2016 International Conference on Collaboration Technologies and Systems

High-resolution interactive and collaborative data visualization framework for large-scale data analysis

Simon Su, Vincent Perry, Nicholas Cantner
US Army Research Lab, Aberdeen Proving Ground
simon.m.su.civ@mail.mil

Dylan Kobayashi, Jason Leigh
University of Hawaii, Honolulu, HI

Abstract — **We present a high-resolution interactive and collaborative data visualization framework capable of supporting multiple user interaction on a large screen display. The data visualization framework takes advantage of SAGE2 collaborative visualization workspace. SAGE2 is designed to take multiple displays and use them as one high-resolution multiuser workspace. SAGE2 users are able to access and manipulate this workspace through a modern web browser. Our data visualization framework, ParaSAGE, supports scientific data visualization by extending ParaViewWeb into a SAGE2 desktop application where a subset of ParaView scientific visualization functionalities is accessible to SAGE2 users. We also developed a SAGE2 desktop application based on D3.js library to support information data visualization efforts.**

***Index Terms* – Visualization, Collaboration.**

I. INTRODUCTION

In a fast paced, rapidly changing, data driven world, it is crucial to be able to quickly analyze large datasets to identify trends, anomalies, and correlations. For problems in the visual analytics domain, identification of these trends, anomalies, and correlations in data from past events improves our ability to anticipate future events to stay at the forefront of ongoing developments. For problems in the computational sciences domain, the ability to visualize and analyze vast datasets can help improve the accuracy of the computational model.

As the problems faced by data analysts and computational scientists today become increasingly larger and more complex, it is no longer a challenge that can be addressed by a single analyst or scientist. A realistic problem requires dedicated team members with interdisciplinary expertise that can offer a diverse perspective to solve the problem at hand. Therefore, the concept of data visualization has evolved into collaborative large-scale data visualization, with collaboration as one of the grand challenges for visual analytics [1].

Data visualization framework capable of large-scale high-resolution visualization opens up the possibility for the user to visualize and interact with more of the data present in a large-scale data analysis problem. In scientific data visualization, visualization of a dataset with millions of data points requires a display system with at least that many pixels for an overview visualization of the data. For information data visualization, the unstructured nature of the data unfolds many different ways to visualize the data, accompanied by a demand for larger screen resolution requirements.

In addition, visualization framework capable of supporting multiuser interaction can also contribute to the collaboration process. All the team members working on a visualization collaboration session will have the ability to interact directly with the content displayed on the visualization system using their own mouse. Team members can interact with the shared visualization workspace as they would their own personal laptop, while sharing and discussing their insight resulting from both the visualization and collaboration efforts.

We developed our data visualization framework as a Scalable Amplified Group Environment2 (SAGE2) [2] application to utilize the collaborative visualization capability provided by the framework. SAGE2 collaborative workspace environment enables the use of a large-scale, high-resolution multi-display wall for concurrent analysis. Multiuser interaction is also supported in SAGE2. For the first SAGE2 application, we extended ParaViewWeb to run on SAGE2 framework for scientific data visualization. For information data visualization, we developed a D3 JavaScript library based SAGE2 application.

We designed our data visualization framework to support co-located, synchronous visualization scenarios [3]. The high-resolution visualization capability of our framework has limited our ability to support distributed collaborative visualization due to the display hardware requirement. Chen listed scalability as one of the top 10 unsolved information visualization problems [4] and our high-resolution visualization framework is capable of processing and displaying large amounts of data. Our high-resolution large-scale display system allows the users to see more of the data in the "Overview first" phase of Schneiderman's Visual Information Seeking Mantra [5].

In the next section, we describe some of the related work in collaborative visualization. The following section describes the hardware and software of our setup. SAGE-IVF, the information visualization application that we built on top of the SAGE2 framework is discussed in the following section. We then elaborate on ParaSAGE, a SAGE2 application for scientific data collaborative visualization, before we conclude.

