SAGE2: A New Approach for Data Intensive Collaboration Using Scalable Resolution Shared Displays

Thomas Marrinan, Jillian Aurisano, Arthur Nishimoto, Krishna Bharadwaj, Victor Mateevitsi, Luc Renambot, Lance Long, Andrew Johnson
Dept. of Computer Science
University of Illinois at Chicago
Chicago, IL, USA

Jason Leigh
Dept. of Computer Science
University of Hawai'i at Mānoa
Honolulu, HI, USA

***Abstract*—Current web-based collaboration systems, such as Google Hangouts, WebEx, and Skype, primarily enable single users to work with remote collaborators through video conferencing and desktop mirroring. The original SAGE software, developed in 2004 and adopted at over one hundred international sites, was designed to enable groups to work in front of large shared displays in order to solve problems that required juxtaposing large volumes of information in ultra high-resolution. We have developed SAGE2, as a complete redesign and implementation of SAGE, using cloud-based and web browser technologies in order to enhance data intensive co-located and remote collaboration. This paper provides an overview of SAGE2's infrastructure, the technical design challenges, and the afforded benefits to data intensive collaboration. Lastly, we provide insight on how future collaborative applications can be developed to support large displays and demonstrate the power and flexibility that SAGE2 offers in collaborative scenarios through a series of use cases.**

Keywords—Large Displays; Co-located Collaboration; Remote Collaboration; Window Manager; Cloud Technologies; Multi-user Interaction; Computer Supported Cooperative Work

I. INTRODUCTION

Today, scientific and industrial data is collected, stored, and analyzed digitally, often in the cloud. Modern science is driven by new types of digital instruments and sensors capable of collecting data at ever-increasing resolutions. Natural phenomena, from global weather systems to chemical reactions at the atomic level, can now be simulated with supercomputers, generating massive volumes of data. These troves of data are invaluable to researchers and businesses as they explore the raw information and evidence needed for new insights, discoveries, and innovations. However, making those insights is an increasingly complicated task as the scale and complexity of data continue to grow at unprecedented rates. Since big data problems frequently require the combined efforts of many individuals from disparate fields, the next generation of data intensive visualization and interaction environments will need to enable collaboration and group work.

To deal with the scale and complexity of data, the 2007 DOE Visualization and Knowledge Discovery workshop report [1] and the 2008 NSF Building Effective Virtual Organizations report [2] recognized that new modalities for accessing more visual information were necessary, and described large shared displays as the type of environments that are crucial for collaborative cyber-enabled exploration. Furthermore, there is now conclusive evidence that large display environments enable collaboration and significantly amplify the way users make sense of large-scale, complex data [3-5].

Since large display environments present technical challenges and unique affordances, specialized software and middleware are needed to allow users to capitalize on these benefits in authentic settings. In 2004, our group developed SAGE, the Scalable Adaptive Graphics Environment [6], for large shared displays without resolution constraints, which we term Scalable Resolution Shared Displays (SRSD).

SAGE is an open-source middleware that provides multiple users with a common operating environment to access, display, and share an assortment of data intensive information. The software allows each user to create a pointer on the SRSD by using their own personal device, or to directly approach the SRSD and interact through a multi-touch interface. In this manner, multiple users can simultaneously add and interact with content. SAGE displays pixel streams from remote sources by utilizing high-speed networks to render content ranging from high definition images and videos to PDF documents and laptop screens. At the time, SAGE, in conjunction with an SRSD, represented a new type of *digital lens* – an ultra high-resolution display that can effectively

Fig. 1. A typical SAGE2 session, depicting multiple applications windowed on a SRSD. This figure illustrates the wide array of content supported by SAGE2, including images, videos, PDFs, 2D and 3D custom applications, and an off-the-shelf application from a remote source.

978-1-63190-043-3 © 2014 ICST
DOI 10.4108/icst.collaboratecom.2014.257337

visualize large volumes of data in a collaborative environment. As a result, over one hundred sites have adopted SAGE around the world over the past decade. However, its architecture was based on a monolithic design that made it increasingly difficult to integrate new capabilities as user requirements grew.

Since SAGE was first deployed in 2004, the technical landscape has shifted. With the advent of HTML5, web browsers have become powerful rendering tools. Contemporary browsers provide access to high-performance graphics and networking capabilities that were formerly only achievable with native applications. Web applications and JavaScript programming are becoming increasingly popular, attracting a large community of developers eager to reach a broader audience. In turn, this audience has embraced web-based applications due to their ease of access.

In this paper, we present SAGE2 (depicted in Fig. 1), a prototype for the next generation of collaborative SRSD middleware that capitalizes on the increasing power of the cloud and web browser. To our knowledge, SAGE2 is the first collaborative windowing environment for SRSDs that runs in a web browser. This paradigm affords many advantages, but required us to overcome an array of technical challenges.

II. Lessons Learned From Deploying SRSDs

Since SAGE was released in 2004, we have worked closely with users in academia, research, and industry to capture authentic collaborative workflows and motivate SAGE development [7,8]. In addition to these direct observations of SAGE in use, we conducted a user survey to capture feedback from 40 sites on how SAGE was used, its benefits, and features users wished SAGE had. Of the desired future features of SAGE, the most requested were 1) integration of multi-user applications, 2) enhanced real-time distance collaboration, and 3) a reduced barrier to entry.

A. Collaborative Workflows Enabled by SAGE

Results from our user survey indicate that our user community is diverse, with 63% from academia, 20% from industry, and 17% government research labs. 70% of these sites reported that their SAGE installation is used for more than one project. Specific uses of SAGE include sharing telemedicine lectures, showing the output of large-scale scientific simulations, and running multiple ultra high-resolution applications simultaneously. Our survey also indicated an upward trend on adoption and usage of SRSD technologies. Currently 55% of sites have more than one SAGE installation, with 62% projecting to have more than one in the next four years. Also, currently 20% of sites have more than four SAGE installations, with 32.5% projecting to have more than four in the next four years. These installations vary in resolution, with many sites having SRSDs below 8 MPixels and many other sites having SRSDs above 100 MPixels.

