DOI: 10.14529/jsfi160402

In Situ Visualization and Production of Extract Databases

Brad J. Whitlock[1]***, Earl P. N. Duque***[2]

© The Authors 2016. This paper is published with open access at SuperFri.org

Simulations running at high concurrency on HPC systems generate large volumes of data that are impractical to write to disk due to time and storage constraints. Applications often adapt by saving data infrequently, resulting in datasets with poor temporal resolution. This can make datasets difficult to interpret during post hoc visualization and analysis, or worse, it can lead to lost science. *In Situ* visualization and analysis can enable efficient production of small data products such as rendered images or surface extracts that consist of polygonal geometry plus fields. These data products are far smaller than their source data and can be processed much more economically in a traditional *post hoc* workflow using far fewer computational resources. We used the SENSEI and Libsim *in situ* infrastructures to implement rendering workflow and surface data extraction workflows in the AVF-LESLIE combustion code. These workflows were then demonstrated at high levels of concurrency and showed significant data reductions and limited impact on the simulation runtime.

Keywords: In Situ, High Performance Computing, Visualization, Extract Database, SENSEI, Libsim, Workflow.

Introduction

Today's large scale simulations run on HPC systems and generate far more data than can be practically saved or analyzed. HPC system design emphasises fast computations and I/O to and from these systems is often a secondary concern, leading to an asymmetry in which computed data often cannot be written to disk without resorting to strategies that sacrifice the temporal resolution of the data (saving infrequently). Recent developments explore the use of node local storage such as *burst buffers* that give applications a fast, convenient buffer to store results while they are staged out to the main I/O system. However, such hardware is not yet commonplace and other strategies such as *in situ* computations are emerging in production software as a mechanism to manage the data problem by reducing data that must be stored. Without *in situ*, the traditional *post hoc* workflow requires large simulation data to make costly trips to and from the I/O system, slowing both the simulation and later the visualization and analysis codes invoked to process the data. *In Situ* works by integrating analysis and visualization with the simulation so these operations may take place while the data are still in main memory, and are thus far less expensive to access. The data products produced *in situ*, whether they are statistics, rendered images, or even surface geometry extracts, are often orders of magnitude smaller than the memory-resident data and can be saved out far more economically.

In situ visualization is usually inexpensive enough to be applied frequently, actually improving access to spatio-temporal data that is often not saved or is undersampled due to time and storage constraints. Sometimes *in situ* does have limitations if the data products saved during the simulation run lack enough information for analysis. For instance, *in situ* is often ideal for creating statistics and rendered images but those data products may be less useful for actual exploration of data unless many images are saved. To permit better exploration of data after the fact, *in situ* can be extremely useful in the generation of extract databases. An extract database consists of extracted polygonal geometry plus scalar and vector field data. Extract databases can drastically reduce the amount of data being saved while still providing enough information

[1]bjw@ilight.com, Intelligent Light, Rutherford, New Jersey, USA
[2]epd@ilight.com, Intelligent Light, Rutherford, New Jersey, USA

to perform useful *post hoc* visualization and data analysis. The use of extract databases enables flexibility since most of the costly data reductions occur *in situ* and the reduced datasets can be visualized on more modest compute resources. By storing the geometry and fields in an extract database, it is still possible to create derived fields, render the geometry from various viewpoints, perform surface integrations, and many other operations that would not be possible on strictly image-based data products. The potential for massive data reduction, massive time eliminated when extracting dataset features, and the ability to later visualize features of interest, as opposed to static rendered images, are what make extract databases compelling.

In this study, we implement a workflow that uses *in situ* to perform both rendering and extract database generation to highlight "interesting" features in a turbulence simulation and save them out for later analysis in a visualization tool. The combination of *in situ* to avoid the time and storage costs of massive I/O and the ability to perform further analysis or rendering on the generated data produces a powerful, streamlined workflow.

1. Background

AVF-LESLIE [11, 12] is a reactive flow solver used for Direct Numerical Simulation or Large Eddy Simulation (DNS/LES) investigation of canonical reactive flows. It solves the reactive multi-species compressible Navier-Stokes equations using a finite volume discretization upon a Cartesian grid. We used AVF-LESLIE to simulate an unsteady, turbulent mixing layer (TML) between two fluids. The simulation demonstrates the evolution of turbulent, braided flow structures (our features of interest) that form as the system breaks down into homogeneous turbulence. AVF-LESLIE can output results to PLOT3D or HDF5/Xdmf format for later analysis and visualization.

