Visualizing a Trillion-Cell Simulated CFD Solution on an Engineering Workstation

Scott T. Imlay[1], Craig Mackey[2] and Dave Taflin[3]
Tecplot Inc., Bellevue, WA, 98006

The size and number of datasets analyzed by post-processing and visualization tools are growing with Moore's law. Conversely, the disk-read data transfer rate is only doubling every 36 months and is destined to be the bottleneck for traditional post-processing architectures. To eliminate this bottleneck during post-processing visualization and analysis, a subzone load-on-demand (SZL) visualization architecture has been developed which loads only the data needed to create the desired plot. Based on the Moore's law growth, the largest CFD problems are expected to be using 1 trillion cells by 2030. In this paper, to prepare for this 2030 scenario, the isosurface for a simulated 1 trillion cell CFD dataset was visualized using the SZL technology. With traditional visualization technology, this would require a super-computer, but with SZL it was possible on an engineering workstation with 128GB of memory. The 120GB of memory used during this demonstration is nearly two orders-of-magnitude less than the 8.5TB file size. The time and memory required to generate this isosurface generally scales with $O(n^{\frac{2}{3}})$, where n is the number of cells in the grid.

Nomenclature

α	=	angle of attack
β	=	yaw angle
a	=	cylinder diameter
C_p	=	pressure coefficient
M	=	Mach number
n	=	number of points in the full grid
Re	=	Reynolds number
t	=	time
τ	=	pseudo time
$\vec{v}$	=	velocity at a point in space
v_i	=	isosurface value of a variable
v_d	=	discriminant value of an interval tree root node or branch node
v_{min}^s	=	minimum value of variable in subzone s
v_{max}^s	=	maximum value of variable in subzone s
x, y, z	=	x-, y-, and z-coordinates
$\vec{x}$	=	(x, y, z) position in space

[1] Chief Technology Officer, P.O. Box 52708, Bellevue, WA, Senior Member AIAA.
[2] Senior Research Engineer, Research, P.O. Box 52708, Bellevue, WA.
[3] Senior Software Development Engineer, P.O. Box 52708, Bellevue, WA.

I. Introduction

THE application of computational fluid dynamics (CFD) in the aerospace design process has increased dramatically over the last decade. This is due, in large part, to the relentless and continuing growth of computer performance. In some cases the enhanced computer power is used to perform high-resolution CFD calculations to analyze the details of complicated unsteady flow fields around complex configurations. In other cases it is used to create a virtual wind-tunnel where hundreds or thousands of lower resolution CFD computations are performed to estimate the aerodynamic properties of a prospective configuration throughout its operating envelope. In either case, the total amount of data read during post processing is doubling every 18 months – in sync with Moore's law.

These trends are consistent with the conclusions of the NASA CFD Visions 2030 Study[18] which forecasts the need for on-demand analysis and visualization of unsteady CFD problems sizes of 10 billion points by 2020, 100 billion points by 2025 and 1 trillion cells by 2030 (Figure 1).

