Web-Based Service of a Visualization Package "Amira" for the Geosciences

Yunsong Wang[1], Gordon Erlebacher[1], Zachary A. Garbow[2], David A. Yuen[2]

[1]School of Computational Science & Information Technology, Florida State University, Tallahassee, FL 32306-4120, U.S.A
Emails: yunsong@csit.fsu.edu, erlebach@csit.fsu.edu

[2]Department of Geology and Geophysics, University of Minnesota Supercomputing Institute, University of Minnesota, Minneapolis, MN 55455-0219, U.S.A
Emails: garbowza@msi.umn.edu, davey@msi.umn.edu

Abstract. Amira is a powerful 3-D visualization package and has been employed recently by the science and engineering communities to gain insight into their data. We discuss a new paradigm for the use of Amira in the Earth sciences, which relies on the client-server paradigm. We have developed a module called WEB-IS2, which provides web-based access to Amira. This tool allows Earth scientists to manipulate Amira controls remotely and to analyze, render and view large datasets through the Internet, without regard for time or location. This could have important ramifications for GRID computing.

1. Introduction

There is an ongoing explosion of data in the geosciences: datasets are growing in size at what seems to be exponential rates because of the acquisition of satellite interferometry and multispectral data, geodetic data collection campaigns and ever more detailed seismic surveys. Large scale, three-dimensional, unsteady simulations of geophysical phenomena over a wide range of spatial and temporal scales contribute significantly to this growth. Aside from the sheer size of the data sets, large scale multi-institution, multi-regional collaborations result in the distribution of this data across the globe. Both these trends make it all but impossible for the Earth scientist to effectively visualize and analyze the available data in a timely manner (Erlebacher *et al.* 2001).

There is a clear need to develop tools that efficiently extract relevant subsets of the data that can be manipulated and analyzed by the scientists. These tools comprise the gamut from computational tools that preprocess the data into more useful forms, analysis tools that can perform relevant statistical studies, and visualization tools to help the scientist gain insight into the structure of the data (Erlebacher and Yuen 2002, 2003).

In this paper, we discuss the use of Amira (http:/www.amiravis.com), a middleware program to make available to the end users the capability of interactively exploring their data, although they may not have at their local sites the necessary software. Amira is a 3D visualization and modeling system that allows researchers to visualize scientific datasets from various application areas, e.g., medicine, biology, chemistry, physics, engineering, and the geosciences. From the perspective of visualization, Amira has already proven to be a very effective tool. Users construct programs visually with flow charts of modules that represent tasks (e.g., isosurfacing, thresholding, volume rendering, etc.) or via Tcl scripts. In both cases, the result is the display of a subset of the original data. Furthermore, Amira uses some advanced algorithms that capitalize on the hardware available on many of the current commodity graphics cards. However, very large datasets cannot, and should not, be loaded into memory in their entirety. For example, data sets can easily exceed one terabyte for a single time-dependent numerical simulation on a grid with 512^3 points, which is only a moderate size when compared to the largest simulations currently underway at resolutions of 4096^3 (Yokokawa 2003). As an illustration, a single variable, at a single time, at this high resolution takes up 0.5 Terabytes. To help navigate the range of technologies involved in this work, we list the acronyms used in Table 1, along with URLs when appropriate.

It is quite clear that aside from developing efficient extraction techniques to act on these datasets (which require themselves extensive computational resources), it is necessary to have at one's disposal the facilities to visualize the results of the extraction process without regard for geographical location. It is with this end in mind that we have developed the initial steps in an ambitious program to provide remote visualization resources that will permit users to interact with their data without having at their disposal the high performance software and hardware that is necessary. The results can be displayed on a client, whose capabilities range from that of laptops to handheld devices.

Acronym	**Definition**	**URL**
ACE	**A**daptive **C**ommunication **E**nvironment	http://www.cs.wustl.edu/~schmidt/ACE-documentation.html
BMP	**Bi**t**map**	http://www.csdn.net/dev/Format/-windows/Bmp.html
CGI	**C**ommon **G**ateway **I**nterface	http://www.w3.org/CGI/
GAT	**G**rid **A**pplication **T**oolkit	http://www.gridlab.org/
GUI	**G**raphical **U**ser **I**nterface	
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol	http://www.w3c.org/Protocols/
JPEG	**J**oint **P**hotographic **E**xperts **G**roup	http://www.JPEG.org
PIL	**P**ython **I**mage **L**ibrary	http://www.pythonware.com/products.pil
Tcl	**T**ool **C**ommand **L**anguage	http://www.tcl.tk
URL	**U**niform **R**esource **L**ocator	http://www.w3.org/Addressing/
VAN	**V**isual **A**rea **N**etworking	http://www.sgi.com/visualization/van
WEB-IS	**W**eb-**B**ased **I**nterrogation **S**ystem	
WEB-IS2	**We**b-**B**ased **I**nterrogation **S**ystem, version **2** (with links to Amira)	
WDI	**W**eb-Based **D**ata **I**nterrogation	
XML	**E**xtended **M**arkup **L**anguage	http://www.w3.org/XML/

Table 1: List of acronyms used in the text

The system described in this paper is called Amira Web-based Service (WEB-IS2), which is a web-based interactive application that allows users on a client machine to remotely control, display and share visual information generated by Amira on a server over the Internet. The name WEB-IS2 was chosen to reflect that the described software is one of a series of several generation of tools in the Web-based Interrogation System (WEB-IS) series (Garbow *et al.* 2003b).

A Graphical User Interface (GUI) on the client browser lets users interact with the system via icons and a pointer rather through text entry at a command line (see Figure 1). The GUI provides a convenient front-end to facilitate the use of Amira from anywhere in the world, based on a reliable, fast network. Most of the computational work is performed on the server, with only user interaction done on the client. As a consequence, our software is accessible from any location with Internet connectivity. The only requirements imposed on the client are that the

browser be able to display Java 1.4 applets, which is easily satisfied. For this reason, WEB-IS2 can be used almost on any platform. Although the demands on the client hardware are modest, the range of screen sizes demands that we pay particular attention to the details of the user interface. While users demand flexibility, they are often put off by a lack of simplicity and ease of use. The number of options displayed on the screen of a PDA must be kept to a minimum, with the only the most useful options directly accessible.

