# D5.2.3 – Post-processing: first prototype tools for exascale interactive data exploration and visualisation

*WP5: User tools*

| | |
|---|---|
| **Project Acronym** | CRESTA |
| **Project Title** | Collaborative Research Into Exascale Systemware, Tools and Applications |
| **Project Number** | 287703 |
| **Instrument** | Collaborative project |
| **Thematic Priority** | ICT-2011.9.13 Exascale computing, software and simulation |

| | |
|---|---|
| **Due date:** | M18 |
| **Submission date:** | 28/02/2013 |
| **Project start date:** | 01/10/2011 |
| **Project duration:** | 36 months |
| **Deliverable lead organisation** | DLR |
| **Version:** | 1.0 |
| **Status** | Final |
| **Author(s):** | Fang Chen (DLR) |
| **Reviewer(s)** | David Lecomber (ASL), Stephen Booth (UEDIN) |

| Dissemination level | |
|---|---|
| PU | *PU - Public* |

# Version History

| Version | Date | Comments, Changes, Status | Authors, contributors, reviewers |
|---|---|---|---|
| 0.1 | 28/02/2013 | First version of the deliverable | Fang Chen (DLR) |
| 1.0 | 20/03/2013 | Final version of the deliverable with review comments considered | Fang Chen (DLR) |

# Table of Contents

# 1 Executive Summary

This deliverable is a software deliverable, providing a software prototype for exascale interactive data exploration and visualisation. The main purpose of this associated report is to present and document the prototype software, which has been developed using a co-design process with the HemeLB code.

This software build on two previous deliverables associated with Task 5.2 within Work Package 5 (D5.2.1 and D5.2.2). These described and studiedthe challenges, system requirements, system architecture and data structure for exascale data post-processing. The first two deliverables served as a theoretical foundation for the upcoming software development and design.

In accordance with the previous deliverable, this software aims to provide in-situ processing of the simulation data, interactive visualisation for exascale CFD simulations, and further computational steering capability of the on-going simulation. In-situ post and interactive visualisation provides the user with the possibility to explore the simulation result on-the-fly, while computational steering allows the user to change and modify an on-going simulation process by modifying simulation parameters.

In this deliverable, we deliver a software prototype which was co-designed and integrated into HemeLB. This prototype provides a fundamental structure of interactive data post-processing for HemeLB, allowing developers to evaluate the design of our proposed post-processing system and visualisation algorithms. We present an initial attempt to visualise a HemeLB simulation with a newly implemented visualisation and steering client. We also outline future plans and on-going work regarding software implementation.

# 2 Introduction

Today's Computational Fluid Dynamics (CFD) software deals with complex geometries. Such simulations output numerical data on an extreme scale. For such extreme simulations it will no longer be possible to store simulation data on disks for post-processing. Therefore, in-situ data analysis and computational steering becomes important in exascale data post-processing. These two major concepts provide the simulation experts with the possibility to inspect their simulation results at run-time, and to further modify and change the on-going fluid simulation.

HemeLB is one of the important applications within CRESTA. This code aims to develop algorithms for blood flow simulation. In a co-design process, with the core HemeLB group, our software focuses on developing post-processing tools tailored for HemeLB, providing the simulation experts with the possibility of inspecting the on-going simulation processes, and to further steer simulation parameters.

Three major goals exist in the development of our post-processing software. The first goal is to provide suitable visualisation of the on-going simulation, visualisation that is able to handle huge data sets at the interactive frame rate. Our second goal is to develop rendering architectures that maximize the performance of current available hardwares, in the sense of visualisation computation. Finally, a steering client should be available that allows the user to modify the on-going simulation with the visualisation output.

In this report, we present the current software architecture, as well as the first prototype of our software. We list the goals and implementation tasks for the on-going software development in section 6. Minimal, but necessary, information on how to test the prototype is provided in section 7.

The actual software is available on the CRESTA SVN service.

