A Subzone-Based In-Situ Technique for I/O Efficient Analysis and Exploratory Visualization

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

The size and number of datasets analyzed by post-processing and visualization tools is 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 loading post-processing visualization and analysis, a subzone load-on-demand (SZL) visualization architecture has been developed which only loads the data needed to create the desired plot. The same I/O bottleneck also affects the writing of data to long-term storage from CFD codes. In this paper, the SZL technique is extended to eliminate the CFD code write bottleneck by writing only those subzones needed to create the desired set of plots. The new technique is a compromise between traditional In-Situ visualization techniques, which export surface-data like slices or isosurfaces extracted as part of the CFD solution process, and traditional full volume data file export from the CFD code. The subzone-based in situ files are larger than traditional surface-based in situ files but, because they contain some volume data, limited exploration and the computation of spatial derivatives is possible during post-processing. The new technique was tested with slices and isosurface. Using this technique, file sizes are reduced to as low as 0.8% of the full PLT volume dataset.

Nomenclature

α	=	angle of attack
β	=	yaw angle
a	=	cylinder diameter
C_p	=	pressure coefficient
C_{pt}	=	total 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^s_{min}	=	minimum value of variable in subzone s
v^s_{max}	=	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.

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.

The data is generally stored on arrays of hard disk drives. Over the last two decades, the storage capacity of hard disks grown in accordance with Kryder's law - doubled 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. The bandwidth for both LANs and WANs are growing more slowly than dataset size, but much faster than sustained disk-read data transfer rates.

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 1. 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).

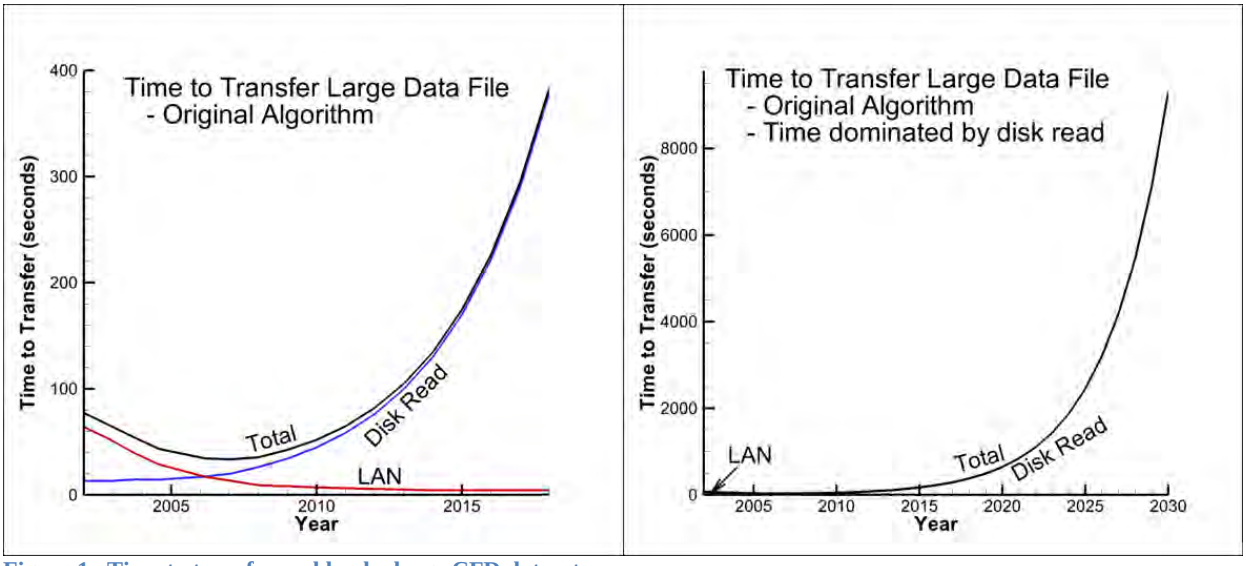

Figure 1. Time to transfer and load a large CFD dataset

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 and 100 billion points by 2025 (Figure 2).

