In-Situ Processing and Visualization for Ultrascale Simulations

Kwan-Liu Ma, Chaoli Wang, Hongfeng Yu, Anna Tikhonova

Department of Computer Science, University of California at Davis, One Shields Avenue, Davis, CA 95616

SciDAC Institute for Ultrascale Visualization (IUSV)

E-mail: `ma@cs.ucdavis.edu`

Abstract. The growing power of parallel supercomputers gives scientists the ability to simulate more complex problems at higher fidelity, leading to many high-impact scientific advances. To maximize the utilization of the vast amount of data generated by these simulations, scientists also need scalable solutions for studying their data to different extents and at different abstraction levels. As we move into peta- and exa-scale computing, simply dumping as much raw simulation data as the storage capacity allows for post-processing analysis and visualization is no longer a viable approach. A common practice is to use a separate parallel computer to prepare data for subsequent analysis and visualization. A naive realization of this strategy not only limits the amount of data that can be saved, but also turns I/O into a performance bottleneck when using a large parallel system. We conjecture that the most plausible solution for the peta- and exa-scale data problem is to reduce or transform the data in-situ as it is being generated, so the amount of data that must be transferred over the network is kept to a minimum. In this paper, we discuss different approaches to in-situ processing and visualization as well as the results of our preliminary study using large-scale simulation codes on massively parallel supercomputers.

1. Introduction

To date, most large data visualization methods rely on an offline pre-processing step to reduce the data. Visualization is almost exclusively done as a *post-processing* step. Even though it is desirable to monitor some of the simulation stages, the cost of moving the simulation output to a visualization machine could be too high to make interactive visualization feasible.

A better approach is not to move the data, or to keep the data that must be moved to a minimum. That is, both simulation and visualization calculations run on the same parallel supercomputer so the data can be shared. Such simulation-time *co-processing* can render images directly or extract features, which are much smaller than the full raw data, to store for later examination. As a result, reducing both the data transfer and storage costs early in the data analysis pipeline optimizes the overall scientific discovery process. In practice, however, this approach was seldom adopted because of two reasons. First, most scientists were reluctant to use their supercomputer time for visualization calculations. Second, it could take a significant effort to couple the parallel simulation code with the visualization code; in particular, the domain decomposition optimized for the simulation is often unsuitable for parallel visualization, resulting in the need to replicate data for speeding up the visualization caculations. Hence, the common practice was to store only a small fraction of the data or to study the stored data at a coarser resolution, which defeats the original purpose of performing the high-resolution simulations.

To enable scientists to study the full extent of the data generated by their simulations and for us possibly realize the concept of steering simulations at extreme-scale, we ought to begin investigating the option of *in-situ processing and visualization.* In fact, many of the scientists we have been working with are convinced that simulation, run-time feature extraction, for example, is a feasible solution to their large data problem. A key fact is that during the simulation time, all relevant data about the simulated field are readily available for the extraction calculations. We have been experimentally studying in-situ processing and visualization for selected applications to understand the impact of this new approach on the simulations, subsequent visualization tasks, and how scientists do their work. This rest of this paper provides a comprehensive discussion of in-situ processing and visualization for ultra-scale simulations using some of the results from our study.

2. Related Work

Integrating analysis and visualization into the simulation pipeline will be a trend, but it is not new. Many researchers have experimented with this approach. For the reasons we gave in previous section, running tightly coupled simulation and visualization on the same parallel computer was rarely done in practice. Even though some have demonstrated such runtime visualization of simulation on a parallel computer but the scale of the system and problem is rather small [16, 11].

The other type relies on a more loosely coupled setting. For example, Vista [19], VASE [7], SCIRun [15, 9], and CUMULVS [3] are among the computational steering environments that can support runtime tracking of simulations in this manner. Vista is an early remote visualization system that offers an environment for effective debugging and program development with little or no instrumentation of the underlying application. VASE defines an abstraction for a steerable program structure and provides tools that create and manage collections of these codes. SCIRun is a problem solving environment that uses an object-oriented data flow approach to enable the user to control scientific simulations interactively by varying boundary conditions, model geometries, and computational parameters. CUMULVS is a library that, in addition to providing access to distributed data at runtime, also supports fault-tolerance to failures by checkpointing. The ability to steer a large-scale simulation allows the scientists to "close the loop" and respond to simulation results as they occur by interactively manipulating the input parameters. As changes in parameters become more instantaneous, the causal effects become more evident, which helps scientists develop intuition, understanding, and insight into their modeling, algorithms, and the data being studied.