The quick evolution of turbulent structures requires access to many simulation time steps to produce a faithful visualization. For the TML case, each time step produced many gigabytes of data. This made it impractical to save enough time steps to produce a times series suitable for purposes such as animation. The sheer size of the saved volume datasets would quickly overwhelm available disk space and make post-processing the results require as much compute resources as the original solver to fit the solution in memory and read it back from disk in a reasonable time.

Previous integration of VisIt's Libsim library [6, 13] into solvers such as CREATE-AV^{TM} Kestrel to enable *in situ* generation of surface extracts yielded good results [14]. When run on 1024 cores and saving isosurfaces of structured and unstructured grid data for helicopter geometries, the coupled Kestrel/Libsim was able to frequently output extracts while using no more then 2-3% of the solver runtime. As the simulation produced extracts, a separate visualization job processed them into images, resulting in an efficient automated workflow.

The same *in situ* workflow would be applicable to combat the challenges of isolating features of interest from AVF-LESLIE's TML flow field. Polygonal surface geometry and the fields defined on those surfaces of interest are extracted and exported to FieldView *eXtract DataBase* (XDB) files for later *post hoc* analysis and rendering using Intelligent Light's parallel FieldView [2] software. XDB files are designed to save line and mesh geometries and the associated scalar and vector fields defined on those geometries. XDB files preserve numerical precision of the stored data making it possible to perform accurate analysis such as surface integration using the extract data in lieue of the original volume data. The resulting workflow decouples feature extraction from rendering. This enables scheduling flexibility and it lets each phase of the workflow use different amounts of compute resources.

The separation of feature extraction and actual visualization into distinct phases that run asynchronously is a feature that is shared by certain in transit infrastructures. In Transit infrastructures such as ADIOS [8] provide an I/O interface to the simulation, which then "writes" its data to other compute resources and then continues. This has the advantage of not stalling the simulation while extracting data or rendering. However, this approach requires additional compute resources to receive the simulation data and further process it, which might be a downside when operating at the limits of the compute resource. A more problematic downside of I/O based systems is that they can be too low-level. I/O interfaces such as ADIOS provide functions for writing arrays. Complicated data structures such as meshes often consist of multiple arrays to represent coordinates and connectivity. Such data structures are encoded into multiple data arrays using various conversions. By necessity, analysis routines that run on the other side of the in transit pipeline must support conventions to reassemble array data back into useful mesh structures. Paraview Catalyst [1, 5] is a powerful *in situ* infrastructure that also supports in transit. Data are exposed to Catalyst as VTK [10] datasets via user-provided adaptor code that is developed for each simulation. The adaptor code defines how simulation data are translated into VTK datasets and permits simulation data arrays to be wrapped as VTK data arrays, allowing zero-copy data passing. Once the data are represented as VTK datasets, they can be operated on by user-defined rendering and data analysis pipelines or they may be shipped to other compute resources for in transit visualization and data analysis. Though it does not have its own in transit mode, VisIt's Libsim otherwise provides similar functionality to Catalyst and provides rendering, data analysis, and extract functionality. VisIt's large set of plots and operators enable the creation of complicated visualization pipelines that can be used for rendering and data extraction. In addition, VisIt provides an export plug-in to the FieldView XDB file format, which enables analysis and rendering of extract data using FieldView.

These infrastructures can all form the building blocks of higher-level infrastructures. For example, Cinema [3] assembles images that are produced *in situ* into databases that can be interactively explored in a lightweight viewer. The images are produced by a simulation instrumented with Catalyst, or another suitable *in situ* infrastructure. SENSEI [4] is another higher-level infrastructure. SENSEI provides a unified interface to multiple *in situ* infrastructures, including ADIOS, Catalyst, and Libsim. SENSEI simplifies the process of *in situ* instrumentation by providing data abstractions that enable creation of a write-once simulation adaptor to expose simulation data structures to SENSEI using the VTK data model. Once inside SENSEI, the VTK data can be passed to any of the coupled *in situ* infrastructures, even combining multiple infrastructures in a single analysis. For the purposes of this analysis, we coupled AVF-LESLIE through SENSEI to enable future flexibility in connecting to infrastructures such as Catalyst or ADIOS. Once instrumented using SENSEI, we selected the Libsim analysis back end to permit the creation of FieldView XDB files. XDB files were read and processed as they were created by separate FieldView jobs running on a workstation.

2. Instrumentation

Producing a working instrumented version of AVF-LESLIE capable of running on a large scale HPC system required changes to several software packages.

