A Subzone-Based Client-Server Technique for I/O Efficient Analysis and Visualization of Large Remote Datasets

Scott T. Imlay[1], Dave Taflin[2] and Craig Mackey[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 loading 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. The same I/O bottleneck also affects the transfer of datasets from remote to local storage systems. In this paper, the SZL technique is used to create a client-server architecture where only the volume subzones needed to create the plot are transferred from the remote computer. The new technique differs from traditional client-server architectures where surface-data, such as slices or isosurfaces, are extracted by the remote server and transferred to the local client. The subzone-based client-server method transfers more data initially but can re-use the transferred volume subzones for other data extractions during further exploration. The new technique was tested with slices and isosurfaces. The initial data transfer from the server was found to be 3-17 times more than equivalent plots that transfer only surfaces, but one to two orders of magnitude less than the full volume data size. Server memory usage was substantially reduced compared with traditional client-server techniques.

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 Software Development Engineer, P.O. Box 52708, Bellevue, WA.
[3] 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 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 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).

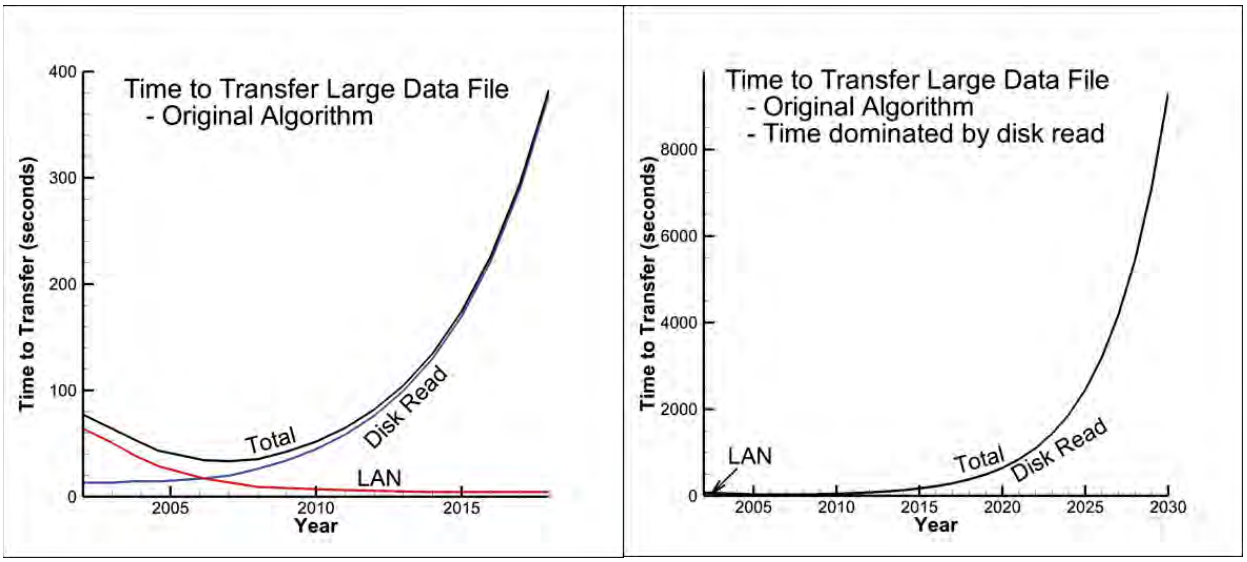

Figure 1. Time to transfer and load a large CFD dataset

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).

