In-Situ Visualization of 10-Billion Cell Transient Data via Subzone Writing

Scott T. Imlay[1], Craig Mackey[2], and David E. Taflin[3]

Tecplot Inc., Bellevue, WA, 98006

The subzone load-on-demand (SZL) visualization architecture has been presented in prior work. Subsequent work has extended it to support in-situ visualization for steady-state CFD simulations. In this work, we further extend it to support in-situ visualization of unsteady CFD simulations. We analyze its performance using four unsteady data sets: an unsteady multi-block structured CFD simulation of a wind turbine, an unsteady unstructured-grid hurricane simulation, and synthetic structured and unstructured data. We then examine performance scaling and project the utility of this approach in view of the anticipated growth of CFD simulations in the next decade. Compared with full data set output, the data reduction due to in-situ extraction ranges from 83% for the hurricane simulation to 99.5% for the largest structured synthetic data. This advantage is shown to increase as data set size increases. The data size required for unsteady iso-surfaces are shown to scale with $O(n^{\frac{2}{3}})$, where n is the number of cells in the grid, making this approach a viable candidate for the trillion-cell simulations expected in the next decade.

[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

NASA's CFD Vision 2030 study projects the need for on-demand analysis and visualization of 10 billion-point unsteady CFD simulations by year 2020, and of 1 trillion-point unsteady simulations by 2030.[1] Supporting this goal requires analyzing and addressing the various bottlenecks in the visualization pipeline.

Simulation data are 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 all network types is growing more slowly than dataset size, but faster than sustained disk-read data transfer rates, such that disk read is overtaking network bandwidth as the primary bottleneck in simulation post-processing.

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 are loaded on a remote computer with a high-bandwidth access to the data, important data abstractions are extracted, and the geometry and data on these abstractions are transferred across the slow network to a local client. In this context, "abstractions" may include model geometry, slices, iso-surfaces, 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 three 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 long-term storage of 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, such as reduced memory requirements and reduced network bandwidth requirements. This is one architectural approach taken by Tecplot Inc. for large-data visualization.

In a previous paper[2], 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 iso-surfaces, 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[3], 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. In this paper, the subzone-based in-situ will be applied to unsteady data.

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.

The term "in situ visualization" covers a wide variety of methods. According to the "In Situ Terminology Project"[4] these methods can be characterized using six axes: integration type, proximity, access, synchronization, operation controls, and output type.

Our in situ method is directly integrated into the simulation code using a general-purpose library. This is the most common integration type, incorporated by popular in situ tools such as ParaView Catalyst[5] and Libsim[6]. Other in situ tools, such as ADIOS[7] and GLEAN[8] use a shared protocol to indirectly connect the two components.

The other major category that more distinguishes our in situ technique from the others is output type. Most in situ techniques, such as Catalyst and Libsim, export images or, in rare cases, extracted surfaces. Our in-situ technique outputs the minimum number of subzones (subsets of the volume grid) necessary to create the desired images.

B. Subzone-Based Architecture

The subzone load-on-demand technology is described in the following subsections. The basic technology, described in the first subsection below, dramatically reduces the time and memory required to visualize a large data-set. Subzone-based in-situ, described in the second subsection, is used to reduce the disk space and time required to write the data-file from a CFD code running a very large case.

Subzone Load-On-Demand

The basic subzone load-on-demand architecture was described in a previous paper[2] 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. Node subzones are used to store nodal data, while cell subzones are used to store cell-centered data, plus cell connectivity for unstructured grids. Subzone load-on-demand (SZL) files are 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 subzone variable data and the cell subzone connectivity arrays. These connectivity arrays are compressed by replacing the 64-bit 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 unstructured-grid data.

The variable min/max trees allow the software to only load those node and cell subzones necessary to create the desired slices, iso-surfaces, or streamtraces. For iso-surface extraction, the software loads the relevant iso-surface variable min/max tree, searches for those node and cell subzones with a min-value less than the iso-surface value, and a max-value greater than the iso-surface value, and loads only those subzones. The triangles making up the iso-surface are then extracted using a standard marching-cubes (or marching-tets) algorithm. Slices are treated as iso-surfaces of a coordinate variable.