In addition to receiving feedback through a user survey, we have worked closely with authentic SAGE users and observed its use. Theses observations have given us insight on several types of collaborative workflows enabled by SAGE that are difficult to achieve through other platforms. For instance, SAGE was often used for regular, large group meetings where group members presented short updates to their collaborators. In these meetings, SAGE enabled *lightning collaboration*, where each collaborator could rapidly share content, such as images, videos, and PDFs, or share the screen from their laptop, in order to illustrate their progress. Transitions between presenters were quick due to each user being able to load content and create a pointer from their personal device, rather than taking turns using a master controller.

We also observed collaborators using SAGE in a *parallel investigation workflow*. In this workflow, several collaborators gathered together to investigate a problem using a SRSD. Each user had specific domain expertise that pertained to one aspect of a problem they were attempting to solve. Working on personal machines, these users each explored one or two online databases, or loaded data in an interactive application on their laptops to display its content. Seeing these disparate pieces of information together on a SRSD prompted new questions, which each researcher explored in parallel during the meeting, sharing results step-by-step in SAGE. This work was only possible because SAGE permitted multiple users to simultaneously share and control content on the SRSD.

SAGE was used in another type of collaborative work session, which we call *single-driver, multiple-navigators*. In this type of session, the driver would share an interactive application running on his/her laptop to explore a dataset and share findings. Navigators would ask questions, prompting the driver to perform specific operations on the dataset through his/her personal machine. SAGE enabled this collaborative work by allowing everyone to see the content simultaneously on a SRSD and by capturing real-time changes on the user's machine. In addition, navigators could point to content on the SRSD, using pointers created from their personal machines, to highlight aspects of the content.

Lastly, SAGE has been used for remote collaborative work sessions, with participants sharing multimedia content between SAGE environments with *remote gigabit data streams*. These modes of co-located and remote collaboration work remarkably well for a variety of problems, but only act as a step towards improved computer supported cooperative work.

B. Collaborative Roadblocks in SAGE

While the original SAGE has enabled collaborative workflows that were previously not possible, users have identified situations in which they desired more. In order to share and interact with content on the SRSD, users needed to download and install a client application. We noticed that in situations with frequent new users, the requirement to install this client application to engage in a SAGE session acted as a barrier to entry and limited participation for new users in *lighting collaboration* scenarios.

We observed that *parallel investigation workflows* were challenging in places where users wanted to interactively engage the content of their collaborators or synthesize results from multiple investigations, since content on shared screens remained controllable only by the single owner. We observed that this often resulted in duplicated work or diminished participation. We also noticed that while users could share large volumes of information, the lack of integration across

content made it challenging for researchers to combine results from the parallel investigations.

In collaborative workflows involving *single-driver, multiple-navigators*, there was a challenge with collaborators needing to switch roles. Since everyone could not directly manipulate content within the shared application simultaneously, users would be forced to take turns as the driver. Occasionally, one navigators needed to duplicate the work of the driver, so they could interact with the data on their personal devices. This delayed the collaborative session, and in some instances limited participation and work sharing.

Finally, we noted that remote collaboration frequently involves configurations of users that were not adequately supported by the original SAGE architecture. Remote single users (a user without access to a SAGE session or SRSD) were limited to viewing and sharing content by standard teleconferencing systems, which did not adequately allow for participation. In addition, while multiple SAGE sites could freely and independently reposition content to fit the needs of a co-located group, other sites were unable to gain insight on the position of individual items on remote SRSDs. This often resulted in miscommunication due to the lack of context (e.g. "see the image on the right").

Our user survey supports our observations concerning collaborative roadblocks. Users listed over 25 applications that they wished were SAGE compatible for multi-user interaction, many of which were web-based, such as Google Maps. Users also identified enhanced remote collaboration through data sharing and native teleconferencing as a top desired priority.

III. Related Work

Collaboration has been enhanced by advances in different areas of research, such as leveraging web-based applications for real-time communication and utilizing large display systems for rendering data intensive content. While the work described below (outlined in table 1) has made significant contributions, none of them encapsulate a holistic approach to better enable collaboration in data intensive scenarios.

A. Web-based Collaboration

Co-located and remote collaboration has been greatly affected by the advancement of web-based technologies. The web is increasingly used to collaborate in a variety of scenarios such as teaching, corporate meetings and manufacturing. Google Hangouts, Skype, and WebEx [9-11] are examples of successful commercial products designed to enable remote collaboration. While they are all effective tools for single users, none of them successfully address the needs for distributed groups working with large volumes of high-resolution data. Binary Meetings [12], a web-based collaboration tool for lab tutorial classes, enables students to interact with each other and with lecturers through chat rooms, email, discussion forums and webcam teleconferences. However, researchers note that students preferred to physically meet with the instructor when posing questions because of the lack of real-time document sharing. DiCoDEv [13] is a web-based, virtual collaborative platform that can be used during manufacturing product and process design evaluation. The DiCoDEv platform allows multiple users to work in a collaborative and distributed manner, but is limited to a specific domain. CollaBoard [14] uses web technologies to create a video, audio, and data conferencing system that imitates life-sized face-to-face meetings. CollaBoard achieves this through very specific hardware configurations and careful calibration of cameras, which therefore does not provide a flexible and scalable solution in authentic scenarios.

B. Large Display Systems and Parallel Rendering

OmegaLib [15,16] uses Equalizer [17] for parallel rendering and provides the tools to develop immersive 2D-3D applications for flexible systems ranging from tiled display walls to CAVE environments. OmegaLib also offers event handling that supports multiple heterogeneous devices. However, Omegalib does not provide an easy way to manage multiple content windows. The Cross Platform Cluster Graphics Library (CGLX) [18] is an OpenGL graphics framework for distributed, high performance visualization systems. CGLX allows users to easily develop new or adapt existing OpenGL desktop applications for visualization clusters

TABLE I. Comparison of Systems that Enable Collaboration

	Simultaneous Multi-user Interaction	***Windowing Environment for Multiple Applications***	***Extensible 2D and 3D Applications***	***Off-the-shelf Application Support***	***Multi-site Remote Collaboration***	***Supports Content with Unlimited Resolution***	***Leverages Cloud-based Infrastructure***
Google Hangouts	X				X		X
Skype, WebEx					X		X
Binary Meetings					X		X
DiCoDEv					X		X
CollaBoard	X			X	X		X
OmegaLib	X		X			X	
Equalizer			X			X	
CGLX	X		X	X		X	
Liveboard				X	X		
Impromptu	X	X		X			
CubIT	X	X				X	
Mezzanine	X	X			X		
Montage		X				X	X
Display Custer	X	X	X	X		X	
SAGE	X	X	X	X	X	X	
SAGE2	X	X	X	X	X	X	X

such as tiled displays and multi-projector systems. CGLX supports collaboration through multiple multi-touch devices. Input events are synchronized with the display environment and scene information is streamed to the mobile device [19]. CGLX only supports Mac OSX and Linux, thereby creating a barrier to entry for Windows users. Yokoyama and Ishikawa used emerging HTML5 features to utilize web browsers for distributed rendering of web applications [20]. However, their approach only achieved scalable resolutions up to 8240x4920. While allowing vast volumes of information to be rendered in a single high-performance application, none of these frameworks support the multi-windowing environments that are necessary for dynamically juxtaposing information from various sources.

