The Tele-Immersive Data Explorer: A Distributed Architecture for Collaborative Interactive Visualization of Large Data-sets

Nikita Sawant, Chris Scharver,
Jason Leigh, Andrew Johnson
{nikita, scharver , spiff, aej}@evl.uic.edu
Electronic Visualization Laboratory
University of Illinois at Chicago

Georg Reinhart, Emory Creel, Suma Batchu,
Stuart Bailey, Robert Grossman
grossman@uic.edu
National Center for Data Mining
University of Illinois at Chicago

ABSTRACT

There exist a number of scientific visualization systems designed to provide a two-dimensional interface to the user. However, little consideration has been given to the development of collaborative virtual environments for visualization purposes. This paper discusses the Tele-Immersive Data Explorer a generalizable framework to facilitate the construction of domain-specific data exploration applications challenged with the problem of having to visualize massive data-sets immersively and collaboratively. In the paper we describe the framework's conceptual organization, its distributed multiprocessed object-oriented architecture, and its application to visualize gridded scalar data.

KEYWORDS

Tele-immersion, collaborative virtual reality, data mining, multivariate data, annotations, persistent environments, design patterns.

1 INTRODUCTION

Research scientists and engineers need to analyze massive data sets generated by intensive computational simulations, geographic information systems etc. Scientific visualization plays an important role in this task; it exploits the low-level human visual system to find trends, clusters and anomalies in these data sets rapidly. There exist a number of general-purpose scientific visualization systems which aide the user in performing a variety of tasks such as data exploration and visualization: namely IRIS Explorer[5], Khoros[6], IBM Data Explorer[8], and AVS [9]

With the existence of high-end visualization systems like the CAVE and the Immersadesk, it is now possible to allow users to perceive three-dimensional structures of the data being analyzed. These Virtual Reality (VR) devices provide stereoscopic head-tracked displays that give users the illusion of being immersed in a three dimensional world and allow the user to interact spatially in three dimensions with objects in the environment. Research scientists at Iowa State University have experimented with visualization of statistical graphics on virtual display devices and on traditional two-dimensional monitor and the benefits of using the former over the latter in visualizing multivariate data[4]. The aforementioned visualization systems were not designed to take advantage of the rich interface provided by VR devices. Efforts have been made to add VR interfaces to AVS [10], however these extensions only display the final visualization on a VR device thus limiting the users interaction with the data.

Immersive Scientific Visualization systems like Cosmic Worm[11], Virtual Wind Tunnel[12], CAVEvis [13], and CAVEstudy[14] have been developed to take advantage of the rich interface the CAVE provides. However these systems are not collaborative. They do not allow a user to share his/her view with a remote user. If a group of research scientists wish to collaborate during a data visualization session they would have to share one interface to interact with the data; this can prove to be quite cumbersome and inconvenient especially if the people involved are geographically at different co-ordinates. Wood et al. [3] have emphasized the need for a Collaborative Visualization System and have provided extensions to make IRIS Explorer[5] collaborative, but this is not enough.

Another problem that the scientific visualization community faces is the disability to visualize massive data sets ranging from a few megabytes to several terabytes. Scientific visualization systems that have been designed for data that can fit into the physical memory (in core) cannot be used for the visualization of large data sets.

Very little work has been done to develop visualization systems that allows a user to collectively:

- Visualize data interactively and immersively
- Collaborate with remote users during the data exploration and visualization process
- Explore large data-sets

Developing such a system can prove to be a daunting task. The visualization process involves the following:

computational tasks that operate on the data and rendering tasks that generate geometry to be visualized. Both these tasks can be time consuming and computationally intensive, especially if the data being explored is complex and huge, this invariably leads to very low frame rates. Immersive systems such as the CAVE require that the rendering rate be no less than 10 stereo frames per second, to give the user the sense of being immersed in the environment. Any frame rate below this is not acceptable. Tele-Immersion (TI) is defined as the integration of audio and video conferencing, via image based modeling, with collaborative virtual reality (CVR) in the context of data mining and significant computation. A tele-immersive visualization system for handling large data needs to provide the minimum frame rate while collaborating with other users in parallel.

In this paper we describe The Tele-immersive Data Explorer(TIDE): a framework for building such tele-immersive applications for visualization of massive data-sets. The TIDE framework allows groups of scientists each at a geographically disparate location to collectively participate in a data analysis session, in a virtual environment. The data being analyzed can be stored on data servers, which are at a different location from the clients'[2].

This paper will describe the TIDE framework and how it provides for collaborative immersive visualization of large data. Section 2 discusses the conceptual organization of TIDE. In section 3 we describe the TIDE architecture and section 4 gives a detailed insight into the TIDE framework. An application of the framework is described in section 5. In the subsequent sections we talk about lessons we learnt from TIDE and a detailed comparison is made with other scientific visualization systems. Some of which are mentioned above.

2 The TIDE Concept

Imagine a scenario in which there are three scientists who wish to confer on the effects of ocean currents on the earth's climate. However all the three scientists are located at geographically disparate locations and so is the data. One of the scientists is an oceanographer, all the oceanographic data is situated in the oceanographer's lab. Another scientist is a climatic expert and has the climate data in his laboratory and the third is an environmentalist. The climate data is defined by attributes such as temperature, precipitation and pressure. Whereas the oceanographic data has attributes such as salinity, direction and strength of the ocean current and sea surface temperature. The scientists decide to participate in a collaborative data exploration and analysis session. Each scientist can access both data sets.

The scientists visualize and interact with the data using a rear projection based stereoscopic device such as the CAVE or an Immersadesk. A virtual environment is created wherein each scientist can see the representation of the remote scientist in his own environment. This representation is called an avatar and is nothing but a three-dimensional model that may constitute of various parts of the human body. The position and orientation of each user in the virtual environment is tracked and his/her remote representation is updated accordingly. So if the oceanographer points to a certain location in the data-set, the environmentalist and the climatologist will see his avatar pointing to that location in their own worlds. Digital audio is set up between the sites to allow the participants to speak to each other. The scientists can work synchronously or asynchronously. Consider a synchronous session where all the participants in the session share the visual representation of the data. The scientists can interact with the data using three-dimensional tools.

The oceanographer decides to play the lead role and loads the atmospheric data and chooses to visualize the temperature over the earth's surface. The climatologist notices an anomaly in the surface temperature over Europe. However the oceanographer knows the reason behind this are the strong currents that flow from the hot tropical regions towards the poles and loads the ocean current information from the database. Now the scientists can see the ocean currents that are displayed as vectors, the magnitude and direction of which depend on the strength and direction of the current. The oceanographer correlates this with the temperature information and gives a detailed explanation of the anomaly in the temperature of the European region. During his discourse, he can rotate or scale the data set to focus on certain aspects of the visualization. He/she can also navigate to a particular point in the virtual space. For example say if the oceanographer wants to talk by keeping both the climatologist and the environmentalist in view then he/she can navigate to a suitable position where both the collaborators and the data are within his/her filed of view Now that the environmentalist has gained enough knowledge of the effects of the ocean currents, he/she decides to work asynchronously and leaves the session, the other collaborators see his avatar exit the virtual space. After this point, no information is shared between the environmentalist and the rest of the participants. He then explores some other region specific data and finds similar effects of the ocean currents on the climate. The environmentalist and the