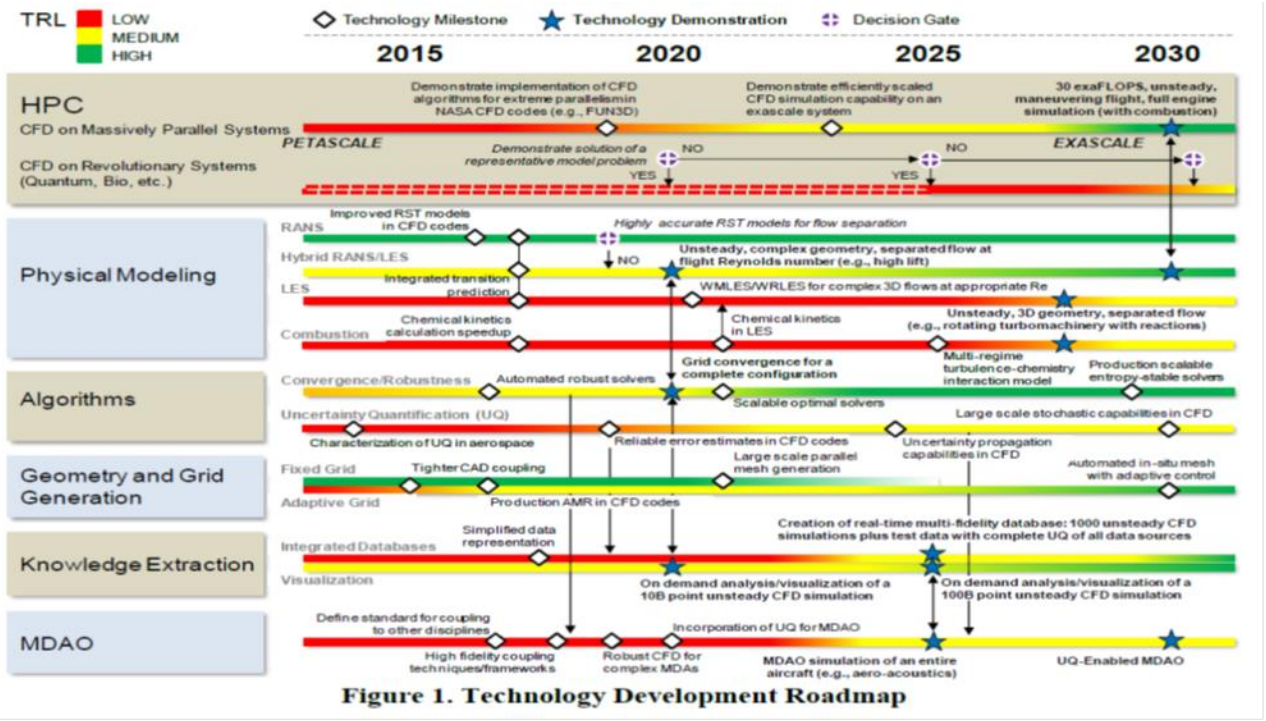

Figure 1. Technology Development Roadmap

Figure 1. Technology Roadmap from NASA CFD Vision 2030 Study[18]. Read the PDF.

The expected dramatic growth of CFD problem size poses a significant challenge for developers of CFD visual analysis software. To prepare for this challenge, Tecplot Inc. set an internal goal to visualize a finite-element dataset containing one trillion tetrahedral cells, using slices, isosurfaces, and streamtraces. Furthermore, Tecplot's goal was to do this visualization using an engineering workstation. The workstation was a Dell Precision T7610 with dual 8-core Intel Xeon processors, an NVIDIA Quadro K4000 video card, 128GB of memory, and a 16TB Raid5 external hard-disk array. This system is probably a little more advanced than what is sitting beside your desk, but systems with these capabilities will be common-place in the near future. When purchased in 2015, this computer system costs less than $10,000.

This paper describes the three-year effort by Tecplot Inc. to complete the trillion-cell challenge. The following sections will detail computer-system and network trends, the new software and technologies that were developed to deal with these trends, and the results of the trillion-cell visualization.

A. Computer I/O and Network Performance Trends

CFD data is generally stored on arrays of hard disk drives. Over the last two decades, the storage capacity of hard disks has grown in accordance with Kryder's law - doubling every 12 months. This is more than sufficient to keep up with the growth in dataset size. Unfortunately, the sustained rate at which data can be read from the hard disk is growing much slower – doubling every 36 months[1]. This is because sustained data transfer rate grows with the lineal density of the magnetic dots on the hard disk while storage capacity grows with the areal density (roughly the square of the lineal density). While hard disk capacity is keeping up with dataset size, the speed at which we can read the data is not.

In the past, the primary bottleneck in visualization software performance was network speed. Over the last decade, the speed of Local Area Networks (LANs) has doubled every 2 years on average. It doesn't change that often, but upgrades tend to yield an order-of-magnitude increase in bandwidth (100Mb/s to 1Gb/s, for example). Likewise, Wide Area Network (WAN) performance is also doubling every 2 years, although it lags substantially behind LAN performance, and internet bandwidth is generally worse than WAN bandwidth. The bandwidth for both LANs and WANs are growing more slowly than dataset size, but much faster than sustained disk-read data transfer rates. For internet connections, network bandwidth is still generally the bottleneck.

Given these trends, a simple analysis of visualization system performance can be performed. Assuming initial values of 100M cells in 2005, 100MB/s (1Gb/s) LAN in 2006, and 75 MB/s sustained read bandwidth for the hard-disk in 2006. The trends in time to load a large dataset are given in Figure 2. Note that the load-time ultimately becomes dominated by the hard-disk sustained read data transfer rate, with the cross-over date a function of the network type (bandwidth).