Our aim is to successfully overcome the limitations of handheld devices to allow them to be used as a "powerful" and portable resource. The client sends Amira commands or scripts to the server for dataset visualization or analysis. The Amira server renders and analyzes the dataset at the request from the client and sends the resulting image or the analyzed results over the Internet to the client to be displayed. As a result, users interact with the client through a web browser, decoupled from both the Amira server and the physical datasets.

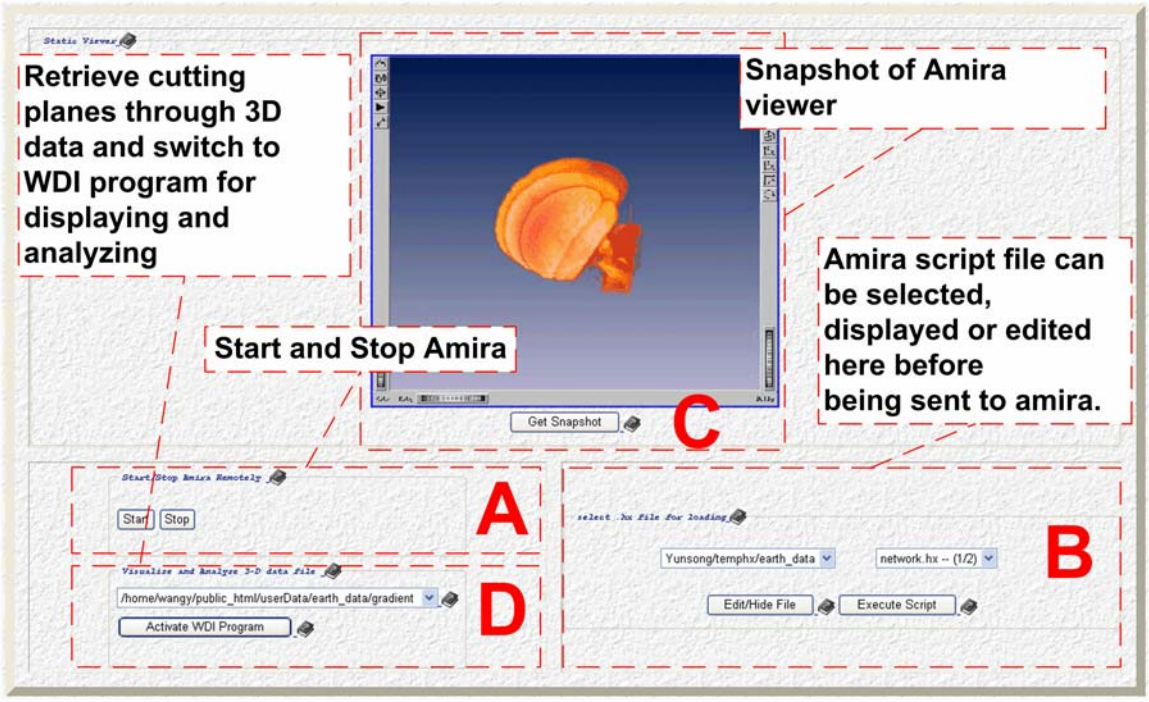

Figure 1 Web-based service for Amira allows user to analyze, render and view (using amira) the large datasets over the Internet from a web browser.

In summary, our goal is to enable the analysis and visualization of the large geophysical datasets stored around the world using client devices that range from laptops to handheld devices used by geographically distributed organizations.

2. Related work

Garbow has developed a web-based interrogative system, called Web-based Data Interrogation (WDI) that permits the users to analyze data remotely over the Internet using a client-server paradigm (Garbow *et al.* 2001, 2003a-b). This system promotes portability, dynamic results on-demand, and collaboration among researchers separated by long distances. A major feature of WEB-IS2 is its capability to generate non-compressed data suitable for analysis by WDI's server, and a JPEG image for display in the WDI client. An interface is provided so that users can analyze the data from Amira with WDI. They have recently expanded on their idea of

web-based data interrogation with the Web-based Interrogation System (WEB-IS), which allows the user to navigate through their 3-D datasets and interactively analyze the data for statistics or apply data mining techniques. In WEB-IS, the developer creates his own visualization tools on a case by case basis. Our software, however, harnesses the visualization power of Amira, although specialized routines can also be written that expand the capabilities of Amira. Upgrading WEB-IS to WEB-IS2 will greatly expand the ability of scientists to explore their large datasets.

Silicon Graphics provides a commercial solution called Visual Area Networking (VAN) (http://www.sgi.com/visualization/van) to advanced visualization and collaboration across multiple sites, allowing scientists around the world to work cooperatively and share visualization information with each other VAN provides instant, visual access to large data sets--independent of the location of the data or the person accessing it. However, the software is massive and very expensive, and also in addition, is designed to run only on SGI machines. These constraints limit its widespread adoption.

The GridLab project (http://www.gridlab.org) is an ambitious project led from Poland that aims to develop a flexible, easy-to-use, generic and modular Grid Application Toolkit (GAT), enabling today's applications to make innovative use of global computing resources. They have already developed remote access tools to visualize huge datasets, monitor performance and simulation properties and interactively steer the simulation (http://www.cactuscode.org). Moreover, GAT is designed to handle dynamic reconfiguration of the grid and a certain level of fault tolerance. However, GAT requires much more software than ours to use it. An extensive, up to date reference on Grid activities has just appeared (Berman *et al.* 2003).

3. HTML Interface to Amira

The panel of the client interface is divided into four areas, labeled *A*, *B*, *C* and *D*. There are two buttons, "Start" and "Stop" in area *A*. They let users initiate and terminate Amira. Once initiated, an Amira script is loaded and run on the Amira server. Area *B* provides a dropdown list to browse directories and retrieve a particular Amira script file. The results of running the script on the server can be checked by capturing a snapshot of the viewer (see "Get Snapshot" button in area *C*), which is displayed in area *C*. In a future version, the snapshot will be generated automatically. The difficulty lies in the inability of Amira to send out messages over the socket, for example acknowledge a task completion. Area *D* allows the user to select a dataset and connect to the Web-based Data Interrogation (WDI) component of the system (Garbow *et al.* 2003a). WDI is then loaded into the current browser ready to manipulate the data contained in the file referred to by WEB-IS2.