C. Large Display Systems for Collaboration

Large display environments have been used to display large amounts of data and utilize the physical space to organize group work. Researchers are able to work independently, as well as perform collaborative data analysis with an entire co-located or distributed group. Liveboard is a large rear projected interactive display for meetings, presentations, and remote collaboration [21]. While limited by resolution and a single lightpen for interaction at the wall, Liveboard still serves as an early design for large display collaborative systems. Impromptu is a more recent interaction framework where users can share applications between multiple display devices ranging from tablets to a large shared display [22]. Since devices in Impromptu pixel stream their content, the window resolution is limited by the resolution of the source device. CubIT is a presentation and public display space running on The Cube, a series of large cluster-driven multi-touch walls [23]. CubIT supports uploading, sharing, and organizing an array of multimedia content, but does not provide an extensible platform for interactive applications. Mezzanine is a commercially available custom-built conference room environment designed for streaming live video and sharing of documents between co-located or remote users. However, Mezzanine is a fixed setup that occupies an entire room and only supports a limited number of connections for both co-located and remote sites. [24]. Recognizing the wealth of content on the web, Montage [25] aims to use large tiled displays to render multiple web pages, using grouping and filtering techniques to make discoveries. Montage is primarily designed as a passive display receiving limited user input in the form of keywords received from mobile devices. DisplayCluster [26] provides a windowing system for multimedia content across SRSDs. Similar to SAGE, DisplayCluster relies purely on pixel streaming from remote rendering sources. While enabling users to juxtapose a variety of content, all systems mentioned in this section impose collaborative constraints, ranging from limiting content resolution to requiring specific hardware configurations.

IV. SAGE2

The prior work discussed above has shown that co-located and remote collaborators greatly benefit from the ability to share and interact with data in real-time, display content in ultra high-resolution, and juxtapose volumes of information to extract insight. However, no system to our knowledge has encapsulated all these features into a single collaborative framework. We believe that other systems do not support these features because the underlying architecture makes it prohibitively difficult to achieve. We investigated a new paradigm, leveraging cloud-based and web browser technologies to drive a SRSD, and enable real-time communication and multi-user interaction. We hypothesized that a novel web-based platform could achieve the performance of stand-alone applications, while better enabling authentic, multi-user, interactive collaborative workflows.

A. Design Rationale

Our goal in designing SAGE2 was to create the next generation system for facilitating data intensive co-located and remote collaboration using a SRSD. Interactive applications with simultaneous multi-user input, enhanced real-time communication, and a lower barrier to entry were three major priorities according to feedback from authentic users from the first generation of SAGE. To address these needs, we aimed to completely redesign and implement SAGE due to the fact that its aging architecture was not well suited to handle emerging technologies. Additionally, SAGE and other platforms for driving large display environments are built as custom standalone applications, which in our experience requires a technical expert with hours of training to install and support, limiting the adoption at new sites. Therefore, we decided to pursue a different approach that leverages cloud-based and web browser technologies for their increasing collaborative power, flexibility, and ubiquity.

The power of web browsers is increasing at an extraordinary rate. Browsers now support native two-dimensional and three-dimensional rendering through HTML5 and WebGL. Hardware acceleration is leveraged for both rendering and CSS transforms. WebSocket communication has been standardized, enabling persistent two-way communication between server and client. WebRTC is currently under development, with many features already integrated into web browsers that allow real-time peer-to-peer communication. Additionally, browsers can capture events from input devices such as a mouse, keyboard, and touchscreen.

Web browsers have become a ubiquitous application found on any visual computing device. They are platform independent and do not require technical expertise to install. The web browser also acts as a portal to the vast amounts of data stored and accessed through the cloud. Numerous APIs allow developers to retrieve static and dynamic data, perform data manipulation, and visualize and store the results. Web-based communication also enables real-time collaboration and data sharing through peer-to-peer connections.

These recent developments to web-based technologies have enabled high performance graphics and networking capabilities that were formerly only achievable with native applications. Therefore, we have designed SAGE2 using the web browser for rendering and user interaction, and the cloud-based infrastructure for data retrieval and real-time communication.

B. Architecture

We decided to explore whether or not the new features of web browsers would allow them to perform as well as standalone applications in regard to high-performance graphics

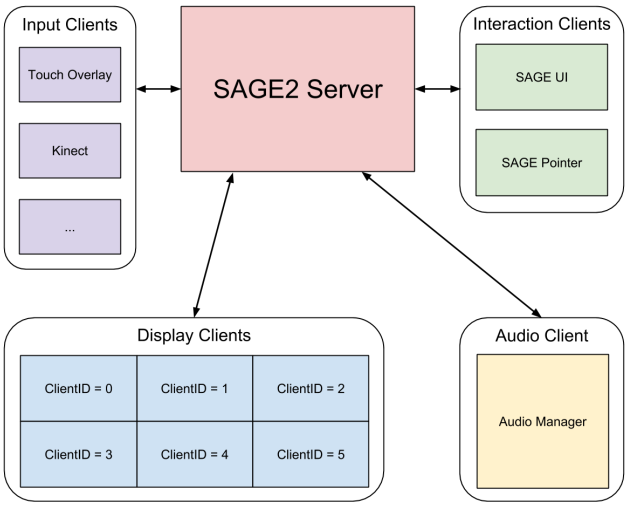

Fig. 2. Illustration depicting SAGE2's architeture and communication scheme. The SAGE2 Server is a customized web server with clients accessed through visiting a URL in a web browser.

and networking. The SAGE2 architecture provides a proof of concept that applications can leverage the web browser runtime environment to achieve the necessary performance to drive SRSDs and support large volumes of data, while also benefiting from the cloud-based infrastructure to facilitate remote collaboration.

