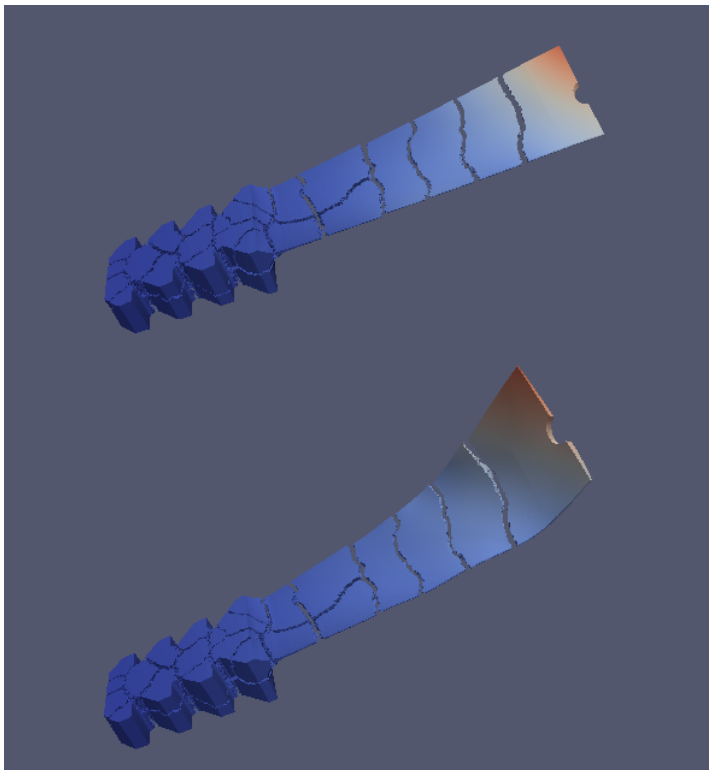
ESPRESO Solver for SoHPC

Anthony Bourached



We adapt the code of the **E**xa**S**cale **Pa**ra**R**allel F**E**TI **SO**lver (**ESPRESO**) in such a way that it is linked with Paraview Catalyst during simulation runtime to produce real time, *In Situ*, analysis and visualisation of the simulations. Furthermore, we use IT4Innovations supercomputer, Salomon, to produce visualizations of a fan blade in real time.

This project is based on the visualisation of the results from **E**xa**S**cale **Pa**ra**R**allel F**E**TI **SO**lver (**ESPRESO**). The Finite Element Tearing and Interconnecting (FETI) method is a practical and efficient domain decomposition (DD) algorithm for the solution of numerical partial differential equations. Domain decomposition methods solve boundary-value-problems[1] by splitting them into smaller boundary value problems on subdomains and iterating to coordinate the solution between adjacent subdomains. The problems on each subdomain are independent, this means that we can effectively solve them at the same time. This makes domain decomposition perfect for parallel computing, for a comprehensive explanation of why this is we must discuss the theory of high performance computing.

High Performance Computing

High Performance Computing (HPC) generally refers to the practise of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer. A powerful modern desktop computer typically has eight cores. The purpose of multible cores is to maximise efficiency of the computer's compute power. It allows the computer to more effectively perform different tasks at the same time.

Maximum program efficiency would be achieved if the program's tasks were divided in such a way that they can be executed by independent cores with, preferably, minimum communication or co-dependency. This practise is referred to as parallel computing. One of the massive aspects of HPC is the use of Supercomputers: massive, highly maintained clusters of computing nodes. Simulations in this project have been produced on IT4Innovations supercomputer, named Salomon.

Parallelism in ESPRESO

From figure 1 we can see how the use of Parallel programming is used to most effectively solve problems with ESPRESO. This is a simulation of a steal block, we shall discuss what ESPRESO is doing with it but first lets discuss its appearance. The gaps that divide the block into many pieces do not physically exist in the structure but show us the domain decompostion of the problem. Each subdomain may be solved in

[1] A differential equation with a set of constraints called the boundary values. A solution to a boundary value problem is a solution to the differential equation that also satisfies the boundary.

parallel.