The total size of data required to generate a particular iso-surface 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.

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 data-set were written.[3]

Extension to Unsteady Data

The extension of in-situ subzone load-on-demand to transient structured data is straightforward. For convenience of generation, each time step for the synthetic data was generated as a number of separate blocks, which become separate "zones" when loaded into Tecplot 360, and are organized into "strands" over time. This added a trivial amount of overhead to the process.

For unstructured data, grid coordinates and connectivity are output for each time step. We note a possible optimization for the case of grid-steady solutions, where grid coordinates and connectivity could be shared between time steps. But given that the feature of interest—an iso-surface, for example—is likely to be moving over time, the subzones needed for the grid will be different for different time steps. Sharing the output grid would then require a consolidation of the grid-related subzones over all time steps, which would require either additional processing each time step (adding new subzones to an existing grid file or stored memory) or a post-processing step to consolidate all output grid files. Neither of these approaches was pursued in this work. In addition to avoiding the extra processing that the optimization would require, another advantage of the current approach is that it allows restarting and adding time steps as needed.

III. Results

Four transient cases were studied: Two CFD solutions (one structured and one unstructured) and two cell synthetic data sets (again, one structured and one unstructured).

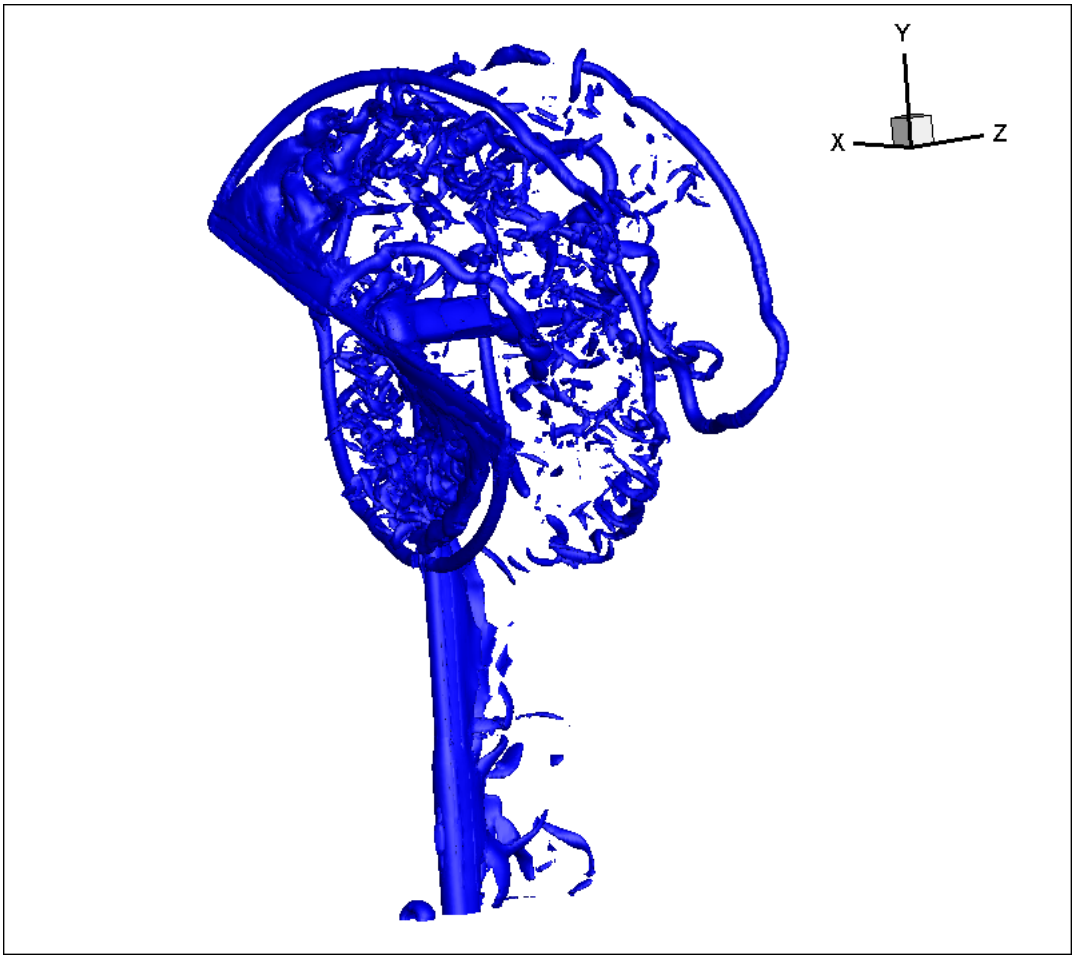

Figure 1. Wind Turbine Data Set

The wind turbine data set, shown in Figure 1 and supplied courtesy of Chris Nelson, Innovative Technology Applications Company, LLC, is 158 time steps of structured overset data with 13 variables. The data varies between

409 blocks (“zones”) for the first time step and 5,863 blocks for the final time step. The full data is 370GB, and the in-situ subzone data file size was 24.5 GB for a single iso-surface, but including all 13 variables for further exploration. In addition to reducing the subzones by the iso-surface value, blanking was also considered to remove entirely blanked regions and further reduce the data file.

Wind Turbine CFD Dataset: Structured, 158 time steps			
Dataset Size (Cells/Time Step)	Full Data File Size	In-Situ Data File Size	Reduction
Varies from 38 million to 263 million	370 GB	24.5 GB	93.4%

