Van Gogh painting, but is a only a small result of the ginormous visualisation capabilites our modern day supercomputers have.(So can this be termed as Modern Art?) The image, shows free fall of breaking of square section of water, (say, dam) under the infulence of gravity, obstructed by a small protrusion at the bottom, at 0.5 s of the flow.

What bother with CFD?

Consider a flight taking off from an airport. Why do you think, the next one waits a little bit after the previous one is already long gone in the air?. The answer is simple. The previous flight has left a huge whirlpool of air behind it, strong enough to make the next plane wait in line. A following good question would be how does one know when will these whirlpool of air diminish?, or rather, how weak of a whirlpool strength, would be good enough for the next plane to take off?

It is questions like these that underline the importance of a CFD study. Knowing the answers of which could lead to a correct take off time for your next flight. Well that is of course, if Airport support staff doesn't harass you with something else.



The three pillars

In short one could just think of CFD simulations or rather simulations, in general, as tools that provides the means, which the complexities of theories and experiments aren't able to. And this is obviously not limited to Aerospace applications as can be seen in Fig 1.[2] These range from, but not limited to, Automotive, Health research, HVAC, Chemical processing , Sports, Marine applications, Power generation, etc

Why are HPC's involved?

When we talk about CFD simulations, they are usually a set of very long equations that are run to obtain some variable of interest, say pressure, or velocity for instance. These equations could be thought of as similar long equations that you would enter in a calculator to calculate, lets say some variable 'x'. Only slightly more complex, sometimes coupled, and quite often dependent on previous iteration.

The calculators do it on a very small scale, while the supercomputers do it on a think-as-big-as-you-possibly-could, ginormous scale. Think not only CFD simulations, but even those involving the supernova explosions and galaxy studies are performed with the help of Supercomputers. Surely they must be useful. Very very useful.

The handshake between the two

So now, we know CFD are basically a set of calculations, which HPC help them

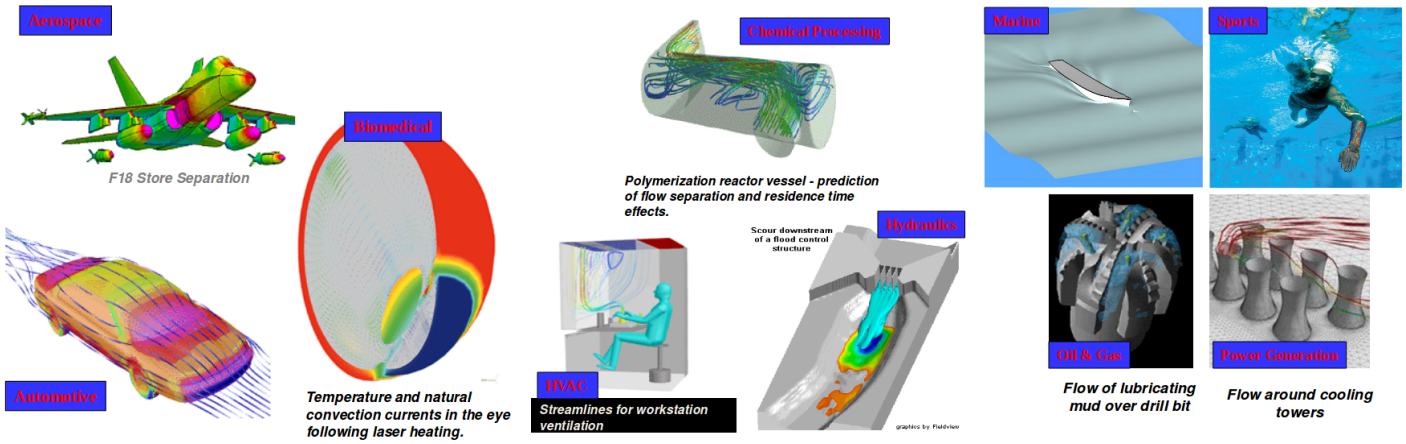
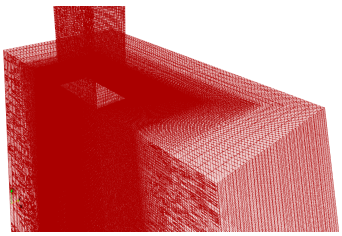


Figure 1: Some applications of CFD

calculate. The way this is done by employing by one of the oldest trick in the book, for whatever the problem may be. "Divide and Rule". Lets see how.