SAGE2 consists of several components: the Server, various Display Clients, the Audio Client, various Interaction Clients, and the Input Client. Fig. 2 illustrates how clients connect to the Server, which handles all inputs and outputs to maintain and synchronize content. The Server is built upon Node.js [27], a platform for building network applications in JavaScript. Node.js is cross-platform and comes with a package manager that downloads all dependencies. This greatly reduces the time for installation and no longer requires a technical expert. The only two prerequisites for running SAGE2 are that Node.js is installed on the machine that hosts the Server, and up-to-date web browsers are installed on any machine running a client.

C. Design Challenges

To explain how we achieved this novel approach, we describe the process of overcomming numerous technical design challenges that were presented throughout the development process.

1) Challenges of supporting cluster environments

One of the priorities for designing SAGE2 was hardware flexibility, so that it could be configured for scalable display resolutions. A SRSD can be run by one or multiple machines each connected to one or multiple monitors. Any of the SAGE2 components can run on the same machine or distributed across a cluster. Fig. 3 shows a range of single machine and cluster-based SRSD configurations.

Supporting a cluster environment presented multiple challenges. First, we needed to make all monitors appear as one seamless graphical environment, regardless of the number or configuration of Display Client machines. The SAGE2 Display Clients are instances of a web browser that connect to the SAGE2 Server by visiting a URL. To address this challenge we attach a parameter to the URL in order to identify each Display Client with a unique ID, mapping it to a specific row and column on the SRSD. Given its row and column, each Display Client shows its own viewport by offsetting content, based on its position on the grid. This results in users viewing content that spans multiple Display Clients that appears continuous and can be moved across the display seamlessly.

Synchronizing audio with multiple video feeds across the SRSD presented another challenge. We determined that a separate Audio Client was necessary in order to mix all the audio sources. Like the SAGE2 Display Clients, the Audio Client runs in a web browser and gets initialized by connecting to the SAGE2 Server by visiting a URL. When a video file is loaded into SAGE2, the Audio Client will output the sound and synchronize the Display Clients' video with its audio. We determined that it was most important that audio be uninterrupted, whereas video could withstand minor modifications in order to synchronize with the audio. Therefore, we designed the Audio Client to control the playback of a video, and the Display Clients synchronize their video feed with the audio.

Another challenge with supporting cluster-based systems was how to properly handle distributed rendering with synchronized animations for interactive applications. To address this, each Display Client renders a portion of the overall application depending on the application's window size and position. Using WebSockets, the SAGE2 Server handles animation synchronization by broadcasting instructions to all Display Clients to redraw. Each Display Client responds to the server when it has finished rendering its frame. Once the Server receives responses from all Display Clients, it

Fig. 3. Range of SRSD configurations. Panel A shows a single 4K monitor connected to a single machine running SAGE2. Pannel B shows eight monitors connected to two GPUs on a single machine sunning SAGE2. Fig. 1 is an exmaple of tiled wall with eighteen displays connected to a six machine cluster, with the SAGE2 Server running on the head node. Lastly, Panel C shows a cylindrical tiled wall with seventy-two displays connected to a thirty-six machine cluster, with the SAGE2 Server running on the head node.

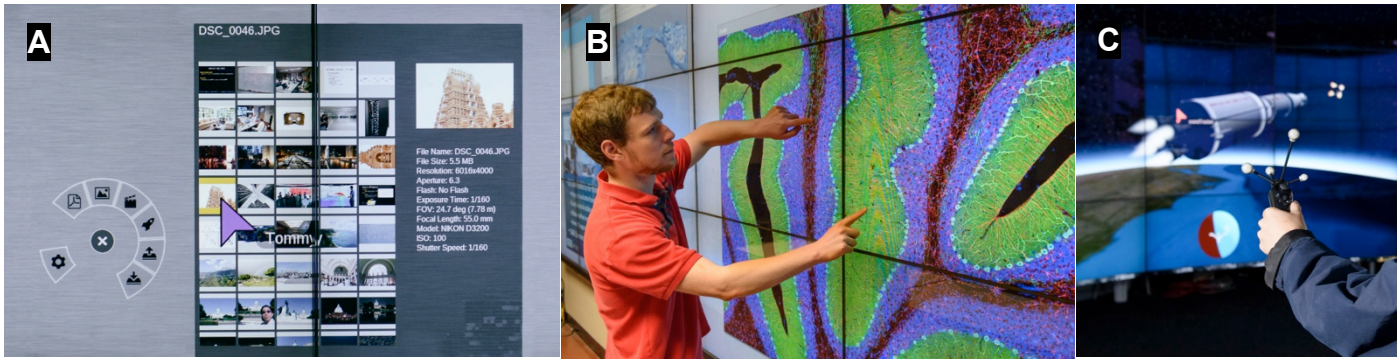

Fig. 4. Mutli-user interaction with expanded input modalities. Panel A shows a radial menu system that allows multiple users to simultanesouly interact with a media brower. Panel B depicts a user interacting at a SRSD using a multi-touch overlay to perform a pinch-zoom gesture. Panel C shows a user interacting with SAGE2 using a video game controller that has been supplemented with motion tracking reflectors.

broadcasts the next redraw command. In this manner, applications remain visually synchronized.

2) Challenges of enabling real-time multi-user interaction

Web pages rendered in a browser serve as both our Display Clients and our Interaction Clients. This allows users to interact with content on the SRSD by opening a browser and connecting to the SAGE2 Server by visiting a URL. This greatly reduces the barrier to entry for new users since there is nothing to install, and they simply use an application that they are already familiar with. By visiting the SAGE UI Interaction Client page, a user can see an overview of the SRSD and can load any of the supported data types or applications. The user can also easily and quickly share local documents, show their screen, or add a pointer to the SRSD.

Since the Interactions Clients are running on personal devices, there was a challenge in correlating interaction events with desired actions on the SRSD. Additionally, web browsers are not designed to handle simultaneous inputs from multiple distinct users. To address this, we capture events from the Interaction Client devices, such as a laptop or smartphone, and forward them to the SAGE2 Server. The Server then broadcasts these events to each Display Client. This enables remote devices to be used for interaction and removes the constraint of one pointer per machine.