In line with this strategy, others have also demonstrated the power of run-time visualizing, monitoring, and steering of specific large-scale scientific applications. Beazley and Lomdahl [1] took a command-driven approach for controlling, analyzing, and visualizing large-scale molecular dynamics simulations. Their primarily focus was on issues including memory efficiency, long running jobs, extensibility, and building steerable applications from existing code. Harrop et al. [5] presented a tool called TierraLab that is built on top of a MATLAB interface and extends the geoscientists' existing tomography code with runtime interaction capabilities. Modi et al. [13] introduced a new, light-weight steering system based on a client/server programming model and demonstrated the effectiveness of the system on a real-time wake-vortex simulations for multiple aircraft running on a parallel Beowulf cluster. More recently, Insley et al. [6] described an application that aids in the monitoring and analysis of a simulation of the human arterial tree. Using multiple views of data, their application provides visual feedbacks about the status of the on-going simulation at different levels of detail.

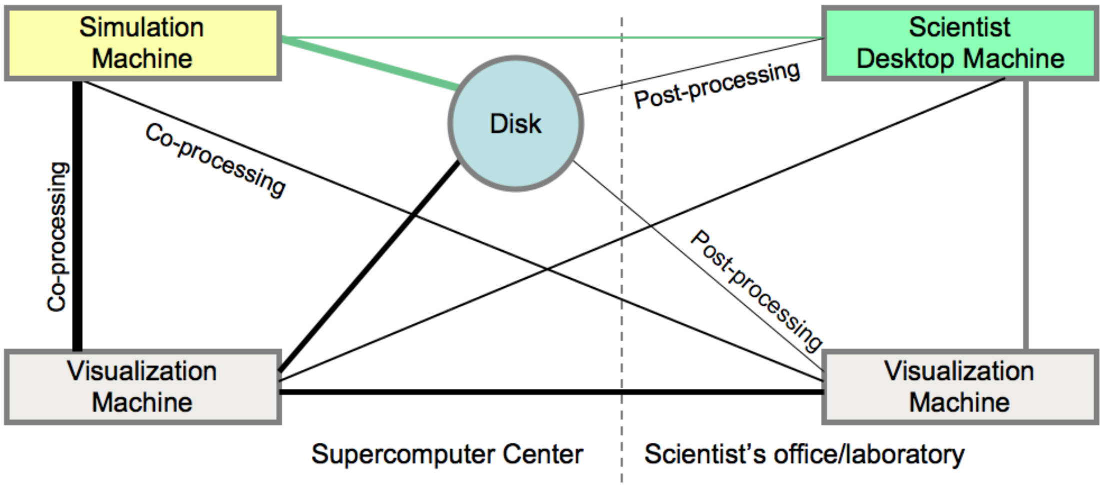

Figure 1. Post-processing, co-processing, and in-situ processing and visualization in the context of ultra-scale scientific simulations. The blue connections may be used for in-situ processing and visualization.

3. Co-Processing and Post-Processing Visualization

So far, routine data analysis and visualization of large-scale simulations running at a supercomputing facility has been done almost exclusively with co-processing or post-processing. Co-processing using a dedicated visualization machine, nowadays a PC cluster, connected to the supercomputer with a fast network. Simulation output is directly transferred to the visualization machine for immediate processing, and the resulting imagery is either stored to a disk or shipped to a desktop machine for viewing. For simulations that can take many days or even weeks to finish, or in the case a dedicated visualization machine is not available when running the simulation, the simulation output is simply dumped to a mass-storage system and studied at later time. Such post-processing, if done over a long-area network, very much limits what aspect and how much of the data that scientists can look at. Figure 1 show different settings of post-processing, co-processing, and in-situ processing/visualization. Exactly which kind of processing should be used depends on many factors including the performance of disk and network, the nature of the simulation, the type of visualization desired, and the openness of the scientists to new data analysis and visualization options.