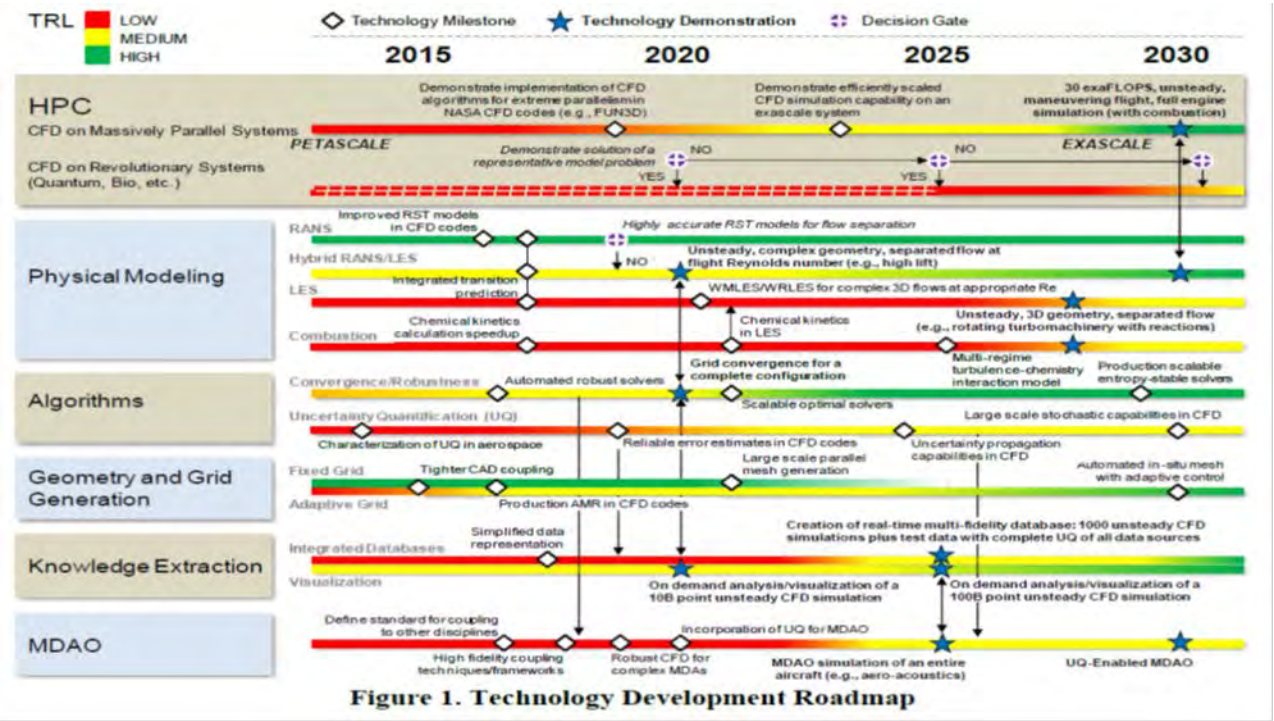

Figure 1. Technology Development Roadmap

Figure 2. Technology Roadmap from NASA CFD Vision 2030 Study[18].

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 compute with a high-bandwidth access to the data, important data abstractions are extracted, and the geometry and data on 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 do nothing to overcome the new bottleneck, sustained disk-read data transfer rates (SDRDTR).

The current hardware-based solution to this problem 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 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, interval trees can be 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 this paper, the approach is extended to the writing of the dataset from the CFD code. Instead of writing the full dataset, in situ processing is performed to determine which subzones are necessary to create the desired abstractions. For slices, isosurface, and value blanking (used, for example, in overset grids to eliminate unused nodes) the needed subzones are those satisfying a query: X, Y, or Z = value; Isosurface Variable = IsoValue; or IBlank <= 0. The query may also use ranges for the X, Y, Z, or Isosurface Variable so that limited exploration of the dataset may be performed. This is especially useful in volume-based vortex detection techniques like the Q-criteria or Lambda-2, where the best value of Q or Lambda-2 are not known a priori.

II. Approach

A. Related Work

In situ visualization has become a popular area of research over the last few years[19]. Traditional in situ visualization is the coupling of visualization software with a simulation CFD solver or other data producer to process the data "in memory" before the data are offloaded to a storage system. The product of the in situ visualization is either 1D and 2D data abstractions (like slices, isosurfaces, and streamlines), or images of the desired visualization. Both of these approaches effectively circumvent the impending data-write bottleneck by dramatically reducing the amount of data written to long-term storage. However, there are significant downsides to the traditional in situ visualization approach. The first is that the data is transient and therefore most in situ extractions are in “batch mode,” where no interaction with the visualization is possible. Also, this traditional approach competes with the CFD code for both CPU cycles and memory, potentially increasing the computation time and/or reducing the potential grid size.

