Enabling in situ pre- and post-processing for exascale hemodynamic simulations

– A co-design study with the sparse geometry lattice-Boltzmann code HemeLB

Fang Chen*, Markus Flatken*, Achim Basermann†, Andreas Gerndt*,
James Hetherington‡, Timm Krüger‡, Gregor Matura† and Rupert W. Nash‡

* Software for Space Systems and Interactive visualisation, Simulation and Software Technology,
German Aerospace Center, Lilienthalplatz 7, 38106, Braunschweig, Germany
Email: fang.chen@dlr.de; markus.flatken@dlr.de; andreas.gerndt@dlr.de
† Distributed Systems and Component Software, Simulation and Software Technology,
German Aerospace Center, Linder Höhe, 51147 köln, Germany
Email: achim.basermann@dlr.de; gregor.matura@dlr.de
‡ Centre for Computational Science, University College London, 20 Gorden Street,
London, WC1H 0AJ, United Kingdom
Email: j.hetherington@ucl.ac.uk; t.krueger@ucl.ac.uk; rupert.nash@ucl.ac.uk

***Abstract*—Today's fluid simulations deal with complex geometries and numerical data on an extreme scale. As computation approaches the exascale, it will no longer be possible to write and store the full-sized data set. *In situ* data analysis and scientific visualisation provide feasible solutions to the analysis of complex large scaled CFD simulations. To bring pre- and post-processing to the exascale we must consider modifications to data structure and memory layout, and address latency and error resiliency. In this respect, a particular challenge is the exascale data processing for the sparse geometry lattice-Boltzmann code HemeLB, intended for hemodynamic simulations.**

In this paper, we assess the needs and challenges of HemeLB users and sketch a co-design infrastructure and system architecture for pre- and post-processing the simulation data.

To enable *in situ* data visualisation and analysis during a running simulation, post-processing needs to work on a reduced subset of the original data. Particular choices of data structure and visualisation techniques need to be co-designed with the application scientists in order to achieve efficient and interactive data processing and analysis. In this work, we focus on the hierarchical data structure and suitable visualisation techniques which provide possible solutions to interactive *in situ* data processing at exascale.

Architectural challenges and road-maps will be presented as the major focus of this paper. We sketch a software architecture which integrates pre- and post-processing techniques that can provide *in situ* analysis and ultimately computational steering to HemeLB.

I. Introduction

There is increasing evidence that the development of cardiovascular diseases, such as aneurysms, is caused by certain blood flow patterns. As intracranial aneurysms are quite common ($1-5\%$ of the entire population are affected), it is desirable to understand their formation and which particular aneurysms are at enhanced risk of rupture. HemeLB is an application for simulation of blood flow in the larger arteries. Its ultimate goal is to contribute to patient-specific treatment, e.g., real-time risk assessment of cerebral aneurysm rupture. To this end, HemeLB has to produce reliable predictions on the time scale of one hour. Additionally, due to the high resolution required in time and space, HemeLB has been designed as a highly parallelised algorithm.

The need for a parallel implementation of a sparse fluid solver requires sophisticated algorithms for the domain decomposition. Ideally, such a decomposition should be optimised for load-balancing with respect to computation, communication and visualisation. Currently, the ParMETIS library is employed to generate computationally load-balanced domain decompositions. Approaching the exascale with millions of processes is expected to impede an efficient simulation initialisation. It is an open question how scalable the current ParMETIS implementation is.

Parallel pre-processing is, first of all, seen as initialisation of calculation data. Load geometry or mesh data from the file system without stressing it too much. Optimise it for a better result. Arrange it for a fast(er) calculation. Proceeding to an exascale regime, this certainly holds true; yet it is extended by a new main target. Load balance becomes a crucial factor.

Most applications are aware of this fact and meet this requirement by applying a partitioner to the pure calculation costs. This is a good start for load balance. Yet, it does not cover costs of other simulation parts. Visualisation, for example, becomes even more important at exascale because some form of interactivity is desired. Therefore a successful pre-processing has to include all parts of the simulation to guarantee an overall load-balance.

