D5.2.1 – Post-processing: analysis and system definition for exascale systems

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:	M6
Submission date:	31/03/2012
Project start date:	01/10/2011
Project duration:	36 months
Deliverable lead organisation	DLR
Version:	2.0
Status	Final
Author(s):	Christian Wagner (DLR), Fang Chen (DLR), Achim Basermann (DLR)
Reviewer(s)	David Lecomber (ASL), Lorna Smith (UEDIN)

Dissemination level	
<PU/PP/RE/CO>	*PU - Public*

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	24/02/2012	First version of the deliverable	Christian Wagner (DLR), Fang Chen (DLR)
0.4	01/03/2012	Added User requirements section	Christian Wagner (DLR), Fang Chen (DLR)
1.0	09/03/2012	Revision	Achim Basermann (DLR)
2.0	22/03/2012	Add reviewers´ comments	Christian Wagner (DLR) Fang Chen (DLR)
2.0	26/03/2012	Final check after internal review	Achim Basermann (DLR)

Table of Contents

Index of Figures

1 Executive Summary

This deliverable gives an overview how post-processing can be designed and implemented to support interactive data exploration and visualisation in exascale environments. Post-processing is concerned with extracting visualisation primitives from solver results, for example computing a stream surface given a flow field. The related workpackage task 5.3 handles the subsequent rendering of these primitives to produce a screen representation (cf. [1]). Within workpackage task 5.1, methods for pre-processing are explored which partition an input data set to minimize communication in the solver phase (cf. [2]). The post-processing algorithms then have to operate on data in the given partitioning.

While the performance of extreme-scale simulations is a key aspect, pre- and post-processing are additional important steps. Mesh creation and partitioning define the accuracy of the simulation results, and visualisation is used to finally analyse the simulated phenomena.

Furthermore, real-time visualisation of the simulation mesh, its partitioning and intermediate simulation results is also important during a simulation run. This not only makes it possible to analyse intermediate results, but also to detect and solve problems arising during a simulation. This avoids wasting a lot of CPU time for inappropriate parameter configurations.

Interactive visualisations during runtime, so-called online-monitoring, come with additional challenges and require for a combination of different solutions described in this document.

In order to further analyse requirements of interactive online-monitoring solutions, the following two applications are discussed:

HemeLB would benefit from elaborate and scalable visualisation and steering approaches that do not interfere with its specific lattice Boltzmann simulation data layout. Introspection of simulations is important to deliver information to developers and users about the detailed status during simulation runs.

OpenFOAM as a traditional grid based solver is considered to scale well and will be prepared for exascale architectures during this project. Therefore, the visualisation tools used have to scale to the same order of magnitude.

2 Introduction

Application domains, such as fluid dynamics, meteorology, nuclear physics, or material science, heavily rely on numerical simulations on High Performance Computing (HPC) resources. To improve the performance of extreme-scale simulations, post-processing of the numerical data has become an important tool for analysing and monitoring simulation processes.

Partitioning and mesh creation are critical for the accuracy of numerical simulations and are normally pre-defined before a simulation run. However, it is often the case that these automatically pre-defined meshes are not appropriate or have to be changed while the simulation progresses. With increasing complexity in exascale simulations, validating meshes becomes a challenging task, which requires innovative visualisation techniques.

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 part of the data through a post-processing pipeline, typically consisting of data extraction, filtering, mapping and visualisation stages. This is a time-consuming process so that the requirements for efficient interactive exploration like the ability to move freely through the data will be hard to meet. A solution to maintain interactivity in exascale environments is to distribute the phases of the post-processing pipeline to a high extent.

The aim of this work package task 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, multi resolution data formats, and data streaming approaches.

This document contains an overview of techniques, which, in combination, allow for interactive explorative post-processing in an exascale environment. In chapter 3, general requirements are stated and further explained by the examples described in chapter 4. Finally, interactive explorative post-processing techniques are explained in detail in chapter 5.

3 Analysis of Requirements for Interactive Data Exploration on Exascale Systems

Visualisation will be one of the key aspects in exascale environments in order to analyse the huge amount of generated data. Visualisation, however, will face many problems. Among these the scalability of visualisation algorithms is of great importance.

Exascale simulation clusters with peak performances over 10^{18} flop/s are expected to be available in 2018 [3]. The realisation of exascale systems will, unlike past hardware trends, rely on massively parallel hybrid systems and high concurrency per node [4].