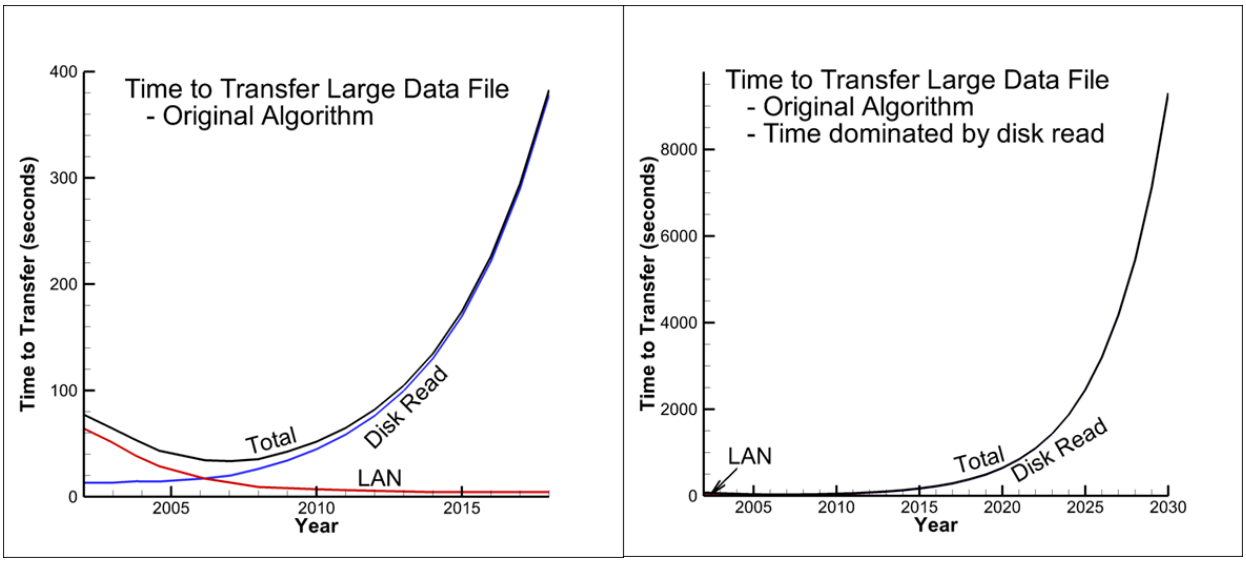

Figure 2. Time to transfer and load a large CFD dataset

These trends have a significant impact on the optimal visualization architecture. Traditional client-server architectures were designed to overcome network bandwidth constraints. In these architectures, the data is loaded on a remote computer with a high-bandwidth access to the data, important data abstractions are extracted, and the geometry and data for these abstractions is transferred across the slow network to a local client. In this context, "abstractions" may include model geometry, slices, isosurfaces, streamlines, vortex cores, and any other one- or two-dimensional data extraction the user may desire. A modification of the client-server architecture is to render the plot remotely and transfer the image at video-like frame-rates. These client-server architectures are important for overcoming network

bandwidth limitations but do nothing to overcome the new bottleneck, sustained disk-read data transfer rates (SDRDTR).

The current hardware-based solution to the SDRDTR bottleneck is to increase the number of spindles (hard-disks) used in the parallel file system. If the number of spindles in the file system doubles every 3 years or so, the time to read a data file will remain constant. However, increasing the number of disks in the parallel file system is counter-intuitive, as the hard disk capacity will match the file size increases without adding disks. As such, that solution will likely meet with some resistance. Longer-term hardware-based solutions, such as solid-state disks (SSDs) are not yet economically viable for collections of large CFD datasets.

The software-based solution to the SDRDTR bottleneck is to read and write less data. Generally, only a small percentage of the total dataset is needed to create the abstractions the user wishes to view, so this solution seems viable. To be sustainable, the percentage of the dataset written by the CFD code and loaded into the visual analysis application must decrease over time (halved every three to four years). This solution also has other benefits, like reduced memory requirements and reduced network bandwidth requirements. This is one architectural approach taken by Tecplot, Inc. for large-data visualization.