Currently, HemeLB implements a proper pre-processing. It reads in the initial data, i.e. the blood vessel geometry, calls a partitioner and redistributes the data accordingly. A quite good

load balance is established.

If, however, visualisation comes into play the situation changes. Constraints on load and simulation sequence alter the balance equation significantly. visualisation costs have to be considered now. A repartitioning may be necessary. A new approach is needed evolving the partitioner to a stand-alone part of the simulation.

Large-scale simulations bring forth huge data sets. It is impractical and inefficient to store all data for later evaluation. Instead, *in situ* visualisation and feature extraction are promising approaches to reduce the amount of data to handle. This involves post-processing routines which extract relevant data at runtime. Physiologically relevant data sets comprise wall stress distributions and streak-lines for the visualisation of the flow field. The challenge is to design new algorithms for scalable volume rendering and line integral convolution methods.

To prevent restarting the simulation process from the very beginning, an on-the-fly results analysing and parameter modifying tool is needed. A steering client can be used to supply the simulation with real-time requests by the user, for example an increase of the visualisation rate, a change of the viewpoint or the extraction of hydrodynamic observables from a user-defined subset of the simulation volume. At the same time, the steering client can receive status informations from the simulation. These may contain visualisation data, consistency and validity checks, or estimates on the remaining runtime.

In this work, we sketch the roadmap for HemeLB co-design, preparing pre- and post processing for HemeLB simulation towards exascale. We present the challenges of pre- and post processing and possible solutions to the presented problems. A co-design system architecture is investigated which provides interactive data post-processing and computational steering. We propose suitable visualisations and making it possible for users to interactively explore their simulation data. We discuss how the loop of pre-processing, simulation and post-processing can be closed and how we are planning to extend the computational steering client within HemeLB.

II. Related work

The Centre for Computational Science at University College London develops, amongst other projects, scientific applications of HemeLB in the field of blood simulation. Recent investigation of HemeLB performance ([1]) has shown that it can scale well to at least 32 thousand cores with more than 81 million lattice sites. In this work, not only the core simulation but also visualisation and steering facilities are examined. Bieferale et. al ([2]) implemented a lattice-Boltzmann algorithm on GP-GPU. Results in this paper are compared with the same algorithm implemented on CPUs.

A large body of literature can be found on partitioning tools. Most of these tools try to balance only the core calculation costs. Several popular choices include: ParMETIS, PTScotch [3], Zoltan [4]. The ParMETIS algorithms are based on the parallel multilevel k-way graph-partitioning, adaptive repartitioning, [5]. Current HemeLB uses ParMETIS for domain decomposition. These three tools provide similar functionality to obtain a balanced partitioning of given data. They are widely used and under active development.

In order to gain insight from simulation data, data post-processing becomes an important step for the application scientists. Several leading research groups have already started to approach the peta- and exascale post-processing problems. Childs presented a system paper ([6]) discussing the possible system challenges and solutions for petascale post-processing. The SciDAC institute for ultra-scale visualisation ([7]) focuses on advancing the state of visualisation techniques in order to enable extreme-scale knowledge discovery.

In the meantime, *in situ* processing has draw a great attention in the visualisation and simulation community. Whitlock et. al([8]) have introduced a new library to $VisIt$, allowing fully featured *in situ* visualisations without changing the application too much. [9] has integrated *in situ* visualisation for large-scale turbulent-combustion simulation.

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 traversing are key issues in using hierarchical data structures. Pascucci and Frank [10] introduced a new global indexing scheme, which accelerates adaptive traversal of geometric data with binary trees, which is also applicable to quadtrees and octrees. 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.