2.1. VisIt / Libsim

VisIt is a visualization and analysis software package that was started at Lawrence Livermore National Laboratory in the year 2000. From the start VisIt was designed to work efficiently on large distributed-memory HPC systems. As such, VisIt's compute server runs in parallel using MPI message passing to coordinate multiple processes that each operate on a subset of the overall dataset. VisIt's compute server is architected such that it can be loaded dynamically from simulations via the Libsim library, a VisIt library that is added to simulations to enable them to perform VisIt-enabled *in situ* rendering and analysis. VisIt is normally built using shared libraries which are dynamically loaded when Libsim detects that *in situ* operations are requested. This approach works well up to a few thousand cores after which the file system may introduce delays while loading VisIt's shared libraries and plugins.

The ultimate target concurrency for this study would be in the range of tens of thousands to hundreds of thousands of cores so long delays incurred loading shared libraries would not be acceptable. To avoid such delays, we modified to VisIt's CMake-based static build process to create a statically-built version of Libsim. The static Libsim includes all VisIt libraries and plugins in a single library that simplifies addition of VisIt functionality into simulations such as AVF-LESLIE.

2.2. XDB Library

VisIt's FieldView XDB export capability uses an export plugin that passes VisIt's internal VTK datasets to the XDB library for writing FieldView XDB files for later consumption by FieldView. The XDB library enables the client program to create extract entities such as streamline rakes or polygonal surfaces, record relevant metadata, and define scalar and vector fields on those geometries. The format preserves the precision of the data when saved so it can be used for analyses that would have been appropriate for the original volume data. The metadata saved by the XDB format provides clues to Fieldview about how the surfaces were generated during the data extraction process. For instance, an isosurface extract will record the variable and isocontour value used to generate the surface.

Prior enhancements to the XDB export plugin in VisIt enabled support for *write groups* (described in [7]), which let VisIt perform partial data aggregation during production of XDB files. However, improved support for parallel writes was desired and the XDB library was not designed for parallel. Consequently, we undertook a redesign of the library that would decouple the data model that describes the XDB surface extracts from the methods used to write and read the data. This effort yielded a second version of the XDB library, which could read/write the XDB format while providing a future path to using parallel data transports such as ADIOS. The resulting XDB library also can represent its data objects using zero-copy constructs that support various memory layouts, a feature necessary to reduce overhead when running *in situ*. The VisIt XDB export plugin was enhanced to use the new version of the XDB library and this was the version used during instrumentation of AVF-LESLIE.

2.3. Integration with AVF-LESLIE

SENSEI enables simulations to integrate with multiple *in situ* infrastructures while creating just one code adaptor. An adaptor exposes simulation data structures so their data may be used by the *in situ* infrastructure. We modified previous adaptor code that had been created to integrate directly with Libsim so AVF-LESLIE could also integrate SENSEI, opening the door to future workflows where ADIOS is used for in transit data staging to a separate "endpoint" analysis program running on an alternate set of compute nodes. At this time, we have not yet attempted using that feature but have verified that our XDB workflow continues to operate using the SENSEI infrastructure.

The SENSEI adaptor follows a pattern that requires just a few insertions of extra adaptor functions into AVF-LESLIE's main routine: an *initialize* function, a *coprocess* function, and a *finalize* function. The initialize function is used to set up SENSEI, which in turn performs the relevant initialization for its subservient *in situ* analysis infrastructures. The coprocess function passes pointers to AVF-LESLIE arrays to some book-keeping code that associates the buffer addresses and sizes with variable names. The coprocess function's role is to package the simulation's buffers (zero-copy when possible) as a VTK curvilinear grid dataset with various fields defined on its nodes. For the purposes of this study, the adaptor also calculates the vorticity field, which is not computed in the solver, yet is needed to identify vortex features of interest for *in situ* rendering and data extraction. After packaging the simulation data as a VTK dataset, the VTK dataset is passed through SENSEI to a secondary analysis adaptor that shares the dataset with an analysis infrastructure, in this case Libsim. The analysis adaptor created for Libsim accepts the VTK dataset from the SENSEI analysis adaptor and exposes that data to VisIt via Libsim function calls and adaptor callback functions. In addition, the adaptor creates a set of VisIt plots based upon commands from an extract input file and uses those plots to export the desired extract surfaces into XDB files. Plots can also be restored from a VisIt session file which is an XML file that is useful for creating visualizations that can be rendered since plot attributes and colors are preserved. The finalize function is called when AVF-LESLIE has completed its main loop and needs to clean up prior to exiting. The overall schematic for the instrumented simulation is shown in fig. 1.

