D5.2.2 – Post-processing: data format (hierarchical, multi-resolution) and algorithms definition

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:	M12
Submission date:	30/09/2012
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)	Jussi Timonen (JYU), Stefano Markidis (KTH)

Dissemination level	
PU	*PU - Public*

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	31.08.2012	First version of the deliverable	Fang Chen (DLR)
0.2	04.09.2012	Document slightly revised	Achim Basermann (DLR)
0.3	26.09.2012	Reviewed version	Fang Chen (DLR), Achim Basermann (DLR)
1.0	28/09/2012	Final version of the deliverable	Fang Chen (DLR), Achim Basermann (DLR)

Table of Contents

1 Executive Summary

This deliverable focuses on the data structures and algorithms in post-processing exascale simulation data. We also present our co-design work with workpackage 6.

A general overview of data structures and post processing algorithms as they are applied to exascale data sets is presented. We elaborate and discuss the advantages and disadvantages of each algorithm before applying as an exemplar to the HemeLB application from workpackage 6.

Multi-resolution data structures and in-situ post processing are two key methodologies when working with exascale data. The basic principals of these two methods will also be applicable and related to task 5.3 of workpackage 5, which focuses on remote rendering techniques. Furthermore, task 5.1 (pre-processing) and task 5.2 (post-processing) propose to use similar data structures and partitioning in order to minimise the communication costs between cores.

Co-design with HemeLB has been established in this workpackage. Interface clients and visualisation algorithms are planned and designed which can be integrated into HemeLB as a post-processing tool. As a result HemeLB will benefit from elaborate and scalable visualisation and steering approaches that do not interfere with its specific Lattice Boltzmann simulation data layout. The choice of data structures and post-processing algorithms has been presented and compared in this deliverable.

2 Introduction

Interactive exploration and visualisation methods have proven to be successful in analysing large-scale, complex simulation data. The process of analysing simulation results, however, requires the computed raw data to be transformed to suitable representations by passing all or parts of the data through a post-processing pipeline, typically consisting of data extraction, filtering, mapping and visualisation stages. This is a time-consuming process making the requirement for efficient interactive exploration, such as the ability to move freely through the data, hard to meet.

In-situ, multi-resolution, level of detail, and region of interests are the keys aspects of interactive post-processing. The fundamental structures as well as algorithms provide the user with the possibility to reduce the data to be visualised, to obtain preliminary visualisation results on a coarser mesh, and to further allow result refinement with higher resolution data.

The aim of task 5.2 in workpackage 5 is to provide interactive visualisation tools to the on-going simulation process. A combination of different scalable techniques will be presented, such as in-situ processing and multi resolution data formats. In this document we present data structure definition and choices of post-processing algorithms.

This document gives a survey of data structures and post-processing algorithms for interactive explorative post-processing in an exascale environment. In section 3, we introduce hierarchical data structures and multi-resolution data formats. Section 4 describes the post-processing techniques that are suitable for exascale data. In section 6, we present the co-design work with workpackage 6.

3 Data structure

3.1 Multi-resolution

The explorative and interactive examination of data is always disturbed and interrupted by long waiting times. Multi-resolution data structures offer the possibility of reducing the processed data size and thus minimising total run time.

Multi-resolution data structures are often combined with context and detail approaches. A lower resolution data is normally used for context geometry and a higher one with more details. The approach allows the user to load a subset of the whole data in an initial step, inspect this subset, and apply further refinement on certain regions.

Visualising the simulation result with a lower resolution subset enables an immediate and preliminary graphical representation. A major benefit of presenting preliminary results to the user is the perception of a shorter total computation time. Even when the presented data does not yet convey useful information, it can improve the perceived level of interactivity. Integration of an interface to cancel a command or restart it with modified parameters almost approaches explorative analysis.

However, one can observe that within the context of interactive exploration even simplified, approximate results can be sufficient to decide whether to cancel the simulation, to change parameters for the next iteration step or to wait for the final result.