B. Subzone-Based In Situ

The subzone-based in situ technique is a compromise between writing the full three-dimensional dataset and the traditional in situ visualization techniques. It couples a light-weight library with the CFD code that exports just those subzones needed to generate the desired visualization. The subzones are exported are those that satisfy a query such as “Q = 0.0”, for the 0.0 isosurface of the variable Q, or “X = 5” for the x-slice at x=5. Queries may be combined with conditionals to export the subzones for both an isosurface and slice (“Q=0.0 OR X=5”) or for the intersection of an isosurface and slice (“IBlank=0 AND X=5”).

One big advantage of the subzone-based in situ is that it allows limited exploration of the data. For example, ranges of iso-values may be specified in the query and all subzones with at least on value satisfying that query (“Q>0.0 and Q<0.1”) will be exported. This allows the user to interactively explore the isosurfaces within the specified range.

An example of the value of interactive exploration is the application of Q-criteria to the visualization of vortices. In theory, vortices exist in regions of the flow field where Q is greater than zero. In practice, Q must be adjusted to values just above zero to get a clear picture of the vorticies. For example, Figures 3 through 6 shows four isosurfaces of Q for a LES of the wake of a wind turbine. The isosurface of Q=0.0 is so complicated that it obscures the underlying structure of the vortices. The ideal value of Q for visualizing the vertical structures is probably somewhere between 0.003 and 0.01. With subzone-based in situ, you could specify that all subzones containing Q between 0.001 and 0.1 be exported to the file, and then you could explore as an interactive post-processing step the isosurface that yield the best visualization.

Subzone-based in situ also enables analysis, like that computation of spatial derivatives, that cannot be done as a post-processing step with traditional in situ approaches. This is because the subzones are actually small regions of volume data whereas traditional in situ visualization techniques save surfaces at most. An application of this capability would be to compute the vorticity magnitude on a slice.

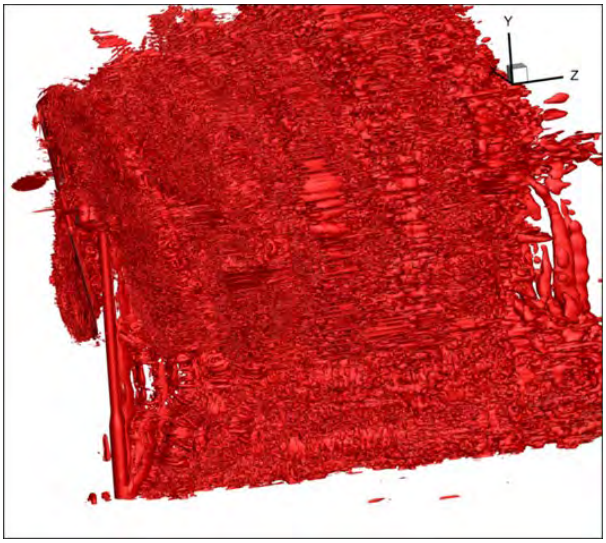

Figure 3. Isosurface of Q=0.0 for Wind Turbine Wake

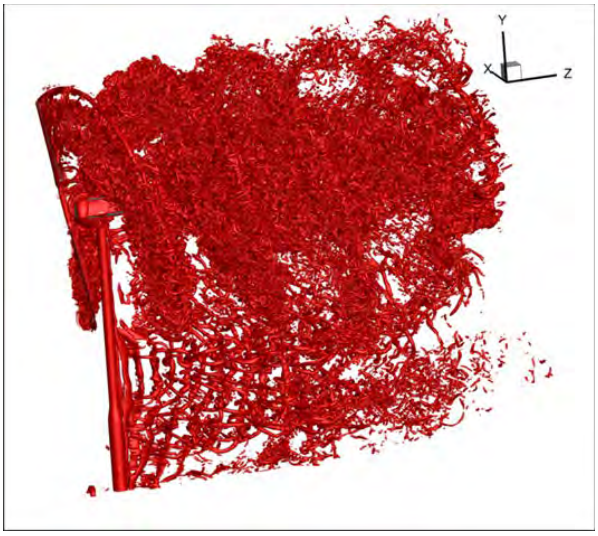

Figure 5. Isosurface of Q=0.003 for Wind Turbine Wake.