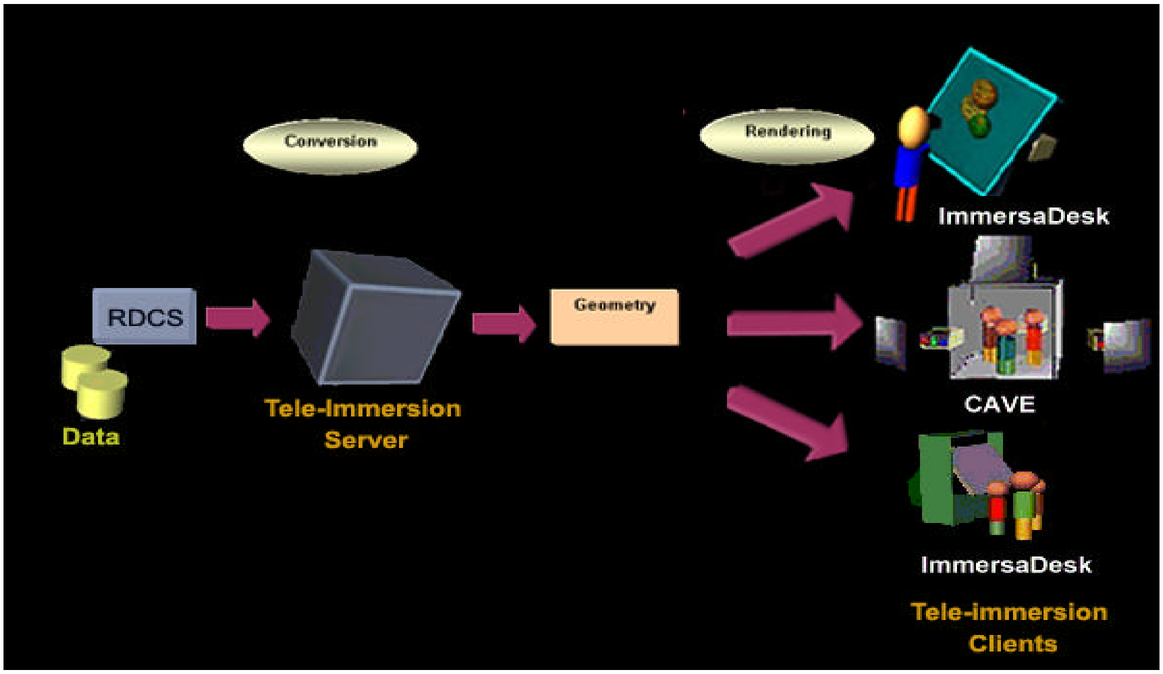

Figure 1. TIDE's Visualization Process

climatologist, could access the oceanographers expertise even though they were not physically in the same space. Neither the climatologist nor the environmentalist needed to have the oceanographic data being visualized to be stored locally, as visualization is generated from remote data and loaded into the environment.

Scientific visualization involves the conversion of raw data into a geometric representation (polygons, cubes etc.) that can be rendered onto a display device as shown in **Figure 1.** The conversion process is responsible for fetching the data from the data source, representing it in the computer's physical/virtual memory using a domain specific data model, performing operations such as feature extraction and decimation (if required) and finally generating a geometrical representation of the model. The rendering process is responsible for drawing the geometrical representation on a display device. Both these tasks when performed by a single application can prove to be computationally intensive and would generally use up most of the resources such as memory, disk space and CPU cycles, especially for large complex data. This results in very low frame rates, which is unacceptable by immersive systems such as the CAVE.

TIDE separates the conversion process from the rendering process by allocating the tasks to different computing environments. The conversion of the data and the rendering of the visualization are done by separate applications.

As shown in **Figure 1** we have split the visualization process by implementing a client/server architecture. The server application does the conversion and generates the geometry that is sent to the client, which renders it on a display device to be visualized by the user. In this way the conversion of the data into a three dimensional visualization is done on a high end supercomputer (the tele-immersion server), with the resources to carry out intensive computations, and the geometrical model is rendered on immersive display systems such as the CAVE (tele-immersion client). The server application also handles collaborative visualization sessions, wherein you have a group of clients collectively analyzing/visualizing the data.

Splitting the visualization process this way allows the server process to focus on the data conversion and manipulation and the client process on rendering and user interaction.

Thus the conceptual organization of TIDE consists of three primary components: the Remote data and computation services (RDCS); the Tele-Immersion Server (TIS); and one or more Tele-Immersion Client (TIC). This is shown in the **Figure 1.** There can be one or more tele-immersion clients, which connect to a central server called the Tele-immersion server. The Tele-immersion clients are the visualization end points that allow the user to participate in the collaborative virtual environment. The Tele-immersion server is in turn connected to one or more remote data and computation services, and mediates the interaction between the client and these services.

2.1 Remote Data and Computation Services

RDCS refer to external databases (data mining servers) and/or simulations/compute-intensive tasks running on supercomputers or compute clusters. The databases hold data generated by computational simulations and digital instrumentation systems. Since the size of the data may vary from a few megabytes to several terabytes, the data may be distributed over several such nodes. It is impossible to visualize all the data , as it will not fit in core memory, and only a subset on the data can be visualized at a time. Hence the data may be processed in such a way that, from the entire data set, a smaller data set is extracted, which is a coarser version of the original large data set, which can be rendered by the TIC at a desirable frame rate. Therefore a trade-off is made between the resolution of the data set and better interactivity. The coarse version can be obtained by averaging i.e. by replacing a set of values by their average and/or by decimating. In addition to this several attributes can be mapped to distinct visually perceptible cues such as stereoscopic depth, hue, size and opacity.

2.2 Tele-Immersion Server

The TIS mediates interaction between the TIC and the RDCS and also serves as a persistent entry point for the clients so that they may initiate long-running data-intensive queries and come back at a later time to view their progress. The TIS handles multiple clients and synchronizes their interaction with the RDCS. The TIS allows each TIC connected to it to operate on it own local sub-set of the data. It is the TIS that actually retrieves the raw data from the RDCS and converts it into a three dimensional geometrical representation that can then be visualized by the TIC.

2.3 Tele-Immersion Client

The Tele-Immersion client (TIC) consists of the virtual reality display device (such as the CAVE, Immersadesk, PowerWall etc.) and the interface to allow collaborative retrieval and visualization of data. The TIC handles the rendering of the three dimensional visualization received from the TIS, on the VR display device. It allows the user to interact with the data and to specify any input parameters required for generating the next visualization step. The TIC also provides the user with the tools to directly manipulate the visual representation of the data. In addition to this the TIC also manages the rendering of avatars, allowing participants to collaborate effectively with each other.

3 The TIDE Architecture

The distributed client/server architecture of TIDE is as diagrammed in **Figure 2**.

3.1 The Tele-Immersion Client

The process of data visualization is one that involves several steps to reach the end rendered result. A user must first query a data archive for specific information. Filtering operations provide more specific details about the particular aspects of the data to visualize. These operations may include partitioning data, specifying correlations between different types of data, or other filtering methods. All of these specifications are consolidated to produce a rendered visual image. It is this rendered image that visually represents the data. A user may, depending upon the visualization application, perform additional operations based on the visualization itself. These include translation and rotation transformations as well as data-related functions like zooming in on specific parts of the visualization.

TIDE has been designed to maintain a conceptual separation between these two modes of interaction: **data querying** pertains to communicating with data archives to specify and obtain information about the data; and **visualization interaction** is any interaction performed in the context of the tele-immersive environment. These latter actions by the user could also communicate with the data servers. Information about actions in the tele-immersive environment must be communicated to other users to indicate the user is performing an action. The query interface addresses communications relating to the data itself, while the visualization interface addresses interactions related to the shared environment and the visualization geometry.

The visualization environment utilized by TIDE is a shared virtual environment inhabited by multiple users. When collaborating in a virtual environment, the user should be able to interact with the remote participants. This is done by streaming audio between the participants to allow them to talk to each other and by providing a representation of the remote user, an *avatar*, that reflects the users position in space. Depending on the tracking abilities of the VR system, the users' position and orientation can be tracked by electromagnetic sensors. Any changes to the client's position and orientation also need to be broadcast to everyone in the environment. This information needs to be updated regularly at high frequencies. An environment server leverages the flow of this information between various clients. A separate environment server is dedicated to this task, as long delays degrade the effectiveness of the collaborative experience.