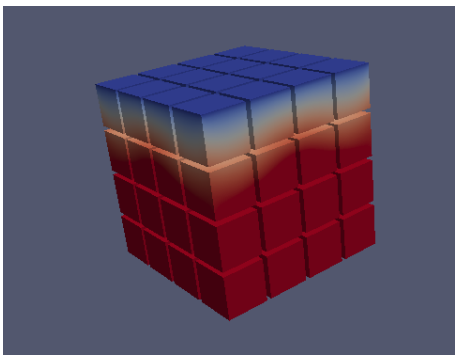


Figure 1: Steal block. Gaps represent the boundary of different subdomains which can be solved independently.

The main object attribute visualised in this project is displacement. We may consider displacement in this context as the distance of each given point of the structure from its original position. Even a rigid structure such as a steal block deforms under force, such as gravity, though this is often far too slight to be detected by the naked eye. However, using Paraview we may warp vectors. This means that we can emphasize the magnitude of that attribute without changing anything else about the structure or appearance of the object. For example, warping displacement or velocity by a factor of two will make the dsiplacement appear double of what it actually is or make velocity twice as fast as it actually is. For our simulation of this steal cube we warped displacement by 4^{12} (4,000,000,000,000).

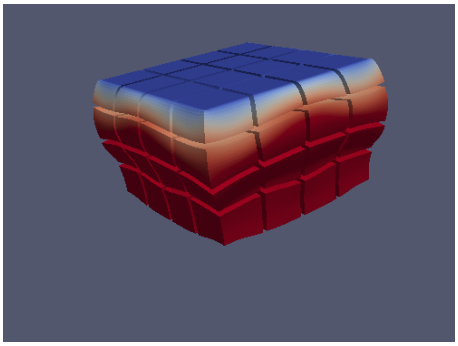


Figure 2: Steal Block. This is the exact same image as in figure 1 (note the colours) except that displacement has been warped by a factor of 4^{12}.

We can see the steal cube squashed under gravity.

In Situ Visualisation

The goal of visualisation is often to find desired/important details within a large body of information. As well as getting a general understandning of the structure and attributes of an object on data sets that are far too large and complex for the human brain to comprehend without such tools. Paraview is often used for such visualisations and is used in this project.

We use Catalyst[2] and Paraview to perform *In situ*[3] analysis. It is common to require that our simulations discard most of the data created in order to maximise efficiency. Since, in this case, it is not possible to store data for many simulations, data analysis and visualisation must be performed *in situ* with the simulation to ensure that it is running smoothly and to fully understand the results that the simulation produces. This is opposed to the traditional workflow where simulation would be ran and data outputed to disk and then finally analysed and visualised (using, say, paraview).

There are many advantages to *in situ* analysis. First, we can begin the analysis and visualisation process without needing to do any input/output of simulation results- this is an especially important point as simulation power is advancing at approximately ten times the rate of input/output efficiency. Second is that rather than performing our visualisation/analysis on another machine we have the full computational power of the supercomputer available to do this processing. Third, we expect that the the data produced by the analysis/visualisation will be much smaller than that produced by the simulation itself.

For these reasons I believe that co-processing will have a massive role in the future of HPC and thus was great motivation for this project.

Method

My task was create the API (Application Programming Interface) for the ESPRESO that enables it to run with Catalyst and hence visualise the results in real time.

Before I started my project, ESPRESO only enabled post-processing; It output results to a vtk file. **V**isual **T**ool **K**it (vtk) is a file type which is readable by Paraview. Two important data types give the structure of the object: 1) Points: we must have an array of points which outline the shape at, for simple cases, uniform intervals. Each point has three associated values: its x, y and z coordinates. 2). Cells: these will be the most fundamental element of the image.

So far this gives the structure of the object. Furthermore, ESPRESO records attributes, such as displacement which is what we have focused on in our visualisations.