WEB-IS2 makes extensive use of Python (http://www.python.org), the Adaptive Communication Environment (ACE) (http://www.riverace.com), Tcl (http://www.tcl.tk), and C/C++. CGI scripts are written in Python, the message communication between the Common Gateway Interface (CGI) and the Amira server are written with the help of the ACE Application Programmer Interface (API); and the communication program interfaces to Amira through the intermediary of Tcl scripts. (Amira console only accepts external commands written in Tcl.) Statistical processing and other analysis routines are written in C or C++ to maximize performance.

The images in the client interface are two-dimensional bitmaps sent from the server in compressed JPEG format (http://www.JPEG.org). As a consequence, client-side storage and network resources are relatively low, which is an ideal match for handheld devices, since their main bottleneck (when compared to capabilities of current laptops) is low-performance software and hardware. The client sends command information to the server, which is then responsible for passing on the message to the proper end-point: either Amira or the computational engine attached to WDI responsible for the statistical calculations. Finally, the server sends either a compressed image or computed statistical data to the client.

4. Design and Implementation

Client-server architectures seek to separate out user functions (user interface and display) from the more computationally intensive tasks. Figure 2 is a bird's eye view of the architecture. Multiple clients, ranging from desktop computers to handheld devices, each with a browser that supports Java 1.4 applets, have the ability to connect to a server capable of running high level visualization software. As implemented, the server plays multiple roles: it is a repository for the datasets to be analyzed, it hosts the Amira software, and it runs the statistical software used by WDI. Clearly, as an increasing number of users access the system, it will become overloaded. It will therefore be important to separate out these functions as much as possible, and allow them to run on separate servers. Although this has not yet been implemented, the proposed architecture and software tools have been selected with this generalization in mind. In what follows, the word server refers to a single machine that handles data storage, computation and visualization.

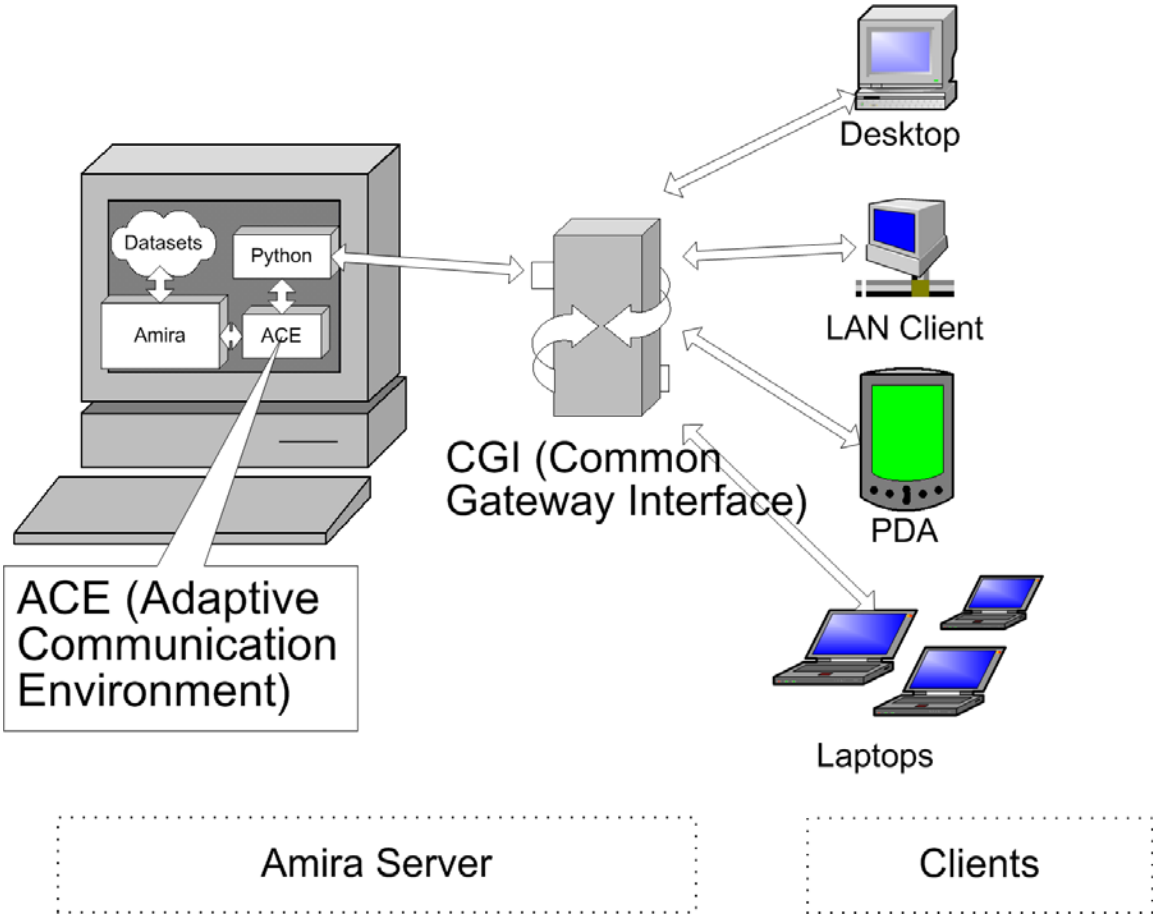

Figure 2 Schematic diagram of the "client – Amira server" setup used for WEB-IS2

The client interfaces with the server through the intermediary of CGI scripts written in Python (http://www.python.org). Python is an interpreted language with high-level data structures and a simple but effective approach to object-oriented programming. Python is open source and is supported by a large number of third party modules and libraries. To speed up development, Python provides powerful debugging utilities that generate detailed error diagnostics directly within the client's browser. The Common Gateway Interface (CGI) is a standard for interfacing external applications with an information server. CGI scripts execute in real-time and output dynamic information (http://www.w3c.org/CGI). They are invoked by an HTTP server, and then

executed. Output is returned to the client. During execution, a CGI script sends commands to service programs (such as Amira), retrieves their output, and returns them to the client. Clients communicate with the CGI scripts on the server through input parameters encoded into a URL.

An important restriction on the use of our system, as opposed to WEB-IS (Garbow *et al.* 2003b) is the current lack of off-screen rendering capability for the Amira software package. As a consequence, the server cannot be a machine used by the researchers for routine work, visual or otherwise, else there may be interference with remote usage. Furthermore, the rendering appears on the screen, which necessitates that some user be logged onto the server console to ensure that Amira can be activated remotely. It should also be noted that our current version of WEB-IS2 has only been tested on Linux systems. Recent research has demonstrated the possibility of off-screen rendering of graphics programs without the need to modify their source code (Stegmaier *et al.* 2002).