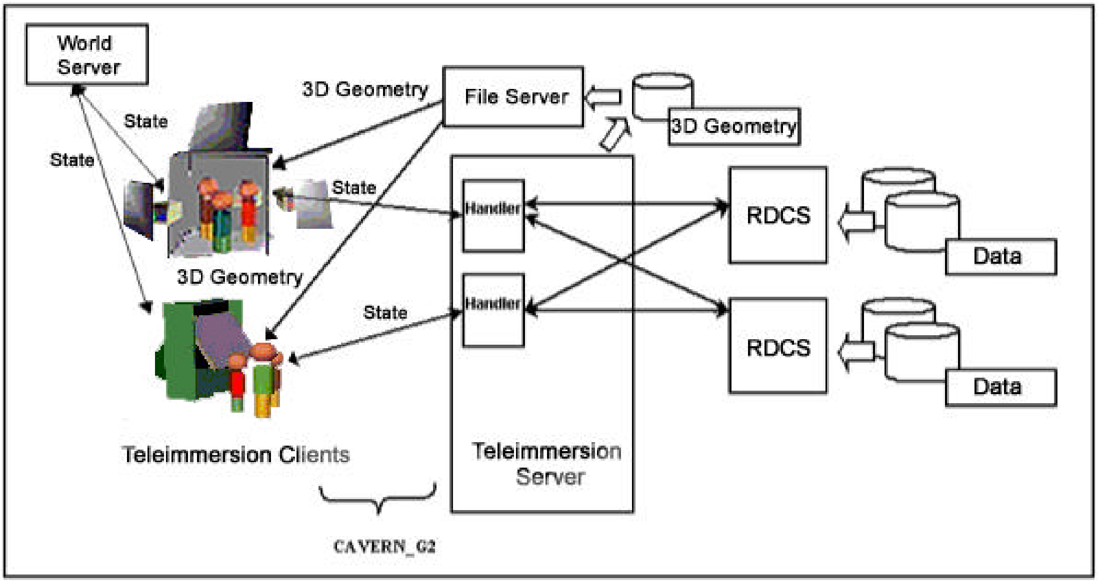

Figure 2. Distributed Architecture of TIDE

After a query has been completed the visualization geometry is sent to each of the connected clients, who load it into their environment for viewing. The user can scale and rotate the visualization to get the correct view of the data, or to bring a particular region of interest into focus. There is a time interval between the moment that the user submits a query and the visual image is rendered on the screen. This delay depends on the amount of information queried, the available bandwidth of the network, the time taken to convert the data into a visual model, file download time and time taken for rendering the image. This time interval varies from a few milliseconds to a couple of minutes (or may be even hours and days), depending on the size of the queried data. For improving the performance of the system, the client starts the process of loading the new visualization in a new thread, which runs in the background. The old visualization is removed only after the new one is ready. This way the user does not have to stay idle for long intervals. Whenever any client changes the state of the visualization, this change is propagated to all the other clients participating in the collaboration.

Although the visualization has been reduced in complexity from the fidelity of the original data, the geometry can still be quite intricate. Complex geometry can hinder the performance of the interaction due to increased time between frame updates. To reduce the polygon count of the geometry during manipulation, level of detail features have been introduced. This switch to a significantly lower number of polygons makes the manipulation updates more fluid in response to the user's actions. As soon as a manipulation begins, the geometry view switches to a lower-resolution bounding box. This switch takes place for all users in the environment. It also serves to provide an indication that a user has initiated a manipulation operation on the geometry.

In order to provide additional status information about the current state of operations, the client uses audio cues. These cues indicate when a visualization download has begun. Additionally, a 3D watch icon indicates that the client is performing processing--in this case, background loading the geometry into the scene graph.

3.2 The Tele-Immersion Server

The TIS abstracts the TIC from the actual data, the TIC need not be aware of how and where the data is stored. It only needs to specify to the server what data it is interested in and how it wants to visualize the data i.e. as a three-dimensional plot, extract an isosurface from the data or correlate several attributes by generating a histogram etc.

In this way the TIS server acts as a mediator decoupling the TIC from the source of data, ideally allowing any client to visualize any data. The TIC can then concentrate on making the rendering process more efficient by focusing on tasks that improve the interactivity with the visualization and increase the frame rate.

The TIS is multiprocessed i.e. for every new client a separate server handler process is created. When a client connects it can specify what view of the data set it wants to visualize, and the corresponding server process retrieves that data from the data server. Every

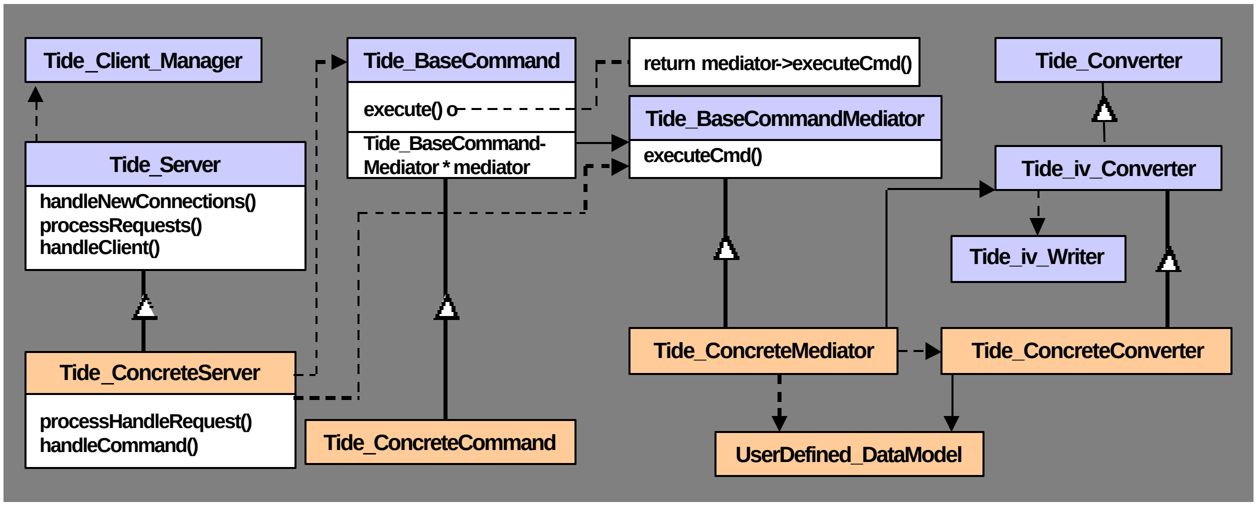

Figure 3. UML Class diagram for the TIS

handler process stores locally the subset of the data being visualized by the client. The client can specify operations on its own subset of the data set like feature extraction, decimation, mapping attributes of the data set to various visual dimensions etc. The handler process performs these operations on the data, converts the data into a three dimensional representation and notifies all the clients that new data is ready. The client then downloads the visualization from the server.

The clients share only the geometric representation in a collaborative session, not the data. The novelty of the TIDE architecture lies here, where it decouples the data from its visualization, thus allowing them to be treated as separate entities. If a new client connects to an already existing session then the TIS automatically sends commands to the new client enabling it to visualize the current data being explored. To visualize another data set a client needs to submit a fresh query, the handler process again retrieves the necessary data from the data source.

Since every handler process has the data for its client, the client can perform multiple operations on this data, without having to query the data servers repeatedly. Huge amounts of data transfers lead to an increased delay in responding to the client. If the dataset the client is interested in fits into the physical memory of the TIS, then the data can be retrieved once and different views can be generated off the same data, this improves the response time as the TIS does not have to execute expensive data retrieval operations always.

For example consider a scenario where there are two clients (A and B) visualizing an atmospheric data set, which is characterized by attributes such as wind velocity, precipitation, temperature and vegetation. Client A is analyzing the effect of precipitation on the vegetation and Client B is correlating the temperature to the wind velocity. Client B discovers an anomalous trend in the correlation and needs Client A's opinion to support his finding, hence he asks client A to participate in the analysis session. Client B then shares his visualization with Client A. After discussing with client B, A can still go back to his own visualization by asking the TIS handler process to generate a visualization from his copy of the data, without having to re-query for the data.