II. RELATED WORK

Visualization tools capable of supporting collaborative visualization have been developed over the years. Su et al. developed distributed collaborative virtual environment to support distributed collaborative visualization of molecular structure in a virtual environment [6]. Users from different virtual environments are able to collaboratively visualize and examine the same molecular structure displayed. The tool

978-1-5090-2300-4 2016

DOI 10.1109/CTS.2016.57

supports equal interaction between the users in the virtual environment by using the menu system to determine which of the two users is in control of the interaction. Users are able to share their viewpoint with the other user, which has the effect of showing the other user the exact view of the protein structure that they are collaboratively investigating. Distributed Collaborative PaulingWorld was developed using VRTool virtual reality toolkit with networking protocol extension to support improved users collaboration experience.

Using VRTool virtual reality toolkit, Loftin et al. presented a distributed collaborative visualization application for training [7]. NASA and European Space Agency (ESA) used the application in a simulated distributed training where they collaborated on a space mission to repair the Hubble Space Telescope. During the training session, the two astronauts were able to exchange virtual objects within the same virtual environment. The astronauts were given equal interaction capabilities with the virtual environment, and protocol for collaboration conflict resolution was not considered in this collaborative training application.

Besides SAGE2 framework, DisplayCluster is another software framework capable of supporting high-resolution collaborative visualization. Johnson et al. developed DisplayCluster software framework to support high-resolution collaborative visualization on an 80-tiled multi-display system at Texas Advanced Computing Center on University of Texas at Austin's campus [8]. DisplayCluster is a cluster driven, interactive visualization environment that is capable of supporting multiuser interaction. Users are able to control their own simulated mouse cursor with an Xbox game controller to interact with the visualization content on the high-resolution display system. Although both SAGE2 and DisplayCluster have comparable support for collaborative visualization, our research pointed to a lower entry barrier for our web based visualization application to be developed with the SAGE2 framework.

A. *SAGE2 framework*

SAGE2 is a software framework to enhance data intensive co-located and remote collaboration. SAGE2, which is a complete rewrite of the original SAGE software, is developed using web-browser technologies. SAGE2 is accessed through its web server component, which is aware of all connected clients. SAGE2 has two types of clients, namely display clients and user interface (UI) clients. The display clients' Uniform Resource Locator (URL) string includes information of the corresponding viewport in the overall multi-display system, while the UI clients' URL string includes uniquely identifiable information. For a display client, the URL address determines the viewport position in the overall workspace. Since the web address determines which view in the workspace a display client shows, it is possible to mirror the view by having another browser access the same address.

A SAGE2 application is a fully contained JavaScript web application that runs on the SAGE2 framework. Although SAGE2 server is aware of all open applications and their general display properties, it has no knowledge of what is displayed by the respective applications. Each display client has to track and allocate resources for all open SAGE2 applications, even if the application is not currently visible on that display client's viewport. The SAGE2 UI page has the overall view of all open applications and their layout. Interaction with displayed content can be done through a SAGE2 pointer or through the UI's overview page. Although the SAGE2 pointer is not a real mouse pointer, it is used to manage the mouse events across all display clients by propagating JavaScript mouse events to the relevant display client.