3. Performance

In this study, AVF-LESLIE was configured to simulate unsteady dynamics of a temporally evolving planar mixing layer at the interface between two fluids, a configuration that gives rise to a temporal mixing layer (TML). This type of fundamental flow mimics the dynamics encountered when two fluid layers slide past one another and is found in atmospheric and ocean fluid dynamics as well as combustion and chemical processing. The two sliding fluid layers are subject to inviscid instabilities and can evolve from largely 2D laminar flow into fully developed, 3D homogeneous turbulent flow as shown in [9]. Visualizations of the TML flowfield in fig. 2 show isosurfaces of the vorticity field, at 10,000, 50,000, 100,000, and 200,000 time steps where the flow evolves from the initial flow field, vortex braids begin to form, wrap and then the flow breaks down leading to homogeneous turbulence, respectively.

We conducted scaling studies using AVF-LESLIE on Titan at Oak Ridge Leadership Class Compute Facility. Titan is a Cray XK7 with 18,688 compute nodes, each containing a 16-core AMD Opteron CPU, 32GB of memory, and an Nvidia Tesla K20X GPU. The scaling studies used a Cartesian grid size of 1025^3 and physical non-dimensional domain size of 4π x 4π x 2π. The size of the 1025^3 grid was held constant during scaling, resulting in a strong scaling study that reduces the average simulation workload per core as the number of cores increases. Two types of in situ computations were performed: a rendering workflow, and an extract-based XDB workflow.

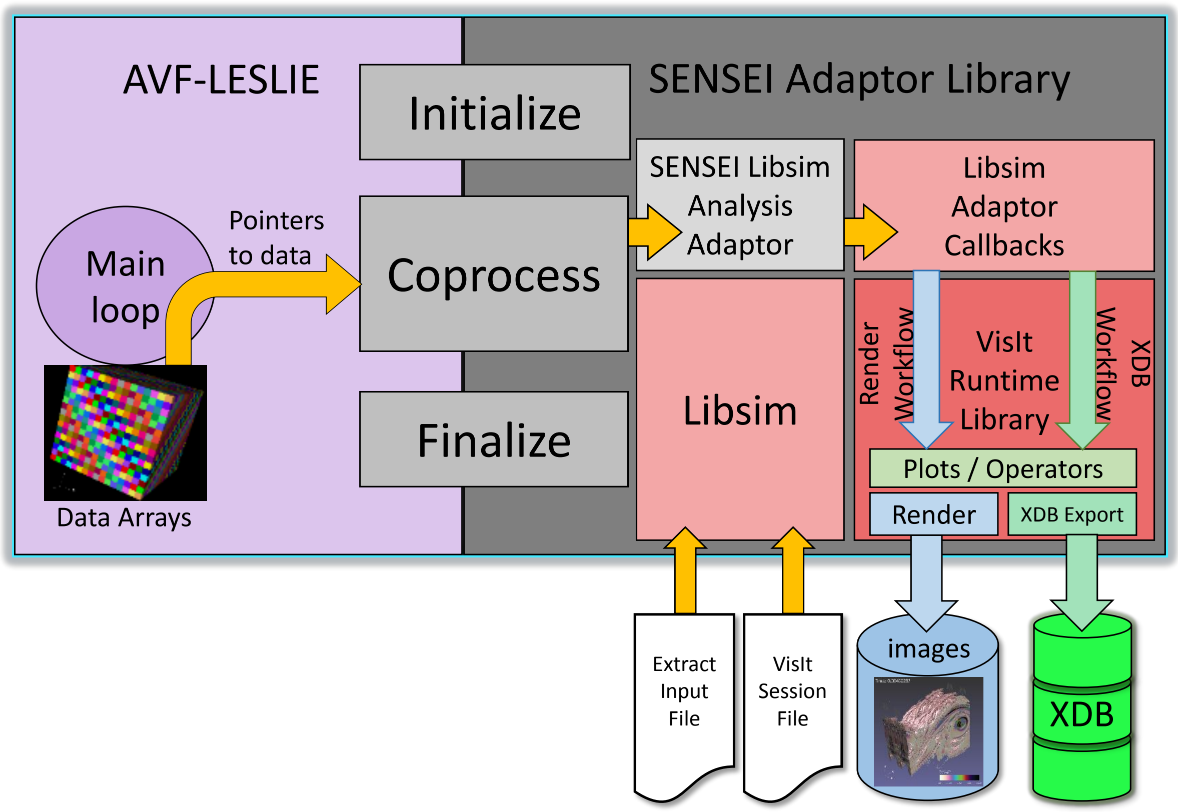

Figure 1. AVF-LESLIE simulation instrumented with SENSEI, VisIt/Libsim, and XDB library

3.1. Rendering Workflow

The rendering workflow was scaled up to 131,072 cores on Titan using all 16 cores in a compute node. The workflow demonstrated that the instrumented simulation can create and render a visualization based on the data directly from the solver memory as in [4]. In this case, the simulation was initialized using a VisIt session file, which directed Libsim to create 3 isosurfaces of the derived vorticity field and 3 planar slices. The visualization was rendered into a 1600x1600 pixel image image and saved to PNG format. Each measurement in the scaling study was performed using 100 time steps of AVF-LESLIE where *in situ* rendering was performed every 5th solver time step. Timing measurements were obtained from log files produced by the instrumented solver, which called the *MPI_Wtime()* library function around blocks of code being timed.