A shared memory arena is used for communication between the various handler processes and the parent process. Communication between the TIC and TIS is established using CAVERNsoft G2, a C++ toolkit for building collaborative networked applications [1].

3. 2.1 Co-Servers of the TIS

To efficiently handle the communication between various participants in the collaborative session, separate dedicated servers are allocated the tasks of handling exchange of shared information and file operations, namely the world server and the file server.

3.2.1.1 The World Server

Information about the immersive environment is communicated among multiple users through the TIDE world server. This server allows for collaboration between researchers examining a particular dataset. Each user has hand and head tracking data available from the CAVE library. This information is shared among users to provide the location information for each avatar within the environment. Whenever a manipulation is initiated from within the visualization, the other users need to be aware of that operation as well.

The TIDE world server is comprised of three primary components. First, a UDP reflector transmits UDP packets from one client to another. Second, when information being transmitted is essential, a TCP reflector transmits reliable TCP packets. Finally, a database component manages state information such as the transformations of the visualization geometry. An identifier string identifies each of these manipulations, this allows for multiple manipulations within the environment.

The CAVE library handles the tracking of the user's position and orientation. This information is transmitted during each frame to the TIDE world server. The high frequency of the transmission does not require a very reliable service. Therefore, a UDP connection is made between each client and the server. During each frame update, the client packages the position and orientation of the user's hand and head, and sends them via a UDP connection to the server. The server reflects this information to each of the other connected users in the environment. Each client in turn updates the avatar for the particular user.

Client commands involving direct manipulation of the interface are signaled with reliable TCP transmissions, as any loss of this information could lead to misunderstandings in the communication between participants. For example, when a user begins to rotate an object, the client sends a rotation initialization command to the server. When this command is reflected to the other remote clients, they each switch their views to the bounding box representation of the geometry. After manipulation is completed the client sends a rotation complete command. Each remote client receives the reflected command from the server and switches the visualization back to the full-resolution geometry. In the time duration between the beginning and the end of a rotation the client sends state information that reflects the amount/degree of rotation to the remote clients, which use this information to rotate their view.

The TIDE world server's primary function is to serve as a central connection point for each of the clients. The server reflects commands and operations to each of the connected clients to enforce the shared states of the environment.

3.2.1.2 The File Server

A dedicated file server allows multiple clients to down load the geometry files generated by the TIS. This improves the performance of the TIS as it does not have to deal with uploads to all the clients, additionally the architecture can be further modified to allow the clients to download the files only if they wish to.

The file server is based on CAVERNSoft's remote file I/O networking class.

4 The TIDE Framework

TIDE provides an application developer with the infrastructure to build collaborative teleimmersive applications to explore large multi-dimensional data sets. TIDE has a distributed command driven architecture as shown in **Figure 2.** A message-passing interface is used for exchanging information between the TIS and the TIC over the network. A typical application of TIDE would consist of a number of TICs on various VR display devices requesting services from the central TIS. The TIDE framework defines how the client handles collaboration, user interaction, rendering of the visualization and querying the TIS for information. It specifies how the TIS handles: multiple requests from multiple clients, data representation and conversion. Communication with the TIC is established using a reliable TCP channel.

The implementation of the framework is done in C++ and effectively uses design patterns**[20]** such as Template method, Command, Observer and Mediator.

4.1 The Tele-Immersion Server

It is not possible to define every operation that the TIS needs to carry out, as some tasks are application specific. However, some key tasks remain common to all applications and these are: handling multiple requests from multiple clients, data representation and conversion. "What" the client requests are and "how" they are to be handled may be application specific. Similarly "what" the data is and "how" the geometric modeling is done may depend on the application domain. The TIS provides a network of interrelated abstract classes, the task of providing the application specific functionality is deferred to specialized sub-classes, which need to be implemented by the application developer.

Figure 3 shows the interrelationships between various classes using the UML notation. The classes belonging to the framework are blue in color. The Tide_Client_Manager maintains a client database for the Tide_Server. Every client request is packaged into a command object and a handler is implemented to execute one or more commands. For example if a client wishes to visualize a subset of the data then the TIS gets a message from the client that specifies all the parameters required for data segmentation. The TIS implements a segmentation command to wrap the client request into an object and then identifies a handler which is aware of the data source and knows how to get the subset of the data the client is interested in. So every time a client asks for a data subset the segmentation command-handler pair is sufficient to handle the request. Command-Handler pairs can be created based on the abstract Tide_BaseCommand and Tide_BaseCommandMediator classes.

The framework also has an abstract converter class that specifies the algorithm for converting the data into a geometric model and sending the reply back to the client. Hooks are provided wherever application specific code needs to be added, this code is provided by concrete converters. The handlers that do geometric modeling use these converters. For example: if the client wishes to extract an isosurface based on a threshold value it sends a message to the TIS, the TIS sets the isosurface_command and isosurface_handler objects to execute the request, The isosurface_handler uses an isosurface_converter to generate the isosurface, the geometry information for which is then sent back to the requesting client. Currently the framework has a Writer object that generates the geometry as an Open Inventor file. Writers for other formats can easily replace this one.

4.1.1 Communication with the Clients

For every client that connects, the TIS forks a new handler process, which is responsible for carrying out any requests made by a client. However the type of requests made by a client, are application specific, hence all the tasks performed by the handler process cannot be defined a priori. A template method, handleClient(), is used to define the function associated with the handler process. This method defines the steps to : read incoming requests from the TIC, handle those commands (a hook method handleCommand() is provided for this purpose) and send the reply to the main parent TIS process to be sent back to the client(s). A hook method processHandleRequest() is provided to allow concrete server classes to interpret the reply , and package it and send it to one or more clients.

The concrete sub-classes can define the behavior that can vary by implementing these handleCommand() and the processHandleRequest() functions, which are called when a new client request arrives and before sending data back to the client (i.e. unmarshal/marshal the client request/reply). The Tide_Server class therefore provides the basic framework to handle multiple client connections and to send commands, state information and data to these clients. The TIC uses a reliable TCP connection to send commands to the server using a message-passing interface.

4.1.2 Handling Client Requests

The task of interpreting the client requests and performing the operations needed to carry them out is deferred to application specific concrete subclasses of Tide_BaseCommand and Tide_BaseCommand-Mediator, these classes are based on the Command pattern[20]. The concrete subclasses understand the client requests and are aware of the operations to be performed to fulfill those requests. The TIDE framework provides a generic way in which requests can be coupled to handler objects that execute the request.

The Tide_BaseCommand provides the interface to wrap the client request into an object and stores the receiver of the request as an instance variable. Once this is done a client of this class invokes the command by calling the execute() function, which internally calls the executeCmd() function of the handler (instances of Tide_BaseCommand-Mediator). Concrete subclasses of Tide_BaseCommandMediator override the executeCmd() operation and perform application specific tasks to handle the requests.

In this way an application can handle any user request by implementing

- A concrete subclass of Tide_BaseCommand that stores the state of the user request and implements execute() (e.g. Tide_ConcreteCommand)
- A concrete subclass of Tide_BaseCommand-Mediator that implements executeCmd() and has the knowledge to carry out the request (e.g. Tide_ConcreteCommand-Mediator)

The Tide_ConcreteServer class creates both the command and the corresponding mediator object and links them together. Thus it is easy for an application to add new commands, as it does not have to modify existing classes.