In a previous papers[11], a new architecture was described for visualizing large CFD datasets. It was based on loading subzones (spatially correlated sub-segments of the full dataset of less than 256 nodes or cells) on demand (only as needed). To support this algorithm, variable min-max trees are created to rapidly select the needed subzones. The architecture is sustainable: for slices and isosurfaces, the number of subzones loaded is approximately $O(n^{\frac{2}{3}})$ and for streamtraces it is approximately $O(n^{\frac{1}{3}})$. In a more recent paper[20], the same benefits have been demonstrated for a subzone-based in-situ technique where only those subzones needed to create desired abstractions are written to file from the CFD code. Once subzone in-situ and subzone load-on-demand are adopted, the network bandwidth and latency often become the dominant bottleneck, particularly during the visual analysis of remote data. In our most recent paper, a subzone-based client-server architecture is presented to overcome network bandwidth and latency limitations. These techniques, and more, are used in the current trillion-cell challenge.

II. Approach

A. Related Work

The work is based on the subzone load-on-demand technology described in a series of previous papers and summarized in the following section.

B. Subzone-Based Architecture

The subzone load-on-demand technology is described in the following three subsections. The basic technology, described in the first subsection below, dramatically reduces the time and memory required to visualize a large dataset. Subzone-based in-situ, described in the second subsection, is used to reduce the time required to write the datafile from a CFD code running a very large case. The final subsection describes subzone-based client-server, which is used to visualized data on a remote system.

1. Subzone Load-on-Demand

The basic subzone load-on-demand architecture was described in a previous paper[11] and is summarized here. The approach requires a file whose data is partitioned into subzones of no-more-than 256 nodes each and cell subzones of no-more-than 256 cells each. In the trillion-cell dataset, the variable data is stored at the nodes of the tetrahedra, so the only data stored in the cell subzones is the connectivity (the numbers of the nodes that make up each cell). This subzone load-on-demand (SZL) files is composed of a header which contains a tree of variable min/max values for each variable of each node and cell subzone followed by the actual node subzone variable data and the cell subzone connectivity arrays. These connectivity arrays are compressed[21] by replacing the full node numbers with a (node-

subzone, subzone node offset) pair. The subzone node offset is an 8-bit integer (256 nodes per subzone) and the node-subzone is a reduced precision offset into a look-up table of node subzones referenced by that cell subzone. By using the reduced precision offset instead of the actual node subzone numbers, the size of the data file can be reduced by up to 50% for tetrahedral finite-element data.

The variable min/max trees allow the software to only load those node and cell subzones necessary to create the desired slices, isosurfaces, or streamtraces. For isosurface extraction, the software loads the relevant isosurface variable min/max tree, searches for those node and cell subzones with a min-value less than the isosurface value, and a max-value greater than the isosurface value, and loads only those subzones. The triangles making up the isosurface are then extracted using a standard marching-cubes (or marching-tets) algorithm. Slices are treated as isosurfaces of a coordinate variable. For each step of a streamtrace, it searches for node and cell subzones that contain the required (x,y,z) point.

The total size of data required to generate a particular isosurface generally scales as approximately $O(n^{\frac{2}{3}})$, where *n* is the number of cells in the grid. The benefit of reduced data transfer thus increases proportionally as problems grow larger. For streamtrace generation, the required data generally scales with roughly $O(n^{\frac{1}{3}})$.

2. Subzone-Based In-Situ

The I/O bottleneck is also a problem when writing data from a CFD code. To reduce the required time to write the data, the CFD code can write just those subzones needed for the desired visualization (or set of visualizations). The min/max trees are also compressed to eliminate entries for subzones that are not written to the file, so the files are generally much smaller than if the full dataset were written.

3. Subzone-based Client-Server

The subzone-based client-server approach uses a server process on a remote machine to load subzones from a data file local to the server machine and transfer those subzones over the network to client software running on the user's local machine. The server benefits from the same advantages that the client does in the data-local case—it requires only enough memory to load the necessary subzones, and reads only those subzones from disk. This contrasts with other client-server architectures where the server must load the grid plus some number of solution variables in their entirety in order to extract the desired surfaces. The server transfers the subzones required to encompass the desired surface to the client, which then extracts and renders the surface.

Another advantage of subzone-based client-server is that it allows small adjustments to the surface location with no, or limited additional data from the server. As the surface is moved, only those additional subzones required to encompass the surface's new location are transferred from the server.