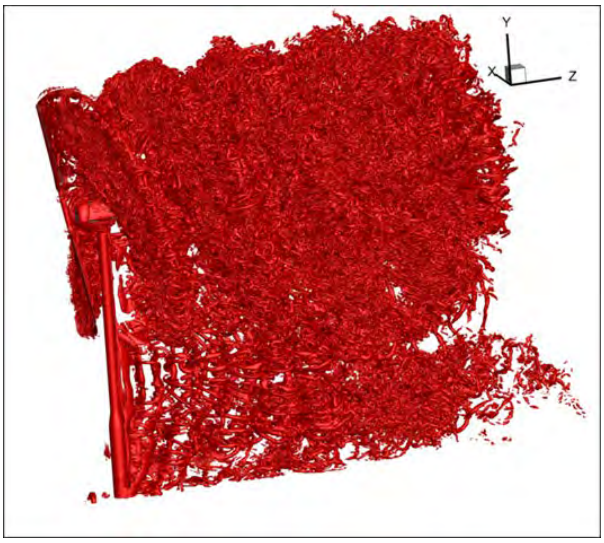

Figure 4. Isosurface of Q=0.001 for Wind Turbine Wake

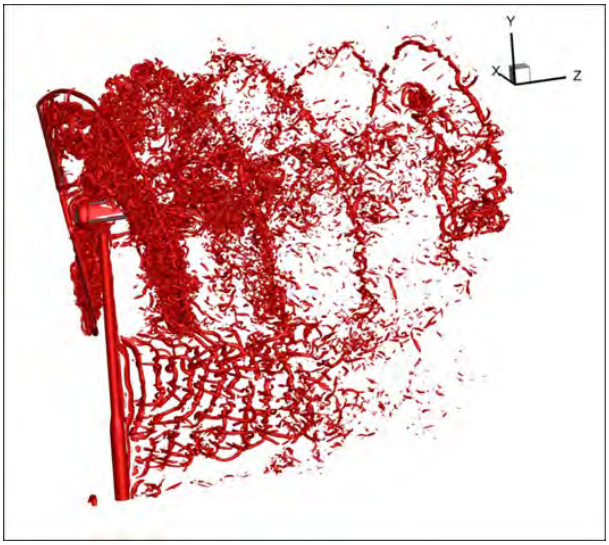

Figure 6. Isosurface of Q=0.01 for Wind Turbine Wake.

III. Results

The subzone-based in situ was tested on three CFD datasets: a large-eddy simulation of the wake of a wind turbine, the NASA trapezoidal wing dataset in the High Lift Prediction workshop[11,12],

Timing tests were run on a Windows 7 64-bit workstation having 128GB of memory and dual Intel Xeon E5-2630 6-core processors. The results are given in the following section.

A. LES of Wind Turbine Wake

The first example is a large eddy simulation of the flow in the wake of a wind turbine. This case was run in OVERFLOW 2.2 in OVERFLOW-D adaptive mesh refinement mode using 2nd-order differencing near the body and 4th-order differencing off body. At the time-step used for this test, the grid is composed of 260 million nodes in 5600 blocks.

Figure 7. NREL 10m Research Wind Turbine

Isosurfaces were extracted for a range of values of Q using both subzone-based in situ, where subzones containing the isosurface are exported, and traditional in situ, where the actual isosurfaces are exported. The results are given in Figure 8.

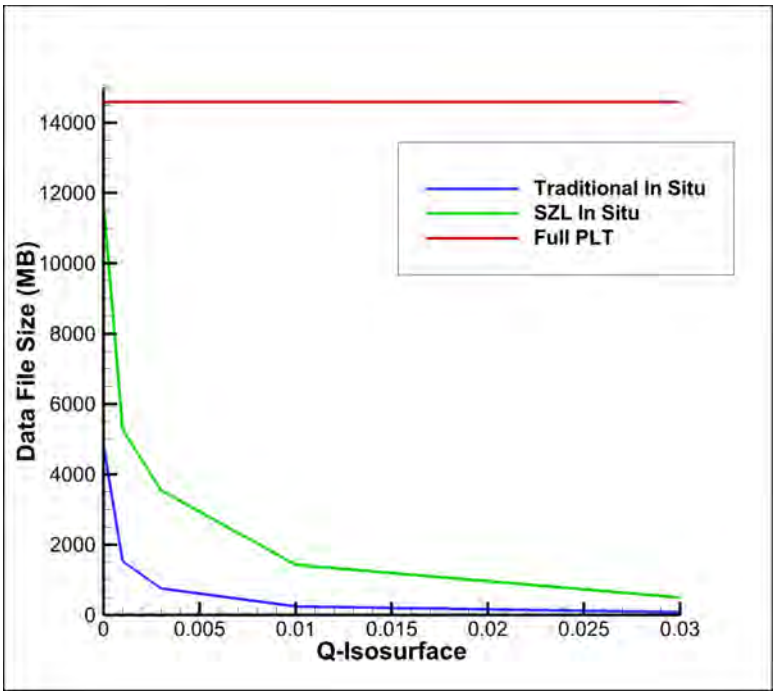

Figure 8. Comparison of data files sizes for subzone-based and traditional in situ.