High-quality visualisation is important for many application domains. To be successful in analysing large-scale, complex simulation data, interactive exploration and visualisation methods have proven to be useful [5]. This involves complex tasks such as volume rendering of scalar fields and streamline integrations on vector fields. These complex extraction and rendering approaches need to be performed in a highly parallel and scalable way in order to be applicable to the massively parallel simulations and distributed data on future exascale systems.

3.1 Online Monitoring and Computational Steering

Visualisation is of increasing importance in the context of exascale applications and not only dedicated to classical post-processing tasks. With increasing complexity simulations need to support introspection capabilities. Developers and users need to be aware about the detailed status of a running simulation. Therefore, information about timings, numerical stability and simulation progress must be collected. Furthermore, intermediate results are important to justify the state of a running simulation (online monitoring). In complex simulations interactive exploration is beneficial in order to support the ability to freely move through the current data and identify regions of interest or critical situations.

Nowadays, a common online monitoring solution is to copy the current simulation data to a dedicated visualisation cluster. This will not be an option for exascale environments as the amount of raw data will be too huge to be transferred in a reasonable time, considering the available network bandwidths. Using in-situ visualisation is a key component for online monitoring in order to convert raw data to meaningful visual representations. Since compute power is cheap compared to network transmission, data reduction as early as possible appears to be the only viable solution. However, visual representations are still huge. In order to provide interactive exploration, multi-resolution data structures are required. While they can be streamed for further processing or visualisation a progressive visualisation can be supported. First intermediate results are rendered at once; details increase when additional data arrives.

Online monitoring allows justification the state of running simulations. Since exascale systems will add additional requirements on efficiency and power consumption to simulation applications [6], computational steering is a very promising approach to reduce pre- and post-processing efforts by providing the possibility to directly interact and steer a running simulation.

Computational steering introduces additional technical and usability challenges. A suitable software framework needs to be designed. In the past, computational steering systems were developed to interact with on-going simulations by enhancing existing

visualisation tools [7] [8] or by developing specific computational steering frameworks [9]. These solutions mainly concentrate on data management and data interfaces [10]. However, the main drawback of both approaches is that all steerable parameters as well as callable methods have to be known at compile time [11]. To address this problem, new software frameworks are desired which will allows on-the-fly steering of the simulation process.

A computational steering solution for CFD simulation, developed at DLR, inherently couples the simulation code with visualisation algorithms, rendering systems, and user interaction methods. First tests of this environment with DLR CFD codes were promising (cf. Figure 1). From these experiences, interaction and manipulation techniques also appear to be an appropriate means to enable intuitive and effective online monitoring and computational steering of exascale simulations with e.g. HemeLB or OpenFOAM.

3.2 Mesh Visualisation and Interaction

The quality of a simulation mesh is crucial for the simulation run. The mesh cells directly influence the accuracy of simulation results, domain partitioning of the mesh cells has an effect on simulation performance and efficiency. Although the maintenance of high mesh quality is desired throughout the entire simulation, automatic mesh adaption algorithms use local heuristics and do not guarantee a certain global mesh quality.

Suitable visualisation approaches are required in order to assess the quality of a simulation mesh. Common solutions, such as cut-away representations, make use of the mesh wireframe representation. In exascale simulations, even with future high-resolution displays, these approaches are not usable anymore. Furthermore, simulation meshes in exascale environments are too large to be rendered by a traditional frontend system.

A next point is mesh interaction. How can mesh portions be interactively selected and manipulated even if raw data is still present only on the simulation side and maybe only a small fraction of the mesh is visible?

Furthermore, what are suitable manipulation strategies for those large meshes? They will also be different for different mesh types. How will the manipulation be synced with the original mesh on the simulation side?

Another point is steering the domain partitioning and checking for unbalanced partitions. Here cooperation with workpackage task 5.1 (pre-processing, cf. [2]) and workpackage 6 (co-design via applications) is required.