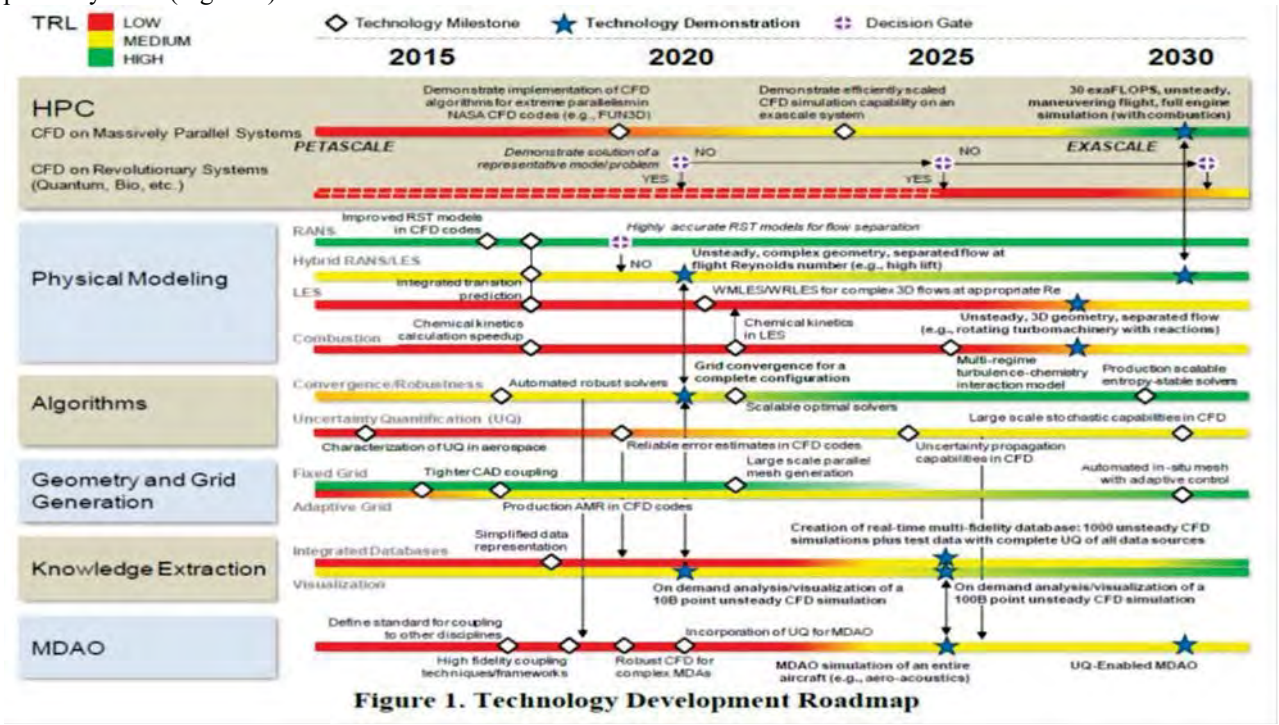

Figure 1. Technology Development Roadmap

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 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 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 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 this paper, a subzone-based client-server architecture is presented to overcome network bandwidth and latency limitations.

The new subzone-based client-server architecture transfers, from the server to the client, only those subzones needed to create the desired abstractions. The architecture is sustainable: for slices and isosurfaces, the number of subzones transferred is approximately $O(n^{\frac{2}{3}})$ and for streamtraces it is approximately $O(n^{\frac{1}{3}})$, where n is the number of cells or nodes in the volume grid. This means that the bandwidth requirements are dramatically reduced for large data files, and that the benefits of the architecture increase as the dataset size increases. Once the needed subzone data is transferred the latency experienced by the user is also dramatically reduced since all of the needed data is local on the client computer. With traditional client-server architectures, even a slight change in slice position or isosurface value triggers a re-extraction and resend of the entire surface. Also, because subzones are volume data, computations such as spatial derivatives that require volume data can be performed without transferring more subzone data. On the other hand, the subzone-based client-server does generally transfer more data than a well-designed traditional client-server architecture, which should transfer the minimal required to represent the desired abstractions. We will analyze this tradeoff more fully in the results section of this paper.

