A Web-Based Visualization and Reposition Scheme for Scientific Data

R. Ananthuni,[1] B. B. Karki,[1] E. F. Bollig,[2] C. R. S. Da Silva[2] and G. Erlebacher[3]

[1]Department of Computer Science, Department of Geology and Geophysics, Louisiana State University, Baton Rouge, LA 70803, USA

E-mail: karki@bit.csc.lsu.edu

[2]University of Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, MN 55455, USA

[3]School of Computational Science & Information Technology, Florida State University, Tallahassee, FL 32306-4120, USA

Abstract:

We have developed a scheme for remote visualization of large collections of scientific data by supporting an expandable database and providing clients the option of an on-line data repository. A fat client approach within client-server paradigm in which the visualization software and database always reside on the server is exploited. The application classes are deployed on client's machine over the network using both Java Database Connectivity and Java Web Start technologies. Clients visualize unique data via submission through web browsers or by accessing previously submitted data on remote server. Client data, with client's permission, are subsequently validated and added to the server data so that they can be made available for other interested clients. To demonstrate our capabilities, we consider our recently developed software, which allows the interactive visualization of multivariate elastic moduli and wave propagation in an anisotropic crystal under the influence of pressure, temperature and compositional factors.

I. INTRODUCTION

Rapid advances in computer modeling/simulation along with a considerable progress in experimentation are making ever-larger sets of a wide variety of scientific data available. It is highly desirable that researchers can access the data of their interest through fast network and visualize them remotely. As such, numerous works can be found in literature in this endeavor, e.g. [1-4]. Like we have geographically distributed clients who are interested to access/analysis the remote data, we also have those clients who generate such data and are possibly willing to make their data available to others. In other words, there are server data at which a client may be interested and at the same time there are client data, which may be interesting to other clients. By providing an option for a client while he/she is executing remote visualization, to add his/her own data to the server database, it is possible to systematically expand the database over the time. To the best of our knowledge, not much work has currently been done towards supporting simultaneous remote visualization and on-line reposition of the data.

In a typical client-server based remote visualization, the server stores the relevant data and visualization software [2]. The user initiates a request for information or action through the client software. The request travels over the network to the server, which takes some action accordingly. The results are then sent back to the client. There are three different models in which visualization process is shared between the client and server. In the so-called fat server model, all programs including visualization are executed on the server's side and only the program's parameters are controlled on the client's side. The user sets the parameters and then activates the visualization process on the server's side via the network. The only rendered image is sent back to the client and displayed in the user's browser. In other so-called fat client model, for instance, a complete application can be deployed on the client machine through network network. Visualization is completely done on the client's side and the server only acts as a storage for the database. Any new version of the visualization application needs to be installed on client's side. Unlike these two models where the client and server parts of the visualization process are strictly split, a dynamically distributed model can also be designed.

In this paper, we have exploited a fat client approach for web-based implementation. The database always resides in the server. Java Web Start technology is used to deploy the application classes over the network [5]. JDBC is used to access the data from the database. Our system provides an interactive GUI for data entry/selection and for subsequent control of visualization. It is designed in such a way that every time the client locally opens the application, which is either currently downloaded or previously downloaded, the entered/selected data need to be accessed from the server database through the network. This way ensures that all data including those that have been entered by the client are always saved in the server side and, any client data, if

applicable, can be added to the server's database eventually. Thus, our approach supports an option for online reposition of the data so that the database can be expanded over the time with the help of clients who are the potential users as well as the potential sources of the data.

For illustration, we choose to support the remote visualization and reposition of elasticity data. Recently, we have initiated the development of an efficient visualization system, named as *ElasVis,* to facilitate understanding of various aspects of the mineral elasticity datasets produced by an increasing number of both theoretical calculations and experiments [6]. *ElasVis* allows us to visualize and analyze the elastic constant tensors and the additional data (for wave-velocities and anisotropy), which are generated on the fly. It has exploited a combination of the parallel coordinates, star plots, scatter plots and polygon-surface rendering techniques to visualize the original and extracted data. Elasticity is of substantial interest and importance across several scientific and engineering disciplines [7-10]. It is a key factor, which characterizes a wide range of mechanical properties and processes of materials, which could be Earth-forming minerals or technologically useful ceramics, or interesting biomolecular systems.