Consistent with data-local subzone loading, total size of data required to display a particular surface generally scales as approximately $O(n^{\frac{2}{3}})$, where *n* is the number of nodes or cells in the grid. The benefit of reduced data transfer thus increases proportionally as problems grow larger.

For streamtrace generation, the required data generally scales with roughly $O(n^{\frac{1}{3}})$. There is a performance penalty for streamtraces, however, because the client cannot know a priori which subzones will be required to enclose the complete path of the streamtrace given only its starting location. The client must request subzones to encompass the starting location and integrate the velocity field from there until it encounters subzones not yet loaded, then request those additional subzones, repeating the process until the integration is complete (by whatever criteria the user has specified). Each of these requests incurs a latency penalty.

III. Results

The subzone load-on-demand was tested for isosurface generation on a 1 trillion cell (167 billion node) simulated CFD dataset shown in Figure 3. Since there is currently nobody doing trillion cell CFD solutions, a simple simulated

dataset is used. The geometry is a cylinder and the isosurface variable is a simple quadratic polyhedral. The data set is 8.5 TB spread across 16 files. It took 120 GB of memory and about 45 minutes to render. The isosurface is 190 million cells. The orange lines are the bounding boxes of the 1024 zones that make up the data.

.

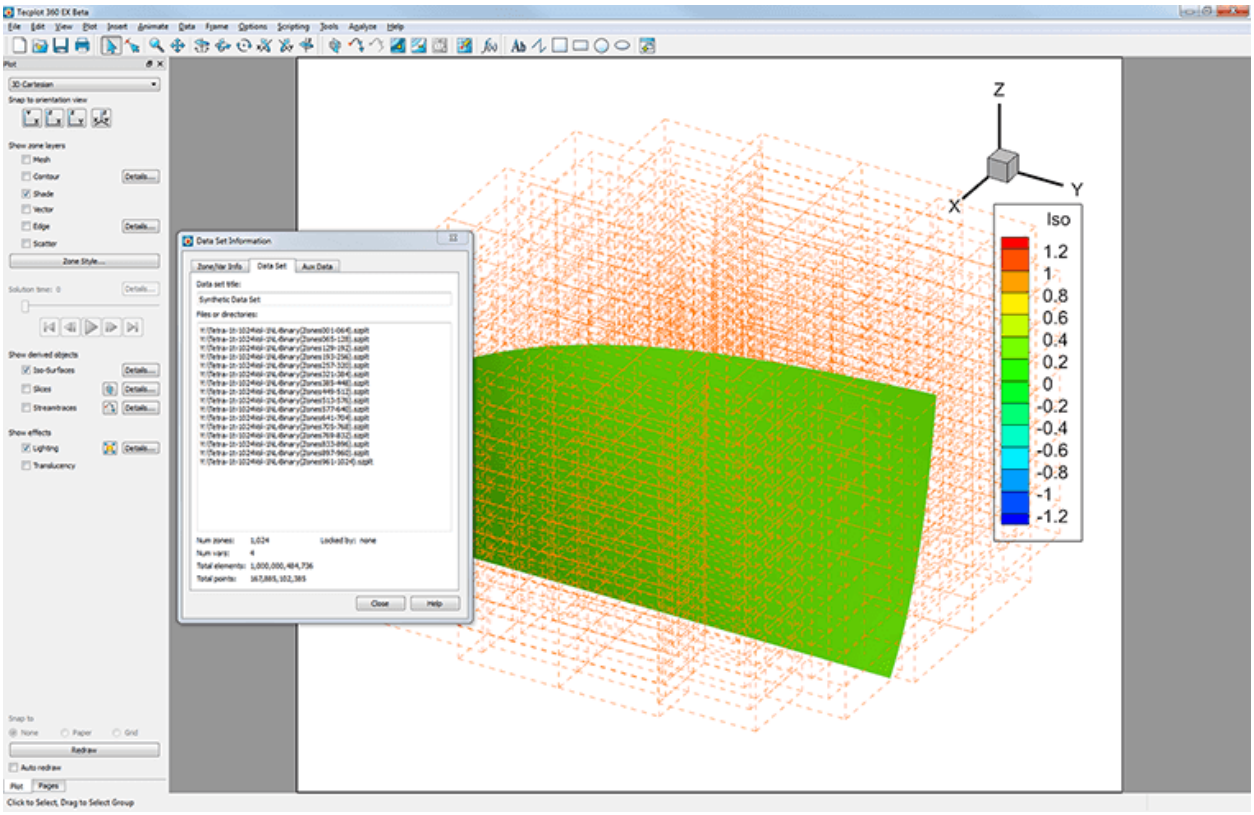

Figure 3. Isosurface for the trillion-cell tetrahedral dataset. Read the [Trillion-Cell Challenge blog series](#).