The timings measured time spent in the solver time step and overall time performing *in situ* rendering, which includes data extraction, rendering, image compositing, and image saving. The time spent in the solver decreased as the number of cores was increased due to the strong scaling induced by holding the grid size constant. It is worth noting that the complexity of the visualization resulted in long rendering times. The total time spent rendering is amortized over 5 solver time steps, resulting in an image saving cost on the order of 1-2 seconds when compared against the average solver time step. Also, the time spent on *in situ* increased with scale, largely due to image compositing.

Image compositing is an operation whereby full scale images from each MPI rank are combined in a tree-based reduction among all MPI ranks ultimately resulting in one MPI rank having a single image containing contributions from all of the input images on other MPI ranks.

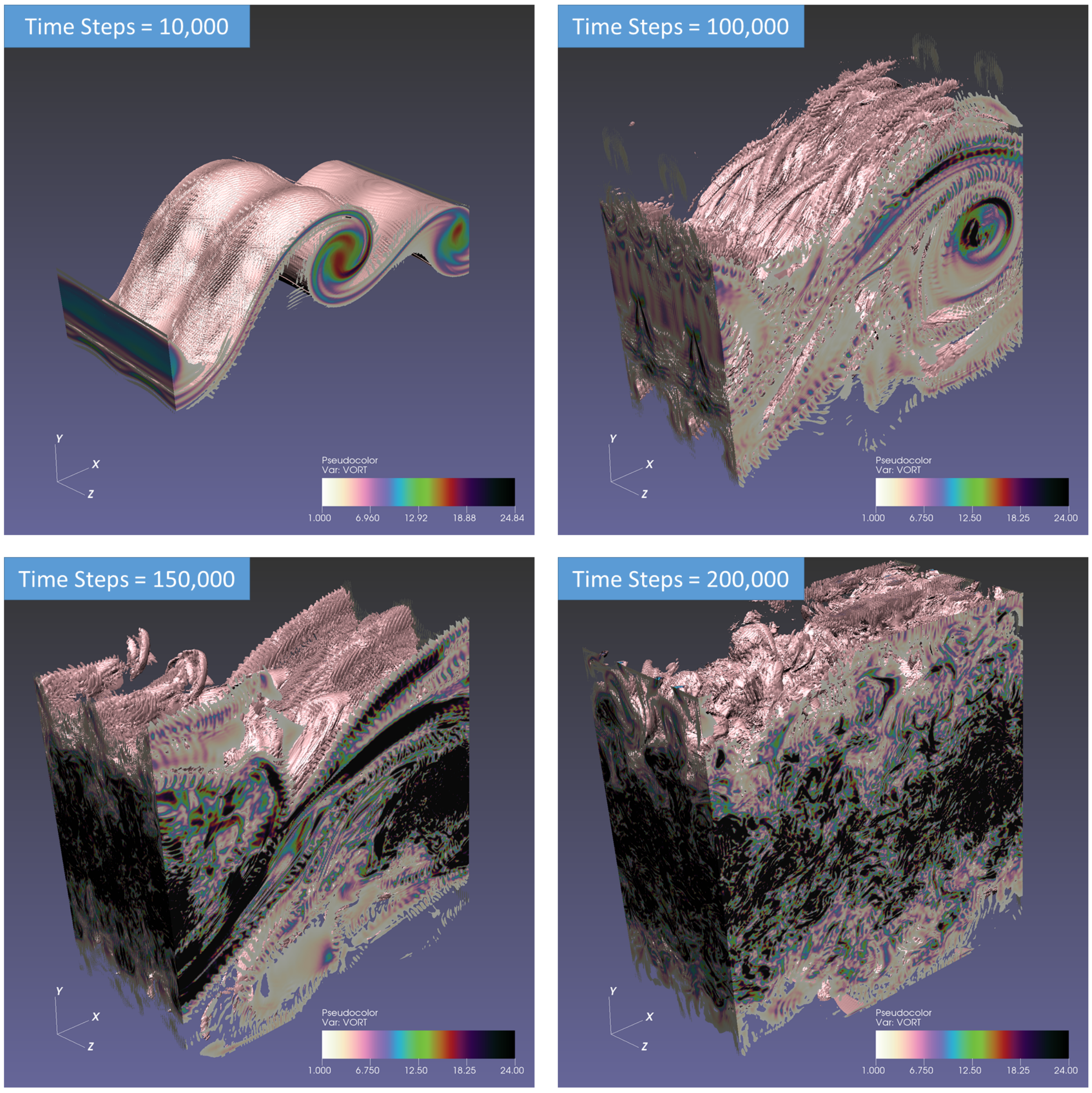

Figure 2. The Evolution of Temporal Mixing Layer from Initial to Vortex Breakdown

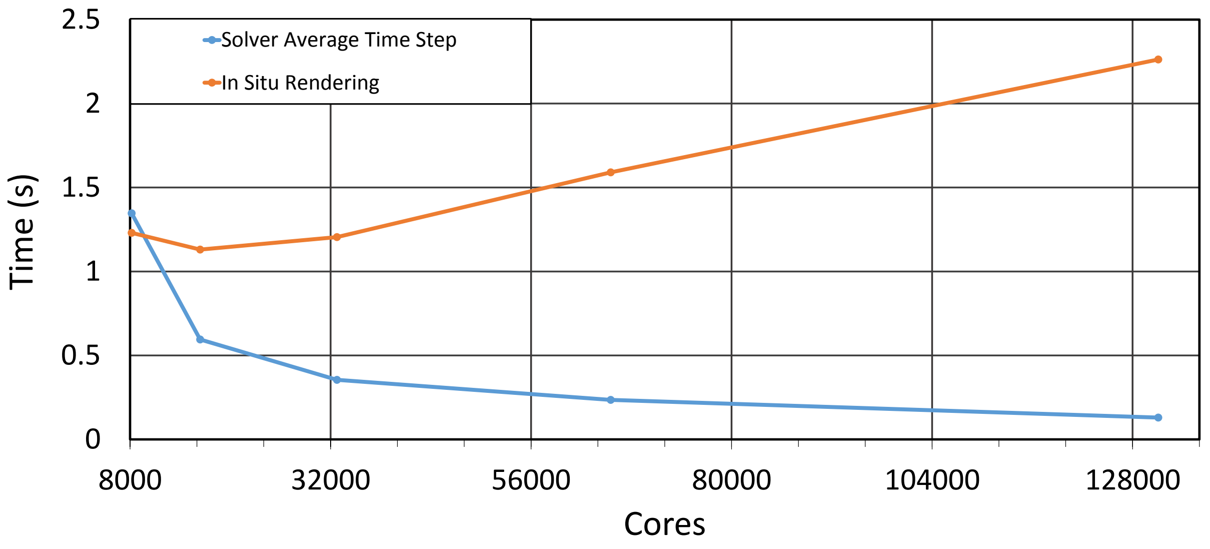

Figure 3. Strong Scaling Performance of AVF-LESLIE and *In Situ* Rendering Workflow for IsoSurface and Coordinate Cuts Rendered to 1600x1600 Pixel Image

Image compositing represents a different type of workload from the solver computation. As the number of processors increases, the number of images to be composited increases, as does the depth of the communication tree of processors which must communicate their images and depth buffers. Thus, image compositing can grow more expensive