In-situ Processing and Interactive Visualization for Large-Scaled Numerical Simulations

Fang Chen * Markus Flatken † Ingrid Hotz ‡ Andreas Gerndt §

Simulation and Software Technology, German Aerospace Center (DLR)

Abstract

With the increasing power of the HPC hardware systems, numerical simulations are heading towards exa-scale computing. Early inspection and analysis of on-going large simulations enables domain experts to obtain first insight into their running simulation process and intermediate results. Compared to conventional post-processing, such in-situ processing has the advantage of keeping data in memory, avoiding to store the large amount of raw data to disk, providing on-the-fly analysis, and preventing early failures in the simulation process. In this poster we present a distributed and scalable software infrastructure, which provides distributed in-situ data processing, feature extraction and interactive exploration at user's front-end. We have integrated and extended our system to multiple simulation applications, ranging from Lattice-Boltzmann blood flow simulation to grid based simulation for propulsion systems. A user-interactive front-end is integrated to our system, allowing to directly interact with the visualization of running simulations, gain insight, and make decisions.

1 Introduction

Given the complexity and scale of today's simulations, it is no longer a viable solution to store all simulation data to disk. Limited system I/O capacity hinders the simulation from outputting intermediate results. Therefore, it has become a common practice for large simulations to throw away results from intermediate time steps. To prevent simulation failure at an early stage, in-situ data analysis and visualization is becoming a necessity to enable domain experts to monitor whether a simulation is running smoothly and to obtain first insight into the resulting data.

The challenges of performing in-situ processing differ from those of traditional post-processing. First, a deep integration of the analysis and visualization algorithms into the simulation is required, which allows a conflict-free sharing of data. Second, the additionally memory requirements should be kept minimal. Third, the communication costs between cluster nodes should be kept low, which excludes global algorithms. E.g., algorithms searching through the whole data, distributed over a huge of number of parallel machines, are very expensive. Last but not least, network bandwidth becomes a bottleneck when the user is interacting with the data at the front-end application. Latency issues arise whenever huge amount of information exchanges happens between the user and the in-situ process.

A first insight into the ongoing simulation normally does not require complicated visualization algorithms. However, domain experts should have the possibility to use their domain knowledge to steer the visualization to focus on regions that they consider as important and to identify critical information which might lead to a modification of the next simulation design. Therefore, we consider user interaction as a key component of our system.

The presented system infrastructure extends previous work called Viracocha [1], a distributed post-processing system. The result is a more flexible and powerful system, which permits in-situ processing with the distributed simulation, supporting on-demand data analysis, and interactive exploration with current instance of simulation data.

As a pilot case, the proposed system has been integrated with HemeLB, a Lattice Boltzmann based software that simulates the flow of blood in intracranial aneurysms. Given the current problem sizes, our approaches yield promising results with respect to interactivity and scalability.

Network bandwidth becomes a bottleneck in two places: the bandwidth between simulation cluster and rendering machine/cluster, and the bandwidth between the rendering cluster to front-end display. The first one limits the size and complexity of features to be rendered, while the second constrains the image quality to be displayed. To further reduce latency, we have integrated multiple image compressing techniques.

2 System Architecture

Figure 1 illustrates the layout of our proposed software infrastructure. The body contains three major parts. First, in-situ data processing and parallel feature extraction take place on the large-scale cluster system together with the ongoing simulation. Next, depending on the size of resulting extracted features as well as the complexity of rendering algorithms, rendering is performed either on a

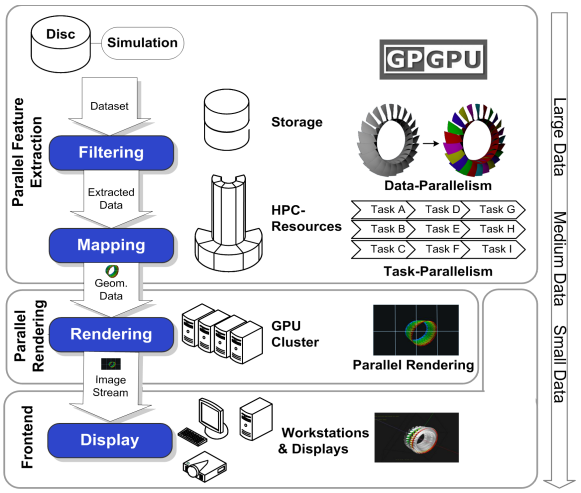

Figure 1: Infrastructure: The simulation and parallel feature extraction is executed on large cluster systems. Depending on the amount of extracted features rendering takes place on a smaller GPU-cluster or directly on the user's desktop.

*e-mail:fang.chen@dlr.de
†e-mail:markus.flatken@dlr.de
‡e-mail:ingrid.hotz@dlr.de
§e-mail:andreas.gerndt@dlr.de

IEEE Symposium on Large Data Analysis and Visualization 2014
October 9–10, Paris, France
978-1-4799-5215-1/14/$31.00 ©2014 IEEE

small scaled GPU-cluster or directly on a single front-end machine. Thirdly, resulting images are displayed either on a single desktop or in a virtual environment accompanied with different type of user interaction techniques.

For such a distributed software infrastructure, communication and data transfer are essential factors to enable interactivity and scalability. Data must be exchanged with minimal latency between different software layers, applications, and resources. Taking HemeLB as an example (Figure 2), we illustrate the data and communication flow within our system.