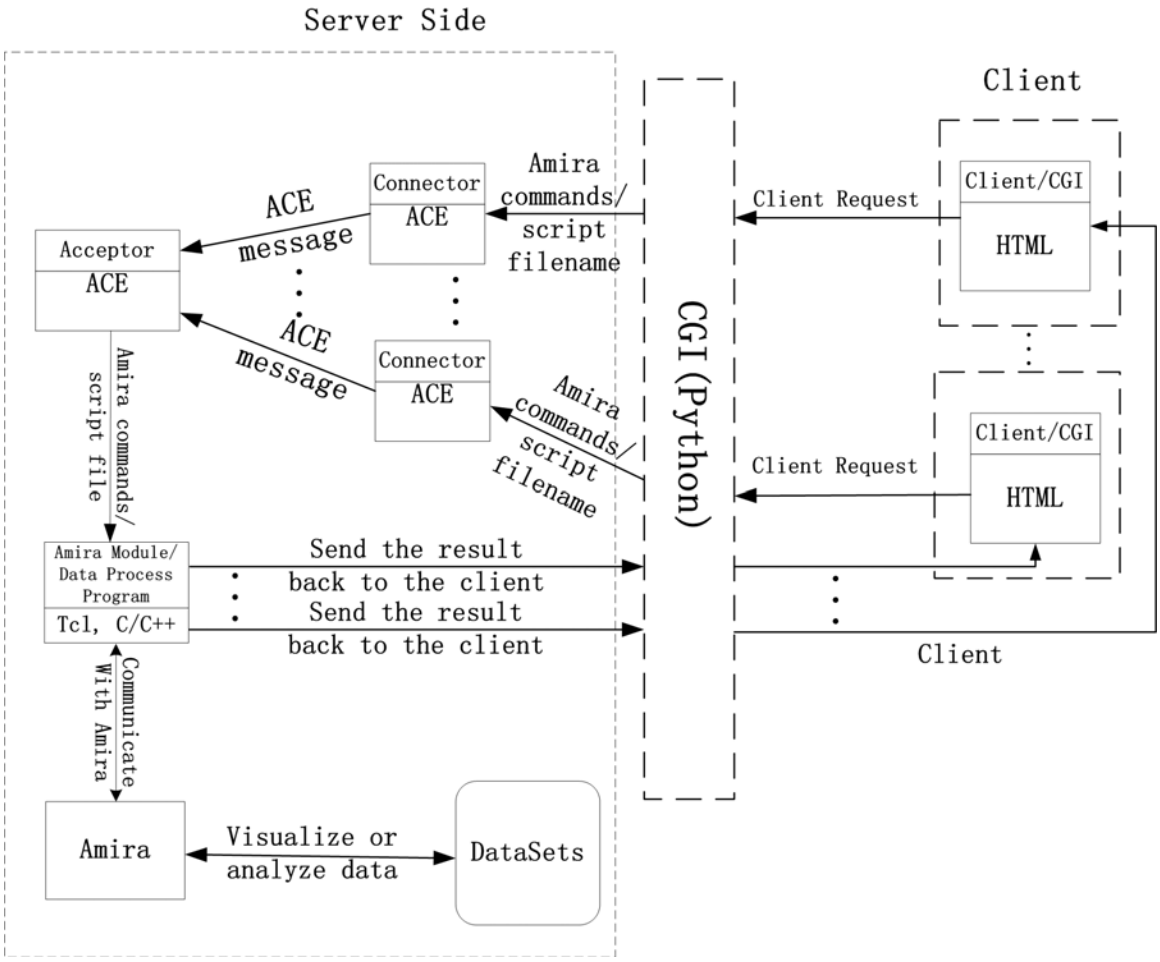

Figure 3 Functional setup for WEB-IS2 that allows remote visualization and data analysis with Amira over the Internet. The client interacts with the CGI script located on the web browser (right), which provides an interface to interact with the server (left) to analyze, render and view datasets with Amira.

A more detailed flowchart of our implementation is illustrated in Figure 3. The underlying message communication between CGI and the Amira application makes extensive use of the Adaptive Communication Environment (ACE), a modern, high-performance application framework and C++ class library that implements multi-threaded socket-based communication, and is heavily based on patterns (Schmidt 2000). ACE includes a wide variety of classes and

patterns that simplify the development of communication software, thereby enhancing flexibility, efficiency, reliability and portability (http://www.riverace.com).

Currently, all of WEB-IS2 is on a single server, and supports multiple clients. However, we allow for the possibility of activating multiple instances of Amira on the server. To this end, the CGI script opens a socket connection, referred to as a `connector`, which in turn communicates with an `acceptor`. The `acceptor` is a daemon whose role is to redirect the incoming messages to the correct Amira instance. Messages sent to Amira can originate either from WEB-IS2 or from WDI. The `acceptor` passes the message received to the Amira application, which is for producing a bitmap image (BMP format), that is sent to WEB-IS2 or to WDI.