To study the scaling of the algorithm with grid size, the same isosurface was generated for a range of grid sizes from 1 thousand cells to 300 billion cells. A set of 60 streamtraces was also generated for the same datasets. The resulting time and memory required is shown in Figure 4. These are log-log plots, so the exponent m in the scaling $O(n^m)$, where n is the number of cells, is the slope of the line. For convenience, the lines for linear $O(n)$ scaling and $O(n^{\frac{2}{3}})$ scaling are shown. The disk cache has a big impact on data-load performance, so the plots show both uncached results (first load after a restart of the computer) and cached results (final load after loading several times in a row). As you can see, the time and memory scale with $O(n^{\frac{2}{3}})$ up through 100 billion cells. After that, the cached results for isosurfaces and streamtraces converge to the uncached results as the disk cache becomes overwhelmed due to the problem size.

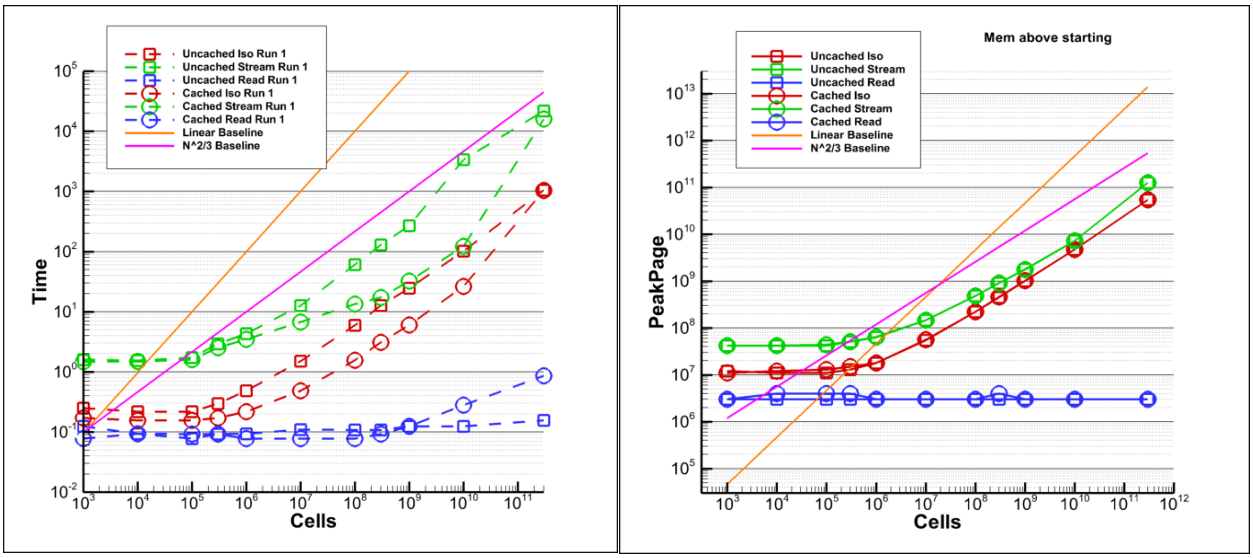

Figure 4. Isosurface for the trillion-cell tetrahedral dataset. Read the Blog.

As a final note, the size of an in-situ data file for the isosurface of the trillion-cell case would be roughly the same as the 120 GB required to visualization the isosurface. This is nearly two orders-of-magnitude smaller than the full trillion cell dataset.

IV. Conclusion

The largest CFD problems are expected to be using 1 trillion cells by 2030. To prepare for this scenario, the isosurface for a simulated 1 trillion cell CFD dataset was visualized using the subzone load-on-demand technology. With traditional visualization technology, this would require a super-computer, but with subzone load-on-demand it was possible on an engineering workstation with 128GB of memory. The 120GB of memory used during this demonstration is nearly two orders-of-magnitude less than the 8.5TB file size. The time and memory required to generate this isosurface is shown to scale with $O(n^{\frac{2}{3}})$, where n is the number of cells in the grid.