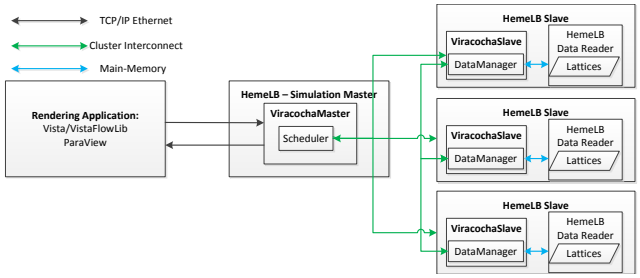

Figure 2: Communication and data flow: Every Viracocha process is embedded into HemeLB as a concurrent thread. The Viracocha scheduler is responsible to receive user requests from a front-end over TCP/IP and initiate the algorithm execution on the Viracocha slave instances. The data manager within the slave's have direct access to simulation data over main-memory and are able to communicate over the cluster interconnect with similar ones to exchange data when required.

A user sends filtering requests to the Viracocha application embedded into the HemeLB solver. The scheduler directly distributes this request to the slave processes. Each executed algorithm within a Viracocha slave is able to access the HemeLB simulation data over a data manager component. Furthermore, it is also possible to access data from other processes. As soon as the requested data is extracted, the results are streamed to the rendering application for immediate presentation.

2.1 Coupling the simulation with in-situ processing in a distributed system

To access the simulation data in-situ and perform on-demand filtering, we need to access the main-memory of each computing node. Since the numerical simulation is running as a parallel application, each process holding data has to pass the current simulation data to the cohabitant Viracocha process. A solver specific data extractor utilized by the Viracocha data manager is integrated into HemeLB software (Figure 2).

To guarantee high bandwidths and low latencies we have chosen to couple both parallel applications in the process-space (in-situ). Therefore, Viracocha is executed concurrently by threading within each solver process. Viracocha is based on the master/slave paradigm where two major types of instances exist. The Viracocha slave is in charge of algorithm execution including data access while the Viracocha master is responsible for receiving filtering requests and scheduling algorithm execution of the Viracocha slaves. Once the master receives a filtering request, the simulation is shortly disrupted after the actual iteration to access data snapshots and to apply desired data conversion. Then the simulation proceeds while the snapshot is used by the filtering operation to extract user defined features. We have chosen a snapshot approach where the simulation is only interrupt for a negligible time and the simulation process is not interfered further by Viracocha. Altogether, this concurrent execution and easily extendable data exchange allow for quick integration into other simulation codes.

2.2 Rendering of filtered data

Interactive visualization of large-scale simulation results not only requires parallel feature extraction, it also requires scalable rendering solutions. The major challenges for the rendering application are the preparation of the received data at high throughput, and the massive amount of main and GPU memory needed for the rendering. If the extracted features are huge, a single desktop machine becomes quickly overwhelmed. In this case, we place a scalable parallel rendering architecture between the feature extraction and the user's front-end. On top of this parallel rendering technique, our developed rendering application makes heavy use of multi-threading per process to prepare the data rendering. Therefore, it exploits today's and future multi-core CPU architectures.

2.3 Allowing user interaction

We developed an front-end application based on Vista and Vista FlowLib [2]. This application allows the user to interact with the in-situ visualization in a virtual environment (as shown in Figure 3). Using a fly-stick in front of a powerwall display, the user is able to perform tasks including sending request to couple or decouple data filtering, choosing data mapping and rendering algorithms, and navigating through the resulting visualization. As alternatives, we also support single desktop as front-end using mouse and keyboard as input devices which send out interaction commands.

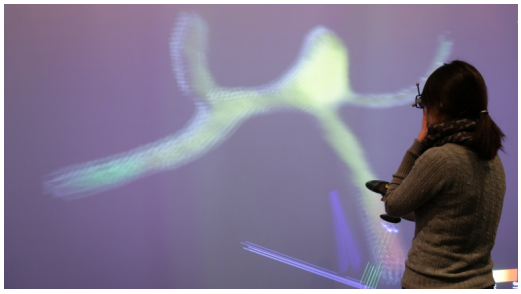

Figure 3: A scientist is interactively exploring the blood flow of an aneurism during an on-going HemeLB simulation.

3 Future Work

Work in progress includes extending Viracocha's caching mechanism to accommodate data from multiple time steps. This will enable in-depth exploration of data's temporal characteristics. Further effort will be put into the development of effective global visualization algorithms, where initial data decomposition plays an important role in minimizing communication overhead.

Acknowledgements

The authors want to thank University College London for the collaboration on HemeLB. Our research has been supported by the CRESTA project funded within the European Community's Seventh Framework Programme (ICT-2011.9.13) under Grant Agreement no. 287703.

References

[1] 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 *Supercomputing, 2004. Proceedings of the ACM/IEEE SC2004 Conference*, pages 50–50, Nov 2004.

[2] M. Schirski, A. Gerndt, T. van Reimersdahl, T. Kuhlen, P. Adomeit, O. Lang, S. Pischinger, and C. Bischof. Vista flowlib - framework for interactive visualization and exploration of unsteady flows in virtual environments. In *Proceedings of the Workshop on Virtual Environments 2003*, EGVE '03, pages 77–85, New York, NY, USA, 2003. ACM.