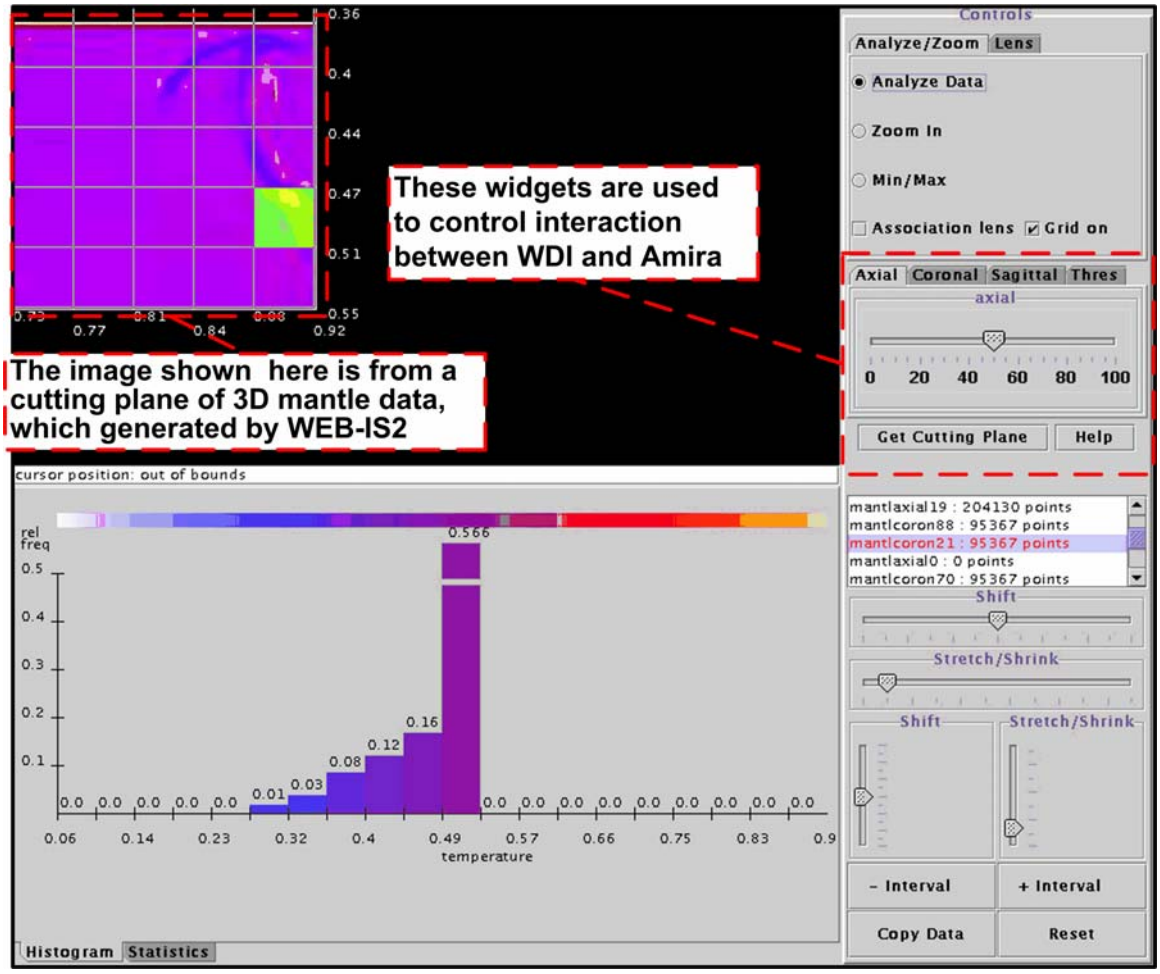

Figure 4. A snapshot of WDI client interface analyzing data from WEB-IS2. The image area displays the compressed image sent by server, which is based on uncompressed datasets in server. The user can view and analyze the desired datasets by sending request to server. The server utilizes Amira with WEB-IS2 to generate the image and data suitable for WDI. The data shown were generated by numerical simulations of 3D convection in the Earth's mantle (Dubuffet *et al* 2001).

One advantage of the `Acceptor/Connector` pattern is a clear separation between the task of creating a socket connection from the computational task that follows. As a result, connection establishment policies and subsequent tasks can be fine-tuned independently (http://www.cs.wustl.edu/~schmidt/ACE-documentation.html). In addition, the use of ACE should make upgrading the current underlying connection mechanism to support multi-threaded

connections, necessary for a robust high performance scalable network-distributed system, a relatively straightforward endeavor. Finally, use of ACE as the underlying communication scheme will allow us to separate the Amira server from the WEB server for improved security and more flexibility.

We have written several Amira modules in Tcl that are preloaded into Amira when it is activated by the remote client. These are meant to simplify control scripts. Examples include routines to take snapshots and to quit the program (with appropriate cleanup). As our experience with WEB-IS2 grows, we will replace selected low performance Tcl modules with C++ implementations, and add additional modules to perform specialized functions, such as wavelet transforms and other kinds of analysis (Erlebacher and Yuen 2003). Currently, Tcl (http://www.tcl.tk) is the only means of interacting with Amira without accessing its C++ API.

Each instance of Amira has a unique port number associated with it. The software is capable of accepting Tcl commands to execute via this port. The port number is therefore useful for the acceptor to uniquely tag clients using Amira. Currently, a single instance of Amira can only be controlled by a single client. In the future, we would allow multiple users to control a single instance of Amira, via token sharing; in the simplest case, the token and the port number are the same. More elaborate schemes become necessary when operating multiple Amira servers.

5. Capabilities

WEB-IS2 has two modes of user interaction. The first is based on the HTML interface shown in Figure 1. The second is provided by the interface to WDI, which can instruct Amira to slice the data and perform statistical analysis. An image of the WDI interface is shown in Figure 4.

5.1 Capabilities of client based on HTML

WEB-IS2 can activate multiple Amira instances from one or multiple client browsers and control them individually, although all copies of Amira are instantiated on a single server. Each session is identified by the port number associated with a particular Amira instance. When an Amira starts up, its startup script sends the port number to the requesting client. Users can switch among different Amira instances easily by switching port numbers. The port number behaves like a token through which each user is identified. Given enough hardware resources, memory in particular, many instances can be supported since the computational load for a single idle instance is relatively small. In our case (dual XEON 1.7GHz,, one GByte memory), Amira consumes approximately ten percent of the CPU resources and nearly 2.5 percent of the memory (25 MBytes) in the first few seconds, after which CPU utilization drops to zero and the process status switches to sleep mode. However, memory consumption is unchanged. The main bottleneck for supporting additional Amira instances is therefore memory and the particular user applications. Complex scripts involving several large datasets on a system with insufficient memory will severely limit performance of even a single user. In practice, there should be a cluster of servers, each running its own version of Amira. With a fixed amount of hardware, it becomes necessary to pay more attention to optimizing visualization and data processing algorithms, multitask scheduling and reducing the mutual influence among different instances. This last point concerns the problems encountered by two users simultaneously invoking Amira on a system that does not yet support off-screen rendering. Unless precautions are taken, the displays will interfere with each other.

The client GUI provides facilities to enter Amira commands or to select an Amira script file, which contains a sequence of commands. In either case, the commands are sent to the Amira server to be executed. A startup script, executed when Amira is instantiated, allows us to preload additional modules specific to WEB-IS2. We take advantage of this to overload certain commands. For example, “exit” has been overloaded to clean up our environment and to notify the client browser that the corresponding port number is “dirty” and should be returned to the

pool of available ports. In the future, we will provide an interface to allow users to use their own startup script, which would enable them to load modules with functionality unique to their particular tasks.