II. ELASTICITY VISUALIZATION SYSTEM

We first describe the basic features of *ElasVis*, which deals with visualization/analysis of multiple sets of elasticity data. While the original implementation was done in C with support of OpenGL [11, 12] and GLUI [13], the current implementation is done in JAVA [14], as it is more convenient for remote extension and interaction. JOGL [15] is used to render the data on the client machine. *ElasVis* exploits the application-based approach so that several specific visualization needs can be fulfilled. Its architecture consists of several components including data management, visualization and interactivity modules, which are related to each other.

The *ElasVis* data management component encompasses the mechanisms, which import data from external sources into visualization system and manage the data internally with generation of additional data. The original input data are the calculated or measured values of elastic moduli, which represent multivariate physical quantity defined by a symmetric 6x6 matrix, C_{ij} with $i,j = 1, 2 \ldots 6$. The maximum number of independent C_{ij} is 21, which is for a triclinic crystal [8, 16]. The presence of crystal symmetry reduces the number of independent C_{ij}'s. For instance, a

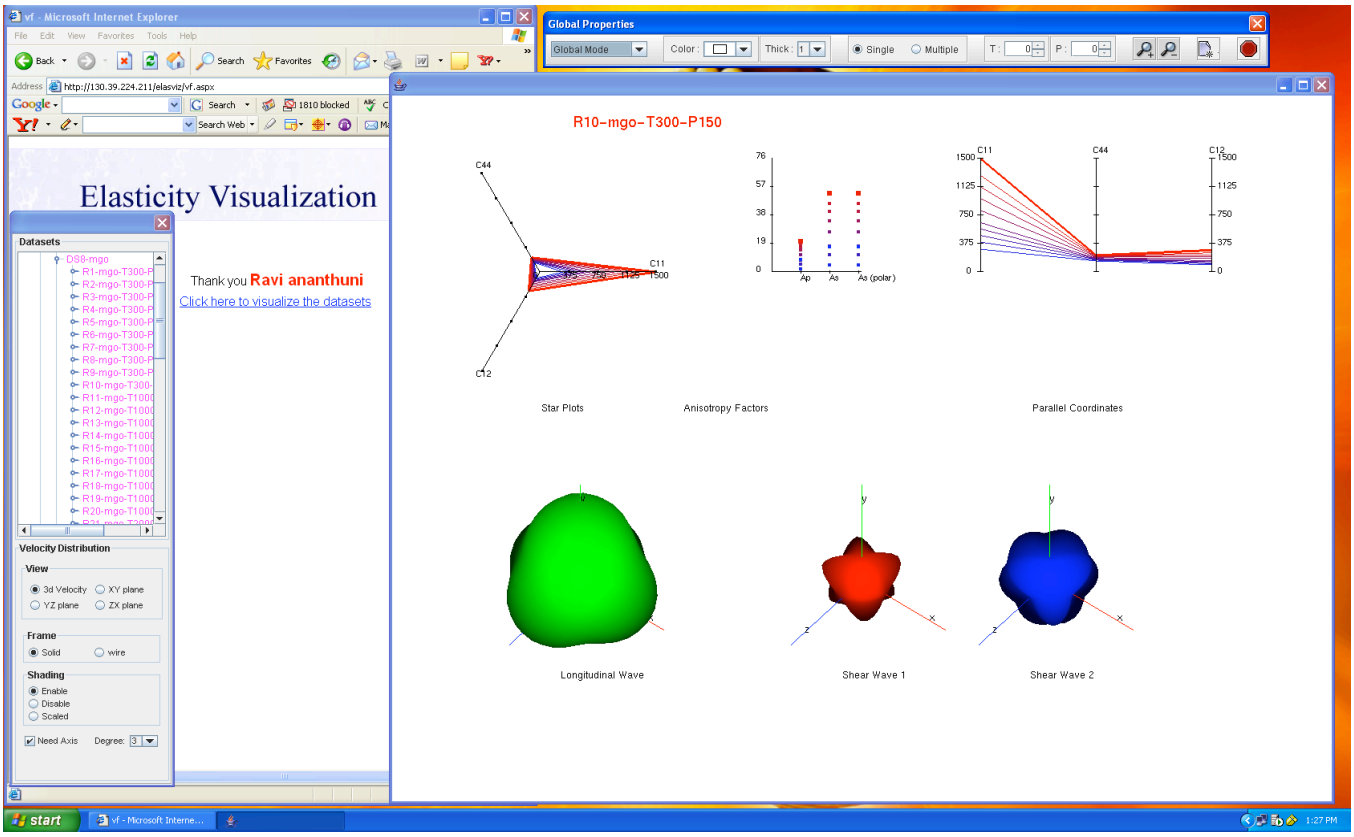

Fig. 1: Global view of *ElasVis* system deployed in the client's machine. Starplots and Parallel Coordinates are used to represent the elasticity data. The wave velocity distributions are shown by 3D surfaces. Also shown are detachable toolbars, which provide various interactive features and HTML interface.