with larger numbers of processors as shown in fig. 3. In this study, VisIt relied on custom image compositing code which overlaps image communication with the actual pixel compositing operations. Unfortunately, the custom image compositing code appears to degrade in performance after 16K cores. In future work, Ice-T, a powerful image compositing library that is known to scale well and provides many communication optimizations could be used to further reduce the overhead associated with image compositing.

3.2. XDB Workflow

The XDB workflow was scaled up to 32,768 cores on Titan using all 16 cores of the allocated compute nodes. As with the rendering workflow the scaling numbers were obtained by running AVF-LESLIE for 100 time steps, with *in situ* operations every 5 solver time steps, and analysing its output logs. At each *in situ* time step, the AVF-LESLIE adaptor made Libsim function calls to create isosurfaces of vorticity and save them to a FieldView XDB file. The surface was specified using an input file to the adaptor which was interpreted and translated into Libsim function calls for creating a VisIt plot to generate an isosurface. Export to XDB was performed by partitioning the total MPI ranks with geometry into smaller write groups of 96 MPI ranks. Each write group aggregated isosurface geometry locally within the group to a single MPI rank responsible for writing a XDB file. When all groups completed writing their XDB file, the lead MPI rank wrote a FieldView layout file, which contains a list of XDB files to later read in parallel to recreate the entire surface extract geometry.