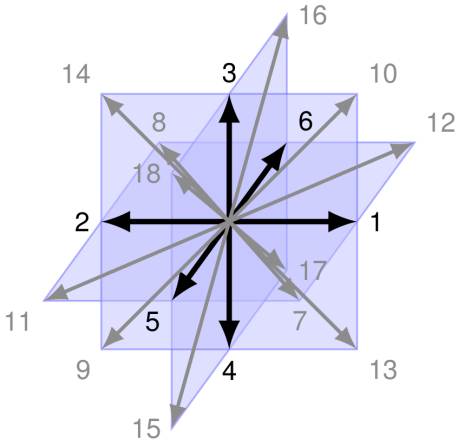


Fig. 1. The lattice-Boltzmann method does not use an unstructured meshes; instead, it uses a regular lattice structure [11]

III. Major challenges of pre- and post-processing at Exascale

The advanced Scientific Computing Advisory Committee [12] predicted that the major challenges in exascale computing will be:

1) Moving data around. At exascale, moving data between simulation and pre- or post-processing would be costly. Algorithms need to be designed to minimise the data movements.
2) Limitation of memory bandwidth.
3) Resiliency problem. Computation with millions and billions of cores will pose a challenge to error resiliency.
4) Power consumption.
5) I/O bandwidth.
6) Latency.

When it comes to pre- and post-processing, we expect the following challenges while developing algorithms for the exascale:

1) Parallelism. How to distribute the data, how do apply domain decomposition in order to achieve a optimal load balance will be a major issue.
2) Data storage: disk or memory. At exascale, it is preferable not to store all the data anymore. On-the-fly analysis of simulation results will be desired. However, how to cache the data in memory and what type of data is need for further processing are questions for scientists who design post-processing algorithms.
3) Data structure. Multi-resolution data structures provide an opportunity to downsample data fields, thus reducing the volume of data to be processed. How to access the data required also poses a challenge to data structure algorithms.
4) Scheduling. Parallelism for post-processing can be defined in different ways. Whether to use a load-on-demand approach or static distribution heavily depends on the choice of post-processing algorithm. How to speed up communication and optimise scheduling is of great importance.

IV. System Architecture

In this section, we described the work flow and architecture of our system. We primarily focus on the communication and integration of processing with the simulation component. The general work flow and system architecture is described. Following this, description of how this architecture can be realised within our co-design collaboration is provided.

A. Simulation

The core of HemeLB provides a lattice-Boltzmann solver for flood simulation with sparse geometries [1]. A pre-partitioning library ParMETIS is employed to decompose computational domains in order to allow parallisation. This domain decomposition is based on a load-balance approach.

B. Pre-processing

Pre-processing serves as a preparatory step for simulation, in order to create and optimise the geometry or computational effort for the actual simulation. In a general sense, pre-processing of the simulation data may contain the following steps

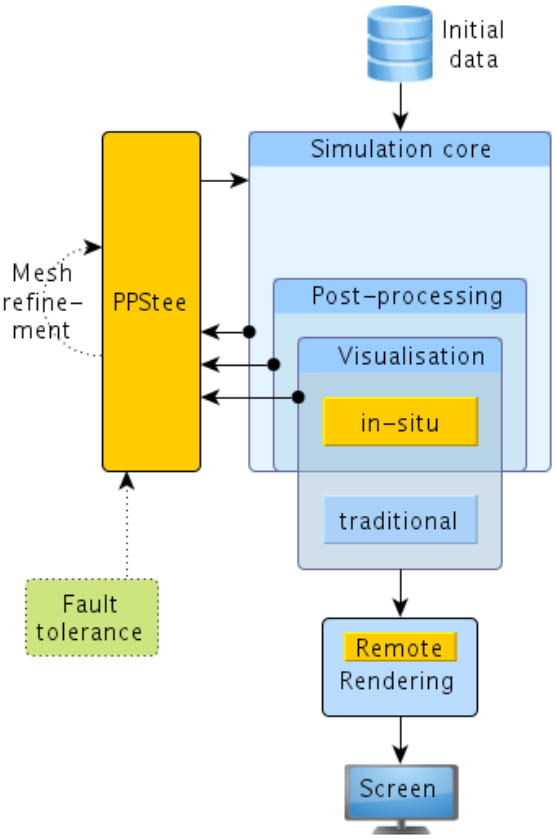


Fig. 2. System architecture of the co-design

1) Initialise geometry or computational mesh.
2) Load or distribute the initial mesh
3) Apply optimisation on geometry, such as mesh refinement in a certain region
4) Check and evaluate load balance
5) Repartition after certain simulation time steps, which can be conducted either at each single time step or every few steps.