cubic crystal is characterized by three constants, C_{11}, C_{12} and C_{44}. Crystals with lower symmetry possess higher number of independent constants, for example, nine for orthorhombic crystal. The C_{ij} values are used to generate additional data such as velocities and anisotropy factors, which are needed for the understanding of wave propagation in an anisotropic material. Elastic wave velocities are obtained as a function of propagation direction by solving the Christoffel equation [16]:

$$\left|C_{\alpha\beta\gamma\sigma} n_\beta n_\sigma - \rho V^2 \delta_{\alpha\gamma}\right| = 0$$

where n is the propagation direction, ρ is the density, V is the velocity and δ is the Kronecker delta function. The calculated eigen-values and vectors define three unique waves for each propagation direction; one longitudinal (P) wave, b) two shear (S1 and S2) waves. Thus generated velocity data (eigenvalues) are then used to generate different types of anisotropy data such as the azimuthal, polarization and transverse anisotropy factors.

The visualization module is a set of software components that realize rendering of different sets of data. It deals with manipulation of the geometric primitives (points, lines or polygons) required for rendering and determines where and when objects are displayed; supporting animation, selective, global and multiple viewports. *ElasVis* adopts the parallel coordinates and star plot techniques to visualize the C_{ij} data. In both techniques, the number of axes is equal to the number of independent elastic constants, which is determined by the symmetry of the material under consideration; for example, an orthorhombic crystal will have nine axes. The computed velocity-direction data are graphically displayed using the polygon-based surface rendering technique. Two approaches are used to enhance the three-dimensional view of the velocity surfaces; shading and scaled-coloring. Three velocity surfaces are represented by different colors: green for longitudinal wave, red for shear wave 1, and blue for shear wave 2. Thus, the shape and brightness of the closed 3D surface together represent the velocity-direction distribution. Other data are displayed using line or scatter plots.

Finally, user interface module deals with the exploration of the data in a flexible and interactive mode. The GUI has been developed using Swing classes in conjunction with AWT classes. To handle large number of datasets various interactive features are provided. These include detachable toolbars to provide more space for the application, personalized color and thickness options where a client can have his own thickness and color for the samples or datasets selected, single and multiple mode options where a client can compare samples within one dataset or more than one dataset, zooming options to increase or decrease the size of the objects on the screen, buttons to clear the present screen and stop the application, tree structure to select the dataset, samples within the dataset and elastic constant values per sample, spinners to select the datasets with respect to pressure and temperature, tool tips providing additional information corresponding to the dataset, sample and values within sample. The client is also provided with options to view the velocity distribution waves in 3D mode or 2D mode (i.e., *xy, yz* and *zx* planes), to choose solid or wire frame representation of the wave velocities, to change lightening options, to view wave velocities with or without axis and to increase or decrease the resolution of the wave velocity distribution. In addition to these various keyboard and mouse motion options are also provided. Figure 1 shows *ElasVis* deployed in the client machine and various interactive features discussed above.

III. WEB-BASED EXTENSION

A. Demands and Software Support

Our goal is to support remote access to the elasticity database and subsequent interactive visualization, with an option for online reposition of the data, irrespective of geographical location. To achieve this goal the system should fulfill the following requirements: It should be able to implement the client – server architecture effectively, being completely platform independent.

- It should support online reposition of the datasets and centralized data management. An efficient database is needed to organize the data in a structured format and provide additional functionalities such as expanding the database, client tracking, storing the datasets for future use, editing the data, etc.
- The client part of the system should make use of OpenGL graphics library and other visualization possibilities offered by the client.
- It should be insensible to network or server bottlenecks.
- All the system client components should be installed dynamically over the Internet without requiring any additional software installation.
- Any changes made to the system on the server should be dynamically updated on the client machine. Thus, the system architecture should allow easy extension and exchange.
- It should provide an interactive and straightforward GUI with easy to access controls to add and visualize the datasets.