II. Approach

A. Related Work

Client-server architectures are a common approach for visualization of large remote datasets with low network bandwidth or high network latency. Traditional client-server approaches typically overcome the network limitations in one of two ways: by transferring images or by transferring data abstractions such as slices, isosurfaces, and streamtraces. Both of these approaches dramatically reduce the amount of data transferred for a single visualization. However, there are significant downsides to the traditional client-server approaches. First, multiple visualizations require that each image or abstraction be sent over the network in their entirety, not allowing any reuse of previously transmitted data. Second, the typical server architecture loads the entire volume data and generates all or most of the visualization on the server thus consuming large amounts of potentially valuable server memory and CPU time.

B. 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.

A server process is launched by Tecplot via an SSH connection, with the user providing authentication to the SSH server on the server machine. The server connects back to Tecplot via SSH's port forwarding mechanism. This procedure ensures that the user is allowed access only to those data files granted to the login account on the server machine, and that all data passed between client and server are encrypted.

The same code module ("add-on," a shared library/dynamic-link library) that Tecplot uses to load local data is used to read data files on the server. This add-on is dynamically loaded by the server process, which handles the communication between the add-on and a client add-on loaded by Tecplot itself. When Tecplot requests data to create a plot, the client add-on passes this request over the SSH connection to the server process, which calls into the loader add-on to load the data. The server transmits the data back to the client add-on, which returns it to Tecplot.

Latency issues were encountered in certain cases, such as data sets with many zones or many meta-data items ("auxiliary data"). Such data resulted in many round trips between client and server, causing a significant delay in loading. This problem was addressed by batching these requests.

III. Results

The subzone-based client-server 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] and an unsteady simulation of a jet impinging on a flat plate.

Timing tests were run on a Windows 7 64-bit workstation having 32GB of memory and dual Intel Xeon E5-2630 6-core processors. All tests were run with each of four data locations: (1) local hard disk; (2) network disk; (3) server on local network, an 8-core virtual machine; and (4) remote server, a 4-core machine running in Amazon's EC2 cloud. The first two of these loaded the data directly (not using client-server) while the last two tested client-server. Each test was timed. The client-server times included the time to connect to the server machine and launch and establish a connection with the server, which added a few seconds to these cases. The total amount of data transferred in the client-server test cases was also measured and compared with the minimum amount of data required to display plot (if the surface in question were extracted to a file, for example; this includes connectivity). Finally, the memory occupied by the server—resident set size—was noted.

A. LES of Wind Turbine Wake

The first test case is a large eddy simulation of the flow in the wake of NREL's 10-meter research wind turbine. The grid is composed of 280 million nodes in 5800 blocks. The total file size, including 13 single-precision variables, is

15 gigabytes. Tests were performed for two different plots of this data—a slice through the data and an isosurface of vorticity magnitude.

The slice consisted of about 360,000 nodes and 352,000 quadrilateral elements, representing 11MB of data. The client-server cases transferred 49MB, including all meta-data. The server consumed 425MB of RAM.

The isosurface consisted of about 3.4 million nodes and 6.6 million triangular elements, representing 131MB of data. The client-server cases transferred 335MB of data, and the server occupied 1.4GB of RAM.

The average times in seconds for each case are displayed below; individual test times varied by no more than 2 seconds.

Test	Local Disk	Network Disk	Network Server	Remote Server
Slice	5	13	36	38
Isosurface	18	93	91	156

It is interesting to note that a server running on a local network machine resulted in a faster time for the isosurface test case than the network disk. The remote server took roughly 7 times as long to perform each plot compared with loading the data from local hard disk.