Pre-processing starts with data preparation. Initial data like geometries or meshes has to be prepared for simulation computation. It is typically stored on the filesystem. A parallel file access by only a subset of all nodes helps reducing stress for the filesystem while providing sufficient speed. On the other hand, generation of mesh data could be done on-the-fly if the problem extent could be described simply enough.

Geometry optimisations are now applied if necessary or useful. Mesh refinement improves certain simulation hot spots; or globally generates intermediate grid points thus enhancing result precision. Other mesh quality improving techniques are possible. In an exascale environment, automating these procedures offers many advantages, including lower communication costs, less memory consumption, smaller data set sizes, to name a few.

The main aim of pre-processing is to balance load as evenly as possible across the system. If this primary requirement is

not matched, overall system performance is lost. Every parallel code deals with this burden, but for an exascale simulation this effect is clearly worse, as hundreds of thousands of cores possibly wait for only a couple of cores. Thus more effort is needed to lower the imbalance. Additionally, exascale magnifies another aspect of this load balance. Huge systems increase in performance and so does the wish for interactivity. Simulation steering, interactive visualisation and mesh adaptation become part of the simulation. A direct consequence is more sources of workload and communication costs in the whole simulation chain. It does not matter if these are tightly coupled to core computations or independent. These costs of other simulation parts, like visualisation, must be involved in the balance equation.

Also, this interactivity brings further tasks for pre-processing. Current simulations are governed by a one run policy. Simulations calculate lots of time steps and put out the result which is finally visualised. Interactivity, on the other hand, requires an intermediate result. Thus, only a small part of all timesteps is computed in one simulation cycle covering pre-processing, calculation and post-processing. The opportunity to adjust the partitioning mid-term is introduced. This repartitioning helps to improve load balance greatly.

In practice, when it comes to pre-processing for HemeLB, the following steps are specified:

1) Read in the geometry for blood vessel model
2) Apply partitioning on the geometry. This step only accounts for the fluid dynamics calculation and no other simulation parts.
3) No repartitioning will be applied. In this case, the geometry set up as well as distribution and loading are fixed.

HemeLB reads data from a two-level file format, where coarse grained blocks are described solely by the volume of fluid with each one. This data is used to perform an initial approximate load balance. A subset of the cores then read the detailed geometry data and distribute the data to those cores that require it. This approach minimises stress on the filesystem. Additionally, the number of reading cores enables control over the balance between file I/O and distribution communication.

This initial data distribution is a first start, but does not claim to be ideal. The geometry information now present is then passed to the partitioner, ParMETIS, which computes an optimised partition and HemeLB redistributes the data accordingly. This domain decomposition is then fixed for the duration of the simulation.

C. Post-processing

The process of analysing simulation results when simulation is finished is called post-processing. This process requires the computed raw data to be transformed to suitable representations by passing all or parts of the data through a post-processing pipeline, which typically consists of data extraction, filtering, mapping and visualisation stages.

Interactive exploration and visualisation methods have proven to be successful in analysing large-scale simulation results. This is a compute-intensive process making the requirement for efficient interactive exploration, such as the ability to move freely through the data, hard to meet.

Virtual Reality is a helpful tool in visualisation extreme-scale CFD data. Gerndt et. al [13] have presented a framework which provides parallel CFD post-processing in virtual environments (VR). Improved depth perception and free navigation in VR allow the scientist to explore their data in a more natural and informative way. Therefore, we also plan to include the use of virtual environment in our post processing.

1) In situ processing with HemeLB: Unlike most of traditional off-line data post-processing, *in situ* processing has become important in large scale simulation. It enables monitoring and analysis of the on-going simulation state. Understanding the science behind exascale blood simulation, requires the extraction of meaningful data from dataset of hundreds of terabytes and more [14]. With the ever increasing size of the simulation, the cost of moving the results from simulation to visualisation machine also dramatically increases. Kwan [15] pointed out it is preferable to not move the data at all, or to keep the moved data to minimum.