Keeping in mind the above specific requirements, the following software support is provided. SQL server 2000 is used to develop the backend database. A web application has been developed which provides an interactive HTML interface to load the datasets to the database on the server. Java [14] has been chosen as the underlying programming language to develop the visualization software. Being platform independent, Java is a powerful object oriented language that supports OpenGL [11] through JOGL [15] (Java Bindings for OpenGL), which, in turns, supports integration with the Java platform's AWT and Swing widget sets while providing a minimal and easy-to-use API. The client server architecture is implemented using Java Web Start [5] Technology with which standalone Java software

applications can be deployed with a single click over the network. Moreover it ensures that the latest version of the application and requested version of JRE is installed on the client machine.

B. Design

Our remote elasticity visualization system is designed based on a flat client model. Figure 2 gives the schematic diagram of the client – server architecture used for the *ElasVis* system. Server acts as a repository for data, web application files, Java standard application classes, external libraries, JNLP [17] (Java Network Launching Protocol) files developed with each client request and other scripts. The datasets are added to the database using a HTML interface. Application classes and other dependencies are deployed in the client machine using Java Web Start technology. For each client request a JNLP file will be saved on the web server. The client connects to the JNLP file using a link on the web browser. Based on the JNLP descriptor, application classes and external libraries are signed and deployed dynamically on the client machine. If any pieces of the application are already cached on the client machine, those components are not deployed again, unless they have been updated on the server machine. Thus, any changes or developments to the server will be dynamically updated on the client. The external libraries include JOGL JAR file to visualize the datasets. These deployed application classes connect back to the database using JDBC to access the data. The data is then rendered on the client machine using JOGL.

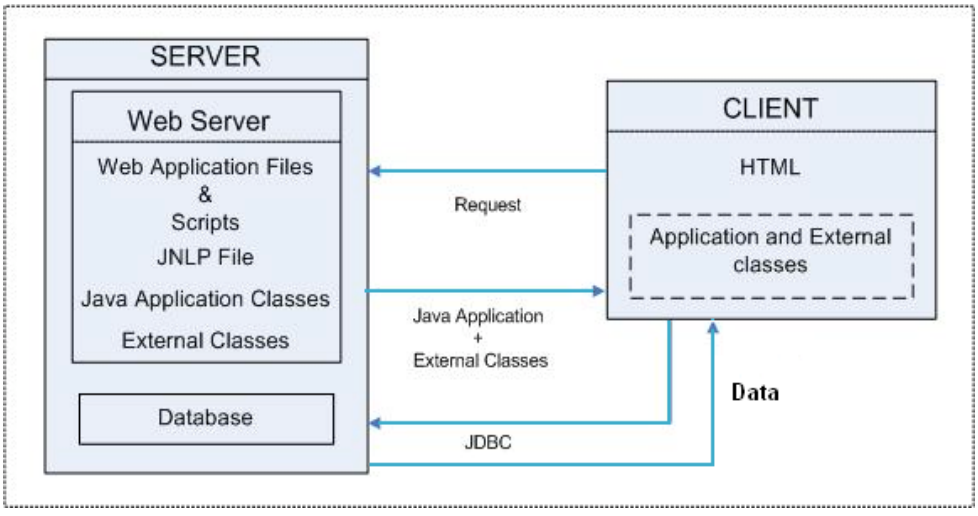

Fig. 2: Client – Server architecture

C. Online Data Reposition

The elasticity database is dynamically expanded with the option for online data reposition. A similar interactive interface is provided to both client and admin to add the data. The datasets added by the admin, called the server data, are public and are accessible to all clients. Data added by the client, called client datasets, are private and are subsequently accessed only by him or her. However, if the client claims that his/her data is genuine and can be made public, it waits for the approval of the administrator. At some regular time intervals, each client-claimed genuine dataset is tested for the accuracy and reality before it can be added to the server dataset. The client data after the test remain on the server for other clients and for future access.