3.2 Hierarchical data structure

Data hierarchy is an important issue when handling distributed large-scaled datasets. Popular choices of hierarchical data structures are quadtrees and octrees. Each level on the tree corresponds to a set of data at a certain resolution. Effective ways of searching and traversal are main issues in using hierarchical data structures. Pascucci and Frank [1] introduced a new global indexing scheme, which accelerates adaptive traversal of geometric data with binary trees, which is also applicable to quad and octree trees.

The usability of hierarchical data structures also depends on the choice of post-processing algorithms. The partitioning and the hierarchy of data blocks should fit the requirements from the post-processing algorithm.

3.3 Region of interest

Another way of reducing data sizes is to use region of interest approaches. Combining

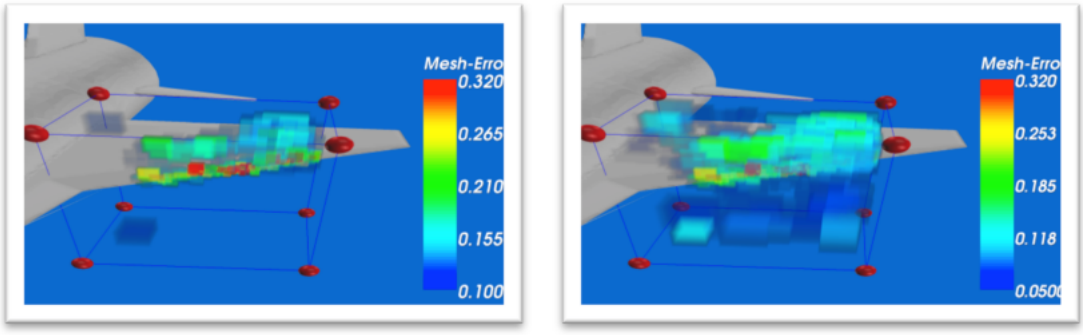

Figure 1 Region of interest to reduce data size

this approach with user interactivity, we are able to reduce the data to an interesting sub-region, therefore a much smaller amount of data is processed.

Figure 1 is an example of the DLR tool which applies a region of interest method to reduce the data size. In order to limit the amount of mesh data to be visualised, a user-defined box of interest has been implemented, with which the user can navigate smoothly through the whole data. The knowledge and experience of the user will also speed up the localisation of poorly defined mesh regions.

Region of interest approaches can also largely reduce the amount of data input for the visualisation front-end. This approach can further be combined with the level of detail structure for an optimised data load.

4 Architecture and workflow

In this section, we described the workflow and architecture of our system. We primarily focus on the communication and integration of processing with the simulation component.

The general workflow and system architecture is described. Following this a description of how this architecture can be realised within our co-design collaboration is provided.

4.1 Architecture

Figure 2 shows the architecture of our simulation and post-processing system. A CFD simulation is running in parallel on a cluster of simulation machines. A visualisation front end is connected to the simulation, extracting current simulation output, visualising this, and then returning signals for further modification. A user is added to the visualisation front-end. The user can interact and navigate through the dataset and select interesting regions using region of interest boxes. Therefore, only the selected subset of data will be transferred from the simulation to the visualisation machine, resulting in a reduced amount of data and time.