# 3 Project History

We have previously delivered two related deliverables (see Table 1). These served as the theoretical basis for our research, while this current deliverable and associated report provides a first software prototype that demonstrates our software development practice in data post-processing.

**Table 1 Main functions of the three delivered work packages**

| Deliverable | Main function |
|---|---|
| WP5.2.1 | Identify problems and challenges in exascale post processing |
| WP5.2.2 | Define system algorithms and data structures |
| WP5.2.3 | First software prototype developed in a co-design process with HemeLB for post processing |

Deliverable 5.2.1 [1] studied the challenges in post-processing at the exascale. It provides a foundation and set of guidelines for post-processing experts in the design of these post-processing algorithms.

The second deliverable [2] provided a detailed study of interactive visualisation tools and data structures. It highlighted the importance and necessity of applying in-situ and interactive visualisation for exascale simulation processes. This deliverable also established a series of co-design tasks with HemeLB.

This deliverable (D5.2.3) is the first software prototype developed by DLR in collaboration with the HemeLB team. It acts as a demonstration of the proposed algorithms and system architectures that were presented in the previous deliverables.

# 4 Software Architecture

The current architecture of the prototype software is shown in Figure 1.

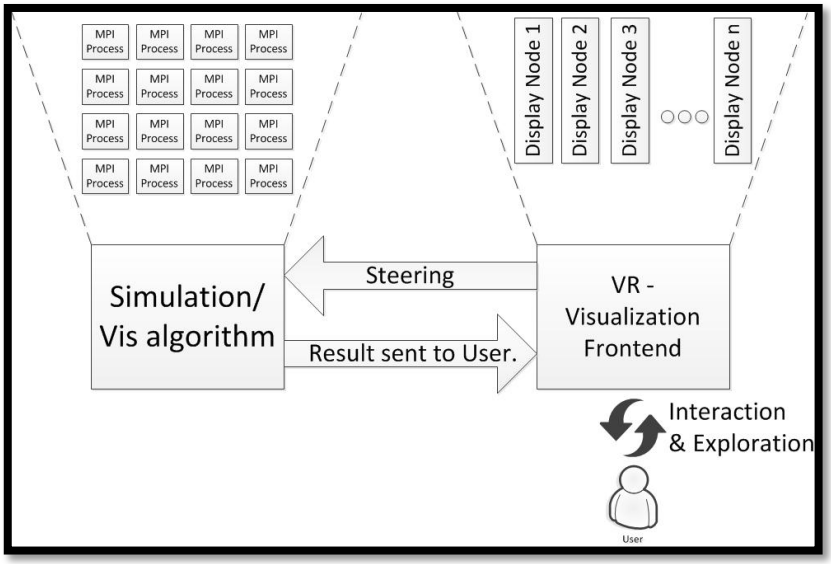


**Figure 1 Software architecture.**

In this framework, the simulation and visualisation algorithms share the same cluster. Simulation geometry is divided into blocks and distributed among nodes. While the current simulation step is done, a desired visualisation is applied on the same block of data and produces an image data. Then, all the images are collected by the master node and presented at the front end.

The main HemeLB code is the simulation computation, which remains unchanged. Our post-processing extends and modifies the visualisation and steering tools associated with the HemeLB code. We aim to maintain the visualisation data structure as well as the parallel computation structure of the original HemeLB code. Minimal changes are made to the steering client in order to achieve a better interactive and in situ visualisation.

A large part of this software is aimed at designing and implementing suitable interactive visualisation algorithms that aid the simulation experts in exploring the on-going simulation. Interactive visualisation involves visualisations of the simulation data being computed and rendered at interactive frame rates. In large-scale fluid simulation, it is crucial to provide interactive visualisation such that simulation experts can inspect and study their simulation process at run time.

Interaction with users can be incorporated together with virtual reality (VR). Improved depth perception and free navigation in VR allow the scientist to explore their data in a more natural and informative way. As a long term goal, we also plan to include the use of VR in our post-processing.