D. Remote Visualization

To remotely visualize the data, client needs to create a Wish-List. A Wish-List is a set of datasets the client would like to visualize. A Wish-List can be created from

- New datasets created by the client in a session
- Datasets that are previously generated by the client but not yet approved by the administrator
- Server datasets created by the administrator or genuine client datasets approved by the administrator and added to server datasets

The client is also provided with an option to edit the datasets that have been created by him. More over he can save these datasets for future reference and allow other clients to visualize them. These datasets are to be approved by the administrator before they can be visualized by other clients. The information corresponding to client preferences is stored in session variables and cookies during the client navigation. Before the application classes are downloaded they are packed in to a JAR file, signed and saved on the web server. A JNLPfile is created and the information corresponding to the location of the JAR file, client preferences (list of selected datasets: Wish-List) and access is passed to it. The JNLP file is saved on the server and the client executes it through a link on the web.

Before accessing the application classes, the scripts check for Java Web Start software in the client machine. If not available it automatically downloads the software to the client's machine. Depending upon the JNLP descriptor, Web Start software downloads application classes and its dependencies (JOGL and other resources), as well as downloads the Java Runtime Environment (JRE) if the requested version is not available locally. The client can also download this software's manually. For the first time, usually it takes couple of seconds for JRE and application classes to get downloaded and run locally in the client machine. The web part of the system that involves creating a Wish-List, creating a JNLP

file, creating a JAR file, signing the JAR file, checking for the soft ware support is handled by the web application developed in ASP.Net. The application classes run locally in the client machine. They access the data from the database using JDBC. JOGL classes then help in rendering the data in the client machine. Figure 1 shows the *ElasVis* system deployed in the clients machine. It also shows the HTML interface used for online reposition of the datasets and a link to JNLP file.

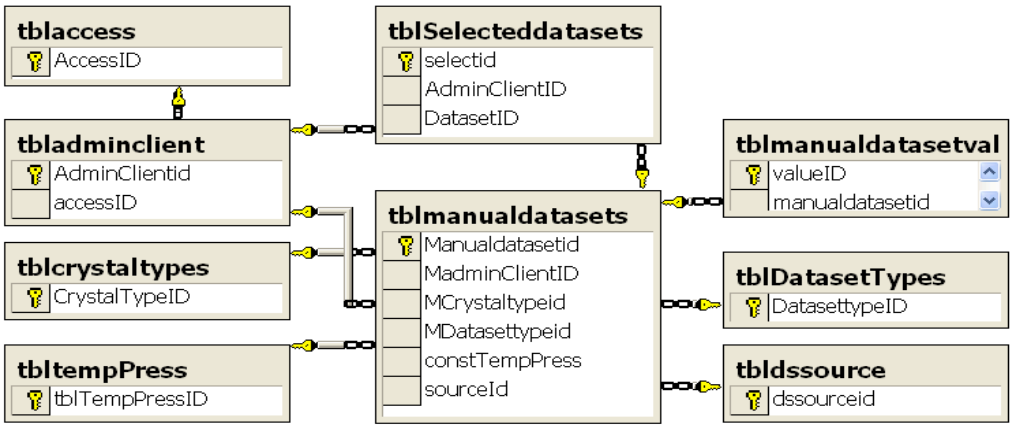

Fig. 3: Entity Relationship Diagram of *ElasVis* database

IV. SERVER DATABASE

The *ElasVis* database acts as a repository for elasticity datasets. In general, a database adds more flexibility to the system and improves the security and integrity of the system. It is centralized and helps in providing better service to the users. Back up and recovery options help in keeping the data safe. Keeping in mind these advantages, *ElasVis* database has been designed and developed in SQL server 2000, to do the following:

- Support large datasets of different types with ease.
- Allow a client to modify his datasets and others to visualize them, once the administrator approves it.
- Allow a client to create a temporary Wish-List out of the existing datasets. Once the request has been completed the Wish-List is made empty.
- Accommodate both the client and the server datasets in the same table to avoid redundancy

Figure 3 shows the Entity Relationship Diagram of the *ElasVis* database designed to accommodate the above functionality. The *tbladminclient* table is used to accommodate personal information of the client and the admin. *tblmanualdatasets* table has all the information corresponding to the datasets created by the client or admin. It tells us whether the dataset created is a server dataset or a client dataset, genuine or not, source of the dataset (Experiment or calculation) and other information related to the dataset. It allows a dataset to be saved for future reference and made public or not. Since each dataset can have more than one sample values, all these samples are saved in *tblmanualdatasetvalues* table. *tblselecteddatasets* table helps in creating a Wish-List for the client. This table mainly helps in selecting more than one dataset per request. All the datasets that are chosen by the client during a session are saved temporarily in this table till the request is processed. Once the request is processed the Wish-List is made empty. Note that as the wish list is made empty, actual datasets in *tblmanualdatasets* table and their values in *tblmanualdatasetvalues* table are not affected. A system may contain different types of datasets. *tblDatasetTypes* table has the list of different types of datasets and the properties associated with each type of dataset. *tblaccess* has the permission levels (client and admin) present in the system. *tblcrystaltypes* has the list of various crystal types a dataset can be associated with. *tbldssource* gives the list of various sources (experiment, calculation, etc) through which the dataset values can be obtained.