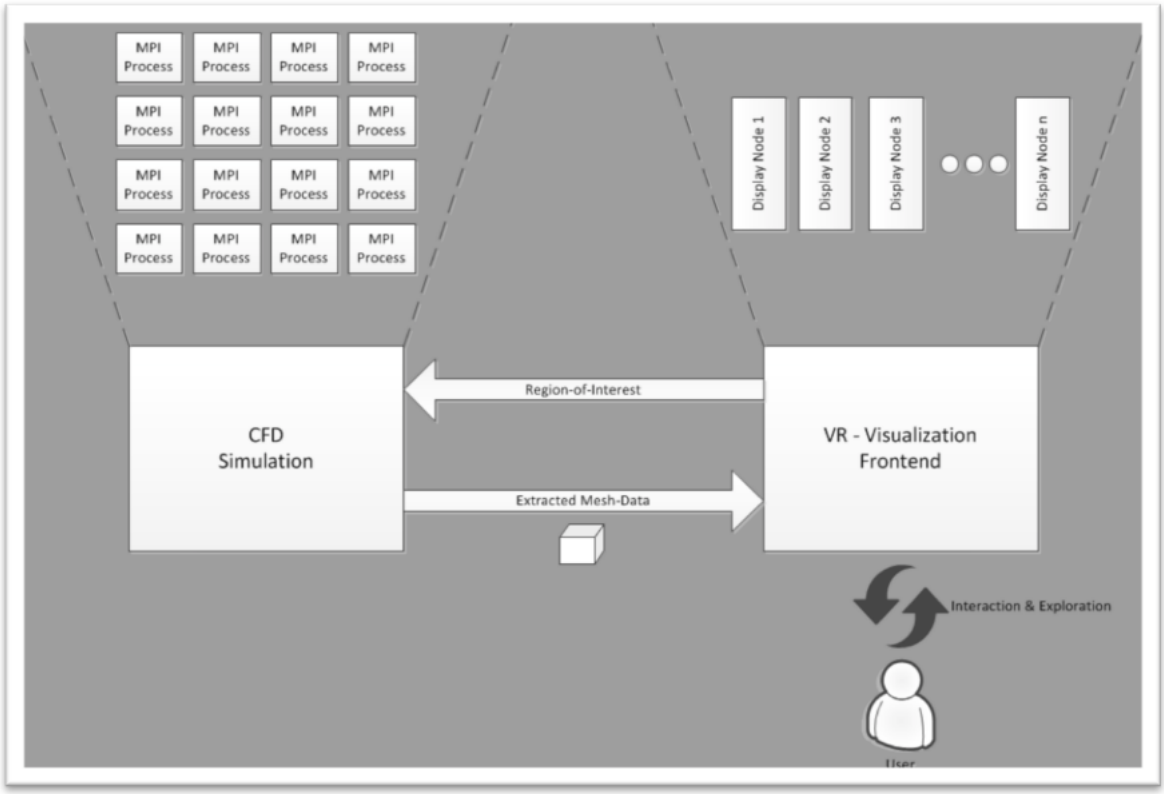

Figure 2 System architecture

Pre-processing (Task 5.1 in workpackage 5) can be connected from the left of the simulation. An agreement on the data partitioning has to be reached between pre-, post-processing and simulation. In order to avoid re-distribution of data among the nodes, the current post-processing system uses the same data distribution as the simulation itself.

4.2 Co-design architecture

The general system structure and workflow can be directly applied to our co-design application HemeLB (workpackage 6). However two main aspects need to be considered.

Firstly, HemeLB has a special data structure and the simulation output is not stored at each time step. The output of the HemeLB data is a kind of field information on the lattices. Therefore, for the purpose of visualisation, a data converter is needed for computing the field information into flow information. Originally, HemeLB simulation outputs field information, which describes a kind of flux in each direction. The resulting flow fields such as velocity and pressure can be calculated from this field information.

Secondly, a client is needed to connect the post-processing tool to the simulation cores. The main concept is to pack the post-processing and visualisation library, allowing this to be called by the simulation. This library will provide functionalities such as visualisations and mesh handling. This client or interface should allow the user to select a field of the data or certain time steps.

Co-design interfaces are elaborated in section 6.

5 Post-processing algorithms

In this section, we define and discuss possible post-processing algorithms for use with exascale datasets.