To perform our real time visualisation all these values had to be passed from ESPRESO to Paraview Catalyst by an Adaptor code which constituded the most significant part of the API.

ESPRESO is written in c++ so we also used this language to write the Adaptor code. The structure of the object is unchanging so the Adaptor code could be configured such that we only need to build the grid points and cells on the first iteration, hence only requiring for the values of displacement to be updated at each timestep. This ensured that communication between ESPRESO and Catalyst was as fast as it could be. Already this means that we save hugely on the amount of data communication required each timestep as Catalyst keeps the object structure in its memory.

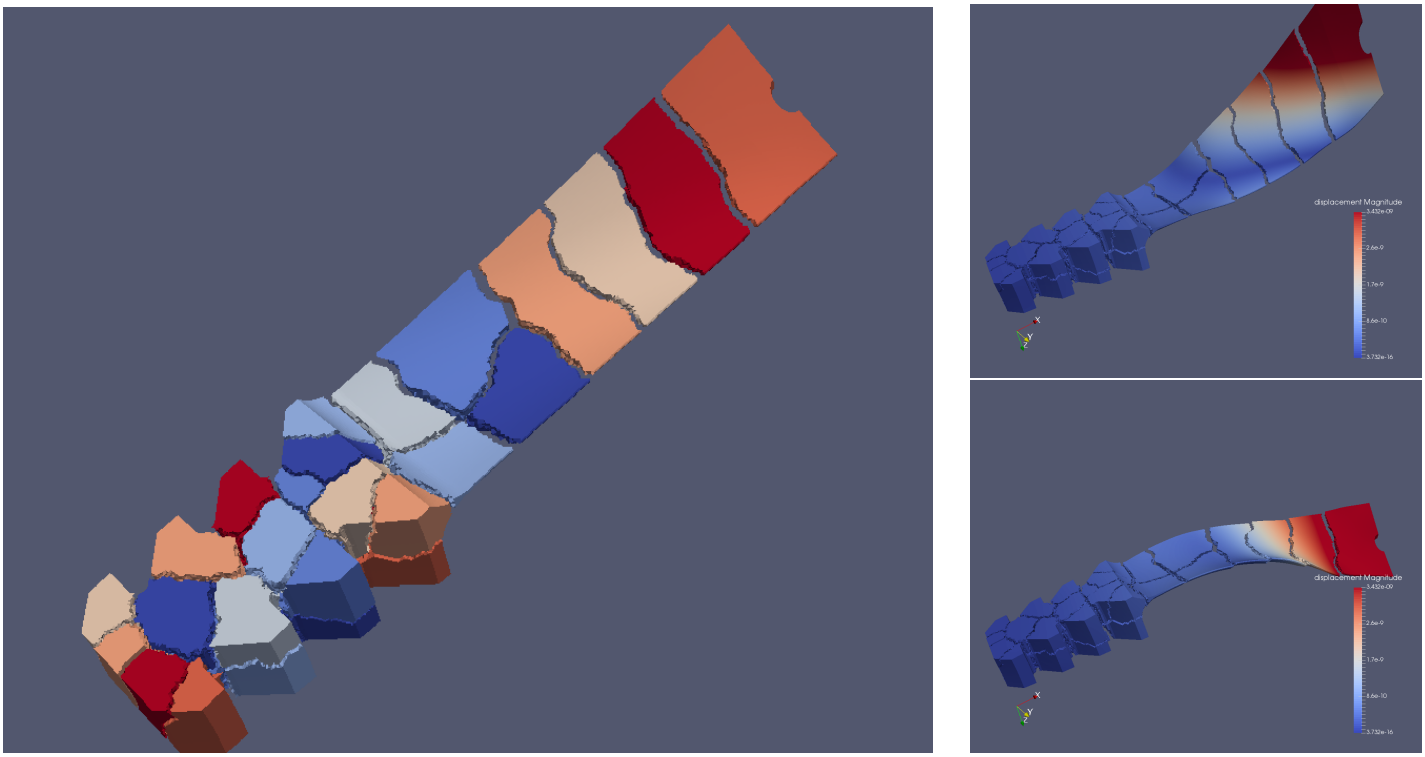
When images are rendered in Paraview from a vtk file one usually uses Paraview to show the exact aspect of the data that is required. For example, one might like to slice the object to show a transverse image of its interior. These tools are infinitely useful and use of them is one of the main motivations for visualisation. A python script was written/exported by Paraview Catalyst which enables us to render a real time image with all our specified Paraview configurations. One of the specification of this script was to save a png of each timestep which we later used to make avi videos.