V. RESULTS AND ANALYSIS

A. Performance

The performance of the system involves download, computation and rendering times. The download time depends upon the speed of the internet. Usually it takes 2 to 3 seconds to download the application classes at 100 Mbps WAN. Computation time can be considered as the time taken to read and process the data before the objects can be rendered on screen. Data processing involves recursive division of triangles on the surface of the icosahedron used to represent 3D wave-velocity surface and generation of other additional data. The computation time plays an important role at the start of the system and when the client wishes to change the resolution [Figure 4]. At the highest resolution (denoted as level 4), the time per sample is around 50 milliseconds. A lower resolution 3 takes around 12 ms whereas the resolutions 2 and 1 take around 4 and 1 ms, respectively. In Figure 4 the anisotropic behavior (shape) can better be seen as the resolution increases.

Rendering time usually depends upon the number of viewports generated (the maximum of 8 x 8 = 64 viewports) and the resolution maintained. As the number of viewports is increased, the rendering speed decreases. Moreover, rendering speed also decreases with an increase in the resolution. To provide interactivity for the user, a minimum of about 10 frames has to be generated per second. Thus, a compromise has to be made between the speed and resolution as the number of viewports increases. Keeping

this in mind, an automatic resolution adjustment is recommended depending upon the number of viewports used at a given time. Table 1 gives how the resolution is varied to maintain an interactive frame rate.

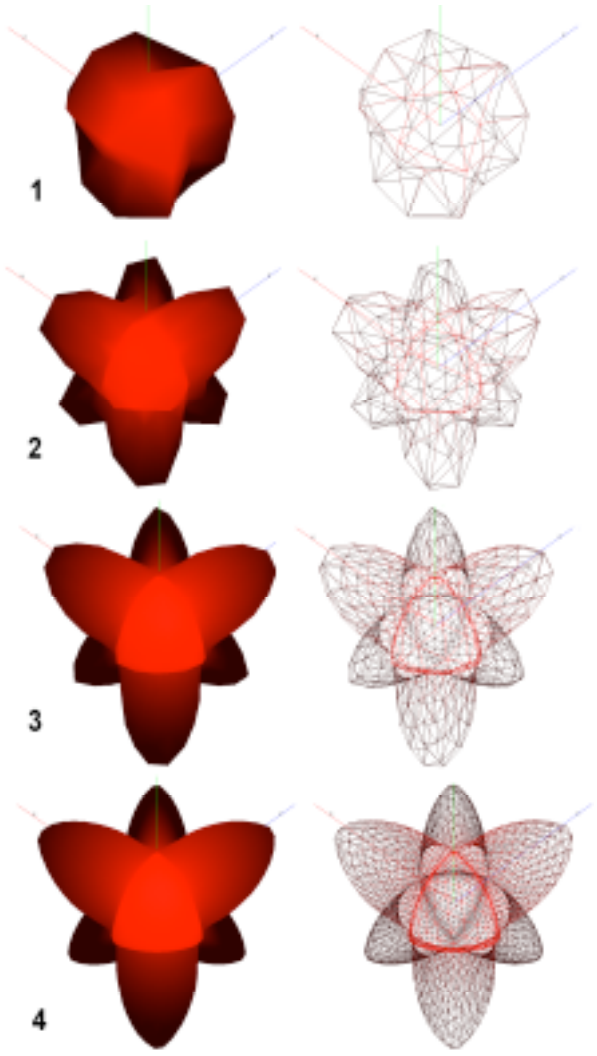

Fig. 4: Different resolutions for calcium oxide (CaO) for shear wave 1 at temperature of 300 kelvins and pressure of 80 gigapascals.