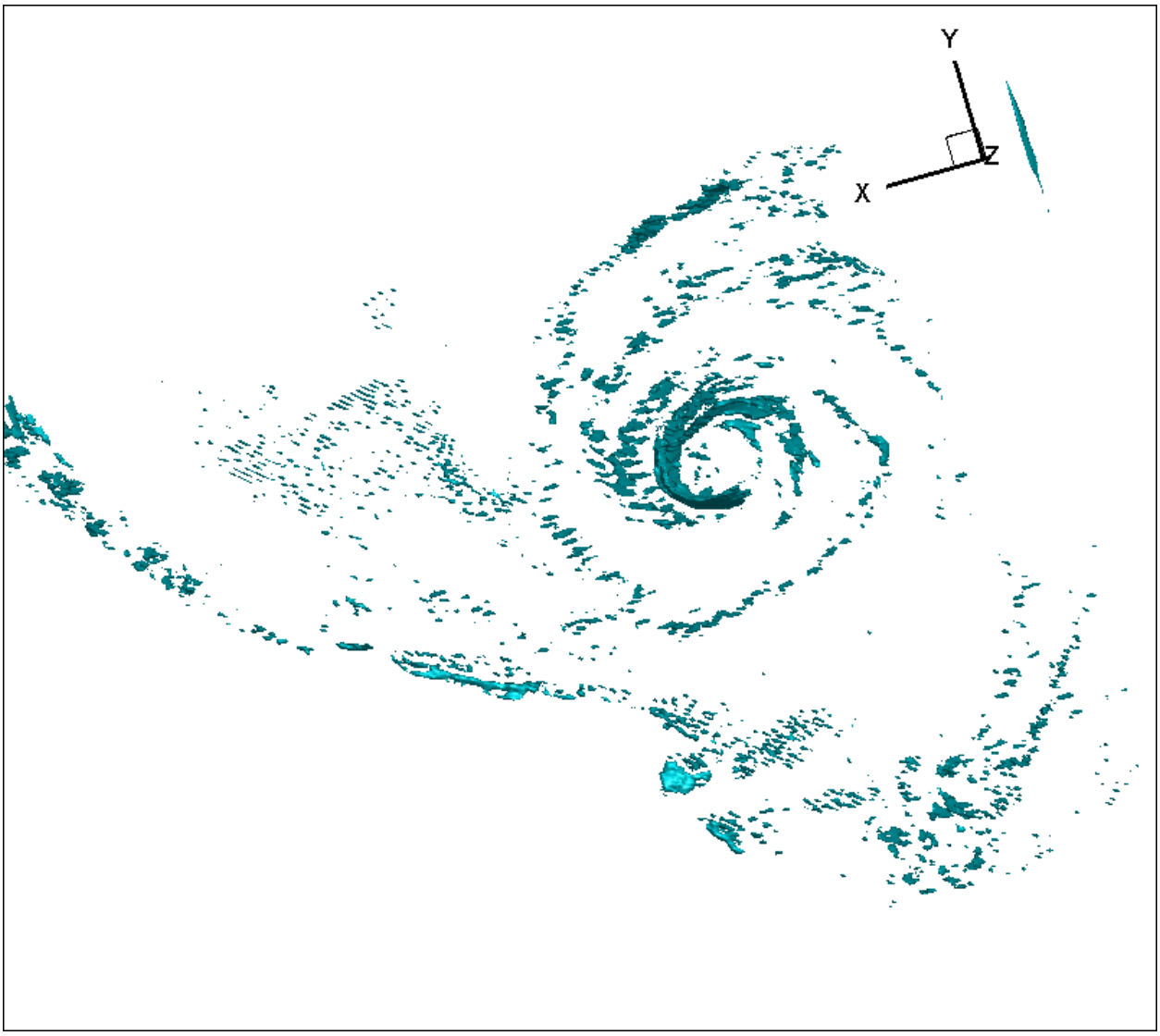

Figure 2. The Hurricane Data Set

The hurricane simulation data set, shown in Figure 2, is 18.8 million tetrahedral cells over 48 time steps. It was generated from the SciVis 2004 Hurricane Isabelle visualization contest data. The data is organized into a single block (“zone”) with 16 variables for each time step. As this data was unstructured, the data shared connectivity and coordinate variables, and was 8.16 GB. (Not using this sharing would have resulting in a total data set size of 23.7

GB.) The in-situ subzone data file size was 1.41 GB for a single iso-surface, including all 16 variables for further exploration. This represents a reduction of 83% from the original data (and 94% from the unshared version).

Hurricane Simulation Dataset: Unstructured (tetrahedral), 48 time steps			
Dataset Size (Cells/Time Step)	Full Data File Size	In-Situ Data File Size	Reduction
18.8 million	8.16 GB	1.41 GB	83%