4. In-Situ Processing

Traditionally, the focus of supercomputing has been on the inner kernel of scientific simulations, i.e., the solver. The following data analysis and visualization tasks are typically relegated to a separate computing environment, such as a desktop PC or a PC cluster, which is much less powerful than the supercomputer. As we move into the era of petascale computing, the enormous amount of data produced by large-scale scientific simulations must be processed, decimated, and summarized, resulting in a much condensed representation of the data for subsequent analysis and visualization in an effective and efficient manner.

For large-scale simulations, in-situ processing is the most effective way to reduce the data output since the reduction can benefit all the subsequent data movement and processing. A variety of data reduction methods can be utilized to enable interactive data analysis and visualization. On the other hand, in-situ processing may also be performed to extract selected features of interest, which can then be used for offline examination and assessment. Compared to the conventional post-processing approach, performing data reduction and feature extraction in situ is a more appealing approach, provided that the processing calculations would only take a small portion of the supercomputer time. With in-situ processing, if done by carefully utilizing domain knowledge, there is no need to keep the complete original resolution data, which saves a considerable amount of disk space for data storage and network bandwidth for data transmission.

4.1. Data Reduction

Common data reduction techniques include subsampling, quantization, and transform-based compression techniques. The simplest way to achieve data reduction is to subsample the data in either the spatial or temporal domain such that the resulting reduced data matches our capability to store and process it for analysis and visualization. For example, in-situ subsampling is essentially what scientists have been using to keep their simulation output more manageable. The common practice is to skip time steps based on the size of the available storage space and network bandwidth. The number of time steps skipped may be as many as hundreds, which creates a major challenge to temporal-space visualization and animation. This problem becomes even worse if vector field visualization is desired because, for example, tracing of pathlines is anticipated to be inaccurate. Therefore, other data reduction methods should be investigated.

Most scientific simulations output data in the form of single precision (32-bit) or double precision (64-bit) floating point numbers. Although scientists usually prefer to preserve the accuracy of their simulation results, it may not always be necessary to use the high-precision floating point data for data analysis and visualization. For example, if data analysis focuses on the shift of data values through the simulation sequence, then the actual precision of data is less of a concern. To make use of hardware acceleration for rendering, data is often stored in the texture memory at 8- or 16-bit precision. If we would know a priori that only a reduced precision of the data will be used for visualization, quantization can be used as a cost-effective way to reduce data.

The simplest quantization method is direct scalar quantization, which maps the range of original data values to a much smaller range. Scalar quantization is a light-weighted option as it accesses only a single data value at a time and is very fast to perform. Scalar quantization could be either uniform or non-uniform. With uniform quantization, the only information needed is the minimum and maximum data values in the global domain and the number of quantization levels. However, the quantization error introduced by this approach could be substantial when data values are not uniformly distributed in the range. Non-uniform quantization, such as the Lloyd-Max method [4], improves this by modifying the quantizer boundaries to match the data statistics. Another promising alternative is to use adaptive scalar quantization, where the quantization boundaries are selected on the fly online to reflect the local statistical characteristics of the data. For example, the Jayant quantizer [8] is a good choice because it only requires a one-word auxiliary memory while offering superior quantization performance.

Unlike scalar quantization, vector quantization (VQ) or block quantization groups data values into the form of vectors and encodes them as single units. The most common Linde-Buzo-Gray (LBG) algorithm [10] for VQ is similar to the K-means method [12] in data clustering. The usual VQ model requires a global data sample for codebook training. Given a large input data set, it would take a considerable amount of time to train the codebook and space to store the codebook. An adaptive variant of VQ exists where the codebook is adjusted online according to the local data statistics and no codebook training is required. We have employed vector quantization for the encoding of data values from multiple time steps and even multiple variables in a post-processing step [2]. Based on our test results, even though the compression performance is appealing, VG is too computational expensive as an in-situ processing method.