Script debugging is currently an unsolved issue. Amira does not send return data to the outside world. Therefore, debugging information will have to be sent to the client through the intermediary of a data file. There is no message returned if the script or command that is submitted to Amira has an error. After a predefined time period, a timeout event will occur, and the user can proceed. A possible solution to this problem is to construct a list of commands accepted by the system. Any command entered by the user not on this list will be rejected. A better solution would be to modify Amira to allow two way communications between it and the external world through a socket.

The snapshot of the Amira viewer can be retrieved and displayed in the client browser at any time. This feature allows users not only to examine the result of running a series of commands, but also to trace the status of a dynamic process if the script generates animations in the Amira viewer. Because the process of getting snapshots does not influence the execution of the script (it only interrupts it), the snapshot can be retrieved at anytime without concern for side effects.

A timeout option, controlled by the user, is provided to avoid indefinite periods of inactivity, possibly due to network congestions, server overload, etc. If the client does not receive a response from the Amira server within the timeout period, a message dialog pops up to the client screen and the session in progress is interrupted. This greatly reduces unnecessary resources on both the server and on the client.

5.2 Interaction through WDI

Users interested in generating statistical histograms based on the datasets being analyzed can choose the desired dataset in area *D* of WEB-IS2 and connect directly to WDI whose client interface is shown in Figure 4. WDI then communicates directly with the instantiated Amira executable. WDI is a web-based interrogative system that lets users analyze their data. The image from datasets generated by Amira can be sent to the client side of WDI to be manipulated further with the help of server-side computational modules. Results computed by the WDI server are sent directly to the WDI client interface.

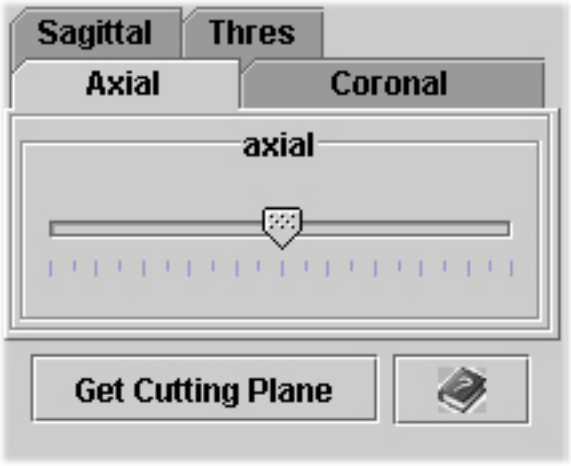

Figure 5. A snapshot of widgets in the interface of WDI program, they are involved with the interaction between WDI system and WEB-IS2.

Currently, Amira is only used to take cuts of a dataset along the *x-y*, *x-z*, or *y-z* coordinate directions, using the ortho module from Amira. The data is assumed to be defined on a Cartesian mesh. Alternative cutting options on more general grids are easily implemented since they are already provided by Amira. The entire process is controlled by a master python CGI script. Its functions are: 1) extract input parameters from URL, 2) send instructions to Amira to generate a slice, 3) take a snapshot in BMP format, 4) construct a color map, 5) convert the BMP image to a compressed JPEG format, 6) store the JPEG image in a location accessible by the WDI client, and 6) return the color map to the client. Selected details of these steps are described below.

A particular slice is determined by two directions (one of axial (*x-y)*, coronal *(x-z)*, or saggital *(y-z)*), and an index along the third coordinate direction (Figure 5). The index is controlled by the slider bar, whose values range from zero (minimum coordinate value) to 100 percent (maximum coordinate value). The slice information is sent to the master CGI script, which is passed on to Amira. The resulting BMP snapshot (uncompressed format) and color map generated by Amira are saved on the server, in a directory whose name is associated with the user. The color map contains five columns: an index, an RGB triplet, and the value of the variable of interest (e.g., temperature).

To convert the BMP image into a form usable by WDI, it is necessary to extract only the physical data. To this end, the background color in Amira is set to a uniform color not included in the color map. All lights in the scene are turned off to eliminate color gradation related to any effect other than the variable itself, and third, choose camera parameters to maximize the size of the image in the viewer. PIL is used to separate the background data from the actual slice data. A JPEG image is generated from this cropped data and is made available to the client for display. The client polls for the existence of the expected file every second. When available, it is uploaded.

The construction of the data array (necessary for the statistical calculation) requires an association of every pixel in the BMP image with a data value. A combination of a quicksort algorithm (Hoare 1962) with a binary search algorithms (Aho *et al.* 1983) were used to quickly construct this array. Unmatched pixels are assigned a special value to ensure that they are not considered during statistical processing.

The interface to WDI program specifically related to slicing is shown in Figure 5. There are four tabs. Three tabs refer to different directions of the cutting plane and the sliders below them specify the translation (in percentage units) of the cutting plane. A simple caching mechanism has been implemented to minimize redundant computations. Each time a new snapshot is taken by Amira, the bmp and JPEG files are stored on the server, and are given a name that encodes the dataset along with the particular slice number and direction. When the master CGI script receives the slicing information, it first checks for the existence of the requested slice. If it is available, Amira is not invoked, and the appropriate JPEG image is immediately available to the client. The threshold tab allows the user to specify a degree of uncertainty in the index slider position. For example, if the threshold value is set to 5 and the current translation value is 45, the program will consider any percentage value in the range of 40 to 50 as identical to 45. If a slice at a percentage value between 40 and 50 is already in cache, it will be used and Amira is not invoked. Once the number of stored slices exceeds 100, they are all deleted. More elaborate mechanisms will be developed in future work.