III. VISUALIZATION SYSTEM HARDWARE AND SOFTWARE SETUP

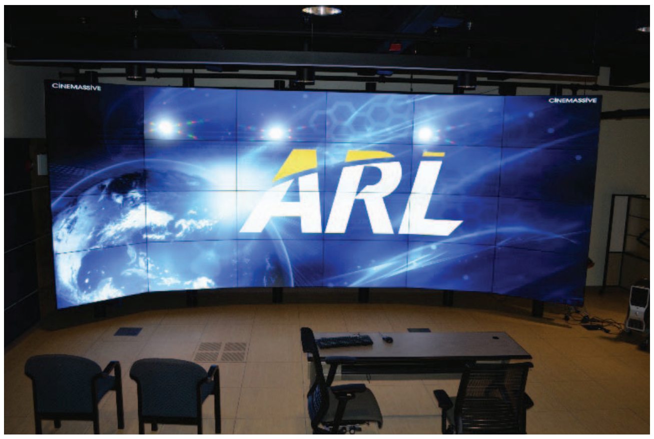

Figure 1. CineMassive display system

As shown in Figure 1, the multi-display tiled system consists of 24 professional grade 42" LCD displays, mounted in a slightly curved configuration of 6 columns by 4 rows. Each of the CineMassive displays has a native HD resolution of 1920x1080 pixels. This surmounts to an entire wall resolution of 11520x4320 across all 24 screens. The current hardware setup has four machines that are responsible for driving the display wall. The server head node hosts and runs the software packages including SAGE2, ParaView, and ParaViewWeb. The other three machines are client nodes, responsible for outputting to the display wall. Each client node has 8 outputs, driven by 2 nVidia Quadro M6000. With three client machines and 8 outputs per machine, there is a total of 24 outputs from the client nodes to the display wall. Thus, each output from the three client machines routes directly to one of the 24 displays on the display wall. All machines have about 128G of memory and run linux CentOS 7. The head node does not have any output connection to the display wall, but is instead responsible for hosting and distributing what is to be displayed to the client nodes. Each client machine is connected to the head node via a 40 GbE Mellanox network that allows for fast data transfer from the head node to the client nodes. The connection between machines also allows for the head node to remotely ssh into each of the three client machines, as well as connect directly to their X-servers. These connections, as well as the head node's access to the wireless outreach network, are vital to running SAGE2 on the head node and displaying across the display wall.

SAGE2 is a web server that runs on the head node server machine. All SAGE2 clients connect to a web page through the browser. From the head node, the user may ssh into the client machines and begin connecting to the web pages through browsers. Users connect to a web page referred to as a UI client.

Because the head node also resides on the Outreach network, other users on the Outreach network are able to connect and interact with the SAGE2 UI. When SAGE2 is implemented on multiple tiled displays, the expected usage is that each display screen has one full screen browser pointing to its corresponding viewport in the workspace. SAGE2 display clients are actually browsers running a web page in full screen. Typically, there are display number of browsers open. Running in the browser allows utilization of the available JavaScript libraries. The SAGE2 server only manages what apps are open, where they are open, and how big they are, as well as relaying the interaction information from UI clients. The SAGE2 server running on the head node knows nothing about the specifics of an app or how it is displayed, but merely hosts the data to be interacted with and viewed on the client machines.

IV. INFORMATION DATA COLLABORATIVE VISUALIZATION FRAMEWORK

SAGE-IVF is our information visualization framework application, designed to support large data visualization. A variety of web-based tools were used in the development of our visualization framework. The framework was constructed as a webpage using both HTML and JavaScript. D3 and jQuery JavaScript APIs were used in the development of the visual component of the framework. Bootstrap API was also used to develop the navigation menu bar for the application, which allows the user to select and deselect files, variables, and the types of charts to be displayed, as well as to build and clear visualizations. We exploited the JSON (JavaScript Object Notation) file format to load the data into our framework. JSON format allows efficient loading of data into memory for further formatting and processing of the data via jQuery. After the data was parsed and formatted, we used D3's (Data-Driven Documents) JavaScript library to build a variety of visualizations.

To manage the complexity of the application development, we first developed the visualization framework independent of the SAGE2 framework. One of the main design goals was to decouple the data from a specific visualization technique. The framework also automatically loads all the JSON files in the application's working directory. Once the data is parsed into memory, all of the uploaded JSON files become available for selection by the user. Once the user selects a file, any of the variables found in the corresponding JSON file and the type of visualization must be selected before building the visualization. For example, after selecting the Wind_Data.json file from the dropdown list of JSON files, all the keys in the Wind_Data.json file will be listed under the variables dropdown list. After selecting the variables WindSpeed and WindDirection, as well as the type of visualization, the framework will then plot each variable on its own separate visualization. Since a typical dataset in our domain is time based, our framework will automatically plot the variables selected against time. Remaining consistent with above, two visualizations will be created; one plotting WindSpeed vs. Time, and the other plotting WindDirection vs. Time. The user can then interact with both visualizations at the same time to deduce conclusions from the data. At this stage of implementation, the responsibility of selecting the appropriate visualization method for the chosen dataset lies with the user.