References

[1]"Hitachi Global Storage Technologies," http://www.hitachigst.com/hdd/technolo/overview/storagetechchart.html.

[2]Moran, P.J., "Field Model: An Object-Oriented Data Model for Fields," NASA TR NAS-01-005, 2005.

[3]Chiang, Y.-J., ElSana, J., Lindstrom, P., Pajarolo, R., and Silva, C.T., "Out-of-Core Algorithms for Scientific Visualization and Computer Graphics," Tutorial Course Notes, IEEE Visualization 2003.

[4]Chiang, Y.-J., and Silva, C.T., "External Memory techniques for Isosurface Extraction in Scientific Visualization," *External Memory Algorithms and Visualization, DIMACS Series*, 50:247-277, 1999.

[5]Chiang, Y.-J., and Silva, C.T., "Interactive Out-Of-Core Isosurface Extraction," *IEEE Visualization 98*, 167-174, Oct. 1998.

[6]Ueng, S.-K., Sikorski, C., and Ma, K.-L., "Out-of-Core Streamline Visualization on Large Unstructured Meshes," *IEEE Transactions on Visualization and Computer Graphics*, 3(4):370-380, Oct. 1997.

[7]Fox, G. C. ``A review of automatic load balancing and decomposition methods for the hypercube," in M. Schultz, editor, Numerical Algorithms for Modern Parallel Computer Architectures, pages 63-76. Springer-Verlag, New York, 1988. Caltech Report C3P-385b.

[8]de Ronde, J.F., Schoneveld, A. and Sloot, P.M.A., "Load Balancing by Redundant Decomposition and Mapping,", *Future Generation Computer Systems*, 12(5):391-406, 1997.

[9]Weinkauf, T. and Theisel, H., "Streak Lines as Tangent Curves of a Derived Vector Field,", *IEEE Transactions on Visualization and Computer Graphics*, Vol. 16, Issue 2, Oct 2010.

[10]Moran, P.J., Henze, C., "Large Field Visualization With Demand-Driven Calculation," ieee_vis, pp.2, 10th IEEE Visualization 1999 (VIS '99), 1999.

[11]Imlay, S.T., and Mackey, C.A., "Improved Performance of Large Data Visualization using Sub-Zone Load-On-Demand," AIAA 2011-1161, Jan. 2013.

[12]Slotnick, J.P., Hannon, J.A., and Chaffin, M., "Overview of the First AIAA High Lift Prediction Workshop," AIAA 2011-0862, Jan. 2011.

[13]Rumsey, C.L., Long, M., and Stuever, R.A., and Wayman, T.R.., "Summary of the First AIAA CFD High Lift Prediction Workshop," AIAA 2011-0939, Jan. 2011.

[14]Isenburg, M. "Compressing Polygon Mesh Connectivity with Degree Duality Prediction," *Graphics Interface*, 2002.

[15]Kronrod, B. and Gotsman, C., "Efficient Coding of Nontriangular Mesh Connecitvity," *Graphical Method,* 63, pp 263-275, 2001.

[16]Pajarola, R. Rossignac, J. Szymczak, A., "Implant Sprays: Compression of Progressive Tetrahedral Mesh Connectivity," *Proceedings of IEEE Visualization 99*, 1999.

[17]Gumhold, S, Guthe, S, and Strasser, W., "Tetrahedral Mesh Compression with the Cut-Border Machine," *Proceedings of IEEE Visualization 99*, 1999.

[18]Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E. and Mavriplis, D., "CFD Vision 2030 Study: A Path to Revolutionary Computational Data Science," *NASA/CR-2014-218178,* 2014.

[19]Mooreland, K., "The Tensions of In Situ Visualization," *IEEE Computer Graphics and Applications, 2016*, Vol. 36, Issue 2, Mar.-Apr. 2016.

[20]Imlay, S.T., and Mackey, C.A., "Subzone-Based In Situ Technique for I/O Efficient Analysis and Exploratory Visualization," AIAA 2017-3806, Jun. 2017.

[21]Imlay, S.T., Taflin, D.E., and Mackey, C.A., "Subzone-Based Client-Server Technique for I/O Efficient Analysis and Exploratory Visualization," AIAA 2018-????, Jan. 2018.