4.1.3 Data Representation and Conversion

The Tide_Converter class is an abstract class that defines the interface for data conversion. The Tide_Converter class provides hooks for concrete sub-classes to override for application specific code. Concrete instances of this class will have a reference to the user defined data model, the result is a geometric representation of the data, which is currently represented in Open Inventor format and is

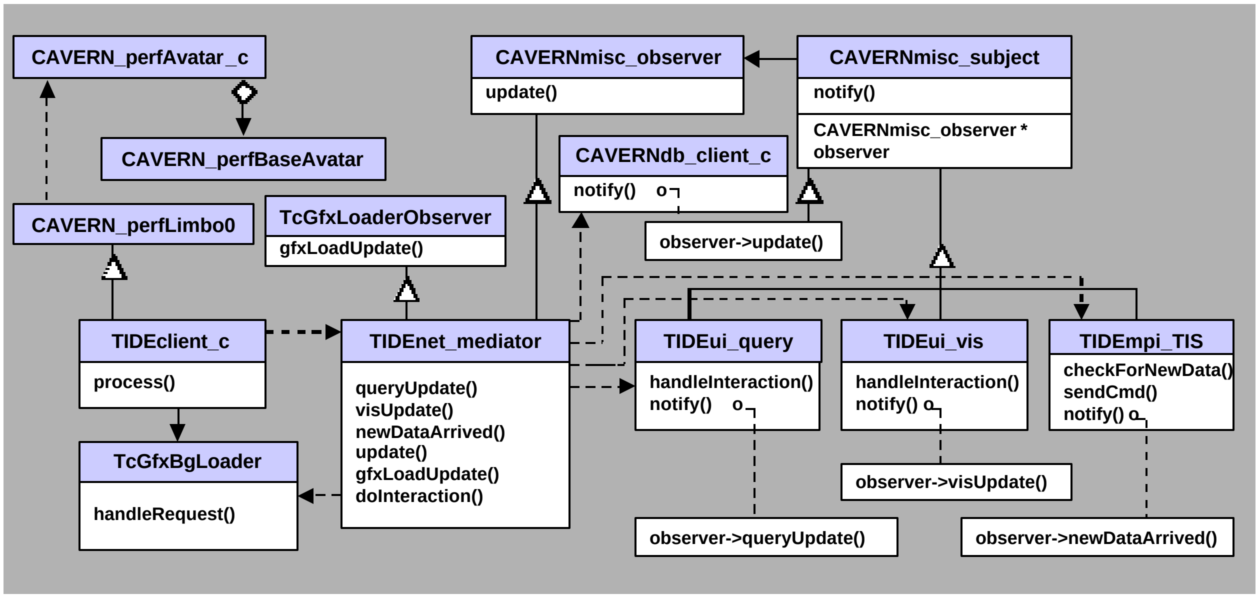

Figure 4. UML Class diagram for the TIC

written to a file. This file is then downloaded by the TIC for visualization. Handler objects use a converter to generate a specific type of model, for example if the TIC needs streamlines to be generated from a data set, then the handler would use a streamlineConverter defined for the purpose and a streaklineConverter if streaklines were required.

4.2 Tele-Immersion Client

The TIC provides an interface to the data, hence its prime responsibility is to handle user interaction and creation of the shared virtual environment for the participants. Currently two types of interfaces are provided for user interaction: a query interface to interact with the TIS and a 3D interface to directly interact with the visual model. The virtual environment is created based on Limbo, which provides the basis for creating tele-immersive applications. A UML class diagram showing the TIC framework is shown in **Figure 4**.

4.2.1 Limbo

CAVERN_perfLimbo0 is a part of the CAVERNsoft G2 toolkit. CAVERN_perfLimbo0 provides the basic shell to build tele-immersive applications. Several other modules included within the toolkit are used for basic functionality within the TIDE client. An avatar module, namely CAVERN_perfAvatar_c, manages connections between multiple users in a virtual environment. The information pertaining to each remote user is wrapped into a CAVERN_perfBaseAvatar object. The avatar module manages this collection and makes sure that information of the local user is transmitted to remote users and vice versa. This module creates two network connections with the world server (which is the central collaboration server for the avatars): a TCP connection is used as a hailing channel to signal the entry and exit of users to and from the environment; and a UDP connection is used as a tracking channel to continuously transmit information about the local users' tracking of body positions. The CAVERN_perfLimbo0 class provides the functionality to load a set of default scene objects into the environment, TIDE uses these objects as elements for providing feedback about the visualization environment. These can be further refined as needed in the future.

The TIDEclient_c class inherits from the CAVERN classes, the basic functionalities to manage avatars and extends this capability to support user interaction and collaborative visualization. This class implements a process() function which is executed once every time after the scene is rendered. It is here that most of the user related events are handled. The TIDE_client_c class also references a background loader, which is a part of the TANDEM [21] VR toolkit. The TcGfxBgLoader is responsible for loading the visualization in a background process. The TIDE_client_c class also references an abstract TIDEnet_mediator class that handles the communication with the TIS and the world server for visualization data and collaboration respectively. So in its implementation of the process() function, a TIDE_client_c object calls the handleRequest() function of the background loader to handle new load requests since the last call to it and the doInteraction()function of the TIDEnet_mediator to handle communication of user requests.

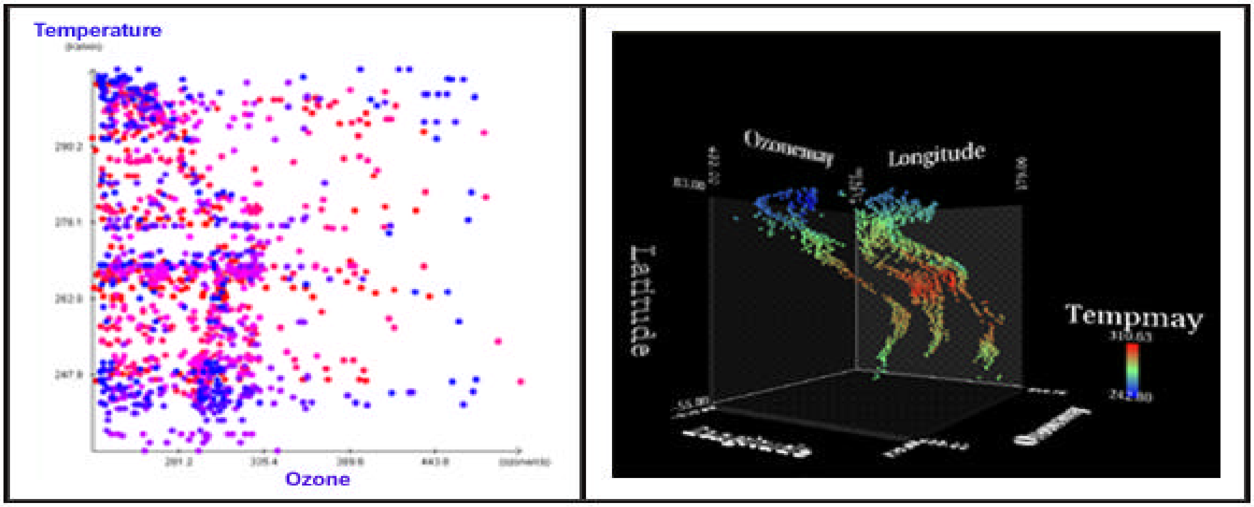

Figure 5(a): A 2D plot of Temperature (Y-axis) vs. Ozone (X-axis), and Latitude (color) on a DSTP client

Figure 5(b): A 3D plot of Latitude (Y-axis), Longitude (X-axis), Ozone (Z-axis) and Temperature (color) on a TIC

4.2.2 User Interface Protocols

Limbo only provides the capability to navigate through a static virtual environment. TIDE has the additional requirements of the ability to query a remote data computation service as well as manipulating the resulting visualization geometry. The framework breaks these interfaces into two classes, TIDEui_query and TIDEui_vis respectively as shown in **Figure 4**. Both these classes inherit from the abstract CAVERNmisc_subject class, and hence are subjects, which are observed by the CAVERNmisc_observer class (see Observer Pattern[20]). These classes implement a basic notify() function, whenever this function is called all the observers interested in this event are updated. Concrete implementations of these classes need to provide application specific details to build a user interface for querying and visualization and to notify observers of all events. These abstract classes also specify a handleInteraction() function that is called to check for any user interaction.

4.2.2.1 Query Interface

The role of the query class is to communicate with the TIDE server. Its implementation will provide a user interface for the user to make queries with the data server. The abstract class leaves the application specific details to be implemented by the application programmer. The query can be done through interaction within the virtual environment, communicating with a handheld device, or using a 2D/3D widget interface. The query interface may or may not have any coupling with the virtual environment itself, and implementing it as a separate class allows for a wider range of extensibility. A concrete TIDEui_query class needs to provide application specific notify() and handleInteraction() functions.