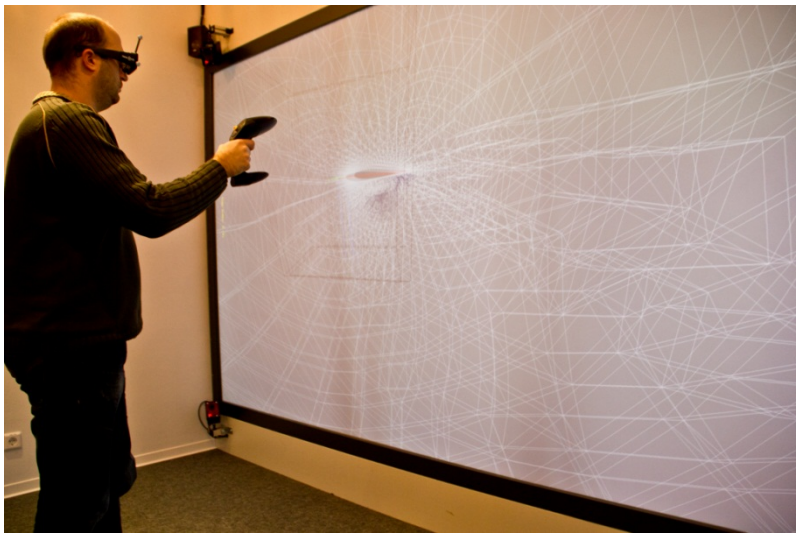

Figure 1: Prototype to visualise and adapt meshes of on-line simulations in a virtual environment.

4 Requirements of the Codes

HemeLB and OpenFOAM are two open-source simulation codes that have the potential to run on exascale systems. The following section focuses on the state of the art of these simulation tools and the arising issues for interactive data post-processing if HemeLB and OpenFOAM are executed on exascale computer architectures.

4.1 HemeLB

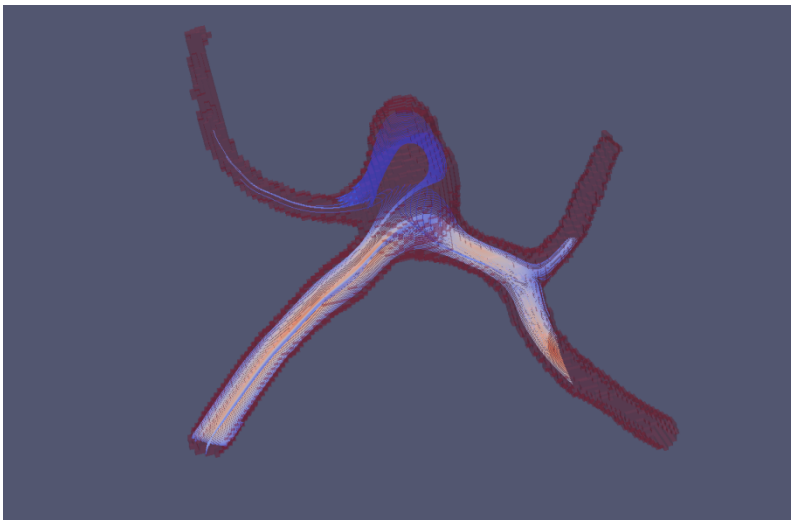

Figure 2: Aneurysm dataset simulated by HemeLB.

4.1.1 Current State

A simple ray-casting approach is available for the current in situ visualisation of HemeLB datasets. Figure 2 illustrates an aneurysm dataset from a HemeLB simulation. Each sub-domain renders a set of pixels, which are then communicated, in a binary tree with non-blocking MPI, the sets of pixels being merged at each level with the final image finishing on a root task. The steering data, which currently only consist of visualisation parameters, is passed from the root task back down the tree.

Aside from ray-casting, a few other visualisation techniques are available, such as streak-lines, volume rendering of the velocity field and surface pressure and stress fields.

4.1.2 Requirements

The goal of this work package is to develop better visualization modules which are scalable to exascale scenarios and have a minimal impact on memory footprint. This requires co-optimisation of solver and visualisation codes and development of interfaces to allow for sharing of data structures between both [12].

A platform approach which requires the solver algorithm to represent its working set in a format pre-described by a visualisation library is not acceptable, as it would interfere with core functionality and potentially reduce solver performance.

The alternative strategy of providing additional code to transform in-memory data to such a format is also not an option as the necessary copying of data implies a large memory cost.

Instead, the goal is to develop a generic visualisation library, which can be configured using domain specific languages (DSLs) to adapt to given data structures. In terms of

functionality, it is desired to provide a superset of the visualisation methods, which are currently available.