Another challenge we addressed was expanding the types of user interaction enabled by SAGE2. We wanted to support application interaction and window management in four major interaction zones for SRSDs: directly at the display using touch gestures, standing near the display using motion tracking or 6DOF devices, seated near the screen using a gyro-mouse or laptop pointer, or indirect control further away from the screen using the SAGE UI [28]. To accomplish this, we utilize the Omicron input abstraction utility library [29]. Omicron is capable of receiving data from different types of input devices including touch overlays, motion tracking systems, game controllers, and speech recognition tools (Fig. 4). Omicron then streams data from these heterogeneous devices in a uniform manner that the SAGE2 Server can interpret.

3) Challenges for supporting multi-user applications

Interacting with ultra high-resolution displays from a distance makes preceision pointing difficult. In standard desktop environments, resizing a window is commonly done by click-and-drag on a corner and moving a window by click-and-drag on a title bar. In order to remove the preceision necessary to accomplish these tasks, we have enabled two interaction modes, window manipulation and application interaction, that a user can toggle between. When in window manipulation mode, events anywhere on an application will be performed on the window (click-and-drag to move, scroll to resize). When in application interaction mode, all events are forwarded to the interactive application to handle.

Another challenge was to support loading custom web applications dynamically. Typically, web applications are only loaded once per browser tab and utilize the browser's default event handling system. In order to create multiple instances of a web application across the SRSD, we have encapsulated all web applications in a JavaScript class. Therefore, an application can be dynamically instantiated as many times as requested. We also forward SAGE2 events to these applications, so that they can respond to multi-user inputs, such as independent keyboards, pointing devices, and touches.

A final challenge with supporting applications in SAGE2, was sharing them across sites. In order to share applications with remote sites and engage in collaborative interaction, we have designed all applications as objects accessible through a URL. This allows a reference to the application to be retrieved from any remote location, whether it is another SAGE2 site or a remote single-user on his/her laptop.

D. Overcoming the Collaborative Roadblocks

Designing the Interaction Clients as applications accessed through a web browser addressed the issues presented in the *lightning collaboration* scenarios. New users are no longer presented with a barrier when wanting to share and interact with content on the SRSD. Creating custom event handlers for managing simultaneous independent inputs addressed both the *parallel investigation workflow* and *single-driver, multiple-navigators* roadblocks. Multiple users can now simultaneously interact with shared content allowing investigators to manipulate content originally shared by a collaborator.

V. Application Development

Many research and industrial collaborative projects take advantage of web-based content. Online data portals and visualization tools have increased tremendously in recent

years, generating demand for direct interaction with web-based content in a group setting. Web development has a large community supporting many APIs, and browsers are becoming as powerful as stand-alone applications at rendering high-quality two-dimensional and three-dimensional content. SAGE2 builds off of these developments and to our knowledge is the first to provide support for both custom and off-the-shelf application integration into a multi-user setting on SRSDs.

A. Open Development

The SAGE2 framework has been designed to enable rapid application integration and development for a community interested in supporting multi-user collaborative environments. SAGE2 provides a rich platform for developing custom applications. We provide a JavaScript API to create native SAGE2 applications. Developers can also make standalone applications SAGE2 compatible by adding a few lines of code to their source. For off-the-shelf applications where users don't have access to the source code, SAGE2 can stream pixels and forward events to a remote machine running the application.

The SAGE2 JavaScript API can be used for developing applications from scratch or porting existing web content. The API allows developers to create applications with scalable resolution. SAGE2 uses the <canvas> element to access native rendering routines, which gives access to two-dimensional and three-dimensional contexts. External libraries, such as kinetic.js, D3.js, or three.js, can be utilized for higher-level graphics abstraction. The API also provides an event handler that supports simultaneous multi-user interaction from a variety of input devices. SAGE2 handles distributing and synchronizing the rendering of an application, hiding the cluster-based compatability from the developer.

SAGE2 also supports pixel streaming and event forwarding for external applications. This provides the ability to utilize SAGE2 in order to view and interact with standalone applications at ultra high-resolution and handle inputs from multiple simultaneous users. Inserting a few lines of code allows a standalone application to communicate with the SAGE2 server. The application then streams its frame buffer to SAGE2 and receives events from the SAGE2 server. The scalable nature of SAGE2 allows these standalone applications to stream output of any size.

Off-the-shelf applications can also run in SAGE2. A remote machine must run a lightweight SAGE2 messaging client in addition to the application. The SAGE2 messaging client scrapes pixels from the remote machine's desktop and translates SAGE2 events into real events for a mouse and keyboard. Even though off-the-shelf applications are inherently single user, multiple users can still interact with these applications through SAGE2. All the events coming from different users will be perceived as being generated from a single user. Though SAGE2 allows applications to render at any resolution, such applications are constrained by the resolution of the display connected to the remote machine. Custom or off-the-shelf applications can stream pixels to remote SAGE2 sites through *remote gigabit data streams*, which helps enable distance collaboration.

B. Prototype applications

We have developed several standard and custom prototype applications for SAGE2. By default SAGE2 provides image viewer, PDF viewer, and video player applications. We have also developed a custom notepad application using the SAGE2 JavaScript API that allows multiple users to simultaneously and independently position cursors and add text to a document. This application demonstrates the usefulness of SAGE2, not only for organizing content in collaborative sessions, but also for collaborative application interaction. This prototype application utilizes the multi-user event handling system of SAGE2, which serves as a template for developing more complex, collaborative applications. Additionally, we have integrated the Google Maps API into a SAGE2 application. This specifically addresses one of the desired features requested in the user survey from section II.B. The Google Maps application provides ultra high-resolution map imagery and allows any users to change location and zoom level, or add additional layers of information, such as weather and traffic.

We have also developed a web browser application that allows users to launch a browser within the SAGE2 display environment and interact with HTML rich content. This is an external C++ application built using the Chromium Embedded Framework (CEF3) [30], which is a library that can render browser content into an off-screen buffer. This buffer is sent to SAGE2 using the pixel-streaming paradigm, while the application receives user interaction from SAGE2. This was necessary to develop as an external standalone application since many web pages prohibit themselves from being embedded inside another web page.