The divisions[3]

The structured lines that you see above in this building like structure is called a mesh. The square or regular shaped volumes formed by these intersecting lines are called cells. These cells are the exact places where our defined equations for a variable of importance will be run. The more number of cells you have, the better approximations your simulations will give. (The denser region in the above picture). Hence, we simply divide the problem into cells, and rule it with whatever equations we want to solve.

So what's the catch?

Well, its time, another oldest ally in the book. How?, Lets just say the divisions or cells that you see from the previous image are in order of millions, and this is quite often commonplace in a CFD study, and add to the the fact that you are running a very complex mathematical equation on each of these cells, which have several unknown variables. It is bound to be time consuming to get to your results.

And when you do, you need this "post-processing" software known as Paraview that helps you see and observe the behaviour of the variable you wanted to study. Not to mention, loading the data to this software after your simulation completes in 3 days (say) will also be a slow process, because there will be loads of data and any software of course has limitations. Which is a traditional way of post processing or rather visualising the results.

What do we do then?

This is where my project comes to play. Want to observe only pressure, then use only pressure as a variable and not anything else. Seems logical, right? Don't observe all the variables, when you only need couple of them. This is exactly what we do inside OpenFOAM, (the software responsible for having those long CFD equations and solvers).

Paraview Catalyst is one such library that takes care of such a handshake between OpenFOAM and Paraview, while the OpenFOAM solver is running. The way Catalyst works, is , lets say if you have only pressure to visualise,then, catalyst lets you define only pressure as a variable.

Now if we are to combine OpenFOAM with Paraview to instantly visualise our results, it would make sense if we only took a part of features that Paraview has to provide, and not the complete software. Which is exactly what we do. Paraview has something called filters, that helps you "filter" out only specific information from the complete results that your simulations produced. But in Catalyst however, we in a way customize Paraview according to our specific needs. If, lets say, you only want to see a slice of the geometry, which shows velocity flow in 2D, you can only select those two parameters. You can then generate a python script using the Paraview-GUI, and define the parameters that saves the image in a seperate folder.

Why is this helpful?

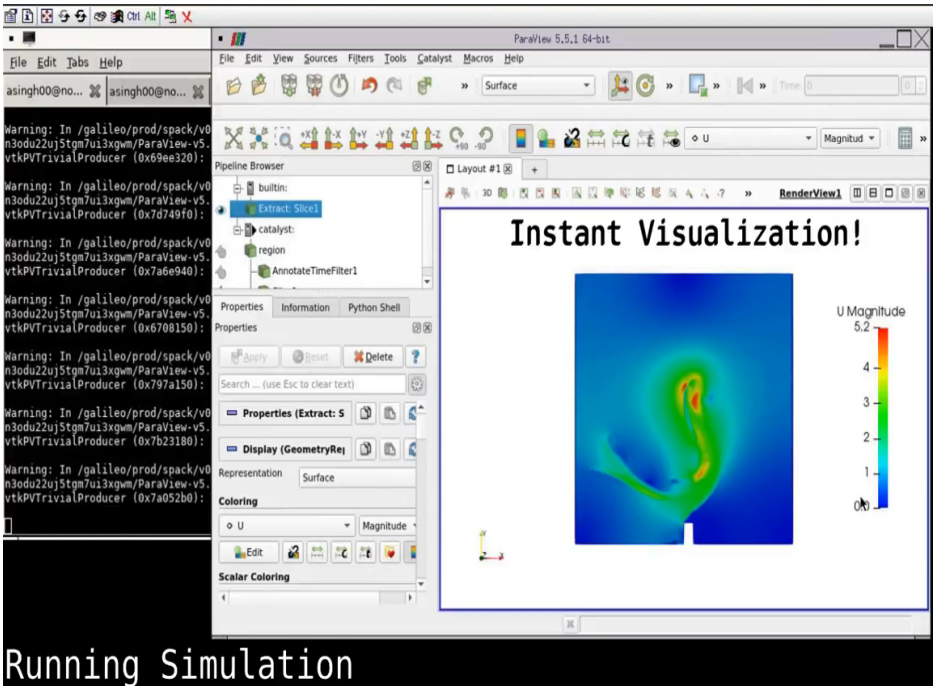
For one, It makes complete sense to use only those filters that are necessary for your results, compared to everything that Paraview has to offer. What this does is it saves memory, as every processor involved will be running the OpenFOAM solver. Having a smaller version of Paraview can make the visualisation code just like one more added calculation for OpenFOAM. Which is much more feasible. This is exactly what happens when we have included a function object as an interface in OpenFOAM to use Paraview.