-

Figure 3. NREL 10m Research Wind Turbine

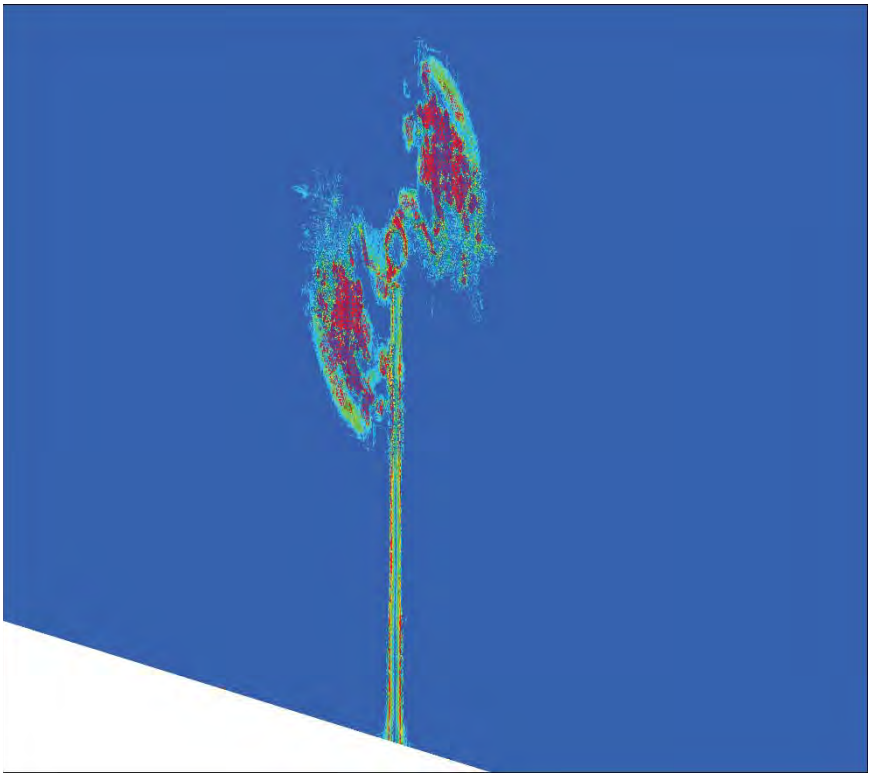

Figure 4. Slice Through Wind Turbine Wake

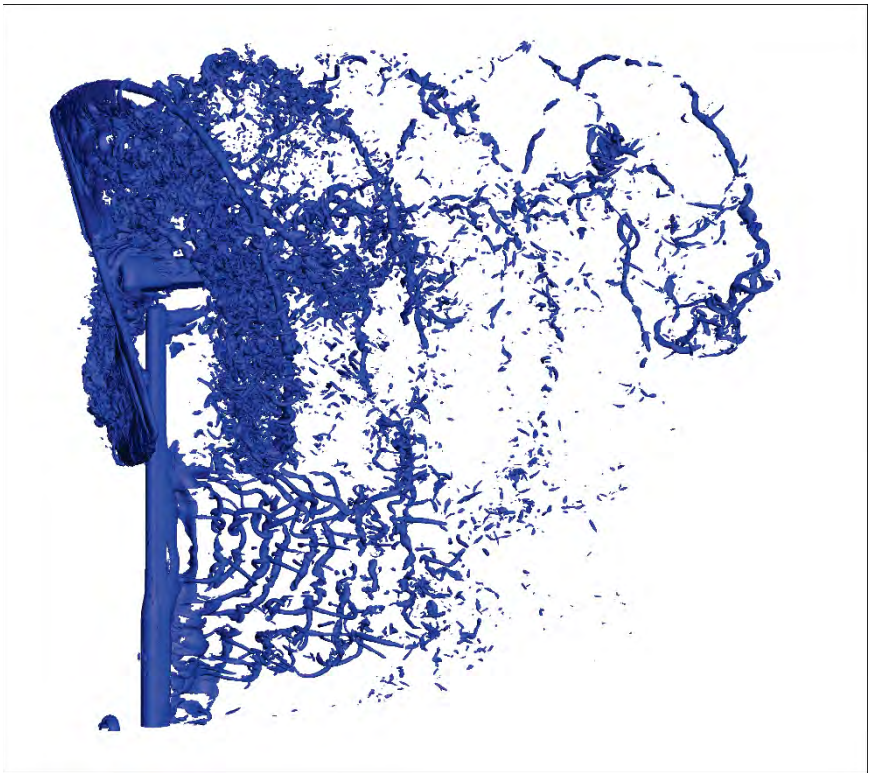

Figure 5. Isosurface of Vorticity Magnitude in Wind Turbine Wake

B. High Lift Prediction Workshop Unstructured Grid

The second example is the NASA trapezoidal wing from the first High Lift Prediction Workshop. This steady-state unstructured-grid dataset has 76 million nodes and 204 million hexahedral elements. The uncompressed file size with 12 single-precision variables is 9.9GB. This becomes 6.2GB when written to the SZL format due to connectivity compression.