We have also installed the SAGE2 messaging client on a remote machine running ParaView in order to allow multiple users to view and interact with this off-the-shelf scientific visualization tool. SAGE2 events, mouse clicks and keyboard presses, from various users are all mapped down to the remote machine's mouse and keyboard. This allows multiple users to control an application that is otherwise single-user.

VI. USE CASES

Data intensive problems require the combined efforts of many individuals from disparate fields. Since these problems are complex, technologies that aspire to support collaborative infrastructures must be flexible enough to aid in solving individual problems of varying domains. SAGE2 is still under development as a prototype, currently used regularly at a few alpha phase installations, and therefore has not been evaluated with a formal user study. In the rest of this section, we outline three use cases based on real-world scenarios that highlight how SAGE2 can uniquely handle the following collaboration sessions: 1) imitate co-located collaboration with a physically distributed team, 2) collaboration between multiple teams working on different aspects of a related problem, and 3) allowing a single remote user to join a collaborative session.

A. Imitate Co-located Collaboration

It is often the case that a team is distributed at two or more physical locations, but wishes to work as though they were co-located. In this scenario, everyone is working on the same problem and looking at the same data. SAGE2 helps bridge the

gap of physical distance by mirroring shared content at all sites, including live video and audio streams. To illustrate how SAGE2 is used in such a situation, we will describe its use during the judging of a research based photo competition conducted at a university. The panel of judges is distributed across two or three campuses and wishes to rank all submissions during a single joint session.

A SAGE2 session is started at one site, and the SRSD is mirrored at all other sites by simply visiting the same URL. All images along with their accompanying research description are uploaded to the SAGE2 Server and displayed on each SRSD. Additionally, the multi-user notepad application is launched, allowing any judge to jot down thoughts or comments at any time. During the first phase of judging, each image and its accompanying description are enlarged to full resolution and viewed one at a time. After a brief discussion, the image is placed in one of three groups – top contender, possible contender, not a contender. Once all images have been discussed, the second phase of judging begins. This stage, depicted in Fig. 5, is more freeform with all judges working simultaneously. Spatial orientation of the images is used to compare and rank the images. A better image is moved up higher on the display – if an image is only a few pixels higher than another it is considered just barely of higher quality, whereas if it is positioned many pixels higher it is considered significantly superior. Any judge at any site can rearrange the images, add comments to the notepad, or communicate with the other judges. This process continues until a consensus is reached and the winners are determined.

SAGE2 provides a number of unique features that enable this type of collaboration. Since SAGE2 provides a windowing environment on an ultra high-resolution SRSD, the judges are able to view multiple high-resolution images simultaneously. The multi-user paradigm allows multiple judges to reorder the windows or add comments to the notepad simultaneously, reducing completion time for the task. The video conference allows judges at different locations to see and talk to each other as if they were present in the same room. This is incredibly important due to the fact that images are submitted from a variety of domains, and each judge has a unique expertise. Since the SRSD is mirrored at all locations, when a judge at one location interacts with the content, all other sites immediately reflect the modification. This gives collaborators at all sites a shared context from which to communicate.

B. Remote Collaborators Working on a Distributed Problem

Another common scenario is when distributed teams work with the same or similar data and want to share related data and discoveries. In this case, each team may have a unique configuration for their SRSD, and not all documents and applications need to be present at each location. To illustrate how SAGE2 is used in this situation, we will describe its use for disaster management planning in the city of Chicago.

Disaster management planning requires multiple groups to work together. In this example, we will illustrate the collaboration between Chicago city officials and researchers at Argonne National Lab. The city officials are responsible for crisis management and direct the disaster management

Fig. 5. Judges at two separate campuses ranking research images for a competition. The content is mirrored at both locations and changes to one site are reflected in the other in real-time.

planning. They are familiar with the city, its infrastructure, and its population. The researchers utilize their supercomputing facilities to simulate and visualize various disasters. Each team uses a wide array of content to analyze their data including interactive maps, PDFs, images, charts, and graphs.

Members of each team simultaneously use a multitude of applications to answer relevant questions. Panel A in Fig. 6 shows city officials reviewing statistics, such as most densely populated area vs. day of week and number of public trains running vs. time of day. Panel B in Fig. 6 shows the researchers running various disaster simulations, such as a tornado or a train derailment, and creating advanced visualizations. Both teams plot data on a map to see the spatial distribution and overall impact of these disasters. Once the city officials have determined a new scenario or one of the researchers have made an interesting discovery, they share their finding with the other team. Each team has independent control over all documents and applications on their SRSD. This allows important information to be shared between sites, while keeping less relevant data private.

SAGE2 allows these teams to view and interact with various types of data simultaneously, such as interactive maps, PDFs, and images. More than one person at a given site can

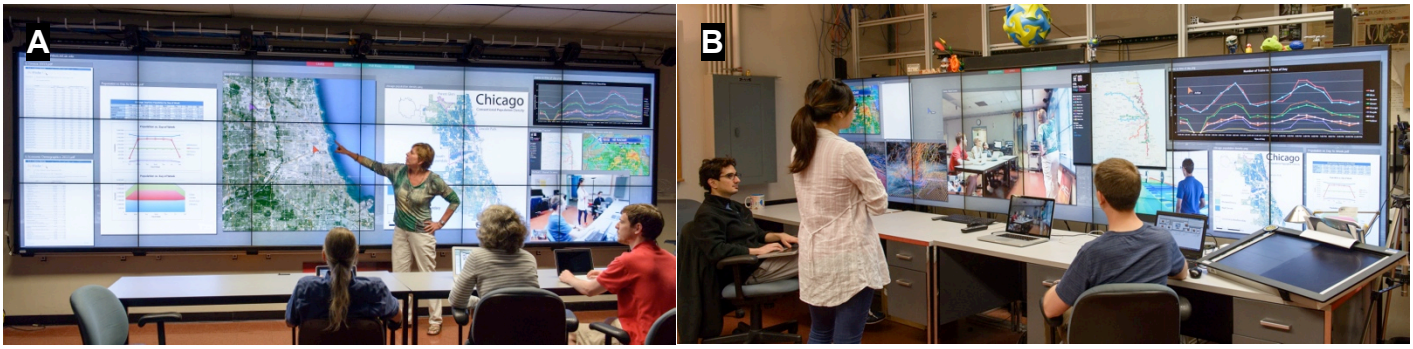

Fig. 6. This figure illustrates Chicago city officials and researchers at Argonne National Lab collaborating by sharing relevant information while maintaining separate focuses. Content is controlled independently, but can be shared across remote sites.

interact with any application simultaneously, which increases productivity and removes the need to switch who is in control. Each location can arrange the documents and applications independently in order to create a layout that is most effective for their team. Also, not all documents and applications are shared between sites, which helps reduce visual clutter, and ensures that only relevant content is available at each site.