4.2.2.2 Visualization Interface

Interaction within the visualization geometry is contained within a concrete implementation of the TIDEui_vis class. Like the query interface class, the abstract base class provides no concrete functionality. It serves only to provide a base class for references with other parts of the framework. The visualization interface can extract information about the user's actions within the virtual environment in order to apply changes to the visualization. These extractions could involve obtaining information such as CAVE Wand button presses or any other input device such as the data glove.

After a query process has been completed, it is typically the visualization interface, which adds the geometry to the virtual environment. Any interactions or manipulations of that geometry are the responsibility of the concrete visualization interface, though some basic manipulations such as rotation and scaling are provided. A concrete TIDEui_vis class needs to provide application specific notify() and handleInteraction() functions.

4.2.3 Network Mediator

The Network mediator (TIDEnet_mediator) handles network communication for the client. It inherits from the CAVERNmisc_observer class and hence is based on the Observer software design pattern[20], the mediator observes instances of each of the TIDE user interface clients. When an operation is performed on any of the interfaces, a notification is sent to the mediator. The responsibility of the mediator is to then transmit appropriate data to the server and other connected users (through the world server), if required.

The network mediator implements a doInteraction() function which is called once between two consecutive frame updates. It is this function that concrete subclasses need to override to handle application specific functionalities. For example: in a simple case this function would call the handleInteraction() functions of the query interface object and the visualization objects to see if any user interaction has occurred, any user interaction would in turn trigger other events i.e. as observers are notified of events. The network mediator is also an observer of CAVERNdb_client_c, which allows a seamless infrastructure for collaboration of visualization states. Whenever a remote collaborator modifies the state of his/her visualization, the update() function of the network mediator is called, as it is a registered observer for this event. Similarly if the local user modifies (i.e. rotates or scales) the visualization, this information is sent to other remote users. The TIDEmpi_TIS provides a TCP communication channel to the TIS. The network mediator implements the newDataArrived() function to be updated when new data arrives from the TIS, this is called when the notify function of the TIDEmpi_TIS object is called.. The network mediator checks if any new data has arrived by calling the checkForNewData() function. Concrete classes of TIDEmpi_TIS need to specialize the notify() and checkForNewData() member functions.

The TIDE client class has a single network mediator to broker all the network communications for the querying and visualization services of the client application. The mediator contains references to concrete instances of the query and visualization interfaces. When the mediator receives network information, it propagates the data to the appropriate interface for updating. For example if an application programmer needed to capture a particular user input say the threshold value for an isosurface extraction module on the TIS the query interface would provide a widget to get the user value when its handleInteraction() function is called by the doInteraction() function of the base network mediator. A concrete CAVERN-net_mediator would implement a specialized queryUpdate() function that will be invoked when a user enters a threshold value. This function would in turn send the information to the TIS through the message-passing interface (i.e. by using the sendCmd() function). The sendCmd() function send a packet of data to the TIS. The TIS would handle this request and the newDataArrived() function is called when the client receive a reply from the server. The applications concrete network mediator will then intercept the reply by implementing the newDataArrived() function to do the needful.

5 A TIDE Application

Data mining servers store large multidimensional data sets such as results of scientific simulations and digital outputs of Geographical Information Systems. These data sets contain information, which is as yet undiscovered and could be potentially useful to the analyst. The TIDE framework can be applied to visualize these “hidden gems” of information.

The Laboratory of Advanced Computing at the University of Illinois at Chicago focuses on providing an infrastructure for mining and exploring remote distributed data[17]. They use a protocol for retrieving data from remote nodes on the Internet namely the Data Space Transfer Protocol (DSTP). Servers on some of the nodes called DSTP Servers perform the task of data retrieval. The data on the nodes is organized in tables, with each row in the table corresponding to a data point and the columns specify an attribute of the data point. Attributes of two data points can be correlated if they have more then one attribute in common; this common attribute is the Universal Correlation Key (UCK).

In the implementation of TIDE that interacts with the DSTP servers, the TIC can specify the attributes of the DSTP data that its interested in visualizing. The TIS then retrieves the data from the DSTP Servers. Once the TIS gets the data the client can specify how the attributes are to be correlated and visualized. One can correlate various attributes of the data set by plotting them against one another on a three dimensional graph or by generating a three dimensional histogram.

Since virtual reality devices such as the CAVE provide one more dimension for visualization, the client can select an attribute for each of the three axes (X, Y and Z) as well as the visual representation of each attribute of the data set. When generating visualization from a multivariate data set, the multiple dimensions need to be mapped to different visually perceptible attributes, so that the user can rapidly and accurately identify trends/anomalies in the data sets. In the current TIDE architecture a user can map three attributes to the X, Y and Z-axes and can specify the color, transparency and the size of the points in the 3D graph to be controlled by three other attributes. In this way six dimensions of the data can be visualized at a time.

DSTP Servers have been populated by gridded weather data provided by the National Oceanic and

Atmospheric Administration (NOAA). The data consists of monthly satellite measurements of global surface temperatures, precipitation, ozone levels and vegetation index. Complete data sets are available for every month of the years 1985 - 95. The UCKs for this application are latitude and longitude, gridded in one-degree intervals. Each data file consists of about 64000 rows (roughly 360 times 180) and the columns of the file are: latitude, longitude followed by a particular attribute. The combined size of all files is approximately 350 MB.

Consider a case where a user wishes to see the temperature and the ozone levels over the surface of the earth. The user can get the temperature values from one DSTP server and the ozone levels from another. The **Figure 5(a)** shows the graph generated by a DSTP client to be rendered on a two-dimensional screen, and **Figure 5(b)** shows one generated by TIDE for rendering on a VR display. In the DSTP graph, a maximum of three attributes can be visualized at a time. As opposed to this a TIC can visualize up to 6 attributes at a time by mapping them to visibly perceptible cues such as opacity, hue and size in addition to the normal X, Y and Z-axis. The temperature is mapped to color and the Ozone to the Z-axis. The longitude and latitude are mapped to the X-axis and the Y-axis respectively. The easily discernible features are the warm equatorial belt, the cold temperatures at the poles and the depletion in the ozone level at the South Pole.

Since the TIS allows for connections from multiple clients to visualize the data, we can have two clients who are analyzing different attributes of a multi dimensional data set. Say Client 1 (from Chicago) is visualizing the temperature over the earth's surface (**Figure 6(a)**), from within a CAVE and Client 2 (from Portland) is visualizing the ozone concentration, on an ImmersaDesk. Client 2 then notices an anomaly in the ozone concentration at the North Pole and needs to discuss this with Client 1. He then calls up his colleague and asks her to participate in the visualization session, they now wish to correlate the two attributes i.e. temperature and the ozone concentration. Client 2 then maps the Z-axis to the ozone concentration and the color of the points in space to the temperature, they also plot a 3D Histogram to find any clusters in the data set. Both of them confer over the reasons for this anomaly (**Figure 6(b)**), after they are done with their discussion Client 1 goes back to her own visualization. In this way multiple clients from any geographic location can connect to the TIS and query for data stored in several DSTP servers, which are located at different geographic co-ordinates.

A demonstration of TIDE/DSTP was given at Supercomputing '99, in Portland, Oregon. TIDE users running on ImmersaDesks at various exhibit booths (notably the National Center for Data Mining, Argonne National Laboratory and Alliance exhibit booths) and a InTENsity PowerWall at the Advanced Strategic Computing Initiative (ASCI) booth, collaboratively queried and correlated 500 MB of atmospheric data (provided by NOAA) distributed amongst several servers situated in Chicago. A snapshot of TIDE at SC99 is shown in **Figure 7**. Two TICs (in Portland) are visualizing atmospheric data from DSTP servers situated in Chicago. Both the TICs are running on an ImmersaDesk.