4.2 OpenFOAM

While the visualisation library being developed should be adaptable to different solver codes, the primary focus of workpackage task 5.2 (post-processing) will be the integration with HemeLB. OpenFOAM integration has been identified as a long-term goal.

4.2.1 Testing Example

A application test case is the simulation of the flow in an entire hydro turbine, see Figure 3.

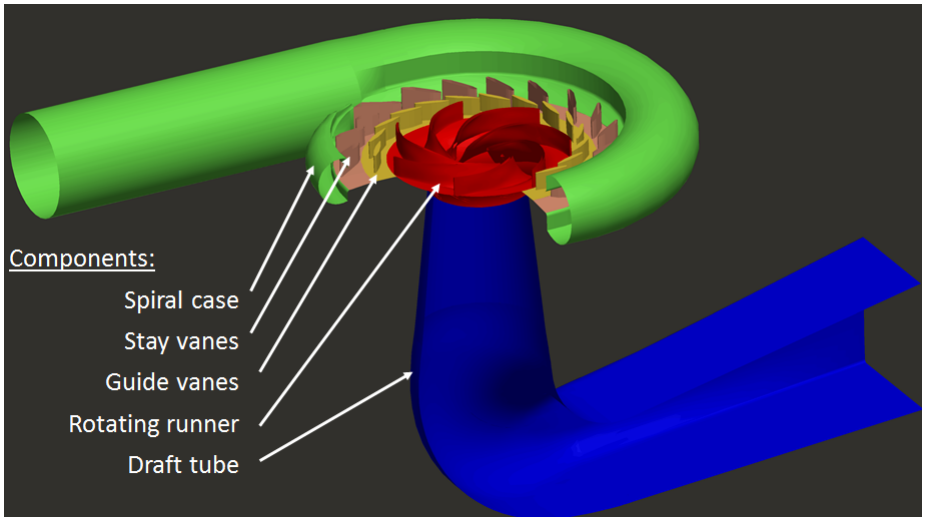

Figure 3: Geometry of a Francis turbine.

A large eddy simulation approach is applied to very fine computational grids. The number of nodes can be estimated to approximately one milliard for all parts of the Francis pump-turbine. Due to the large node numbers and fine time steps, approximately 80 million core hours will be needed for a full-converged simulation.

Resulting simulation data has the size of 250 GB for a single time step only. The number of time steps that have to be saved depends on the visualisation of the instantaneous flow phenomena.

Even though the available data set is only of tera/peta scale, it could be a starting point for modelling exascale systems.

5 Definition of a Post-Processing System Towards Exascale

Post-processing in exascale scenarios implies in-situ extraction of visualisation primitives based on a given partitioning of numerical data. In order to allow for scaling into these regimes, localizing and minimizing communication is of utmost importance.

Communication is required in post-processing either to access input data stored on another node or to transmit partially processed visualisation primitives to another node for further processing. An example for this is streamline computation. Streamlines can be integrated on a node until they leave the local partition. The partially computed streamline can then be transmitted to the node containing the adjacent partition to continue the integration. In such schemes, it is beneficial if the network topology mirrors the topology of the partitioning such that communication between nodes which are responsible for neighbouring partitions is efficient.

In developing the post-processing system, the focus therefore has to be on visualisation methods that require only a minimal amount of communication, which should be limited to the immediate neighbourhood. As scalability is the primary goal, the amount of local computation required is only of secondary importance.

The system will consist of a set of distributed post-processing algorithms that can be controlled interactively by a front-end workstation (cf. Figure 4). Visualisation primitives generated locally by these algorithms are then rendered and composited by the methods described in workpackage task 5.3 (cf. [1]). The purpose of these interactive, in-situ visualisation techniques is to support computational steering by providing insight into the current state of a numerical simulation.