C. *Remote Single User Joining a Co-located Session*

SAGE2 can also be leveraged when a single member of a team is traveling or working from home. In this case, everyone is working on the same problem and looking at the same data. The major difference between this scenario and the first is that the single remote member does not have access to a SRSD, but rather only has his/her laptop or tablet. To illustrate how SAGE2 works in this situation, we will describe a team collectively writing a co-authored publication.

A team of five is outlining a paper and reviewing previously published works. One of the members is attending a three-day conference, so the team schedules its meetings during lunch breaks. The other four members start a SAGE2 session and share numerous PDFs, a multi-user notepad, and the lead author's laptop screen. The remote member joins the session from her laptop, which gives an overview of the content and layout on the SRSD. She can get details of any specific shared application on demand by viewing it in a separate browser tab. Fig. 7 depicts the team reading PDFs and taking notes on the related work, while the lead author integrates relevant content into a draft of the paper on his laptop. When the meeting concludes, the session can be saved, so the team can resume later without having to re-upload and reposition all relevant documents and applications.

SAGE2 allows a remote user to engage in a remote collaboration session without the need for special hardware or software. A browser allows the remote user to view and interact with shared content via an overview of the whole SRSD and detailed applications viewable one at a time. The co-located team and the remote user both receive immediate input from each other. SAGE2 allows both the co-located team and the remote individual to take advantage of the technology at their disposal, rather than forcing a group to fall back to the lowest common denominator. SAGE2 also allows the team to continue their work at a later time by saving the session.

VII. Conclusion

Collaboration has always been an essential dimension of work in research, academia, and industry. In the next decade, collaboration will be even more essential, as multidisciplinary teams tackle complex big data problems. While SRSDs have been shown to be powerful tools for collaborative work, specialized software is required to effectively leverage these environments. We have built a next generation SRSD collaborative platform, SAGE2, that integrates cloud-based and web browser technologies into an environment suited for data intensive problem solving in authentic scenarios. SAGE2 taps into the original SAGE user community and has the potential to expand this community, because of its enhanced support for development and integration of multi-user applications, and its lower barrier to entry. We invite the community to use SAGE2 as a platform to develop multi-user applications for devices ranging from a standard desktop to a cluster-driven tiled display wall. With the ability to provide support for various input devices, its seamless extension towards multi-user scenarios, and its ability to leverage the web infrastructure, we have illustrated SAGE2 as a powerful tool for group work

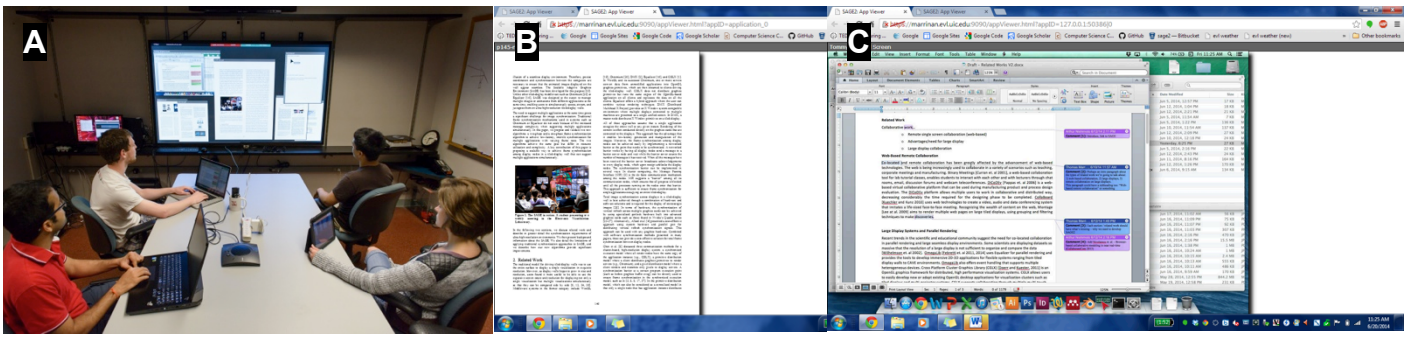

Fig. 7. Panel A shows a team of authors collaboratively writing a paper. A remote single user is able to participate by viewing applications, such as PDF viewers and a shared laptop screen in separate web browser tabs (Panels B and C).

through a series of use cases. More information about SAGE2 can be found at its website, http://sage2.sagecommons.org.

VIII. Discussion / Future Work

Research on SAGE2 is ongoing and currently stands as a functioning prototype. We anticipate an open-source beta release by November 2014. In addition to improving stability of the platform, two of our major future goals are to research linking content and further improving remote collaboration. We plan to explore the possibilities of creating links between applications in order to exchange data, such as the ability of a mapping application to plot the locations of all photos that are geotagged on the SRSD. We will also be exploring various techniques to improve remote collaboration where the technologies at each site are heterogeneous. We plan to synchronize applications across multiple sites and provide a window into a remote site's SRSD to give a visual overview of their content arrangement.

Acknowledgements

This publication is based on work supported in part by the National Science Foundation (NSF), awards ACI-1339772, OCI- 0943559 and CNS-0959053. Any opinions, findings and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agency. SAGE and SAGE2 are trademarks of the University of Illinois Board of Trustees.

References

[1] C. Johnson, R. Ross, S. Ahern, J. Ahrens, W. Bethel, K. L. Ma, M. Papka, J.V. Rosendale, H. W. Shen, and J. Thomas. 2007. Visualization and knowledge discovery: Report from the DOE/ASCR workshop on visual analysis and data exploration at extreme scale. Salt Lake City.

[2] J. Cummings, T. Finholt, I. Foster, C. Kesselman, and K.A. Lawrence. 2008. Beyond being there: A blueprint for advancing the design, development, and evaluation of virtual organizations.