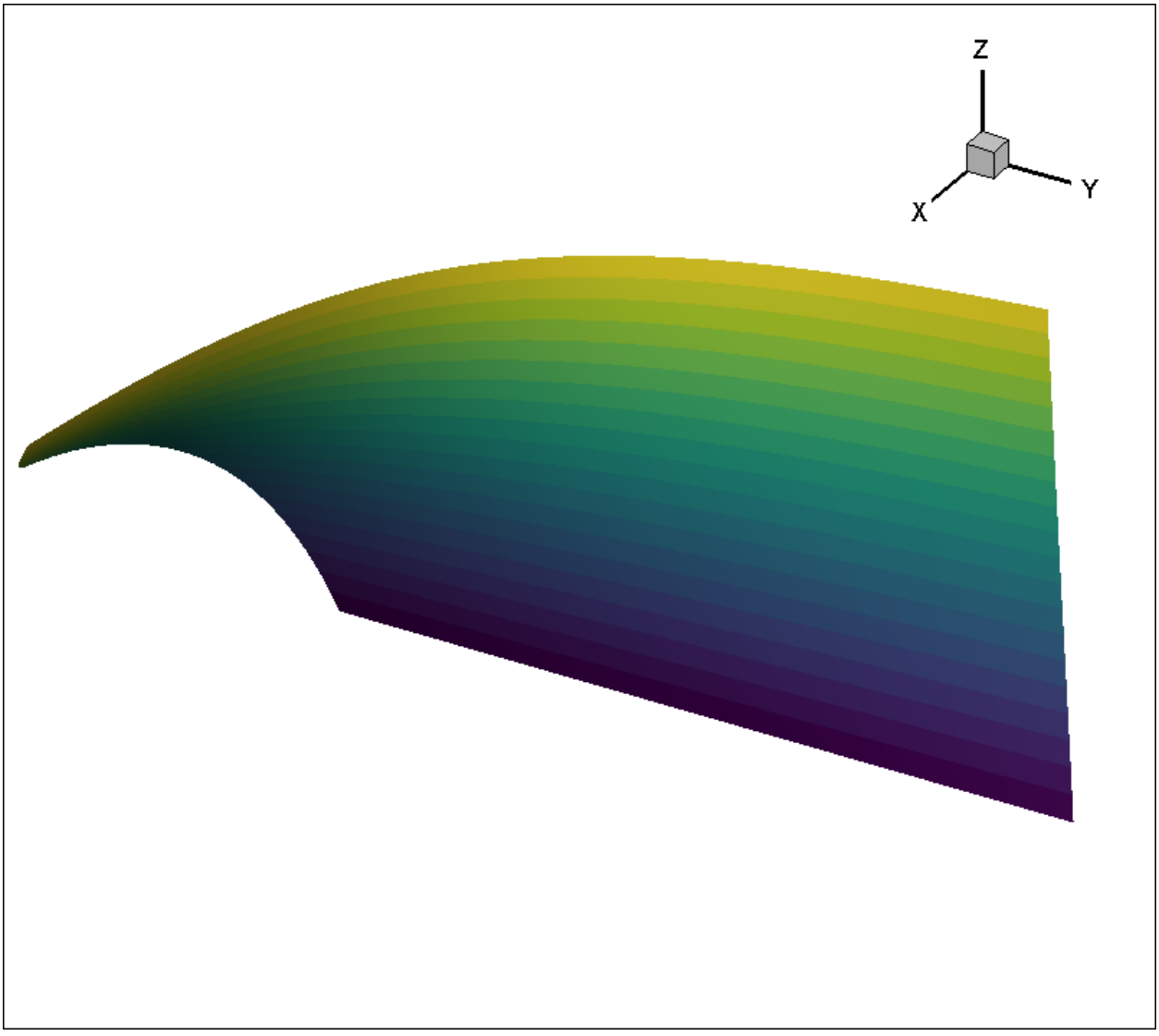

Figure 3: The Synthetic Data Set

To study the scalability of in-situ subzone writing, two series of synthetic cases were constructed: one structured and one unstructured (using tetrahedra). Within each synthetic series, data sets of five different sizes were processed: one-million cells, ten-million cells, one-hundred-million cells, one-billion cells and ten-billion cells. Each case was the same time-varying polynomial calculated within a cylinder over 11 time steps. Each data set had four variables.

For the synthetic structure cases, the data reduction due to in-situ subzone writing was 90.6% to 99.5%. For unstructured cases, the data reduction was 86.4% to 99.4%. The reduction percentage increased as dataset size increased.

Synthetic Datasets: Structured, 11 time steps			
Dataset Size (Cells/Time Step)	Full Data File Size	In-Situ Data File Size	Reduction
1 million	198 MB	18.5 MB	90.6%
10 million	1.88 GB	87.5 MB	95.4%
100 million	18.3 GB	415 MB	97.7%
1 billion	181 GB	1.92 GB	98.9%
10 billion	1.8 TB (est.)	8.88 GB	99.5%

(The 10-billion-cell full data set was not generated in its entirety, so its size is estimated from smaller cases.)

Synthetic Datasets: Unstructured (Tetrahedra), 11 time steps			
Dataset Size (Cells/Time Step)	Full Data File Size	In-Situ Data File Size	Reduction
1 million	97.9 MB	13.3 MB	86.4%
10 million	967 MB	67.5 MB	93.0%
100 million	9.64 GB	291 MB	97.0%
1 billion	95.4 GB	1.30 GB	98.6%
10 billion	950 GB (est.)	5.56 GB	99.4%

(The 10-billion-cell full solution data set was not generated in its entirety, so its size is estimated from smaller cases.)

The scalability of the in-situ data file size is shown to scale with $\boldsymbol{O}(\boldsymbol{n}^{\frac{2}{3}})$ in Figure 4.