5.1 In-situ

The first important idea is to use in-situ processing. Understanding the science behind large-scale simulations requires the extraction of meaningful data from datasets of hundreds of terabytes and more [2]. However, the cost of moving the simulation output to a visualisation machine is increasing with larger simulations. According to [3], it is preferable to not move the data at all, or to keep the moved data to a minimum. This can be achieved by applying simulation and visualisation calculations on the same parallel supercomputer in-situ so that data can be shared.

According to [3], the following processing steps can be performed in-situ and enhance the scientists' research activities.

5.1.1 In-situ rendering

For monitoring and steering purposes a direct rendering of images in-situ can be beneficial to give insight into the simulation without requiring an additional visualisation system.

In [4] in-situ rendering is conducted during a tera-scale earthquake simulation. For the presented ray casting visualisation each processor renders its local data. The same data partitioning created by the simulation can be reused, and thus no data movement is needed among processors. Only an API provided by the simulation is required because all access operations are read-only.

No further changes are needed to adapt the simulation. In the image compositing stage, a new algorithm is designed to build a communication schedule in parallel on the fly.

5.1.2 Feature extraction

A feature is a particular physical structure isolated with domain knowledge. Some examples are vortices, shocks, eddies and critical points in CFD applications. These features can be used to categorise the overall physical phenomenon.

The saving of storage space using feature extraction can be very significant. Scientists, however, do not always know exactly what to extract and track in their data.

In [5], a method is demonstrated for feature tracking using a low cost and incremental prediction and morphing approach to track a turbulent vortex flow. Feature extraction and tracking remains an active area of research, because the high-level data reduction explicitly takes domain knowledge into account. Although many feature extraction and tracking methods have evolved in the last decades, less work has been done to apply them to in-situ processing.

5.1.3 Quality assessment

Most of the presented in-situ processing methods focus on reducing data size during simulation run-time. Therefore, the information loss compared to the original data should be conveyed to the user to identify and quantify the loss of data quality.

Most data quality metrics, such as the mean square error, require access to the original data and are therefore not applicable to large-scale simulations where the original data is too large.

A solution applicable in in-situ processing is shown by [6], which used only statistical information extracted from the original data in the simulation. In the visualisation the distance of the reduced data can be compared with the extracted statistical information in order to indicate quality loss. An improved version extracts statistical information in the wavelet domain and also enables a cross-comparison of different reduction types.

5.2 Streaming for multi-resolution data

Exascale simulation data is large and complex. To study and analyse this type of data, it is beneficial to first look into a lower resolution of the entire data. Further studies can be carried out by refining the level of resolution. A multi-resolution data structure enables a fast representation and an early approximation of the final results.

The main purpose of data streaming, parallelisation and data management is to reduce the total run time. The explorative and interactive examination of flow data is always disturbed and interrupted by a long waiting time.

However, within the context of interactive exploration even simplified, approximate results can be sufficient to decide whether to cancel the simulation, to change parameters for the next iteration step or to wait for the final result.

An additional benefit of presenting preliminary results to the user is the perception of a shorter total computation time. Even when the presented data does not yet convey useful information, it can improve the perceived level of interactivity. Integration of an interface to cancel a command or restart it with modified parameters almost approaches explorative analysis.

5.2.1 Classification of data streaming techniques

Streaming visualisation algorithms are often derived from variants of existing approaches. One class of streaming algorithm transmits already computed parts of the result to the visualisation front-end. This includes viewer-optimised extraction methods. In these approaches, the data is decomposed into multiple blocks which are spatially organised, usually using tree-based meta-data structures. Algorithm execution then prioritises those blocks which are spatially close to the viewer. Some schemes also consider the viewing direction. Those blocks which are further away are processed later as they are assumed to contain less relevant details due to their small representation on screen and potential occlusion.

Another class of algorithms, so-called Level of Detail (LOD) schemes, computes a coarse initial representation of the result which, in contrast to the previous approaches, always gives a global, although coarse, overview of the extracted feature. This can be computationally expensive if these initial results cannot be used in the computation of the next finer resolution step. However if this is possible, the relation between individual detail levels is called progressive. The data structures which are transmitted in this process of refinement from coarsest approximation to the final result are called progressive multi-resolution structures.