Once we completed a functioning prototype, we ported the framework into a SAGE-IVF SAGE2 application on the SAGE2 framework. This allowed our framework to run as an application within the SAGE2 workspace, taking advantage of high-resolution and multiuser interaction capability. In our SAGE2 setup, a SAGE2 application has access to all 49.76-megapixel resolution of our tiled multi-display system. The tiled multi-display system allows our visualization framework to use the enormous amount of screen real estate to produce and view a multitude of visualizations. We have ported line chart, bar chat, and parallel coordinate visualization methods in our SAGE2 application.

Another design goal of SAGE-IVF was to enable data correlation between different visualizations. With various distinct visualizations displayed, the user may select a data point on one of the visualizations and have that same data point highlighted across all other visualizations where that data point resides. After the user selects a data point, SAGE-IVF analyzes all the currently active visualizations generated by the user. If any of the visualizations contain the same data point selected, SAGE-IVF will highlight all of the data points across all of the visualizations. This enables the user to generate a multitude of unique visualizations, but still view the same, shared data at the same time. Highlighting of the same data point across multiple visualizations allows the user to concurrently analyze the data in different visualization formats in order to identify correlations, trends, and patterns among the data. Figure 1 shows SAGE-IVF running on our 24-tiled multi-display system using the VAST 2016 Challenge data.

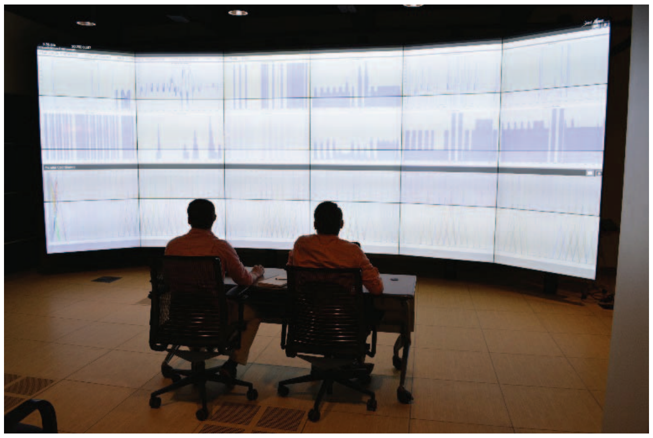

Figure 2. SAGE-IVF Application

A. *Implementation Challenges*

There are many challenges associated with converting a written html document into a SAGE2 application. Since SAGE2 is a browser-based tool, the layout of the html document for each browser is already structured into 'head' and 'body' sections, set on launch of SAGE2's server. Thus, the majority of the html content for the browsers is consistent across any launch of SAGE2, and an html-based application must be sure to only adapt the html code within the application itself.

When an application is launched from the SAGE2 UI, a new html 'div' element is appended to the 'body' section of each browser's html document. Each application launched is contained within its own 'div' element, each with a unique application ID. Since SAGE2 is written completely in the JavaScript programming language, the SAGE2 source folder contains a SAGE2_App base class that provides the structure of the necessary components of an application to be overridden.

Since SAGE-IVF was originally written as its own webpage, it contains its own html document that defines the initial setup of the browser, as well as script reading to load JavaScript files from external source folders. Instead of completely rewriting the html document in the JavaScript language, we were able to adapt the html document itself to be loaded into the app and merged with other JavaScript files. One challenge we faced with loading in SAGE-IVF's html document was being able to call an html reader in JavaScript. To do this, we had to pass the html file path to a new XMLHttpRequest, which then grabbed the html content from the file. Once the html content resided within this XMLHttpRequest, we appended the html content directly to the 'div' element of our application. However, to incorporate the SAGE-IVF's content into a SAGE2 application, much of the original IVF's html document needed to be modified or removed.