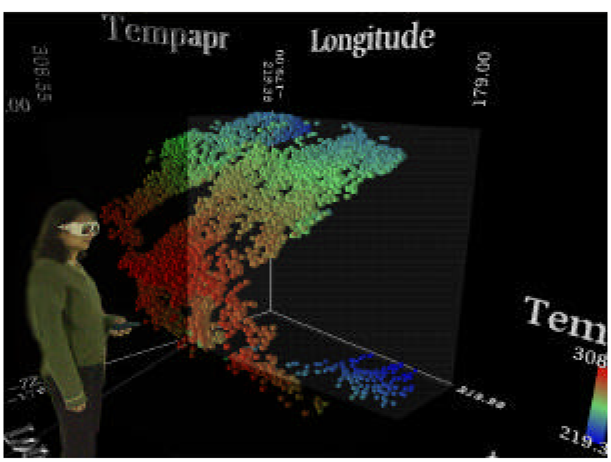

Figure 6(a). A user visualizing Temperature over the Earth's surface in the CAVE

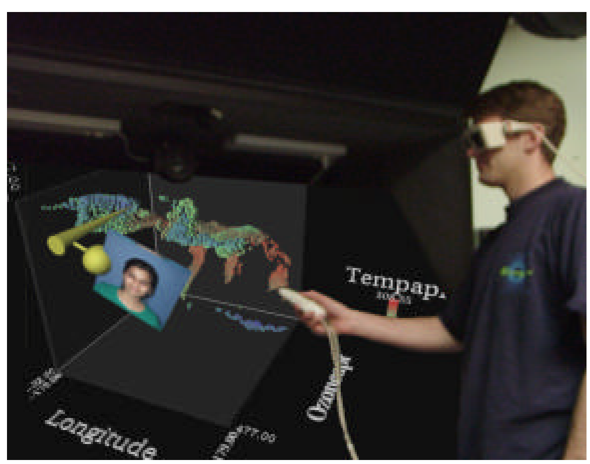

Figure 6(b). A user collaborating from an ImmersaDesk

Although we experimented with an atmospheric data set, TIDE and DSTP can be used to visualize various other multidimensional data. Ongoing research on TIDE is now focused on the visualization of ASCI data.

6 Lessons Learned

TIDE lays the foundation for building tele-immersive data visualization applications. The application programmer does not have to bother about handling issues such as leveraging collaboration amongst the clients. Implementing a distributed multiprocessed Client/Server architecture for TIDE resulted in a number of advantages; some of them are listed below:

6.1 Centralized control of collaboration

The central server mediates all the information exchanged between the clients. All the updates from a client are first sent to the world server which forwards the data to all other participants, similarly the TIS generates the visualization based on a single clients request and forwards it to the rest, this was a central server has control over events. The central server can arbitrate any conflicts and maintain a uniformity to ensure consistency in the shared environment.

6.2 Centralized Location for Data

The TIS handles requests for data from all the clients, storing a local copy for each client. The clients do not have to expend memory and CPU resources performing operations like data retrieval, representation and conversion. The server alleviates the client of performing these tasks, at the expense of increased latency which is introduced due to the delays in the network in fetching the data from the server and talking to the server. This does not affect the rendering frame rate as the client is not blocked in waiting for the data, the user can still interact with the other collaborators, as the server that handles the avatar information is different from the one that is doing the computation.

Since the client only has an interface to the data, it can focus more on improving the rendering task. There is no need to have a monolithic client that handles all the complexities as the tasks are distributed over several processes.

6.3 Multiple Processes

Having a separate process to handle a particular client allows the TIS to pay individual attention to each client. Also since every process runs in its own address space, there is very little possibility that a faulty process will affect the other processes, this increases the robustness of the application.

Since each process has a local copy of the data for the client, it may lead to replication of data and may affect the overall performance of the applications that have a large number of clients connecting to the server. To improve the performance of the processes themselves may be distributed over several processors or a global copy of the data can be maintained in shared memory if all the clients are to visualize the same data.

6.4 Large Data Visualization

The centralized Client/Server architecture also allows the client to specify what subset of the data it is interested in and the server then retrieves only that information for the client. In this way all the data need not be loaded into memory. However for this design to be implemented efficiently the database that hold the data should have a hierarchical structure or should allow for selective retrieval of data based on some input parameters. There is ongoing research on using hierarchical formats for data storage[19].

6.5 Extensibility of the TIDE Framework

The TIDE framework has been designed keeping in mind the requirements of collaborative applications for data visualization. It provides a message-passing interface between the TIC and the TIS. The Command-Handler abstraction allows future applications to support different types of client requests. Hooks have been provided to add application specific code. For example: if an application programmer were to develop an application that allows multiple users to visualize a volumetric data set, the TICs query interface needs to be implemented according to the application requirements, command-handler pairs will have to be implemented for each request from the TIC to the TIS, a suitable data model that captures all the properties of the volumetric data needs to be implemented. Depending on the kind of visualizations to be generated from the data (i.e. isosurfaces, streamlines, isocontours etc.) specialized converters need to be implemented. No code needs to be written for maintaining the shared VR environment or for network communications between the TIS and the TIC. However, the TIC needs to be defined as well as that between the TIS and the RDCS. Implementation of the **networking protocol** between the TIS and RDCS is required.

For the TIDE-DSTP application it was very easy to extend the client queries by just implementing the appropriate command and handler classes. Also it was very easy to plug in the 3D graph converter, which generated a 3D graph based on the user’s selections of mappings for the various dimensions and the Histogram converter, which generated a 3D histogram to correlate three attributes. Only the interface for the

TIC needed programming, as most of the VR framework for collaboration was in place and the client did not have to do any computations pertaining to the data. Since the TIC is a thin client and does not need to have the data stored locally, we did not have to use very high-end machines to run the client at SC99, and it worked well on an SGI O2. The data was stored on DSTP servers in Chicago, the client could

Figure 7. TIDE at SC99: A user running the TIDE Client on a PowerWall, at the ASCI booth. Inset: Another user collaborating from an ImmersaDesk

query for this data from Portland seamlessly.

At SC99 we had two clients participating in a collaborative session for data analysis. Currently the TIS does not arbitrate which client controls the visualization, hence it is possible that both clients can change each other's visualization randomly. This led to situations where a client's visualization would change suddenly when analyzing another data set. This conflict will be resolved in the next version of TIDE.

7. Conclusion

Most Scientific Visualization environments were not designed for tele-immersion, collaborative and VR interface modules were added as extensions later. The goal of TIDE was to provide an infrastructure for developing tele-immersive environments for exploration and visualization of large data sets. It allows a group of scientists to query for subsets of a data set and collaborate on the visualization of the data set. The design aims to keep the data separate from its visual representation by distributing the computation and rendering tasks over several machines. The advantage of this design is that the massive data sets can reside on high-end data servers with the disk capacity to store them; the data can be distributed over several such servers. Thus the visualization end-points can still visualize the data in real time even if they do not have the memory and the disk capacity to hold the data from which the visualizations are generated.

8. Related Work

A number of Visualization Systems exist which have been designed to aide scientists in visualization and exploration of scientific data. Modular data-flow systems such as IRIS Explorer[5], Khoros[6], IBM Data Explorer[8], and AVS[9] provide the user with a collection of modules, the user then uses visual programming techniques to connect these modules together into a visualization network. Data is read in by the initial module in the network, each subsequent module in the pipeline acts on the data transforming it, till the final module which is generally a rendering module renders it on a display device. Most of these systems were designed to run on traditional two-dimensional monitors; efforts have been made to develop rendering modules, which will display the output of theses systems on a VR device. However the user cannot interact with the visualization generated on the VR device. These systems were not designed with collaboration in mind, though efforts have been made to make such systems collaborative[3]. Another draw back of these data-flow systems is that they transform the entire data set, hence they are not well suited for visualization of large data, especially on VR devices, which demand a very high frame rate for interactivity.