Subzone-based in situ files are three to five times larger than traditional in situ files, but are much small than the full volume dataset.

For Q=0.01, load times into Tecplot 360 EX 2016r2 for subzone-based in situ is 11.8 seconds versus 70.3 seconds for the full SZL file. The memory required is 3.5GB versus 5.2GB for the full SZL file.

B. High Lift Prediction Workshop Unstructured Grid

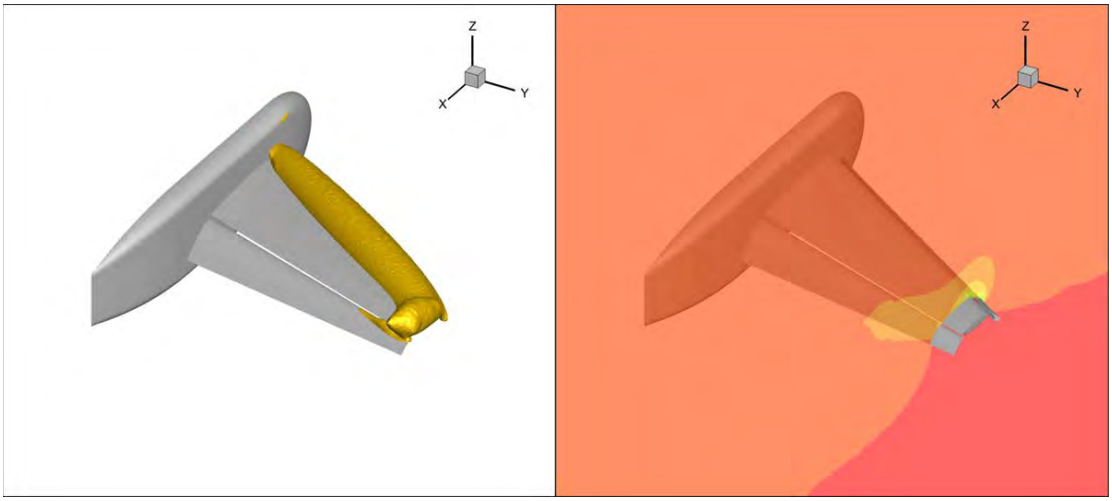

Figure 9. NASA Trapezoidal wing with Cp=-2 isosurface and a slice near the wing tip

The second example is the NASA trapezoidal wing from the first High-Lift Prediction Workshop[5,6] . The data set has 254 million cell and 76 million nodes, with hexahedra near the wall, pyramids in a transition layer, and tetrahedra in the far-field. The geometry, shown in figure 9, is a half body with wing flaps and slats extended. The full Tecplot PLT file for this case is 9.48GB and the full SZL file is 5.92GB.

For the test two visual abstractions are considered: an isosurface at Cp=-2 and a slice near the wing tip (Figure 9). For the isosurface, the subzone-based in situ file is 0.64GB, 7.7 times larger than the traditional in situ file (0.083GB) but just 10.8% of the full SZL file size and 6.8% of the full PLT file size. For the slice, the subzone-based in situ file is 0.075GB, 9 times larger than the traditional in situ file (0.008GB) but just 1.3% of the full SZL file size and 0.8% of the full PLT file size.

C. Animation of Passing Race Cars - Unstructured Grid

The third example is an unsteady analysis of two open-wheeled race cars, one passing the other. The goal of this analysis is to determine the location of the passing race car relative to the wake of the leading car. If the wake of the leading car blocks the air flow around the forward or rear wings of the passing car it will lose downforce and may not have sufficient traction to pass.

The analysis was done in CFD++ with a four overlapping grids: one for each of the cars, one for the passing zone, and one for the far-field. The total number of grid points and elements varied slightly over time but was roughly 71 million nodes and 121 million cells. There are 26 time-steps exported to plot files.