In the IVF's html document, many external JavaScript source files were preloaded using html's 'script' tag. However, when the XMLHttpRequest loaded the SAGE-IVF's html content, the script lines were placed in the application's html code block, but the source files associated with the script elements would never actually load. Instead, after the SAGE-IVF's html document loaded into the app, the script lines needed to be extracted back out of the app's html code, manually defined as new script elements in JavaScript, then appended back into the app's 'div' element. Only then would the source files attached to the 'script' tags be loaded in to the application.

Once we were able to load in the external source files, the next task to overcome was dependency issues. To avoid being plagued with undefined reference errors, all files that our application depended on needed to be loaded in before our application could launch. Because some of the loaded files depend on other files being loaded in, we also needed to nest the file loading process by using callback functions, to ensure that the file loading was asynchronous and would avoid errors. This occurs in the initialization of our SAGE2 app, with the last callback loading SAGE-IVF's adapted html document. So all loading of external source files are completed in the application's initialization before the framework is even available for user input and interaction.

The most challenging aspect of merging the IVF into a SAGE2 application was syncing the SAGE2 pointer actions on the client side to act as actual mouse events on the server side. The SAGE2 pointer is actually an html 'div' element that collects mouse data, such as mouse clicks and the coordinates at which they occur on the client side, interprets their effect on the server side, then relays the effect of those actions back to the client side. When the SAGE2 pointer interacts with an application, the pointer knows which app to interact with based off of the unique application ID. However, within the SAGE-IVF app, each chart needs to act independent of the other charts on mouse interaction. When the SAGE2 pointer dragged or resized one chart, all of the charts would also drag or resize despite not having been clicked. To compensate for this, each chart had to be assigned it's own unique ID, and the SAGE2 pointer had to not only query the ID of the application it was interacting with, but also the ID of the chart within the SAGE-IVF app that it was interacting with. Another setback of the SAGE2 pointer functionality is that of the scrollbar. The SAGE2 pointer currently only registers a mouse scroll event if a scrollbar is present in the html element directly under the SAGE2 pointer. This affects the ability to zoom in on a chart that does not contain a scrollbar element, but still utilizes mouse-scrolling functionality. The last challenge we have faced with SAGE2 pointer interaction is that not all browsers on the display client side register mouse events when they occur. This has to do with how the mouse events propagate across the different number of browsers. There is only one master display, with the rest of the browsers acting as clients. When the SAGE2 pointer invokes a mouse event on the client side within the master display, all other client displays are ensured to receive the mouse event and update accordingly. However, if the SAGE2 pointer invokes a mouse event in one of the non-master display screens, it is not guaranteed that all browsers will update accordingly. This is a current issue we are facing within development, and plan to resolve this issue in the future. Figure 2 illustrates the overall design of SAGE-IVF.