Another category of data reduction is transform-based compression, commonly used in block-based data encoding. In the transform-based encoding, the data is subdivided into blocks and each block is transformed from the spatial domain to the frequency domain. The transform

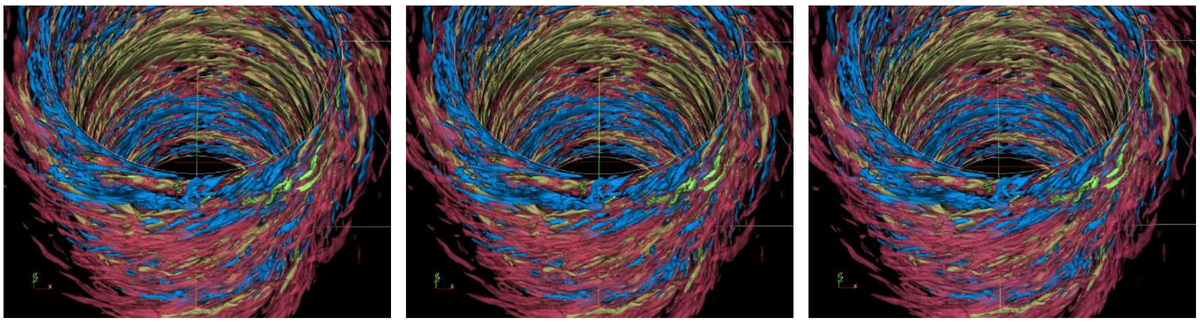
(a) original (b) quantized (c) DCT-compressed

Figure 2. Isosurfaces generated using the original data (32-bit) obtained from a magnetohydrodynamics simulation, quantized data (8-bit), and block-based DCT-compressed data. The isosurfaces are generated for the w-component of velocity with the following isovalues: -0.1, -0.2, 0.05, 0.1.

by itself is reversible, and does not compress the data. The transform puts more energy into fewer coefficients and results in a more energy-compact representation. This allows the less important lower energy coefficients to be quantized more coarsely, thus requiring less storage. The advantage of transform-based encoding is that it operates on each data block independent of other blocks. Therefore, only a single block memory buffer is sufficient and efficient encoding can be achieved with a moderate block size. If the simulation would also use a block-based domain decomposition, then transform encoding could be considered.

The discrete cosine transform (DCT) and the wavelet transform are among the most popular transform-based encoding and compression approaches. DCT has good information packing qualities and tends to have less error at the boundaries of a sequence, which is essential to avoid discontinuities between blocks. The wavelet transform can also be configured for block-wise encoding by imposing boundaries on the transform. For example, a common approach is to apply the wavelet transform recursively within each block, resulting in a local multi-scale or multiresolution data representation. The compression of the wavelet coefficients at different scales provides data reduction. Such a multiresolution representation of the data could be very useful as it allows various levels of detail to be selected according to the visualization requirements and the available computing resources at runtime. Figure 2 shows examples of isosurfaces generated from the original, quantized, and DCT-compressed data. As we can see, the loss of quality due to both quantization and DCT compression is minimal. According to our evaluation so far, transform-based compression is a better choice for in-situ data reduction in terms of both its cost and performance. Additionally, it is possible to employ several data reduction methods (e.g., quantization and subsampling) together to reduce the data to a greater extent.

4.2. Feature Extraction

Large-scale scientific simulations generate massive amount of data that must be validated and analyzed for understanding and possibly new discovery. This often boils down to identifying features in the data. But what is exactly a *feature*? Different disciplines clearly carry different definitions of features. Generally speaking, a feature is a small subset of the raw data isolated with domain knowledge and represents a particular physical structure, pattern, or event of interest. Some examples include vortices, shocks, eddies, critical points, etc. These features can be categorized, and used to characterize and break down the overall modeled physical

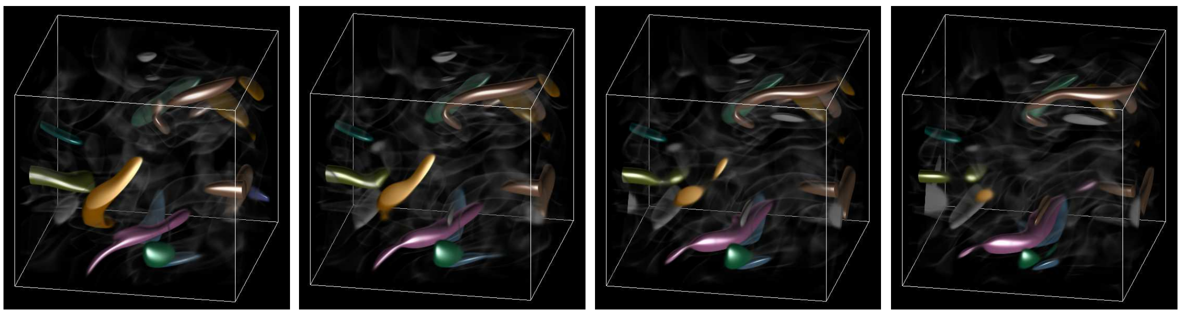

Figure 3. Feature tracking on a turbulent vortex data set [14]. Four selected time steps are shown. Each tracked feature is rendered with a distict color.