B. Case Study

The pressure and temperature variations of the elastic properties of MgO are visualized using calculated data [18], which contain 40 sample points. Figure 5 illustrates visualization of the anisotropic behavior of MgO, where the color intensity and the shape of the 3D objects encode the velocity-direction distribution for the shear wave. The darker regions represent low velocity whereas the brighter regions represent high velocity. The maximum and minimum velocities are denoted by white dots in the brightest and darkest regions, respectively. The more the shape is distorted from a sphere, the more is the sample anisotropic.

Table 1: Managing resolution for interactive frame rate

Viewports	Resolution Preferred	Rendering Speed fps
1 X 1	4	18.9
2 X 2	4	10.1
3 X 3	3	19.6
4 X 4	3	10.8
5 X 5	2	27.4
6 X 6	2	19.4
7 X 7	2	14.5
8 X 8	2	11.1

As shown in Figure 5, at 300 K first 2 samples (corresponding to pressures of 0 and 10 GPa) have the maximum velocity along the direction [100], that is, along the *x*-, *y*- and *z*-axes, while the remaining 8 samples (corresponding to higher pressures of 20, 30, 40, 50, 60, 80, 100, 120, 150 GPa) have the minimum velocity along the [100] direction of axes. Thus, the propagation directions of the maximum and minimum velocities get reversed with pressure between 10 and 20 GPa at 300 K. A similar trend is observed at higher temperatures of 1000, 2000 and 3000 K. However, the pressure at which the anisotropic pattern reverses shifts to a higher value, for instance, it occurs between 30 and 40 GPa at 3000 K (samples from 31 to 40). Also the degree of anisotropy becomes minimum at the transition pressure (note that the 3D object tends to be more spherical) and then it increases with increasing pressure. Moreover, the size of the 3D object becomes larger at higher pressure since the pressure enhances the wave velocity. Similar variations in the magnitude of the velocity and degree and character of the anisotropy can be inferred for the longitudinal wave.

VI. CONCLUSION

We have described design, implementation and application of a remote visualization system that enables a visually aided analysis of large collections of mineral elasticity data. *ElasVis* visualizes the multivariate elastic moduli and the resulting anisotropic wave propagation as a function of composition, pressure and temperature. Data are rendered using a combination of parallel coordinates, star plot, scatter plot and polygon-surface rendering techniques, which are implemented using JOGL. This results in a highly portable and flexible interactive visualization system. The system is web enabled. The client-server architecture of the fat client model is presented and online reposition of the data is also supported. The anisotropic behavior of magnesium oxide compound at various temperatures and pressures is presented for illustrations. We are currently integrating the *ElasVis* system within the overflow of the