Results and Discussions

As the main focus of the project was to create the API for ESPRESO and Catalyst; results are visualisations produced by the ESPRESO. The images shown in

[2]Catalyst is a 'light-weight' version of the ParaView server library that is designed to be directly embedded into parallel simulation codes to perform in situ analysis at run time. Esentially it enables us to use the same analysis and visualization tools that are already available for post-processing using Paraview.

[3]*In situ* is Latin for 'on site' or 'in position'. In this context *In situ* analysis refers visualisation and analysis as the simulation is running. This form of analysis can also be refered to as co-processing or co-visualization.



Similar to the cube, the gaps that look like cracks in the fan blade do not physically exist but show the domain decomposition. This decomposition is emphasized by colour coding in the large image. Displacment is warped in the other two images by ten billion (10,000,000,000). This makes it look like a flag blowing in the wind. These are timesteps 282 and 421 respectively. The first timestep also illustrates the what the fan looks like in real life (with no warped attributes).

this report were rendered in real time-while the simulation code was running. From these visualisations we saved images for each timestep and later compiled them together to make avi (Audio Video Interleave) videos.

Our main visualisation was a fan blade as can be seen above. The large image shows the colour coded decomposition of the structure.

The title image shows two identical similar images. The one on top is a fan blade with a colour scheme that illustrates each points displacement from the idealogical shape of the object. This displacement is far to slight to see so the only evidence of it in this image is the colour. The image immediately below, however, is the exact same image but with displacement warped by 10^{10} (10,000,000,000). This second image therefore enables us to see what the colour in the first image is telling us about the data.

For simulations that produce large amounts of data, Co-processing is becoming a necessity. The use of scripting, python was used in this project, makes it possible to perform co-visualisation without any loss of functionality of Paraview. This means that we can 'have our cake and eat it too' in terms of benefitting from the speed and convenience of co-visualisation/co-analysis while still using all the features of Paraview.

PRACE SoHPCESPRESO Solver

ESPRESO API for ParaView Catalyst to perform In situ Analysis.

PRACE SoHPCIT4Inovations

IT4Innovations,

PRACE SoHPCAuthors

Anthony Bourached, [association,] Czech Republic

PRACE SoHPCMentor

Riha Lubomir
lubomir.riha@vsb.cz

Anthony Bourached-photo

PRACE SoHPCContact

Leon, Kos, Univerza v Ljubljani
Phone: +12 324 4445 5556
E-mail: leon.kos@lecad.fs.uni-lj.si

PRACE SoHPCSoftware applied

Paraview, Paraview Catalyst

PRACE SoHPCMore Information

www.paraview.org/in-situ/

PRACE SoHPCAcknowledgement

I would like to extend my gratitude to my mentor Riha Lubomir for many useful meetings and guidelines throughout the project. I was also like to thank Alexandros Markopoulos for helping to explain the structure of ESPRESO.

PRACE SoHPCReferences

[1] Andrew C. Bauer, Berk Geveci, Will Schroeder Kitware Inc. February 2015. The Catalyst Users Guide v2.0 ParaView 4.3.1

[1] Andrew C. Bauer, Berk Geveci, Will Schroeder ...et al The Paraview Guide, A Parallel Visualisation Application *Kitware Inc.*