Applying the simulation and visualisation processes in parallel in an *in situ* manner allows the sharing of data, hence avoiding unnecessary data movement and output. Furthermore, *in situ* processing makes it possible for the scientist to analyse their on-going simulation, providing insight for further modifications of the simulation, which can then be immediately applied.

Within HemeLB, the *in situ* visualisation process proceeds as follows:

1) A simulation test is started on a cluster.
2) A steering client is connected to the simulation master node.
3) The client sends visualisation parameters (view point, fields to display, etc.) to the simulation
4) The simulation master propagates this to the visualisation component
5) The visualisation component requests the necessary data from the simulation and constructs the image
6) The image is returned the simulation master node and thence to the client.

2) Interactive visualisation:

A comparison of visualisation algorithms at exascale: As mentioned before, volume rendering, line integrals (including streamlines, pathlines, and streak-lines), particle tracing and line integral convolution (LIC) are the most desired visualisation for HemeLB data. We compare the possible limitations and challenges while applying these techniques at extreme-scale.

Table I summarises the pros and cons of each mentioned visualisation.

Visualisation technique	Volume rendering	Line integral	Particle tracing	LIC
Communication cost	low	high	high	medium
Load balance	can be optimised			good
Ease of parallelisation	easy	hard	hard	moderate

TABLE I
PROS AND CONS OF THE VISUALISATION TECHNIQUES

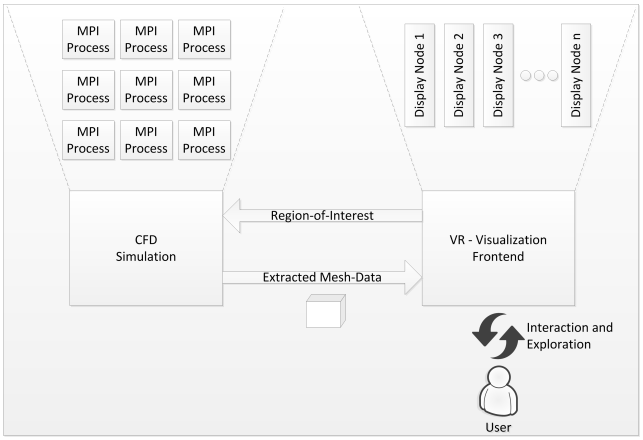
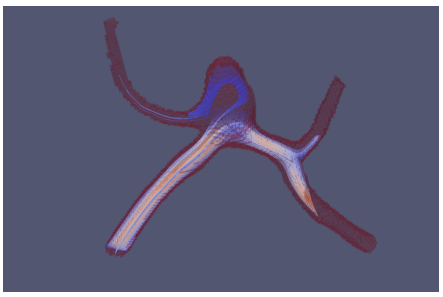
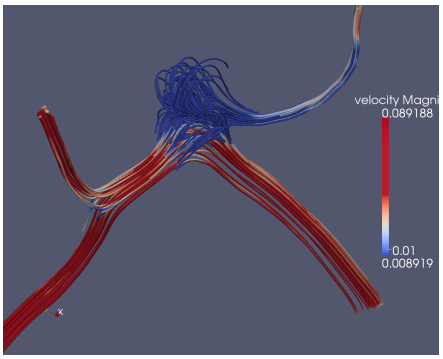


Fig. 3. Post-processing pipeline with user iteration [16]



(a) Example of volume rendering on a small aneurysm data set



(b) Example of streamline visualisation on a small aneurysm data set

Fig. 4. Examples of visualisation techniques

Figure 4(a) 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.

Line integrals, especially stream-lines and path-lines are useful tools for tracing and analysing flow fields. Figure 4(b) shows an example of the path-line tubes while visualising blood flows in an aneurysm dataset. 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.