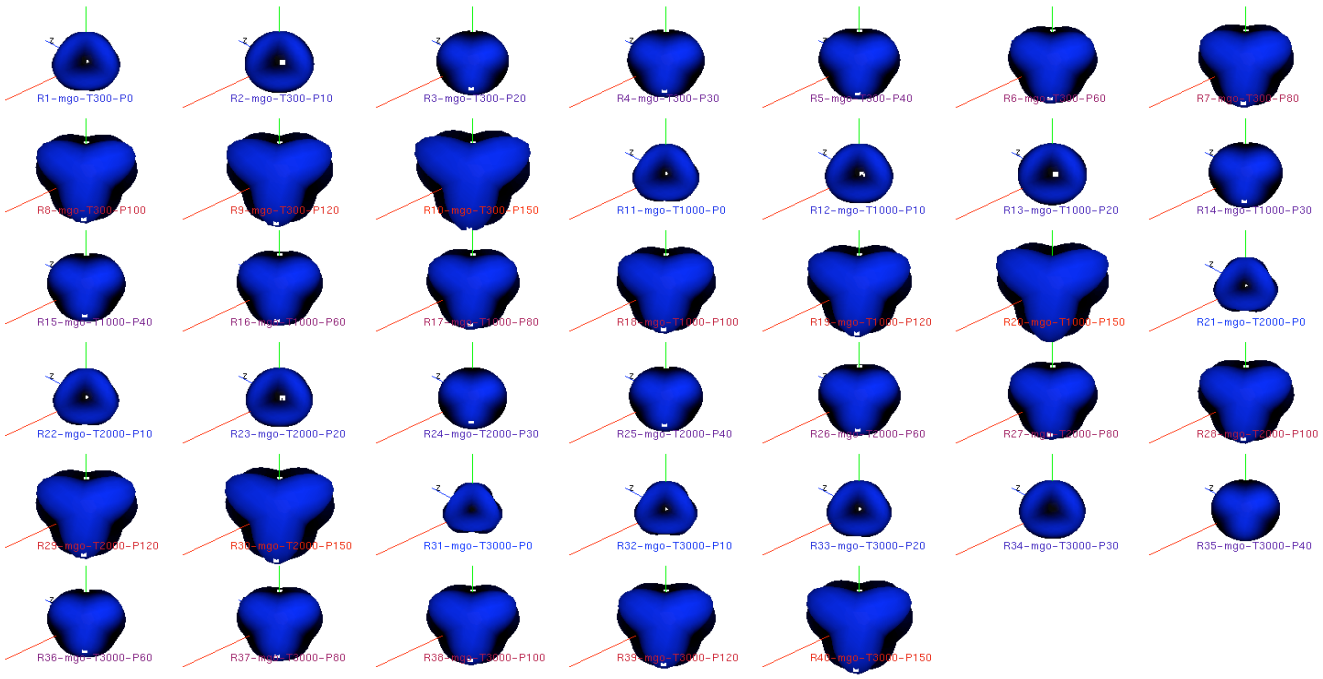

Fig. 5: Shear wave 2 velocity distribution of 40 data samples for magnesium oxide (MgO) as a function of pressure (0, 10, 20, 30, 40 60, 80, 100, 120 and 150 gigapascals) and of temperature (300, 1000, 2000 and 3000 kelvins). The samples from 1 to 10 from top left correspond to 300 K, the 11 to 20 samples correspond to 1000 K, the 21 to 30 ones to 2000 K and the 31 to 40 ones to 3000 K. The Cartesian axes are also shown.

VLab portal, representing a Virtual Laboratory, which has a mission of providing easy-to-use software for the mineral physics community (www.vlab.msi.umn.edu).

Acknowledgements: This work is supported by the NSF Career (EAR 0347204) and is being integrated with the Vlab project funded by ATM 0426601.

REFERENCES

[1] Trapp J.C. and Pagendarm H.-G., A prototype for a WWW-based visualization service, Proc. Eight Eurographics Workshop Visualization in Scientific Computing, 1997

[2] Bender M, Klein R, Disch A, and Edert A (2000) A functional framework for web based information visualization systems, IEEE transactions on visualization and computer graphics, vol. 6. No. 1, 2000.

[3] Stegmaier S, Diespstraten J, Weiler M, Ertl T, Widening the Remote Visualization Bottleneck, In *Procceedings of ISPA '03* . IEEE (2003)

[4] Wang Y, Erlebacher G, Garbow ZA and Yuen DA Web-based service of a visualization package "Amira" for the geosciences, Visual Geosciences (2005)

[5] Java Web Start home: http://java.sun.com/products/javawebstart/

[6] Karki BB, Chennamsetty R, A visualization system for mineral elasticity, Visual Geosciences (2004)

[7] The Elle Mineral Properties Database (2004) www.microstructure.uni-tuebingen.de/mindb

[8] Karki BB, Stixrude L, Wentzcovitch RM, Elastic properties of major materials of earth's mantle from first principles, Reviews of Geophysics 39: 507-534 (2001)

[9] Liebermann RC and Li B, Elasticity at high pressures and temperatures, Reviews of Mineralogy 37: 459-492 (1998)

[10] Sinogeikin SV, Zhang J and Bass JD, Elasticity of single crystal and polycrystalline $MgSiO_3$ perovskite by Brillouin spectroscopy, Geophysical Research Letters 31, No. GL019559 (2004)

[11] chroeder D, Woo M, Neider J and Davis T, OpenGL Programming Guide, 4th Edition, Addison-Wesley (2004)

[12] OpenGL – http://www.OpenGL [19].org/

[13] http://www.cs.unc.edu/~rademach/glui

[14] Java home page: http://java.sun.com/

[15] JOGL home page: https://JOGL [17].dev.java.net/

[16] Musgrave MJP, Crystal Acoustics, Holden-Day, San Francisco (1972)

[17] JNLP Syntax http://java.sun.com/j2se/1.4.2/docs/guide/jws/developersguide/syntax.html

[18] Karki BB, Wentzcovitch RM, de Gironcoli S, Baroni S (1999) First principles determination elastic anisotropy and wave velocities of MgO at lower mantle conditions, Science 286: 1705-1707 (1999).