6 Co-design and post-processing algorithms

In this section, we focus on the definition and discussion of post-processing algorithms, particularly those relevant to ourco-design tasks with workpackage 6.

Currently, the co-design tasks of workpackage 5 have primarily concentrated on the collaboration with the HemeLB application group from workpackage 6. The primary focus of our cooperation is to develop and integrate workpackage 5´s post-processing and steering codes as part of a library or add-on tools for workpackage 6. We also plan to extend our co-design collaboration to the OpenFOAM, Nek5000 and ELMFIRE application groups from workpackage 6 in the coming periods.

6.1 Interfaces and clients

Two interfaces are involved in the co-design task with workpackage 6:

The first one is a visualisation interface. This interface should allow the simulation expert to call the visualisation package as a library by defining which visualisation techniques to use, which field to visualise, and which time step to inspect. It should be in the following form:

```
Initialise:
 MyVisualiser:: Init (
      Proc_t rank,
     Flow Field * field )
MyVisualser::step ()
```

.

Another client or interface that is required is a steering client, which allows the user to give feedback to the simulation, such as to modify a parameter. With the help of this client, we can enable computational steering, and finally close the loop between simulation and visualisation. However, this interface is a long term goal, and it is independent of the first client.

6.2 Choices of visualisations techniques for HemeLB

In this section, we describe the possible visualisation techniques desired by the HemeLB developers. In general, three tools are helpful in analysing the blood flow output simulated by HemeLB, namely volume rendering, path-lines and stream-line. We demonstrate the usage of these techniques using a VTK-based visualisation applied to a relative small dummy dataset, as a preliminary demonstration.

6.2.1 Volume Rendering

The HemeLB simulation produces sparse flow fields output. This type of data is commonly mapped to a volume. Any scalar in the field is mapped to a volume density in the image file, revealing the field density in a 3D image.

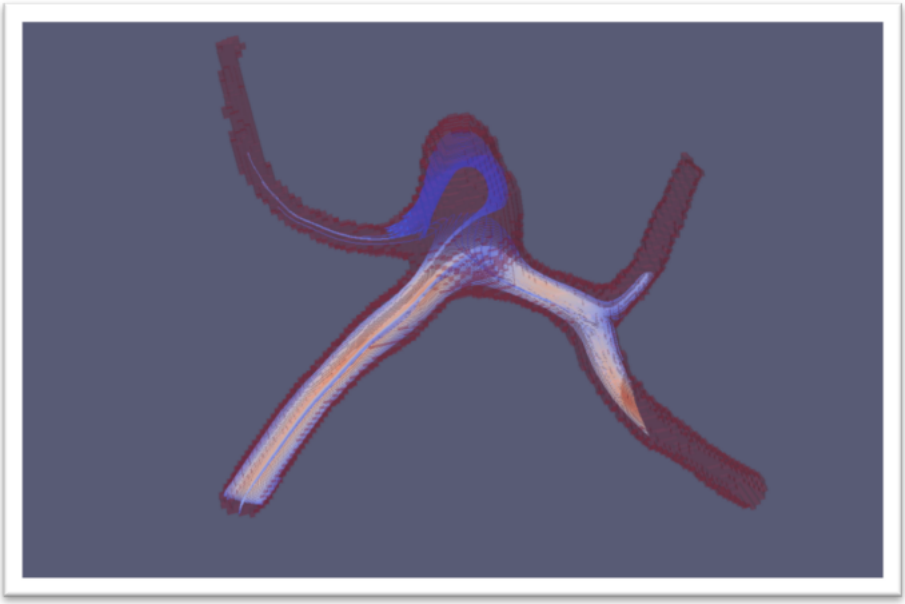

Figure 3 A volume rendering example of the aneurysm data set simulated by HemeLB

Figure 3 shows an example of a volume rendered HemeLB dataset. One advantage of doing volume rendering is that one can distribute the computation on different nodes and carry out the computation in a parallel manner, which is not applicable with many other algorithms.