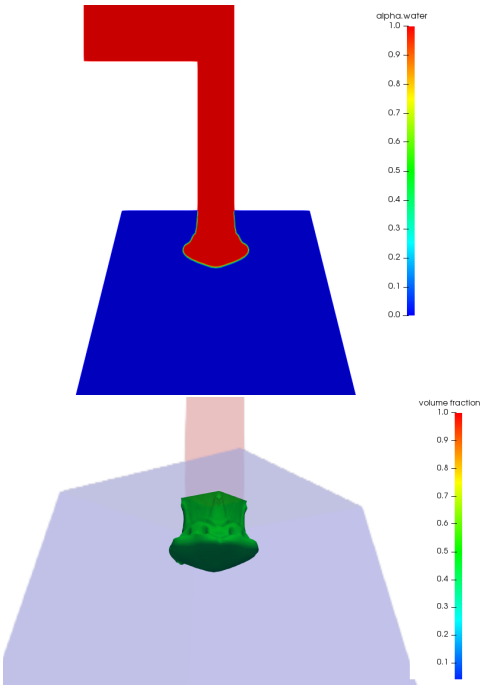
Let's test it

There are basically two steps involved

- The Script obtained from Paraview GUI[5]
- The catalyst file added with the above script

The first step after you have decided onto what visualisation would you want to see is fairly simple. All you need to do is load the Catalyst plugin. This gives a wizard of sorts that helps you generate a script specific to all the filters that you used.
The second step involves, adding a function object to OpenFOAM files, that tells it to process the above generated script





Left: The snapshot when catalyst is running with OpenFOAM. Top Right: Simplified injector tested with Catalyst,showing 2D slice of volume fraction.[3] Bottom Right: Injector with iso surface for the same volume fraction[3]

within its framework. (An example can be seen from this github repository[4]

And finally

The images seen above are the first hand results when Catalyst is being run. It can be seen when the simulations is running , the pipeline browser of Paraview (the place with the smallest blue rectangle are in above picture) has loaded the data that is defined in the script. So it is quite appealing to see the visualisation along with the running CFD simulation.

The script also defines a seperate folder where all the images can be saved. These images are helpful when one wants to make a small movie illustrating how the flow develops during the course of the flow,[6] as is often the case for a CFD simulation. Remember, this was not possible before Catalyst.

Although Catalyst is a briliantly employed solution, it is still under development. The current testing done at Cineca Supercomputing center, under this project was the first instance where it was tested with an Exascale OpenFOAM case.

Of course, the current edition of Catalyst is not without its limitations and the documentation regarding building a custom Catalyst edition, specific to different needs, is still missing. It is also not a trivial task to build and recompile everytime an addition has been made to Catalyst. So there remains lots and lots of scope for improvement.

References

[1] Anderson, J. (2010). Computational fluid dynamics. New York, NY: McGraw-Hill.

[2] https://www.iitk.ac.in/tkic/workshop/FEM/ppt/Introduction

[3] Edelbauer, W. (2017). Numerical Simulation of cavitating injector flow and liquid spray break-up by combination of Eulerian-Eulerian and Volume-of-Fluid methods. Available at: http://www.elsevier.com/locate/compfluid [Accessed 31 Aug. 2018].

[4] https://github.com/Atulsingh92/SouCase-OpenFoam/blob/master/system/catalyst

[5] Paraview Catalyst User Guide

[6] https://www.youtube.com/watch?v=r-u9NJc-pJIlist=PLhpKvYInDmFXUyp_pWBM-h1NCD6GEUfgpindex=9

PRACE SoHPCProject Title

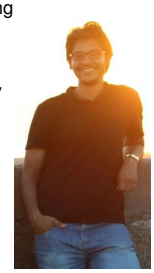
In Situ visualization of CFD data using OpenFOAM.

PRACE SoHPCSite

Cineca Supercomputing Center, Italy

PRACE SoHPCAuthor

Atul Singh, University of Rostock, Germany. contact: atulsingh92@outlook.com



Atul Singh

PRACE SoHPCMentor

Dr Fedrico Piscaglia Politecnico di Milano, Italy

PRACE SoHPCSoftware applied

OpenFOAM 5.0, Paraview 5.5.1, Catalyst, Blender 2.79, Audacity.

PRACE SoHPCMore Information

SummerofHPC website , Paraview website

PRACE SoHPCAcknowledgement

Sincere thanks to Dr Fedrico Piscaglia and Special thanks to Simone Bna. Honorable thanks also to Paola Alberigo, Francesca Delli Ponti, Giuseppa Muscianisi, Massimiliano Guarassi, Luigi Calori, Ivan Spisso, Christiano Padrin and Luis Sanchez. Default thanks to my Parents.

PRACE SoHPCProject ID

1809