Computational steering is the ultimate goal of this software project. It is not only a challenge for post- processing, but also for Task 5.1, pre-processing. Both of these two tasks will work together to enable simulation experts to steer the computational mesh and parameters based on the visualisation results.

Further details of the system architecture is published in [3]

# 5 First Prototype

In this prototype, we have implemented a cut-plane example of the on-going simulation. Installation and compilation of the prototype is presented in the next section. In this section, we describe the functionality and structure of the prototype.

The current HemeLB code already provides a volume renderer of the flow fields, such as velocity magnitude, stress magnitude, etc. On top of the given volume renderer, we have implemented a cut plane example. A cut plane is a user-defined plane which cuts through the dataset. Fluid properties of the intersecting cells will be presented on a pixel-based image on the left top corner. The image will be sent together with the volume rendering pixels as one image to the front-end. With the help of cut planes, geometry cells that are hidden behind the walls can be revealed. It provides the user with the possibility to look 'inside' the blood vessels.

Figure 2 demonstrates the image output of a cut plane visualisation on a test dataset. This graphical window shown in Figure 2 shows the in-situ visualisation for the Regression/*difftest* simulation. While the simulation is still running, the user can trigger the steering client. The steering client will connect to the current simulation time steps, and extract the visualisation output with image transfer. A user-defined cut plane demonstrates the simulation results of the geometry cross-section in an interactive manner. In this test simulation, a blood flow is injected from the bottom of the tube, and the simulation continues until the user pauses it. The four figures shown in Figure 2 represent the stress magnitude of the flow field at different time steps. In this prototype, we have inserted a cut-plane which is orthogonal to the main flow direction. The image shown on the top left corner indicates the stress field projected onto this intersecting plane.

Currently, we have a nearest neighbourhood implementation of the intersecting cell search. We loop over all pixels on the cut plane window, then find the nearest site point on the geometry, extract the field value, and map to a colour. Further implementation of stochastic cut plane sampling [4] and estimation will accelerate the cut plane computation. This approach will accelerate the search of cell locations while computing the intersection between the plane and the geometry.

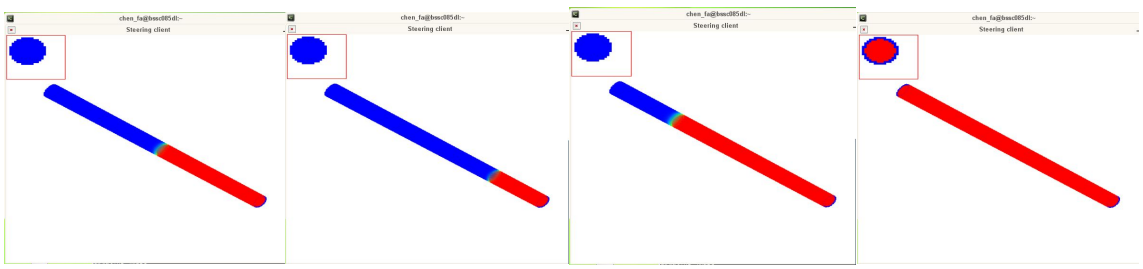

**Figure 2. The prototype of cut plane visualization.**

# 6 On the To-Do List

The current prototype of our post-processing tool consists of a simple in-situ cut plane visualisation of the running HemeLB simulation. In order to achieve interactive post-processing and full steering functionality, there are a number of items on the to-do list.