Two cases were tested: A single slice near the tip, and 10 slices close together near the tip (representing fine adjustments of the slice location). All slices were colored by local pressure coefficient. Each slice consisted of about 120,000 nodes and 165,000 triangular elements. Including the grid variables, one solution variable and the connectivity, 4.4MB of data would be required to produce a single slice, or 44MB for all 10 slices.

The average test times in seconds were as follows:

Test	Local Disk	Network Disk	Network Server	Remote Server
1 slice	2	10	12	13
10 slices	7	25	26	34

The remote server case took 5-6 times as long to produce the slice as the local hard disk case. The server consumed 232MB of RAM for the single slice, and 516MB of RAM for 10 slices. It transferred 78MB of data for a single slice, and 240MB of data to produce 10 slices. The data transferred in the 10-slice case was only about 3 times that of the single-slice case, indicating that the client was able to reuse subzones transferred for the first slice in the extraction of subsequent slices.

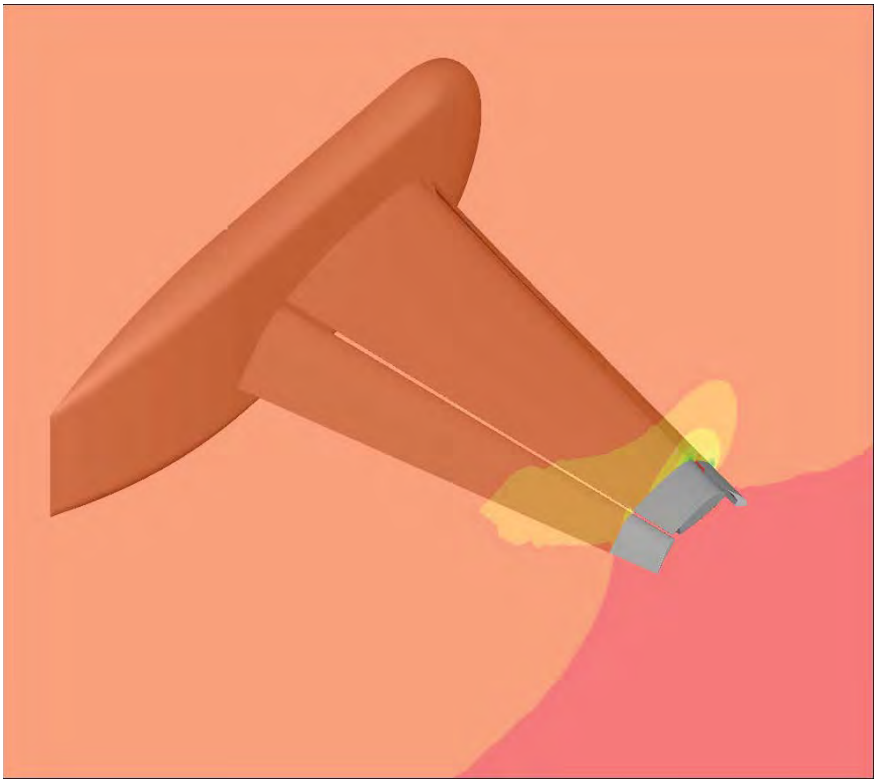

Figure 6. NASA Trapezoidal wing with Cp=-2 isosurface and a slice near the wing tip

C. Jet Impinging on Flat Plate

The final example is a jet impinging on a flat plate. The dataset is grid-steady solution-unsteady, multi-block structured. It has 81 time steps, each of which has 9 million nodes, for a total of 733 million nodes. With 8 single-precision variables, the file size is 15GB.