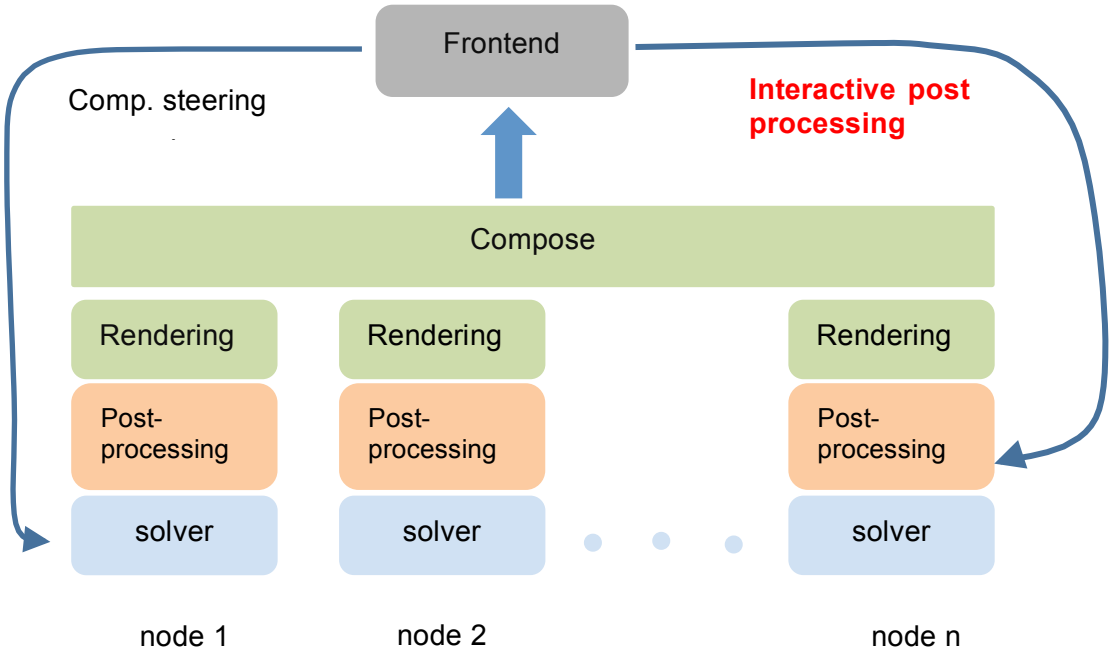

Figure 4: System architecture.

6 Post-Processing Techniques Towards High Scalability

6.1 In-situ Processing

Understanding the science behind large-scale simulations requires the extraction of meaning from datasets of hundreds of terabytes and more [13].

However, the cost of moving the simulation output to a visualisation machine is increasing with larger simulations. According to [12], 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 [12], the following processing steps can be performed in-situ and enhance the scientists' research activities.

6.1.1 Data Reduction

Common data reduction techniques are subsampling, quantisation and transform-based compression. Subsampling is the simplest way to reduce simulation data. A common practice is to skip time steps and select, e.g., every hundredth time steps. However, skipping time steps for simulation output poses a major challenge to temporal-space visualization. For instance, the accuracy of path-lines computation will suffer from large temporal step length.

Simulation data is mostly computed in single or double precision floating point numbers with high accuracy. However, it is not always necessary to preserve this level of accuracy, for example, if relative values are in the focus of research and absolute values are not important. Also, hardware accelerated rendering makes use of texturing hardware with 8 or 16 bit of resolution. In those cases, quantisation makes sense in order to reduce the amount of data to store.

Data quantisation can be performed in many ways. Simplest Quantisation methods are direct scalar quantisation methods, which use only local data and are fast to compute. Data quantisation also contains more elaborate methods making use of data statistics, such as the global Lloyd-Max method [14] or the local Jayant quantiser [15].

Another class of quantisation methods is vector quantisation that groups data values into blocks of data and encodes these blocks. Since these methods, such as the Linde-Buzo-Gray algorithm [16], requires the training of a codebook, vector quantisation methods are computationally too expensive to be used as an in-situ processing method [17].

Finally, transform-based compression is a very effective way to reduce data to store on disk.

This compression transforms the data from spatial domain to frequency domain resulting in energy coefficients for each frequency. Since this representation is often more meaningful for the physical situation, a compression in this domain introduces fewer errors by only quantising the less important lower energy coefficients more coarsely.

Most popular transform-based encodings are the discrete cosine transform and the wavelet transform, later allowing an additional multi-resolution data representation and a level of detail to be selected according to the visualisation requirements.

In terms of cost and performance, transform-based compression is an advantageous choice for in-situ data reduction [12].

6.1.2 Data Extraction

A feature is a particular physical structure isolated with domain knowledge. Some examples are vortices, shocks, eddies, critical points etc. 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 [18], 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 to be 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.

6.1.3 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 are too large.

A solution applicable in in-situ processing is shown by [19], who only used 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 and 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.