We divide these goals and targets according to their functionality in the post-processing pipelines, namely steering, visualisation and user interaction. We list the tasks as follows:

| Category | Feature, task | Priority |
|---|---|---|
| Steering | Select multiple time steps at a time | Medium |
| | Select a region of lattice | High |
| | Set a parameter for steering | High |
| | Send the modified parameter back to Simulation | High |
| | Modify simulation according to new parameters | Low |
| | Connect visualization with steering, define a interface | Medium |
| | Define criteria for modifying simulation | Medium |
| Software achitecture | Incorporate multi-resolution data structure | Medium |
| | Try hybrid rendering | Low |
| | Test remote rendering | Low |
| User interaction | Allow user to defline cutplane location | High |
| | Combine virtual reality into application. Powerwall, display all. | Low |
| | Explore the massive data exploration metaphor on powerall | low |
| Visualization | Implement stochastic sampling for cut plane points | Medium |
| | Fix streak/line code for hemeLB | Medium |
| | Compare different image streaming techniques | Low |
| | Study other suitable visualization tools for hemeLB | Medium |

# 7 Setting up and Running the Prototype

This section gives a minimal description of how the software prototype works. It includes the basic steps that you need to install or change, on top of the HemeLB source code, and the platforms and libraries you might need.

A few things should be checked before starting to install and run the software. The following tools or packages, preferably the latest versions, are required:

*Python-2.7.3*

*Python-argparse*

*PyOpenGL*

*Numpy*

*cmake*

A running version of HemeLB is required (For HemeLB source code, please contact University College London, The Centre for Computational Science).

Check out a working copy of HemeLB. Then build the dependencies with the following commands:

*cd hemeLb-dev/hemelb/dependencies*

*mkdir build*

*cd build*

*cmake.. –DHEMELB_USE_BOOST=ON*

*make*

Additional files or modified files are available at the SVN repository for Cresta*:* https://svn.ecdf.ed.ac.uk/repo/ph/cresta/wp5/postprocessing/trunk

Then replaced the following files with the new ones: (* means replace both .cc and .h files):

*steering.py in …/hemelb-dev/hemelb/Tools/steering/python/hemelb_steering*

*Rendering.* in …/hemelb-dev/hemelb/Code/vis*

*ResultPixel.* in …/hemelb-dev/hemelb/Code/vis*

*Control.* in …/hemelb-dev/hemelb/Code/vis*

Create this folder for cutplane example:

*mkdir ~/hemelb-dev/Code/vis/cutPlane*

add the following files into the …/cutPlane folder

*CutPlane.h*

*CutPlane.cc*

*CutPlanePixel.h*

*CutPlanePixel.cc*

Now it should be possible to compile HemeLB with the Cut plane visualisation. If you have already successfully built HemeLB source code, then you only need to do:

*cd hemeLb-dev/hemelb/Code/build*

*make*

If HemeLB has not previously been built, a build folder must be created first, followed by a cmake before the make, i.e.:

*mkdir …/hemeLb-dev/hemelb/Code/build*

*cmake ..*

*make*

A test simulation is started by:

*cd …/hemeLb-dev/RegressionTests/difftest*

*./diffTest.sh*

While the simulation is running, the steering client with the cutPlane visualization is started with:

*cd …/hemeLb-dev/hemelb/Tools/steering/python/hemelb_steering*

*python steering_gl.py –monitor –show localhost*

# 8 References


[1] C. W. Fang Chen, „Deliverable 5.2.1: Post-processing: analysis and system definition for exascale systems," 2011.

[2] F. Chen, „Deliverable 5.2.2:Post-processing: data format (hierarchical, multi-resolution) and algorithms definition," 2012.

[3] F. Chen, M. Flatken, A. Basermann, A. Gerndt, J. Hetherington, T. Krüger, G. Matura und R. Nash, „Enabling In-situ Pre- and Post-Processing for Exascale Hemodynamic Simulations − A Co-Design Study with the Sparse Geometry Lattice Boltzmann Code HemeLB," in *SC 2012, 10.-16. Nov. 2012, Salt Lake City, USA.*, Salt Lake City, 2012.

[4] C. u. G. A. u. H. C. u. H. H. Wagner, „Interactive In-Situ Online Monitoring of Large Scale CFD Simulations with Cut-Planes. Immersive Visualization Revisited: Challenges and Opportunities," in *IEEE Virtual Reality Workshop,*, Orange County, CA, USA., 2012.