3) Closing the loop: The ultimate goal of our co-design work is to close the loop by enabling computational steering (see figure 2). Post-processing and visualisation of the simulation results provides the user with possibilities to make a decision on how simulation can be changed or modified. Not only simulation parameters, geometries, mesh definition can be further modified, data decomposition and distribution can also be taken into consideration.

Conventional post-processing serves as a stand alone analysis tool for the simulation. With further functionality of computational steering, feedback will be given and put back into the simulation setup, allowing for a better convergence and more steady simulation. Until this point, we have closed the loop in our system, and connected pre- and post-processing with simulation in an interactive manner.

D. Challenges in implementation

When implementing visualisation algorithms in a parallel manner, special attention should be given to data distributions. Algorithms which need a lot of neighbourhood searching, such as path-lines, are challenging to implement in a distributed memory environment. The frequent search between cells results in a huge amount of communication between different cluster nodes, which is slow and energy demanding. On the other hand, techniques which can be implemented on a sub-mesh independent of the neighbouring ones are better suited to such an environment. Volume rendering, for instance, can be performed on each subdomain without any data exchange with the neighbours. Therefore, communication between different cluster nodes is avoided.

V. MULTI-RESOLUTION DATA STRUCTURE AND STREAMING

Moving a large amount of data causes huge latency. Moreover, the overwhelming amount of data waiting to be post-processed makes our data exploration non-interactive. Multi-

resolution data structures offer a solution to reduce the amount of data sent to post-processing, and therefore minimising the 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. This 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. Applying visualisation on the loaded subset of the whole data enables an immediate and preliminary graphical representation. One may argue that the context representation of the data set is not sufficient for a detailed analysis, however, it is commonly true that simplified and approximated results can be sufficient for deciding whether to terminate, or to modify parameters for the next iteration step or to wait for the final result.

The integration of multi-resolution data structures into HemeLB would enable the further types of *in situ* post-processing. At exascale, it will be a major challenge to provide visualisation and analysis interactively. Multi-resolution data analysis will be our only way to largely reduce the data size, to provide insight and to navigate through the whole data set. In order to use multi-resolution data structures, simulation data must be stored or cached in a hierarchical manner. Previous research on data hierarchy points to quadtrees and octrees. Each level on the tree corresponds to a set of data at a certain resolution. To access the data at certain level, one needs to explore the effectiveness of the corresponding tree structures. Effective ways of searching and traversal will be explored when using hierarchical data structures.

It is worth mentioning that the actual data hierarchy should be exploited in coordination with the post-processing algorithms. The partitioning and the data hierarchy must allow the post-processing with a fast data searching and allocation. Moreover, region of interest approaches can also be combined with multi-resolution data structure. First, the user can defined a region to be post-processed. Then, analysis and visualisation can be carried out on a refinable area.

VI. Conclusion

The discussions presented in this paper serve as a roadmap for designing pre- and post-processing for the HemeLB code at exascale. In this paper we have presented the architectural challenges at exascale pre- and post-processing. We sketched a closed-loop-system for the co-design of HemeLB and pre- and post-processing systems or libraries. *In situ* processing and computational steering will be two major directions when working at exascale. Copying data between the simulation cluster and a dedicated smaller scale visualisation cluster becomes impossible. Hierarchical data structures as well as parallel visualisation algorithms are discussed within the HemeLB framework. Parallelism must be pursued with great care at every opportunity and diverse algorithms must be considered simultaneously. For numerical simulations at exascale the combination of techniques such as *in situ* processing, computational steering, hybrid parallelisation and multi-resolution data structures are crucial to perform an interactive exploration of these simulations.

VII. Future work

We plan to extend the steering functionalities within the current HemeLB architecture. The proposed visualisation algorithms should be integrated as an *in situ* post-processing module into the HemeLB code in the near future. To enable a closed simulation loop from pre-processing over the simulation with a concurrent post-processing to a user-interface for simulation steering, we plan to integrate well-defined interfaces to all these parts. To evaluate if these extensions to the HemeLB solver can improve the complete simulation workflow, we will carry out tests and benchmarks on currently available large-scale cluster systems.