Some visualization systems have been designed specifically for high performance visualization on VR devices. The Virtual Wind Tunnel[12], a VR based scientific visualization system developed at the NASA Ames Research Center, is one such system that is designed to study unsteady 3D vector fields. CAVEvis is another distributed system for interactive visualization and exploration of large data sets. CAVEvis was developed at the National Center for Supercomputing Applications to study scalar and vector field data in an immersive environment. This system is similar to TIDE in that, it distributes the rendering and computation tasks over several machines, however it is focused more on the rendering tasks and is not collaborative. In **Table 1** we have compared other such systems; extensibility, collaboration and immersion being the main points of comparison.

Visualization System	Distributed	Collaborative	Immersive	Large Data Viz.	Extensible Framework	Application Domain
AVS	Y	Y	N	Y	Y	Wide
AVS GROTTO	Y	Y	Y	Y	Y	Wide
IRIS Explorer	Y	Y	N	N	Y	Wide
IBM Data Explorer	Y	N	N	N	Y	Wide
Khoros	Y	N	N	Y	Y	Wide
CAVEStudy	Y	N	Y	N	Y	Computational Steering
CAVEvis	Y	N	Y	Y	Y	Time varying scalar and vector fields
Virtual Wind Tunnel	Y	N	Y	N	N	Scalar and Vector CFD Data
Cosmic Worm	Y	N	Y	Y	N	Computational Steering
SGI Mineset	N	N	N	N	Y	Data Mining and Visualization
CAVE6D	N	Y	Y	N	N	Limited
SCIRun	N	N	N	Y	Y	Wide
TIDE	Y	Y	Y	Y	Y	Wide

Table 1. Comparison of Visualization Systems

9. Future Work

Future research on TIDE will focus on supporting:

- **Multiple sessions:** In the current implementation of TIDE all the clients participate in one collaborative visualization session. A collaborative visualization framework ideally should provide for multiple such sessions and allow clients to participate in any such session. This feature can be easily added to TIDE, by modifying the client database to handle groups of clients in addition to singular clients.
- **Persistent TIS:** Collaborative sessions need to be persistent. Provision needs to be made to allow the client to submit an intensive query and come back for the result at a later date/time i.e. the server processes need to be made persistent.
- **Annotating data sets:** TICs can connect from geographically different co-ordinates that fall in different time zones; hence for effective collaboration Clients need to annotate the data set. For example: Consider a TIC connecting from Japan and one in the US but at a different time (Day in Japan => night in America) and the client in Japan notices anomaly in the data set and wishes to convey this to his colleague in America. This can be done in various ways (write about annotations, ongoing research)
- **Visualizing time-dependent data:** When exploring time dependent data sets one needs to visualize a fixed set of attributes of the data set corresponding to different time stamps. For example: A client may want to traverse through their data sets in monthly and yearly time steps. TIDE can provide this as an easy extension to the visualization process.
- **Comparing two visualizations:** In the next version of TIDE we also wish to provide an ability to compare two data sets by juxtaposing them.

10. Acknowledgements

The virtual reality research, collaborations, and outreach programs at the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago are made possible by major funding from the National Science Foundation (NSF), awards EIA-9802090, EIA-9871058, ANI-9980480, ANI-9730202, and ACI-9418068, as well as NSF Partnerships for Advanced Computational Infrastructure (PACI) cooperative agreement ACI-9619019 to the National Computational Science Alliance. EVL also receives major funding from the US Department of Energy (DOE), awards 99ER25388 and 99ER25405, as well as support from the DOE's Accelerated Strategic Computing Initiative (ASCI) Data and Visualization Corridor program. In addition, EVL receives funding from Pacific Interface on behalf of NTT Optical Network Systems Laboratory in Japan.

The CAVE and ImmersaDesk are registered trademarks of the Board of Trustees of the University of Illinois.

ImmersaDesk2, PARIS, Wanda and CAVELib are trademarks of the Board of Trustees of the University of Illinois.

11. References

[1] Leigh, J., Johnson, A., DeFanti, T., "CAVERN: A Distributed Architecture for Supporting Scalable Persistence and Interoperability in Collaborative Virtual Environments". In Virtual Reality: Research, Development and Applications, Vol. 2.2, December 1997 (1996), pp. 217-237

[2] Leigh, J., Johnson, A., DeFanti, T., et al. "A methodology for Supporting Collaborative Exploratory Analysis of Massive Data Sets in Tele-Immersive Environments". 8th IEEE International Symposium on High Performance and Distributed Computing, Redondo Beach, California, Aug 3-6, 1999.

[3] Wood, J., Wright, H., Brodlie, K., "Collaborative Visualization". In Proceedings of the Conference on Visualization 1997.

[4] Arns, L., Cook, D., Cruz-Neira, C., "The Benefits of Statistical Visualization in an Immersive Environment", In Proceedings of IEEE VR'99, pp. 88-95, September 1998.

[5] Foulser, D., "IRIS Explorer: A Framework for Investigation", In Computer Graphics, vol. 29(2), pp 13-16, 1995.

[6] Argiro D., Kubica S., Young M., and Jorgensen, S., "KHOROS: An Integrated Development Environment for Scientific Computing and Visualization", http://www.khoral.com

[7] Young, M., Argiro, D., Kubics, S., "Cantata: Visual Programming Environment for the Khoros System", Computer Graphics, vol. 29(2), pp. 22-24, 1995.

[8] Abram, G., and Ternish, L., " An Extended Data-Flow Architecture for Data analysis and Visualization", Computer Graphics, vol. 29(2), pp 17-21, 1995.

[9] Advanced Visual Systems (AVS) http://www.avs.com/products/index.htm.

[10] Kuo, E., Lanzagorta, M., Rosenberg, R., Julier S., and Summers J., "VR Scientific Visualization in the GROTTO", In Proceedings of the IEEE Virtual Reality, 1998.

[11] Roy, T. M., Cruz-Neira, C., DeFanti, T. A., and Sandin, D.J., " Cosmic Worm in the CAVE: Steering a high Performance Computing Application from A Virtual Environment", Special Issue on Networks and Virtual Environments of Presence: Teleoperators and Virtual Environments, Fall 1994, pp. 121-129.

[12] Bryson, S., and Levit, C., " The Virtual Wind Tunnel: An Environment for the Exploration of Three Dimensional Unsteady Flows", In IEEE Computer Graphics and Applications, July 1992

[13] Jaswal, V., "CAVEvis: Distributed Real-Time Visualization of Time-Varying Scalar and Vector Fields Using the CAVE Virtual Reality Theater", In Proceedings of the conference on Visualization '97, 1997, pp. 301

[14] Renambot, L., Bal, E. H., Germans D., Spoelder, H., "CAVEStudy: an Infrastructure for Computational Steering in Virtual Reality Environments", http://www.cs.vu.nl/~renambot/vr

[15] Leigh, J., Johnson, A., "CALVIN: an Immersimedia Design Environment Utilizing Heterogeneous Perspectives", In proceedings of IEEE International Conference on Multimedia Computing and Systems '96. Hiroshima, Japan, June 17 - 21, 1996, pp. 20-23.

[16] DeFanti, T., Cruz-Neira, C., Sandin, D., "Surround Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE," In Conference Proceedings, ACM SIGGRAPH'93, Anaheim, CA, July 1993.

[17] Creel, E., Grossman, R., Reinhart, G., "DataSpace: Protocols and Services for Distributed Data Mining and Remote Data Analysis", Conference: Knowledge Discovery and Data Mining August 20-23, 2000, Boston MA, submitted for review. http://www.lac.uic.edu/

[18] Schroeder, W., Martin K., Lorensen B., An Object-Oriented Approach To 3D Graphics. Prentice Hall, 1997. http://www.kitware.com/vtk.html

[19] Folk M., A White Paper on "HDF as an Archive Format: Issues and Recommendations", January 27, 1998. NCSA/University of Illinois http://hdf.ncsa.uiuc.edu/archive/hdfasarchivefmt.htm

[20] Gamma, E., Helm R., Johnson R., and Vlissides. J., Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, 1995.

[21] TANDEM: http://www.evl.uic.edu/cavern