phenomenon. The saving in storage space with feature extraction can be very significant; however, scientist do not always know exactly what to extract and track in their data.

In three-dimensional volume visualization, a feature is commonly referred to as a certain space-time coherent region or object of interest [17]. Such a feature could be an isosurface component in scientific data or a classified material in medical data. Basic visualization algorithms exist for feature identification, extraction, and tracking, which incorporate principles from image processing, computer vision, and machine learning. For example, Figure 3 shows the results of feature tracking on a turbulent vortex flow data set using a new prediction and morphing approach [14]. This approach is low cost and incremental so it is perfect for in-situ processing. Another viable approach is to employ machine learning as demonstrated in [20]. The basic idea is to capture scientists' domain knowlege through interactive visualization augmented with an intelligent system, and then apply that knowledge to feature extraction and tracking. Better and better results can be obtained as more knowlege is accumulated over time.

Feature extraction and tracking remains to be an active area of research in the visualizatoin community. For many applications, feature extraction and tracking has yet to be considered. In contrast to traditional data reduction methods described in Section 4.1, feature extraction can be treated as a *high-level* data reduction method that explicitly takes into account domain knowledge to solve specific problems. Focusing on features effectively reduces the amount of data that has to be saved and studied. Beyond this saving, even more importantly, the ability to extract and track features also facilitates quantitative evaluation of modeled phenomena as well as comparative study of features obtained from different simulation runs or codes.

4.3. Quality Assessment

If data has to be reduced, the corresponding information loss must be conveyed to the user of the reduced data set. Quality assessment thus plays a crucial role in large-scale data analysis and visualization since in many cases the reduced versions of data rather than the original version are used for evaluating the simulation and modeled phenomena. In addition, the data itself could also be distorted or corrupted during the transmission over a network. To compare the quality of the reduced or distorted data, it is imperative to identify and quantify the loss of data quality. Most existing data quality metrics, such as the mean square error and the peak signal-to-noise ratio, require access to the original data and are *full-reference* methods. They are not applicable to petascale applications simply because the original data is too large to acquire or compare in an efficient way.

A more promising solution is to take a *reduced-reference* approach [22], where only important statistical information extracted from the original data is used for quality evaluation. If such information can be extracted while the simulation is running, we are able to use it as the basis

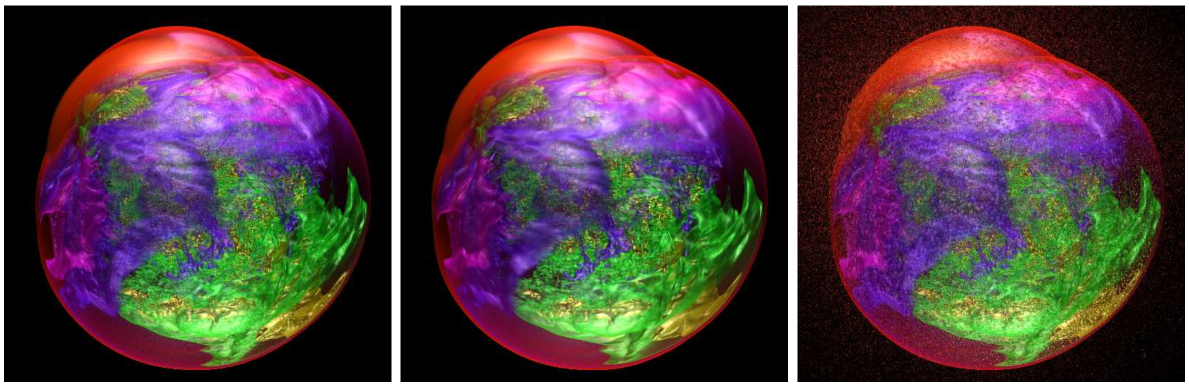
(a) uniform quantization (b) Gaussian filter (c) salt-and-pepper noise