Over the course of the 100 time step runs, the time spent by AVF-LESLIE can be divided into three main categories: solver time steps, I/O, and *in situ*. The solver time steps represented the time that AVF-LESLIE spent solving and updating its physics variables. The AVF-LESLIE runs were configured to output an initial plot file in PLOT3D format, followed by additional plot files every 100 time steps. Over the course of the short run, 2 sets of plot files were created where the size of each plot file was roughly 51842 MB as shown in tab. 1. Over the same run, the *in situ* routines created an isosurface of vorticity and wrote XDB files with the extract geometry every 5 time steps. The size of the average set of XDB files for a single *in situ* time step was between 260 MB and 266 MB, depending on the number of cores. When comparing the size of a single plot file containing volume data and a set of XDB files representing an isosurface extract, the extracts are around 200 times smaller. Given that the *in situ* extracts were written 20 times more frequently than the plot files, adding up the total size of the XDB extracts is still 10 times smaller than a single plot file.

Table 1. File Size Comparison

Cores	1 Volume File (MB)	1 Extract (MB)	1 Extract/ 1 Volume	20 Extracts/ 1 Volume
8192	51842	260	0.005	0.100
16382	51842	262	0.005	0.101
32768	51842	266	0.005	0.102
			˜200x reduction	˜10x reduction

The relative timings of solver time steps, I/O, and *in situ* are shown in fig. 4. The overwhelming majority of time is spent in the 2 calls to the I/O routines that write the full size PLOT3D files. That is followed by the time spent computing 100 solver time steps, followed at last by the *in situ* operations which were the least expensive in terms of time. When considering the actual overhead percentage added to the runtime of the simulation, for each 100 time steps, *in situ* added between 1.6% and 2.4% to the simulation runtime. If one was to consider an AVF-LESLIE run that performed absolutely no PLOT3D I/O then the overhead of *in situ* increases to between 21% and 28% for writing isosurface extracts every 5 solver time steps. As with any configurable operation, the overhead of *in situ* with respect to the overall runtime will depend on its frequency of use. In this case, for 1/30th to 1/50th the cost of full I/O, 10 times better temporal sampling was achieved and much less disk space was consumed by using *in situ* extracts. The time spent further post-processing the XDB extract files in FieldView on a 12-core MacOS X workstation with 64GB of memory was on the order of a few seconds per time step to produce rendered images.

Conclusions

AVF-LESLIE was successfully instrumented for *in situ* using a combination of SENSEI, Libsim+VisIt, and the XDB library. The instrumented version of AVF-LESLIE can produce extract databases in the form of FieldView XDB files, which has proven to be a useful feature that greatly reduces the data that would otherwise need to be saved. The savings were measured in both time and storage costs. Extract databases took 2-3% of the solver runtime when also performing limited volume I/O and would further reduce the time spent during *post hoc* analysis. The entire size of the extract database was 10 times smaller than PLOT3D volume data while saving extracts 20 times more frequently. The surfaces in the extract database were able to be visualized on a workstation instead of requiring a visualization cluster coupled to the