6. Future work

Datasets in the geosciences are becoming increasingly decentralized and distributed. Typically, due to size consideration, the data will remain in archival storage on the supercomputers where they will be generated. When the data sizes prevent them from being sent in their entirety to a processing server, it is necessary to perform some preprocessing and only transmit some subset to one or more visualization servers for display on one or more clients. These datasets generally take the form of extracts, or features the user is interested in, which

usually take a tiny fraction of the storage occupied by the original data (Post *et al.* 2003). Data is also stored in so-called data repositories (Houstis *et al.* 1999), centralized locations whose sole purpose is to maintain huge collections of data, possibly with software support capable of extraction subsets according to a variety of criteria). Data collected from satellites are also stored on Earth in centralized locations and made accessible through web interfaces, also called gateways or portals (http://ioc.unesco.org/oceanteacher/resourcekit/M3/Data/Datasets/EarthData-Pubs/MainServers.htm). There are many situations where a particular analysis might depend on multiple datasets stored at several locations, making it even more important to have the capability to decouple data storage from data analysis. It is, however, necessary that the data storage areas have computational capability to at least extract subsets of the data to send either to computational servers for processing (e.g., for feature extraction, cluster analysis, etc.) prior to the reduced data being sent to the visualization server, and ultimately to the client. Of course, the details of where the data migrates as several tasks are executed should be transparent to the user. To this end, we expect GRID toolkits to play an increasing role.

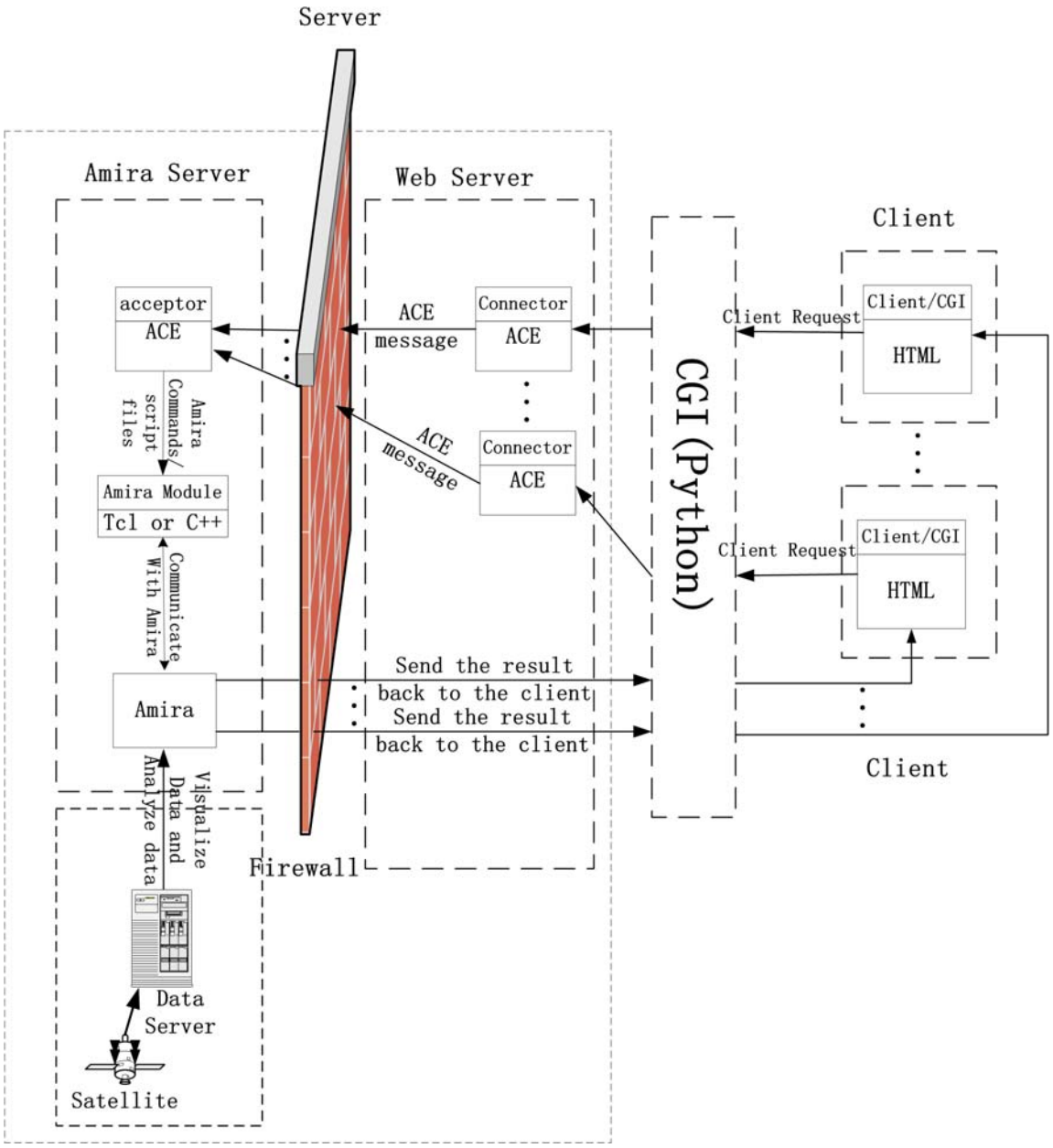

Figure 6. Functional setup for future WEB-IS2, which will decouple the Web server from the Amira and data servers. This will provide enhanced flexibility, higher security, and distributed computing.

In order to meet these complex and challenging requirements, we will decouple the Amira application from data storage from data preprocessing to achieve total distributed capability. Thus, one or more data servers will handle stored datasets and establish a parallel multi-threaded communication with the Amira server. This setup can definitely exploit the characteristics of computational grids, which link together distributed computers and consume their unused cycles for various types of calculations. Most importantly, much effort has gone recently into providing uniform web-based interfaces to these distributed resources (Fox *et al.* 2001, Pierce *et al.* 2002). Since visualization alone is not sufficient to improve our understanding of the very large data sets already at our disposal, it is necessary to couple visualization with advanced forms of data mining, cluster analysis, and other feature extraction techniques. In this regard, the Grid will play an

important role, allowing us to farm out the necessary work to available processors and access backend resources, such as data (Fox 2003). These will transform the data to a more useful form, which will be sent to the visualization servers prior to display on the clients.

Another planned generalization is the separation of the HTTP server from the Amira server. Thus, the Amira server could reside behind a firewall. Its data could be tunneled to a server outside the firewall using a port open for this purpose. Perhaps the ports could be rotated for additional security. We expect the generality of the ACE library to help us implement such a scheme. As a result, we will have enhanced security, and potentially the capability to use resources (inside firewalls) not otherwise accessible. Figure 6 shows the scheme for our future plan. It should be compared with the current implementation illustrated in Figure 3.