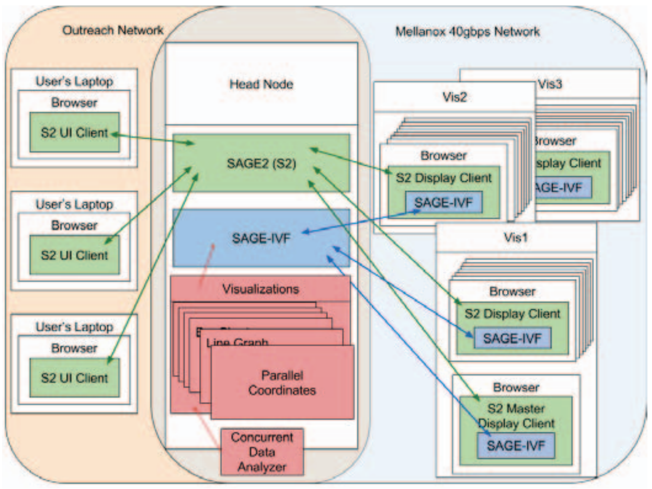

Figure 3. SAGE-IVF architectural design

B. *Discussion and Future Work*

We implemented a time-based information visualization framework within the SAGE2 framework to take advantage of large-scale high-resolution display capability for large data visualization. While working on the VAST 2016 challenge, we constantly ran into the issue of insufficient display real-estate to display the large amount of VAST 2016 challenge data in our original visualization framework. We had to resort to creative visualization to overcome the lack of display real-estate to get an overview of the VAST 2016 challenge data. Even with creative visualization technique, there is just too much data that we need to show in order to have a basic overall understanding of the data. The limitation we encountered working on VAST 2016 challenge is also a contributing factor in our decision to develop a large-screen high-resolution information visualization framework. Visualizing VAST 2016 challenge data using

SAGE-IVF has resulted in new insights into VAST 2016 challenge data.

We plan to continue our efforts to add more visualization and the visualization specific interaction to SAGE-IVF. The JSON file format defined in our SAGE2 application sets a standard for other data to be formatted into similar format to take advantage of information visualization capability offered by SAGE-IVF. For visualization using a different dataset, we will format the data into the same JSON format to take advantage of the visualization and interaction we developed into SAGE-IVF.

V. Scientific Data Collaborative Visualization Framework

ParaView is an open source multi-platform data visualization tool with advanced data analysis capability [9]. ParaView Qt based GUI is providing the user with a standardized interface to VTK. The scalability of ParaView allows large scale data analysis to be performed on the HPC systems used to generate the data. Although Immersive ParaView plugin [10] allows ParaView to be configured to display on a high-resolution tiled display system, user interaction is very different from ParaView running on the desktop.

ParaViewWeb enables visualization and data analysis capabilities of ParaView to be used within web applications [11]. It is a collection of components that employ HTML 5.0 web technologies to enable remote usage of ParaView's visualization and data analysis features. Among those components is a Python script that runs a web server. The server hosts web pages that are able to interface with ParaView through a lightweight JavaScript API. Remote users are able to take advantage of the hardware on the host system for visualization and interact with ParaView through a browser without any software installation. However, ParaViewWeb's user experience was designed for a single user with a single display.

Since ParaViewWeb enables remote interaction of ParaView through a JavaScript API, the necessary technologies are already in place for the development of ParaViewWeb as a SAGE2 application.

SAGE2 enables multiple displays to be used as one multiuser workspace. The framework combines multiple displays (running on a single system or across multiple systems) to act as one large workspace that can be employed by multiple users through a modern browser and network access without needing to install additional software. However, to use the desktop screen shared option of SAGE2, the user will have to install a plugin for Chrome browser. Firefox users will only have to modify some configuration setting to allow desktop screen sharing on SAGE2. Using the modern browser, SAGE2 users are able to access and interact with the workspace.

Using ParaViewWeb and SAGE2, we developed ParaSAGE as a SAGE2 application that allows full resolution utilization and multiuser interaction of the SAGE2 setup. Since the interface provided by ParaViewWeb was designed for a single user and single display, ParaSAGE is a rebuild of the interface in order to correctly display and interact with the data in a multi-display environment. ParaSAGE connects to ParaViewWeb server using ParaViewWeb's packet system in order to avoid ParaViewWeb's detailed information on configuring a connection, process to connect, authentication, and communication packets. Using the same process when we developed SAGE-IVF, the initial prototype was done outside of the SAGE2 framework to reduce the complexity of the implementation. Using the same web technology used by ParaViewWeb, we also used Autobahn|JS library to setup and established the connection to ParaViewWeb. ParaViewWeb uses Autobahn|JS library to simplify packet handling that is compliance with the Web Application Messaging Protocol. The configuration to establish a connection involves specifying transport realm authentication method, authentication ID, and a function that addresses the authentication protocol. For the authentication protocol setup, although an authentication key is needed, it was impossible to specify the key before running the server. ParaViewWeb facilitated the authentication using a predefined vtkweb-secret string.