[3] R. Ball, and C. North. 2005. Effects of tiled high-resolution display on basic visualization and navigation tasks. In CHI '05 Extended Abstracts on Human Factors in Computing Systems. ACM, New York, NY, USA, 1196-1199.

[4] M. Czerwinski, G. Smith, T. Regan, B. Meyers, G. Robertson, and G. Starkweather. 2003. Toward characterizing the productivity benefits of very large displays. In Proceedings of Interact, vol. 3, pp. 9-16.

[5] D. S. Tan, D. Gergle, P. Scupelli, and R. Pausch. 2003. With similar visual angles, larger displays improve spatial performance. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, New York, NY, USA, 217-224.

[6] L. Renambot, A. Rao, R. Singh, B. Jeong, N. Krishnaprasad, V. Vishwanath, V. Chandrasekhar, N. Schwarz, A. Spale, and C. Zhang. 2004. Sage: the scalable adaptive graphics environment. In Proceedings of WACE, vol. 9, pp. 2004–09.

[7] B. Jeong, R. Jagodic, L. Renambot, R. Singh, A. Johnson, and J. Leigh. 2005. Scalable Graphics Architecture for High-Resolution Displays. In Proceedings of IEEE Information Visualization Workshop 2005. Minneapolis, MN, October 2005.

[8] B. Jeong, L. Renambot, R. Jagodic, R. Singh, J. Aguilera, A. Johnson, and J. Leigh. 2006. High-performance dynamic graphics streaming for scalable adaptive graphics environment. In Proceedings of the 2006 ACM/IEEE conference on Supercomputing. ACM, New York, NY, USA, Article 108.

[9] Google+ Hangout. June 20, 2014 from http://www.google.com/+/learnmore/hangouts/.

[10] Skype. Retrieved June 20, 2014 from http://www.skype.com/en/what-is-skype/.

[11] WebEx. Retrieved June 20, 2014 from http://www.webex.com/.

[12] K. Curran. 2002. A web-based collaboration teaching environment. In IEEE Multimedia. 9(3), 72-76.

[13] M. Pappas, V. Karabatsou, D. Mavrikios, and G. Chryssolouris. 2006. Development of a web-based collaboration platform for manufacturing product and process design evaluation using virtual reality techniques. In International Journal of Computer Integrated Manufacturing. 19(8), 805-814.

[14] M. Kuechler, and A. M. Kunz. 2010. Collaboard: a remote collaboration groupware device featuring an embodiment-enriched shared workspace. In Proceedings of the 16th ACM international conference on Supporting group work. ACM, New York, NY, USA, 211-214.

[15] A. Febrett, A. Nishimoto, T. Thigpen, J. Talandis, L. Long, J. D. Pirtle, T. Peterka, A. Verlo, M. Brown, D. Plepys, D. Sandin, L. Renambot, A. Johnson, and J. Leigh. 2013. CAVE2: a hybrid reality environment for immersive simulation and information analysis. In Proceedings of The Engineering Reality of Virtual Reality 2013.

[16] A. Febretti, A. Nishimoto, V. Mateevitsi, L. Renambot, A. Johnson, and J. Leigh. 2014. Omegalib: A multi-view application framework for hybrid reality display environments. In IEEE Virtual Reality, pp. 9–14.

[17] S. Eilemann, M. Makhinya, and R. Pajarola. 2009. Equalizer: A scalable parallel rendering framework. In IEEE Transactions on Visualization and Computer Graphics, vol. 15, no. 3, pp. 436–452.

[18] K. U. Doerr, and F. Kuester. 2011. CGLX: a scalable, high-performance visualization framework for networked display environments. In IEEE Transactions on Visualization and Computer Graphics. 17(3), 320-332.

[19] K. Ponto, K. Doerr, T. Wypych, J. Kooker, and F. Kuester. 2011. CGLXTouch: A multi-user multi-touch approach for ultra-high-resolution collaborative workspaces. In Future Generation Computer Systems. 27(6), 649-656.

[20] S. Yokoyama and H. Ishikawa. 2011. Parallel distributed rendering of html5 canvas elements," in Proceedings of the 11th international conference on Web engineering, pp. 331–345.

[21] S. Elrod, R. Bruce, R. Gold, D. Goldberg, F. Halasz, W. Janssen, D. Lee, K. McCall, E. Pedersen, K. Pier, and others. 1992. Liveboard: a large interactive display supporting group meetings, presentations, and remote collaboration. In Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 599–607.

[22] J. T. Biehl, W. T. Baker, B. P. Bailey, D. S. Tan, K. M. Inkpen, and M. Czerwinski. 2008. Impromptu: a new interaction framework for supporting collaboration in multiple display environments and its field evaluation for co-located software development. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 939–948.

[23] M. Rittenbruch. 2013. CubIT: large-scale multi-user presentation and collaboration. In Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces. ACM, New York, NY, USA, 441-444.

[24] Oblong Industries, Inc. Introducing Mezzanine: The Future of Conference Room Collaboration. White paper. 2013. http://www.oblong.com/mezzanine/overview/.

[25] D. Lee, S. A. Munson, B. Congleton, M. W. Newman, M. S. Ackerman, E. C. Hofer, and T. A. Finholt. 2009. Montage: a platform for physically navigating multiple pages of web content. In CHI '09 Extended Abstracts on Human Factors in Computing Systems. ACM, New York, NY, USA, 4477-4482.

[26] G. P. Johnson, G. D. Abram, B. Westing, P. Navr'til, and K. Gaither. 2012. Displaycluster: An interactive visualization environment for tiled displays. In IEEE International Conference on Cluster Computing, pp. 239-247.

[27] Node.js. Retreived June 20, 2014 from http://nodejs.org/.

[28] J. Leigh, A. Johnson, L. Renambot, T. Peterka, B. Jeong, D. Sandin, J. Talandis, R. Jagodic, S. Nam, H. Hur, and Y. Sun. 2013. Scalable resolution display walls. In Proceedings of the IEEE. 101(1), 115-129.

[29] Omicron. Retrieved February 13, 2014 from http://github.com/uic-evl/omicron/.

[30] Chromium Embedded Framework. Retreived June 20, 2014 from https://code.google.com/p/chromiumembedded/.