Figure 4. Cross-type quality assessment on a low resolution supernova data set ($432\times432\times432$). The data quality degrades as the overall distance increases from (a) to (c).

metric for offline quality assessment without the need to access the original data. This is achieved by computing the distance of feature components of the reduced or distorted version of data with those derived from the original data, which gives us an indication of quality loss in relation to the original data. Such feature information also enables us to perform a cross-comparison of data with different reduction or distortion types. For example, Figure 4 gives quality assessment results on the supernova data set under three different distortion types: uniform quantization (with 1024 levels), Gaussian filter (using a kernel of size $5 \times 5 \times 5$ and a standard deviation of 1.0), and salt-and-pepper noise (with an equal probability of 1/216 for the bipolar impulse). The comparison is achieved using the statistical information extracted from the original data in the wavelet domain as the quality metric [21]. Moreover, by forcing some of its feature components to match those of the original data, we may repair the reduced or distorted data for possible quality improvement for offline visualization.

5. In-Situ Visualization

In many cases, it is also desirable and feaisble to render the data in-situ for monitoring and steering a simulation. Even in the case runtime monitoring is not practical due to the length of simulation run or the nature of the calculations, it could be still desirable to generate an animation characterizing selected parts of the simulation. This in-situ visualization capability is especially helpful when a significant amount of the data is to be discarded. The animation could capture the integrity of the simulation with respect to a particularly important aspect of the modeled phenomenon.

5.1. Issues

Compared with a traditional visualization task that is performed in a postprocessing fashion, in-situ visualization brings some unique challenges. First of all, the visualization code must interact directly with the simulation code, which requires both the scientist and the visualization specialist commit to this integration effort. To optimize memory usage, they have to find a way for the simulation and visualization codes to share the same data structures storing the data. Second, visualization workload balancing is more difficult to achieve since the visualization has to comply with the simulation architecture and builds tightly coupled with it. Unlike parallelizing visualization algorithms for standalone processing where we can partition and distribute data best suited for the visualization calculations, for in-situ visualization, data partition and distribution is dictated by the simulation code. Moving data frequently among

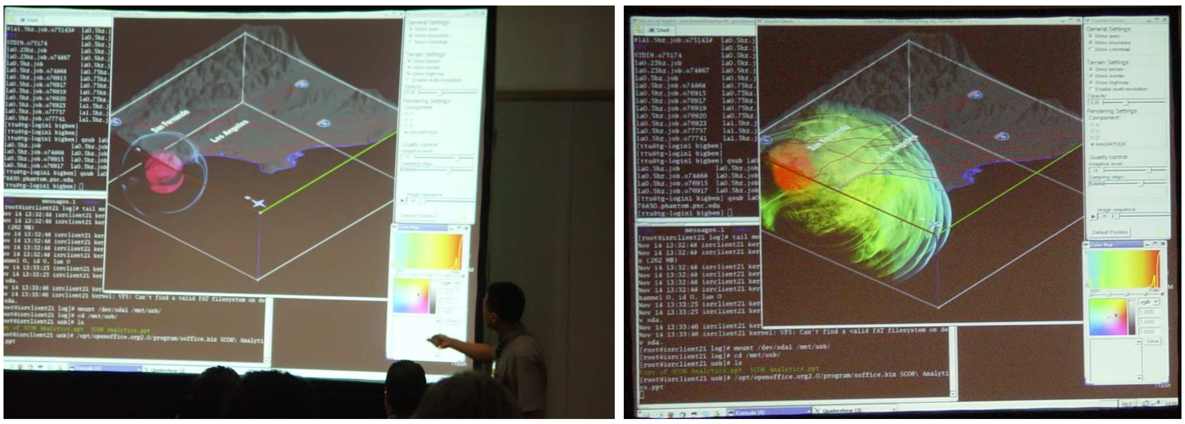

Figure 5. A live demonstration of remote, in-situ visualization of a terascale earthquake simulation at SC '06. This work won the HPC Analytics Challenge.

processors is not an option for visualization processing. We need to rethink to possibly balance the visualization workload so the visualization is at least as scalable as the simulation. Finally, the visualization calculations must be low cost, with decoupled I/O for delivering the rendering results while the simulation is running. Since the visualization calculations on the supercomputer cannot be hardware accelerated, we must find other ways to simplify the calculations such that adding visualization would take away only a very small fraction of the supercomputer time allocated to the scientist.