After we successfully establish a display connection to ParaViewWeb web server, the standalone web application prototype was converted into a SAGE2 application. The conversion involves refactoring the variables and functions to comply with SAGE2 application formatting. After the conversion into SAGE2 application, we also added the interface for the users to specify the web address of the ParaViewWeb server. Furthermore, we also expanded the application to work in a multi-display environment by accounting for changes in display resolution and added interaction supports to the view rendered. Figure 4 illustrates the overall designed of the ParaSAGE application.

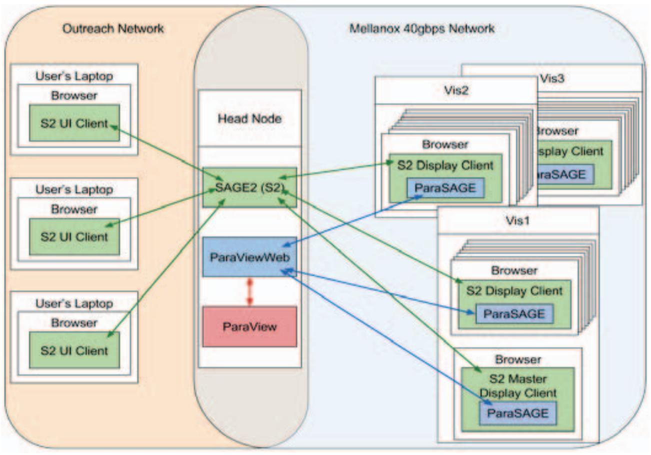

Figure 4. ParaSAGE architectural design

A. *Implementation Challenges*

The development of ParaSAGE shared a lot of the implementation challenges of converting SAGE2-IVF into a SAGE2 application. To support interaction across multiple displays in ParaSAGE, SAGE2 pointer needs to maintain a uniform effect across multiple display clients. User interaction in ParaViewWeb involves camera manipulation based upon mouse cursor location, mouse button status, and mouse movement on the render view. In SAGE2 pointer interaction mode, the raw mouse event data from a UI client's browser are used to control the SAGE2 pointer. However, SAGE2 pointer is not a real mouse cursor. It is just a visual representation showing the move based on the UI client's raw mouse event data. Furthermore, the mouse pointer data is sent to a SAGE2 application only if it was not detected as a server side action (move or resize).

To support mouse interaction for ParaSAGE, the pointer's position also needs to be normalized into the application space instead of SAGE2 space. SAGE2MEP function was added to SAGE2 framework to convert SAGE2 pointer data from the appropriate JavaScript MouseEvent and dispatch that event to the correct node within a SAGE2 application. SAGE2MEP allows using JavaScript event infrastructure to support mouse interaction in a dynamic application. For example, without SAGE2MEP, a mouse click button would require the location of the mouse cursor coordinates to write a bounding box check on the click position. In addition, the developer would also need to account for mouse button movement and scale without the abstraction provided by SAGE2MEP function.

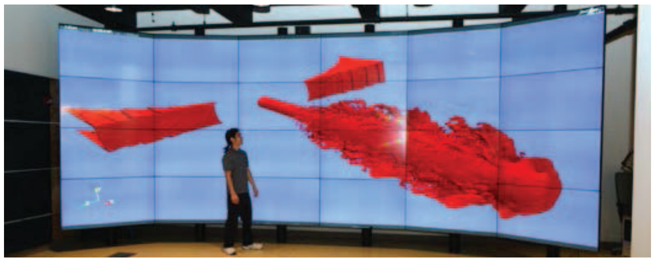

Figure 5. ParaSAGE

B. Discussion and Future Work

As shown in Figure 5, this version of ParaSAGE was tested on a 24 tiled CineMassive system driven by a visualization cluster outputting at 49.76 megapixel total resolution. The dataset rendered shown in Figure 5 is a fuel injection data with 9 million data points. The main motivation of ParaSAGE is to provide an overview visualization capability to computational scientists with large simulation data.