6.2.2 Line integrals

Line integrals, especially stream-lines and path-lines are useful tools for tracing and analysing flow fields. Figure 1 shows an example of the path-line tubes while visualising blood flows in the aneurysm dataset.

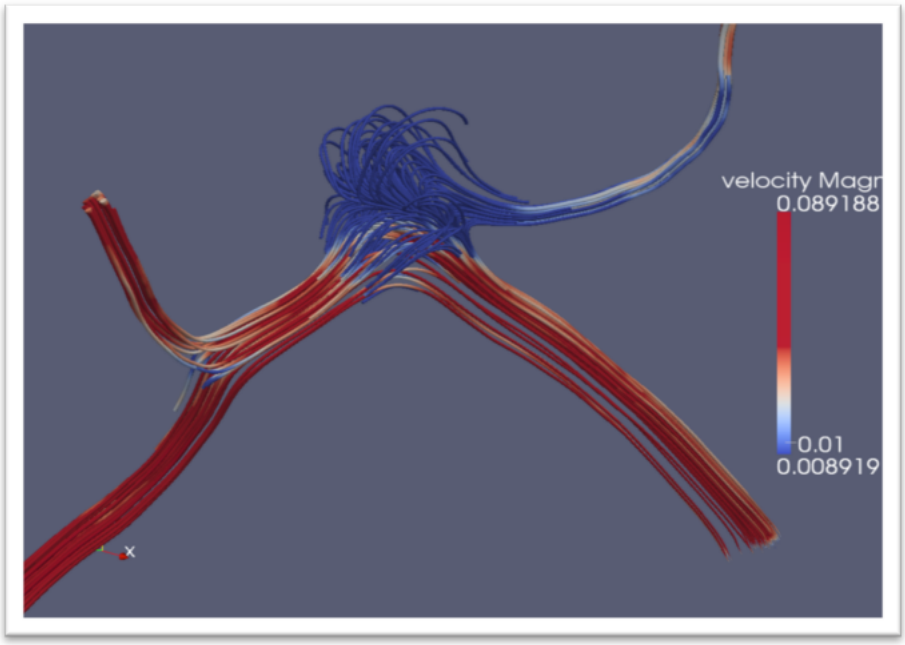

Figure 4 A stream line example of the velocity field in blood flow simulation

This type of visualisation provides the user with an intuitive visual description of how the blood is flowing, revealing not only orientation information, but also features such as vortices and bifurcations.