5.2. A Case Study

We have realized in-situ visualization for a terascale earthquake simulation [18]. This work also won the HPC Analytics Challenges of the SC 2006 Conference [23] because of the scalability and interactive volume visualization we demonstrated. Over a wide-area network, we were able to interactively change view angles, adjust sampling steps, edit color and opacity transfer function, and zoom in and out for visually monitoring the simulation running on 2048 processors of a supercomputer at the Pittsburgh Supercomputing Center. Figure 5 shows photos of the live demonstration at the Conference.

We were able to to acheive high parallel efficiency exactly because we made the visualization calculations, direct volume rendering, use the data structures used by simulation code, which removes the need to reorganize the simulation output and replicate data. Rendering is done in-situ using the same data partitioning made by the simulation, and thus no data movement is needed among processors. Similar to the traditional parallel volume rendering algorithms, our parallel in-situ rendering pipeline consists of two stages: parallel rendering and parallel image compositing. In the rendering stage, each processor renders its local data using software raycasting. Note that this stage may not be balanced given a set of visualization parameters and the transfer function used. In the image compositing stage, a new algorithm is designed to build a communication schedule in parallel on the fly. The basic idea is to balance the overall visualization workload by carefully distributing the compositing calculations. This is possible because parallel image compositing uses only the data generated by the rendering stage and is thus completely independent of the simulation. Furthermore, our new compositing algorithm is optimal in the sense that it not only helps balance the overall workload of rendering and compositing, but also substantially reduces network traffic typically required in post-processing rendering and visualization solutions.

For implementation of in-situ visualization, no significant change is needed for the earthquake simulation code for the integration. The only requirement for the simulation is to provide APIs for the access of the simulation internal data structure, which does not require much effort in practice. Furthermore, because all the access is read operation, the simulation context is not affected by the visualization calculations. The advantage of our approach is obvious: in order to incorporate in-situ visualization, the scientists do not need to change their code, but only to provide an interface for the visualization code to access their data, as everything else is taken care of by the visualization part. This approach is certainly most acceptable by the scientists.

6. Conclusion

By 2011, it is anticipated that high performance computing systems running at sustained petaflop speeds will be a reality for scientists and engineers to perform simulations at the peta- and exa-scale. To possibly understand such extreme-scale simulations and extract meaning from petabytes of the simulation output, it is imperative for domain scientists and visualization researchers to work closely together from now so a a viable solution will be ready by then. In this paper, we discuss the grand challenges facing petascale data analysis and visualization, and propose in-situ processing and visualization as a promising solution for ultrascale simulations. While we have demonstrated some success in realizing this solution, furhter research and experimental studies are needed to derive a set of guidelines and usable visualization software infrastructure for others to adopt the in-situ approach.

Acknowledgements

This research was supported in part by the U.S. National Science Foundation through grants CCF-0634913, CNS-0551727, OCI-0325934, and CCF-9983641, and the U.S. Department of Energy through the SciDAC program with Agreement No. DE-FC02-06ER25777, DOE-FC02-01ER41202, and DOE-FG02-05ER54817.

We would like to thank all our DOE and SciDAC application scientist collaborators. You are all critical in our understanding of the needs of the community and provide invaluable feedback during the research and development process.

References

[1] Beazley, D. M., and Lomdahl, P. S. Lightweight computational steering of very large scale molecular dynamics simulations. In *Proceedings of ACM/IEEE Supercomputing 1996 Conference* (1996).

[2] Fout, N., Ma, K.-L., and Ahrens, J. Time-varying multivariate volume data reduction. In *Proceedings of ACM Symposium on Applied Computing* (March 2005), pp. 1224–1230.

[3] Geist, G. A., Kohl, J. A., and Papadopoulos, P. M. CUMULVS: Providing fault-tolerance, visualization and steering of parallel applications. *International Journal of High Performance Computing Applications 11* (1997), 224–236.

[4] Gonzalez, R. C., and Woods, R. E. *Digital Image Processing (2nd Edition).* Prentice Hall, 2002.