To visualize the wake of the cars we animate isosurfaces of the total pressure coefficient, C_{pt}, at 0.8 as shown in Figure 10.

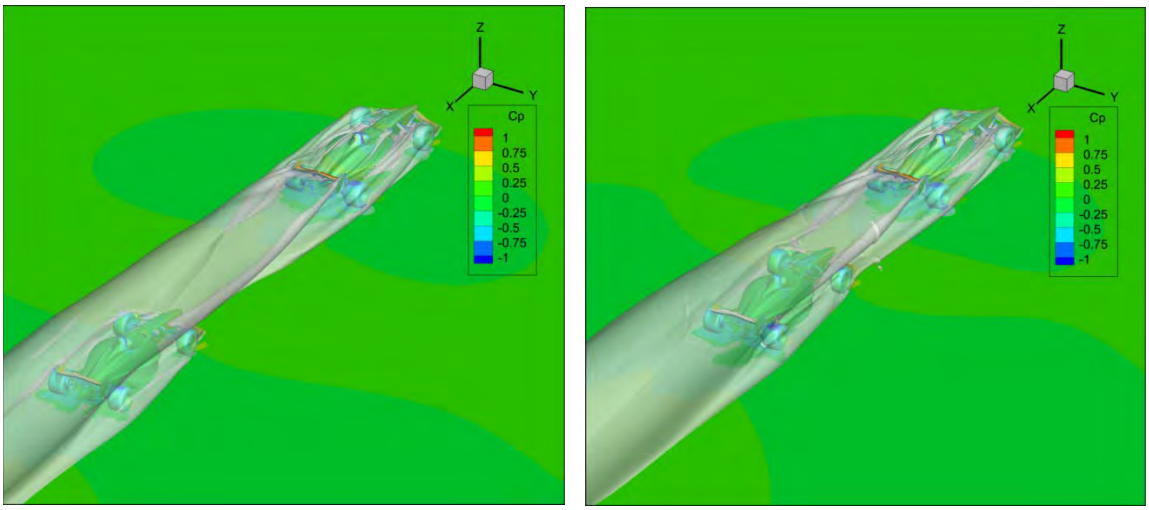

Figure 10. Wake of passing race cars - first and last time steps

For this case we compared volume data file size, animation time, and peak memory usage for Tecplot *.plt files, Tecplot *.szplt (SZL) files, the exploratory in situ files, and extracted isosurface (similar to traditional in situ). The results are show in Table 1. The test computer is a Dell Precision T7610 with 32 cores (16 physical + 16 hyperthread), 128GB RAM, an NVIDIA Quadro K4000, and Windows 7 64-bit. The surface geometry (cars) was 12.4GB and was read from a separate file.

		Animation	
File Type	Volume Data File Size (GB)	Time (Sec)	Peak Page File Size (GB)
*.plt	261.4	2455	10.8
*.szplt (SZL)	214.4	860	6.5
Exploratory In Situ	53.1	563	6.0
Extracted Isosurfaces	20.6	564	4.6

Table 1. Passing race cars in situ performance comparisons

The exploratory in situ file was a factor of five smaller than the traditional *.plt file and a factor of four smaller than the *.szplt (SZL) file. Time to animate was nearly a factor of five faster than *.plt and 53% faster than *.szplt. Interestingly, the time to animate the exploratory in situ was the same as the time to animate the extracted isosurfaces. The peak memory usage (page file size) was only slightly less than *.szplt but showed significant improvements over *.plt. Tecplot has a load-on-demand approach that, by default, retains loaded data until the total memory uses is a substantial portion of the memory available on the computer. To get a better estimate of minimum required memory, the memory model setting was changed to minimize memory usage for the final column in the table.

IV. Conclusion

The subzone-based in situ method has demonstrated significant reduction in file size compared to full volume PLT and SZL datasets. It is larger by factors of between 3 and 9 than traditional surface-based in situ files, but it provides substantially greater post-processing flexibility, including exploration of isosurfaces within a limited range of the isosurface value and the ability to compute spatial derivatives while post-processing the file. Subzone-based in situ provides a compromise between the flexibility, but large overhead, of full volume datasets and the inflexibility, but small data file size, of traditional in situ techniques. Subzone-based in situ can also accomplished with a light-weight library that is less intrusive on the CFD flow code than traditional in situ techniques.

Acknowledgments

The authors would like to thank Chris Nelson for providing the wind turbine LES dataset.

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.M., "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.