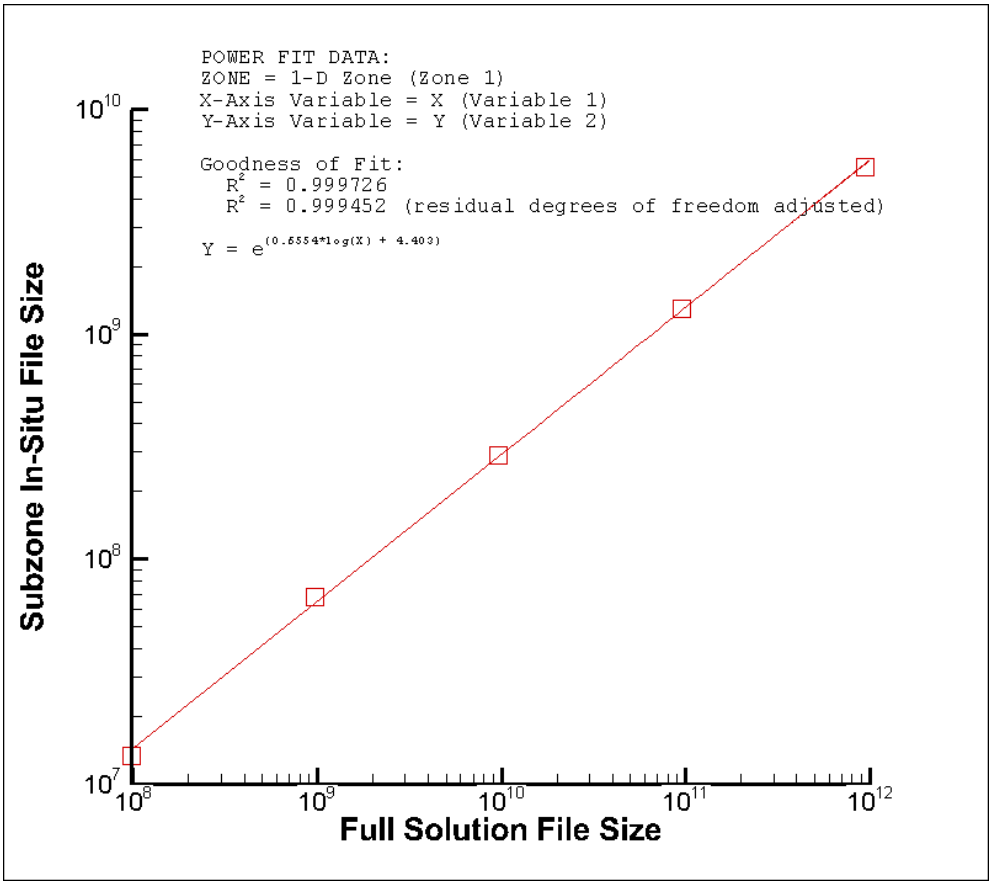

Figure 4. Scaling on In-Situ File Sizes

IV. Conclusions

In-situ subzone writing can significantly reduce the amount of data by CFD applications. Scaling shows that the reduction in data writing increases with problem size. This allows large transient solutions of 10-billion to be easily saved and stored on an engineering workstation.

References

[1]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.

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

[3]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.

[4]Bauer, A.C., Abbasi, H., et. al. "In-situ Method, Infrastructures, and Applications on High Performance Computing Platforms," EuroVis 2016, STAR, Volume 35, Number 3, 2016.

[5]Bauer, A.C., Geveci, B., and Schroeder, W. "The ParaView Catalyst User Guide," Kitware Inc. 2015.

[6]Childs, H., Ma, K.-L., et. al. "In-situ Processing," in *High Performance Visualization-Enabled Extreme-Scale Scientific Insight*, Chapman & Hall, CRC Computational Science, CRC Press/Francis-Taylor Group, Boca Raton, FL, USA, Nov. 2012, pp. 307-329.

[7]Lopstead, J.F., Klasky, S., Schwan, K., Podhorszki, N., and Jin, C, "Flexible i/o and integration for scientific codes through the adaptable io systems (adios)," In *Proceedings of the 6th international workshop on Challenges of large applications in distributed environments*, ACM, pp 15-24, 2008.