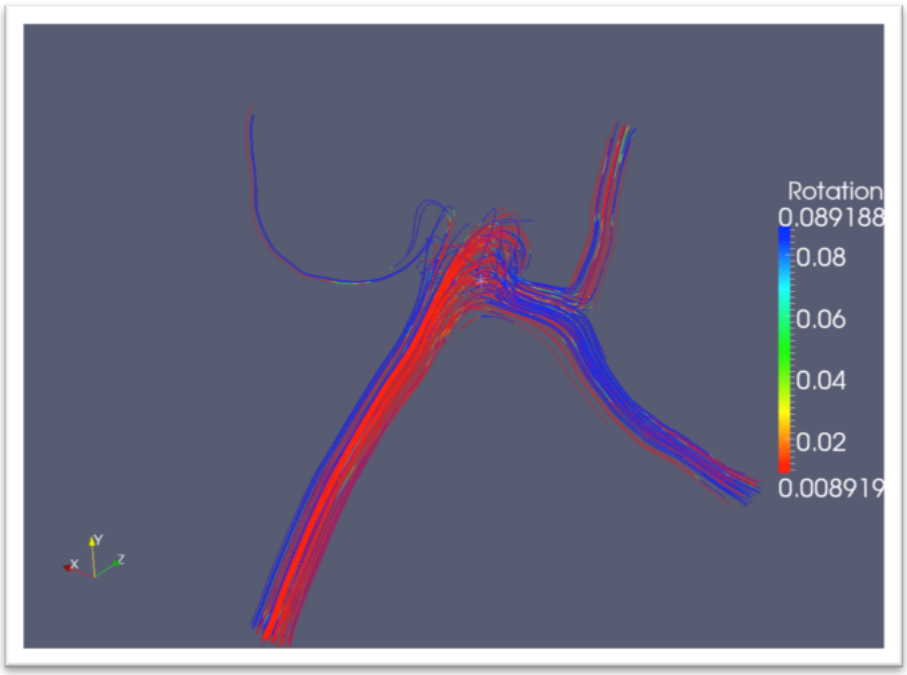

Figure 5 An example of the path-line visualisation of the velocity field in blood flow simulation.

Figure 5 is another example of using path-lines. With this technique, the user is able to trace the flow trajectory, and the colours of the lines are mapped to the rotation of the vector field.

6.3 Challenges in implementation

As mentioned above, the presented visualisation examples are only small dummy datasets generated by HemeLB. To implement the mentioned algorithms into a distributed, parallel process, we face the following challenges:

Parallelisms and data distribution While implementing visualisation algorithms in a parallel manner, special attention should be given to data distribution. Algorithms which need a lot of neighbourhood searching, such as path-lines, are not applicable while doing parallel computing. The frequent search between cells results in a huge amount of communication between different cluster nodes, which is computationally very expensive and slow.

On the other hand, techniques which can be implemented on a sub-mesh independent of the neighbouring ones are better suited in a distributed manner. Volume rendering, for instance, can extract volumes on each partitioning without any data exchange with the neighbours. Therefore, communication between different cluster nodes is avoided.

7 Conclusion

In this deliverable, we have described the data structures and sketched the possible algorithms for post-processing exascale simulation data. In particular, algorithms appropriate to the co-design applications of workpackage 6 are presented.

Data distribution and data reduction are important issues in managing exascale data structures. Multi-resolution data structures are the only possible data structures that can be used for managing large-scaled datasets. Choices presented in this workpackage should also be related to and applicable for certain remote rendering tasks within task 5.3 of workpackage 5.

Post-processing algorithms have been discussed in detail in this deliverable. In particular, in-situ and multi-resolution techniques have been presented as two promising approaches to handle exascale simulation data. Visualisation algorithms, as part of the post-processing algorithms considered, are designed for and adapted to the needs of the CRESTA applications in a co-design process with workpackage 6.

8 References

[1] V. Pascucci and A. J. Frank, *Hierarchical indexing for out-of-core access to multi-resolution data (2001),* 2001.

[2] K. L. Ma, R. Ross, J. H. G. Huang, K. Morel and J. Owens, *Utra-scale visualization: Research and education,* 2007.

[3] K. K. Ma, C. Wang, H. Yu and A. Tikhonova, „In-situ processing and visualization for ultrascale simulations," *Journal of Physics: conference series,* Bd. 78, Nr. 1, pp. 12-43, 2007.

[4] T. Tu, H. Yu, L. Ramirez-guzman, J. Bielak, O. Ghattas and K. L. Ma, „From mesh generation to scientific visualization and end-to-end approach to parrallel supercomputing," in *ACM/IEEE supercomputing conference*, 2006.

[5] C. Muelder and K. L. Ma, „Rapid feature extraction and tracking through region morphing," Computer Science Department, UC Davis, Davis, CA, USA, 2007.

[6] C. Wang and K. L. Ma, „A statistical approach to volume data quality assessment," *IEEE trasactions on visualization and computer graphids,* Bd. 14, Nr. 3, pp. 590-602, 2008.