6.1.4 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 [20] 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.

6.2 Multi-Resolution Data Structures and Streaming

Exascale simulation data are large and complex. To study and analyse these types 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 resolutions. A multi-resolution data structure enables a fast representation and an early approximation of the final results.

6.2.1 Why Use a Multi-resolution Data Structure

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 the long waiting time.

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.

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 brings us very close to explorative analysis.

6.2.2 Classification of Data Streaming Techniques

Streaming visualisation algorithms are often derived variants of existing approaches. One class of streaming algorithms 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 that are spatially organised, usually using tree-based meta-data structures. Algorithm execution then prioritises those blocks that are spatially close to the viewer. Some schemes also consider the viewing direction. Those blocks that 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 flow feature. This can be computationally expensive if these initial results cannot be used in the computation of the next finer resolution step. If this is possible, however, 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.

7 Conclusion

In this deliverable, we have studied the challenges and possible solutions in interactive post-processing of result data from simulations on exascale systems. Techniques that allow for scalable visualisation algorithms were presented, and a system architecture was designed which relates the post-processing component to the numerical solver as well as the distributed rendering system described in workpackage task 5.3 (cf. [1]).

8 References

[1] F. Niebling, J. Hetherington and A. Basermann, “CRESTA D 5.3.1: Remote hybrid rendering: analysis and system definition for exascale systems,” March 2012.

[2] G. Matura and A. Basermann, “CRESTA D 5.1.1: Pre-processing: analysis and system definition for exascale systems,” March 2012.

[3] J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Aloisio and J. Andre, “The international exascale software project roadmap,” *International Journal of High Performance COmputing Applications,* vol. 25, no. 1, pp. 3-60, 2011.

[4] A. Geist, “Paving the roadmap to exascale,” *SciDAC Review, special issue,* 2010.

[5] M. Wolter, M. Schirski, T. Kuhlen, C. Bischof, M. Buecker and P. Gibbon, *Hybrid parallelization for interactive exploration in virtual enrionments.*

[6] S. Borkar, “The exascale challenge,” *International symposium on Vlsi design automation and test (vlsi-dat),* pp. 2-3, 2010.

[7] T. Eickermann, W. Frings, P. Gibbon, L. Kirtchakova and D. Mallmann, “Steering unicore applications with visit,” *Philosophical transactions of the royal society,* 2005.

[8] S. G. Parker, D. M. Weinstein and C. R. Johnson, *The scirun computational steering software system,* 1997.

[9] D. B. M. Jenz, *The computational steering framework steereo,* 2010.

[10] O. Coulaud, M. Dussere and A. Esnard, *Toward a distributed computational steering environment based on corba,* 2003.

[11] C. Wagner, M. Flatken, M. Meinel, A. Gerndt and H. Hagen, *FSSteering: A distributed framwork for computational steering in a script-based CFD simulation environment,* Berlin, Germany: Lehmanns Media-LOB.de, 2010.

[12] K. K. Ma, C. Wang, H. Yu and A. Tikhonova, “In-situ processing and visualization for ultrascale simulations,” *Journal of Physics: conference series,* vol. 78, no. 1, pp. 12-43, 2007.

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

[14] R. C. Gonzalez and R. E. WOods, Digital image processing, third edition, Upper Saddle River, NJ, USA: Prentice-Hall, Inc, 2006.

[15] N. S. Jayant, “Adaptie quantization with a one-word memory,” *Bell system technical journal,* vol. 52, pp. 1119-1144, 1973.

[16] Y. Linde, A. Buzo and R. Gray, “AL algorithm for vector quantizer design,” *IEEE transactions on communications,* vol. 28, no. 1, pp. 84-95, 1980.

[17] N. Fout, K. L. Ma and J. Ahrens, “Time-varying multivariate volume data reduction.,” in *2005 ACM symposium on applied computing*, New York, NY, 2005.

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

[19] C. Wang and K. L. Ma, “A statistical approach to volume data quality assessment,” *IEEE trasactions on visualization and computer graphids,* vol. 14, no. 3, pp. 590-

602, 2008.

[20] 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.

[21] Z. Wang, G. S. H. R. Wu, E. P. Simoncelli, E. Yang and A. C. Bovik, “Quality-aware images,” *IEEE transaction on Image Processing,* vol. 16, no. 6, pp. 1680-1689, Jun 2006.