Acknowledgment

The authors would like to thank all our colleagues from the CRESTA project for their support and for many lengthy and fruitful discussions without which this work would not have been possible. The support of the European Commission through the Seventh Framework Programme (ICT-2011.9.13) under Grant Agreement no. 287703 is gratefully acknowledged.

References

[1] D. Groen, J. Hetherington, H. B. Carver, R. W. Nash, M. O. Bernabeu, and P. V. Coveney, "Analyzing and modeling the performance of the hemelb lattice-boltzmann simulation environment," *CoRR*, vol. abs/1209.3972, 2012.

[2] L. Biferale, F. Mantovani, M. Pivanti, F. Pozzati, M. Sbragaglia, A. Scagliarini, S. F. Schifano, F. Toschi, and R. Tripiccione, "An optimized d2q37 lattice boltzmann code on GP-GPUs," *Computers & Fluids*, no. 0, pp. –, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0045793012002265

[3] "Scotch, software package and libraries for sequential and parallel graph partitioning, static mapping, and sparse matrix block ordering, and sequential mesh and hypergraph partitioning." [Online]. Available: http://www.labri.fr/perso/pelegrin/scotch/

[4] "Data-management services for parallel applications." [Online]. Available: http://www.cs.sandia.gov/Zoltan/Zoltan_phil.html

[5] "Parallel graph partitioning and fill-reducing matrix ordering." [Online]. Available: http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview

[6] H. Childs, "Architectural challenges and solutions for petascale post-processing," *Journal of Physics: Conference Series*, vol. 78, no. 1, p. 012012, 2007.

[7] K.-L. Ma, C. Wang, H. Yu, K. Moreland, J. Huang, and R. Ross, "Next-generation visualization technologies: Enabling discoveries at extreme scale," *SciDAC Review*, no. 12, pp. 12–21, February 2009.

[8] B. Whitlock, J. M. Favre, and J. S. Meredith, "Parallel in situ coupling of simulation with a fully featured visualization system," in *Proceedings of the 11th Eurographics conference on Parallel Graphics and Visualization*, ser. EG PGV'11. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2011, pp. 101–109.

[9] H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K.-L. Ma, "In situ visualization for large-scale combustion simulations," *IEEE Comput. Graph. Appl.*, vol. 30, no. 3, pp. 45–57, May 2010.

[10] V. Pascucci, R. J. Frank, and A. J. Frank, "Hierarchical indexing for out-of-core access to multi-resolution data," Tech. Rep., 2001.

[11] Y. H. Qian, D. D'Humires, and P. Lallemand, "Lattice bgk models for navier-stokes equation," *EPL (Europhysics Letters)*, vol. 17, no. 6, p. 479, 1992. [Online]. Available: http://stacks.iop.org/0295-5075/17/i=6/a=001

[12] S. Ashby, P. Beckmann, and J. C. et.al, "The opportunities and challenges of exascale computing," 2012. [Online]. Available: http://science.energy.gov/~/media/ascr/ascac/pdf/reports/exascale_subcommittee_report.pdf

[13] A. Gerndt, B. Hentschel, M. Wolter, T. Kuhlen, and C. Bischof, "Viracocha: An efficient parallelization framework for large-scale CFD post-processing in virtual environments," in *Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, ser. SC '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 50–. [Online]. Available: http://dx.doi.org/10.1109/SC.2004.66

[14] K.-L. Ma, R. Ross, J. Huang, G. Humphreys, N. Max, K. Moreland, J. D. Owens, and H.-W. Shen, "Ultra-scale visualization: Research and education," *Journal of Physics*, vol. 78, June 2007, (Proceedings of SciDAC 2007 Conference).

[15] K.-L. Ma, C. Wang, H. Yu, and A. Tikhonova, "In situ processing and visualization for ultrascale simulations," *Journal of Physics*, vol. 78, June 2007, (Proceedings of SciDAC 2007 Conference).

[16] F. Chen, C. Wagner, M. Flatken, A. Gerndt, and H. Hagen, "Poster:enabling interactive mesh quality exploration of large scale cfd simulations in virtual environments," 2012.