[5] Harrop, C. W., Hackstadt, S. T., Cuny, J. E., Malony, A. D., and Magde, L. S. Supporting runtime tool interaction for parallel simulations. In *Proceedings of ACM/IEEE Supercomputing 1998 Conference* (1998).

[6] Insley, J. A., Papka, M. E., Dong, S., Karniadakis, G., and Karonis, N. T. Runtime visualization of the human arterial tree. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (1997), 810–821.

[7] Jablonowski, D. J., Bruner, J. D., Bliss, B., and Haber, R. B. VASE: The visualization and application steering environment. In *Proceedings of ACM/IEEE Supercomputing 1993 Conference* (1993).

[8] Jayant, N. S. Adaptive quantization with a one-word memory. *Bell System Technical Journal 52* (1973), 1119–1144.

[9] Johnson, C. R., Parker, S. G., Hansen, C., Kindlmann, G. L., and Livnat, Y. Interactive simulation and visualization. *IEEE Computer 32*, 12 (1999), 59–65.

[10] Linde, Y., Buzo, A., and Gray, R. An algorithm for vector quantizer design. *IEEE Transactions on Communications 28*, 1 (1980), 84–95.

[11] Ma, K.-L. Runtime volume visualization of parallel cfd. In *Proceedings of Parallel CFD Conference 1995* (1995), pp. 307–314.

[12] MacQueen, J. Some methods for classification and analysis of the multivariate observations. In *In Proceedings of the Fifth Symposium on Math, Statistics, and Probability* (1967), pp. 281–297.

[13] Modi, A., Long, L. N., and Plassmann, P. E. Real-time visualization of wake-vortex simulations using computational steering and beowulf clusters. In *Proceedings of International Conference on High Performance Computing for Computational Science 2002* (2002), pp. 464–478.

[14] Muelder, C., and Ma, K.-L. Rapid feature extraction and tracking through region morphing. Tech. Rep. Technical Report No. CSE-2007-25, Computer Science Department, University of California at Davis, June 2007.

[15] Parker, S. G., and Johnson, C. R. SCIRun: A scientific programming environment for computational steering. In *Proceedings of ACM/IEEE Supercomputing 1995 Conference* (1995).

[16] Rowlan, J., Lent, E., Gokhale, N., and Bradshaw, S. A distributed, parallel, interactive volume rendering package. In *Proceedings of IEEE Visualization 1994* (1994), pp. 21–30.

[17] Silver, D. Object-oriented visualization. *IEEE Computer Graphics and Applications 15*, 3 (1995), 54–62.

[18] Tu, T., Yu, H., Ramirez-Guzman, L., Bielak, J., Ghattas, O., Ma, K.-L., and O'Hallaron, D. R. From mesh generation to scientific visualization: An end-to-end approach to parallel supercomputing. In *Proceedings of ACM/IEEE Supercomputing 2006 Conference* (2006).

[19] Tuchman, A., Jablonowski, D., and Cybenko, G. Run-time visualization of program data. In *Proceedings of IEEE Visualization 1991* (1991), pp. 255–261.

[20] Tzeng, F.-Y., and Ma, K.-L. Intelligent feature extraction and tracking for visualizing large-scale 4d flow simulations. In *Proceedings of ACM/IEEE Supercomputing 2005 Conference* (2005).

[21] Wang, C., and Ma, K.-L. A statistical approach to volume data quality assessment. Tech. Rep. CSE-2007-21, Department of Computer Science, University of California, Davis, March 2007.

[22] Wang, Z., Wu, G., Sheikh, H. R., Yang, E.-H., and Bovik, A. C. Quality-aware images. *IEEE Transactions on Image Processing 15*, 6 (2006), 1680–1689.

[23] Yu, H., Tu, T., Bielak, J., Ghattas, O., Lopez, J. C., Ma, K.-L., O'Hallaron, D. R., Ramirez-guzman, L., Stone, N., Taborda-Rios, R., and Urbanic, J. Remote runtime steering of integrated terascale simulation and visualization, 2006. HPC Analytics Challenge, ACM/IEEE Supercomputing 2006 Conference. (http://sc06.supercomputing.org/news/press_release.php?id$\bar{1}$4).