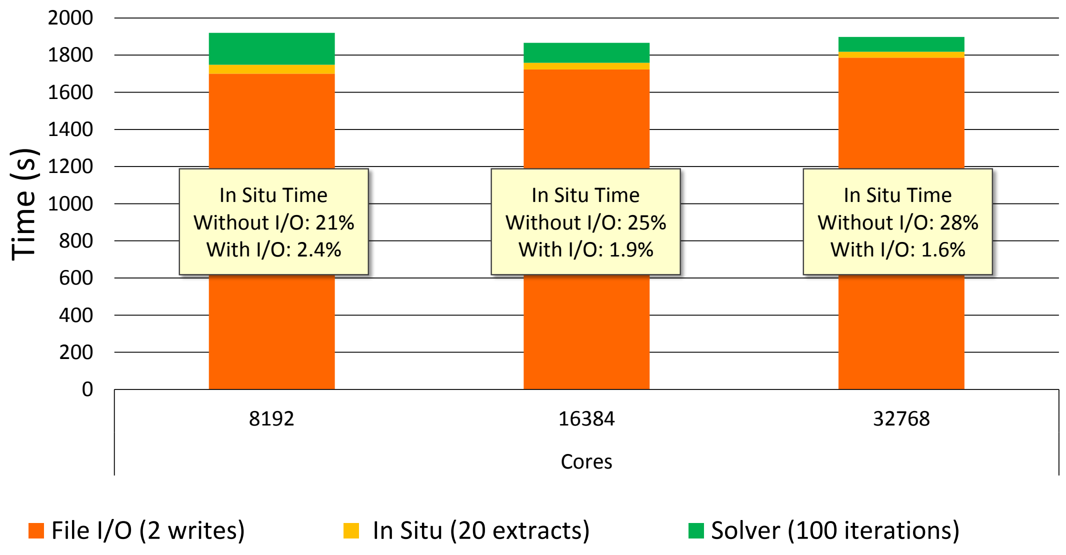

Figure 4. AVF-LESLIE Timings with *In Situ* Data Extraction

supercomputer. *In Situ* technologies are maturing to the point that they can begin to accelerate science and make the best use of HPC systems in spite of whatever I/O limitations might exist.

Acknowledgments

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under award numbers DE-SC0007548 and DE-SC0012449. This research used resources of the Oak Ridge Leadership Computing Facility (OLCF).

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 3.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. ParaView Web Site, http://www.paraview.org, June 2010.

2. FieldView 16, http://www.ilight.com, September 2016.

3. James Ahrens, Sébastien Jourdain, Patrick O'Leary, John Patchett, David H. Rogers, and Mark Petersen. An image-based approach to extreme scale in situ visualization and analysis. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 424–434, Piscataway, NJ, USA, 2014. IEEE Press.

4. Utkarsh Ayachit, Andrew Bauer, Earl P. N. Duque, Greg Eisenhauer, Nicola Ferrier, Junmin Gu, Kenneth Jansen, Burlen Loring, Zarija Lukić, Suresh Menon, Dmitriy Morozov, Patrick O'Leary, Michel Rasquin, Christopher P. Stone, Venkat Vishwanath, Gunther H. Weber, Brad J. Whitlock, Matthew Wolf, K. John Wu, and E. Wes Bethel. Performance

Analysis, Design Considerations, and Applications of Extreme-scale *In Situ* Infrastructures. In *Proceedings of SC16*, Salt Lake City, UT, USA, November 2016. *To appear.*

5. Andrew C. Bauer, Berk Geveci, and Will Schroeder. *The ParaView Catalyst User's Guide v2.0.* Kitware, Inc., 2015.

6. Henry R. Childs, E. Brugger, Brad J. Whitlock, Jeremy S. Meredith, Sean Ahern, Kathleen Biagas, Mark C. Miller, Gunther H. Weber, Cyrus Harrison, David Pugmire, Thomas Fogal, Christophe Garth, Allen Sanderson, E. Wes Bethel, Marc Durant, David Camp, Jean M. Favre, Oliver Rubel, and Paul Navratil. Visit: An end-user tool for visualizing and analyzing very large data. In *High Performance Visualization: Enabling Extreme-Scale Scientific Insight*, pages 357–368. Chapman and Hall, 2012.

7. Earl P. N. Duque, Brad J. Whitlock, Steve M. Legensky, Christopher P. Stone, Reetesh Ranjan, and Suresh Menon. The impact of in situ data processing and analytics upon scaling of cfd solvers and workflows. In *27th International Conference on Parallel Computational Fluid Dynamics*, Montreal, Canada, 2015.

8. Qing Liu, Jeremy Logan, Yuan Tian, Hasan Abbasi, Norbert Podhorszki, Jong Youl Choi, Scott Klasky, Roselyne Tchoua, Jay Lofstead, Ron Oldfield, et al. Hello adios: the challenges and lessons of developing leadership class i/o frameworks. *Concurrency and Computation: Practice and Experience*, 26(7):1453–1473, 2014.

9. Ralph W Metcalfe, Steven A Orszag, Marc E Brachet, Suresh Menon, and James J Riley. Secondary instability of a temporally growing mixing layer. *Journal of Fluid Mechanics*, 184:207–243, 1987.

10. Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit: An Object Oriented Approach to 3D Graphics.* Kitware, Inc., fourth edition, 2004. ISBN 1-930934-19-X.

11. Thomas M. Smith and Suresh Menon. The structure of premixed flame in a spatially evolving turbulent flow. *Combustion Science and Technology*, 119, 1996.

12. Christopher P. Stone and Suresh Menon. Open loop control of combustion instabilities in a model gas turbine combustor. *Journal of Turbulence*, 4, 2003.

13. Brad J. Whitlock, Jean M. Favre, and Jeremy 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*, pages 101–109. Eurographics Association, 2011.

14. Brad J. Whitlock, James R. Forsythe, and Steve M. Legensky. In Situ Infrastructure Enhancements for Data Extract Generation. In *54th AIAA Aerospace Sciences Meeting, SciTech 2016*, pages 1–12, January 2016.