For the test, a slice is placed through the middle of the jet and colored with local density. The plot is then animated through all 81 time steps. A single slice consists of about 150,000 nodes and 144,000 quadrilateral elements, requiring 4.7MB of data, including connectivity. The server transferred 21MB for the initial plot. Each additional time step transferred an additional 4MB of data. The total animation transferred 338MB of data. At the end of the animation, the server occupied 2.1GB of RAM, and Tecplot 360 occupied 909 MB. Run times in seconds are shown in the following table.

Location	Time to First Image	Animation Time	Total Time
Local Disk	2	51	53
Network Disk	7	150	157
Network Server	22	143	165
Remote Server	30	163	193

The remote server total time was less than 4 times that of the local disk case, an improvement probably resulting from the client's ability to re-use the grid for each time step.

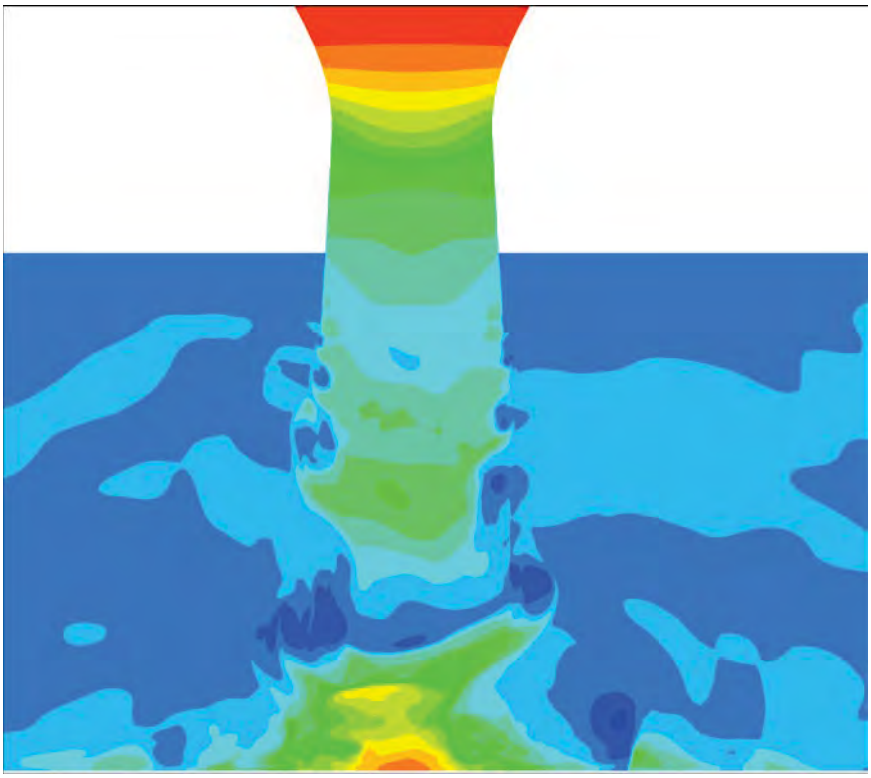

Figure 7. Slice of a jet impinging on a flat plate

IV. Conclusion

The subzone-based client-server architecture has demonstrated significant reduction in network data transfers compared to direct loading of full volume PLT and SZL datasets. The initial data transfer is larger by factors of between 3 and 17 than the data that would be transferred by traditional surface-based client-server architectures, but it substantially reduces subsequent data transfers during many post-processing tasks, including time animation, exploration of slices and isosurfaces near those previously viewed and the potential computation of spatial derivatives on the client computer. It also substantially reduces the server-side resource requirements. Subzone-based client-server provides a compromise between the flexibility with large overhead of full volume datasets transfers and the inflexibility with small data transfers of traditional in surface-based client-server techniques.

Acknowledgments

The authors would like to thank Chris Nelson for providing the wind turbine LES and impinging jet datasets.

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.