In addition to SAGE2 pointer interaction, ParaSAGE can incorporate all rendering settings applied to ParaViewWeb. An informal user study is planned for ParaSAGE, and based on the data visualization usability requirements we gather, we also plan to support additional ParaViewWeb interaction through ParaSAGE. To contribute our work to the general community, ParaSAGE will also be distributed as a standard SAGE2 application available with every SAGE2 installation. Additional user studies are being planned to determine the impact of high resolution visualization in facilitating data exploration and discovery efforts.

VI. Conclusion

We presented a large-scale high-resolution collaborative visualization framework supporting scientific and information data visualization. ParaSAGE extends ParaViewWeb with the ability to natively view unobstructed high resolution rendering, and SAGE-IVF provides existing information visualization with the same rendering capability. The scalability of SAGE2 framework puts the limit of supported display resolution and size to that of the available graphics hardware. SAGE2 framework allows ParaSAGE and SAGE-IVF to keep the same user interaction modality for large-scale high-resolution and desktop display. Furthermore, SAGE2 multiuser interaction capability will also help facilitate collaborative visualization. The seamless transition in user interaction experience should encourage user adoption of the data visualization tools.

References

[1] J.J. Thomas, K.A. Cook, editors. Illuminating the Path: The Research and Development Agenda for Visual Analytics. IEEE Computer Society; 2005

[2] 1. T. Marrinan, J. Aurisano, A. Nishimoto, K. Bharadwaj, V. Mateevitsi, L. Renambot, L. Long, A. Johnson, and J. Leigh, "SAGE2: A New Approach for Data Intensive Collaboration Using Scalable Resolution Shared Displays" (best paper award), 10th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing. 2014.

[3] P. Isenberg, N. Elmqvist, D. Cernea, J. Scholtz, K.-L. Ma, and H. Hagen. Collaborative Visualization: Definition, Challenges, and Research Agenda. Information Visualization, 10(4):310–326, 2011. doi:10.1177/1473871611412817

[4] Simon Su, R. Bowen Loftin, David T. Chen, Yung-Chin Fang, Ching-Yao Lin, "Distributed Collaborative Virtual Environment: PaulingWorld," The Proceedingsof 10th International Conference on Artificial Reality and Telexistence, Oct. 25-27, 2000, pp. 112-116.

[5] R. Bowen Loftin, Patrick J. Kenney Robin Benedetti, Chris Culbert, Mark Engelberg, Robert Jones, Paige Lucas, Mason Menninger, John Muratore, Lac Nguyen, Tim Saito, Robert T. Savely, and Mark Voss, "Virtual Environments in Training: NASA's Hubble Space Telescope Mission," The 16th Interservice/Industry Training Systems & Education Conference held in Orlando, Florida on November 28 - December 1, 1994.

[6] Gregory P. Johnson, Gregory D. Abram, Brandt Westing, Paul Navratil and Kelly Gaither, "DisplayCluster: An Interactive Visualization Environment for Tiled Displays," 2012 IEEE International Conference on Cluster Computing

[7] J. Ahrens, B. Geveci, and C. Law. "ParaView: An End-User tool for Large Data Visualization," January 2005

[8] N. Shetty, A. Chaudhary, D. Coming, W.R. Sherman, P. O'Leary, E.T. Whiting and S. Su, "Immersive ParaView: A community-based, immersive, universal scientific visualization application", Virtual Reality Conference (VR), 2011 IEEE, pp. 239-240

[9] http://paraviewweb.kitware.com/

[10] Ben Shneiderman, The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations, Proceedings of the 1996 IEEE Symposium on Visual Languages, p.336, September 03-06, 1996

[11] Chen, C., Top 10 Unsolved Information Visualization Problems, IEEE Computer Graphics and Applications, 25(4):12-16, July-Aug. 2005