In the future we wish to improve the efficiency of our system from three directions:

1) Network transfer efficiency can be improved by better use of parallelism and more elaborate caching mechanisms and data compression techniques;

2) Data processing efficiency, which could be improved by specialized Amira modules written with Amira Developer and parallel algorithms; and

3) Client side interactive efficiency, which can be improved by providing a more friendly interface and updating the current version to implement some interactive graphics so that some kinds of data manipulation could be implemented locally with very few resources.

We wish to replace our HTML interface to Amira with a Java Applet, perhaps by upgrading WDI. This would provide more fluid interactions, rather be limited by the delays inherent in a stateless HTML protocol. Another area that merits additional work is the need to preprocess data stored on supercomputer to put them in a form suitable for visual analysis. Datasets of 1024^3 and beyond clearly cannot be sent to a visualization server. Solutions are required based on processing routines capable of executing on a variety of platforms. These could be stored in a database and migrated to the appropriate platform. Java provides facilities to accomplish this through reflection, which could greatly enhance the flexibility of any system that aims to provide a maximum amount of transparency to the users.

7. Conclusion

The datasets nowadays in geosciences are growing at what appears to be an exponential rate, which clearly justifies the need for visualization, feature extraction, and compact data representations (Erlebacher and Yuen, 2003). Furthermore, the sheer size of these datasets themselves prevents them being transferred from one machine to another efficiently. Paralleling this trend, visualization software, such as Amira, are increasingly complex and resource consuming. The software we developed allows huge datasets to be visualized and analyzed with Amira remotely over the Internet. Our system has already been used to explore large numbers of earthquake events processed through clustering techniques (Dzwinel *et al.* 2003, Garbow *et al.* 2003b). The design of our implementation using ACES will soon allow multiple users to visually collaborate by manipulating a single dataset with little concern for the client device at their disposal. These clients will only require a browser capable of displaying Java applets. As the deluge of data continues, innovative solutions that maximize ease of use without sacrificing efficiency or flexibility will continue to gain in importance, particularly in the Earth sciences. Major initiatives, such as Earthscope (http://www.earthscope.org), which will generate at least a terabyte of data daily, stand to profit enormously by a system such as WEB-IS2.

8. Acknowledgments

The authors would like to thank Larry Hanyk at Charles University, Prague for useful feedback on the contents of the paper. They also thank the Department of Energy and the National Science Foundation (divisions of Geophysics and CISE) for their support.

References

Aho, A.V., Hopcroft J.E., and Ullman, J.D., Data Structures and Algorithms, 1983.

Berman, F., Fox, G., and Hey, A.J.G. (Editors), Grid Computing: Making the Global Infrastructure a Reality, Wiley Series in Communications Networking & Distributed Systems, 1013 pp., John Wiley & Sons Ltd., 2003.

Brown, C.W. and Shepherd, B.J., Graphics File Formats reference and guide, 1995.

Dubuffet, F., Yuen, D.A., Murphy, M.S., Sevre, E.O., and Vecsey, L., Secondary Instabilities developed in upwellings at high Rayleigh number convection, in EOS, TRANS, AGU, 2001, Vol. 82, No. 47, pp. F210.

Dzwinel, W., Yuen, D.A., Kaneko, Y.J.B.D., Boryczko, K., and Ben-Zion, Y., Multi-resolution clustering analysis and 3-D visualization of multitudinous synthetic earthquakes, Geosciences, Vol. 8, pp. 12-25, 2003. http://link.springer.de/link/service/journals/10069/contents/tfirst.htm .

Erlebacher, G. Yuen, D.A., and Dubuffet, F., Case Study: Visualization and Analysis of High Rayleigh Number -- 3D Convection in the Earth's Mantle. Proceedings of IEEE Visualization, pp. 529-532, 2002.

Erlebacher, G. and Yuen, D.A., Web-Based Visualization Services for the Geosciences, American Geophysical Union, Fall Meeting Talk, 2002, http://www.agu.org/ .

Erlebacher, G., Yuen, D.A., and Dubuffet, F., Current trends and demands in visualization in the geosciences. Electronic Geosciences 4, 2001, http://link.springerny.com/link/sevice/journals/-10069/technic/erlebach/index.htm .

Erlebacher, G. and Yuen, D., A Wavelet Toolkit for Visualization and Analysis of Large Data Sets in Earthquake Research, Pure and Applied Geophysics, 2003, http://www.csit.fsu.edu/~erlebach/publications/erlebach_yuen_maui2_2002.pdf .

Fox, G. C., Ko, S.-H., Pierce, M., Balsoy, O., Kim, J., Lee, S. ,Kim, K., Oh, S., Rao, X., Varank, M., Bulut, H., Gunduz, G. ,Qui, X., Pallickara, S., Uyar, A., and Youn, C., Grid Services for Earthquake Sciences, ACES 2001, Special Issue of Concurrency and Computation: Practice and Experience, 14/6-7, 371-393, 2002.

Fox, G.C., Grid Computing Environments, Computing in Science and Engineering, High-Dimensional Data, Vol. 5, No. 2, 2003.

Garbow, Z.A., Olson, N.R., Yuen, D.A., and Boggs, J.M., Interactive Web-Based Map: Applications to Large Data Sets in the Geosciences, Electronic Geosciences, Vol. 6, 2001. http://link.springer.de/link/service/journals/10069/papers/webbased/index.htm .

Garbow, Z.A., Erlebacher, G., Yuen, D.A., Boggs, J.M., and Dubuffet, F., Web-Based Interrogation of Large-Scale Geophysical Datasets from Handheld Devices, Visual Geosciences, Vol. 8, 2003a.

Garbow, Z.A., Erlebacher, G., Yuen, D.A., Bollig, E. and Kadlec, B., Remote Visualization and cluster Analysis of 3-D Geophysical Data over the Internet Using Off-Screen Rendering, in press, 2003b.

Hoare, C.A.R.: Quicksort. The Computer Journal 5(1): 10-15, 1962.

Houstis, C., Nikolaou, C., Lalis, S., Kapidakis, S. and Christopides, V., Towards a next generation of open scientific data repositories and services, TR98-0219, 1998, http://citeseer.nj.nec.com/235622.html .

Morse W., Petroleum, B., A Tcl/Tk and Expect Tutorial, Landmark Graphics Corporation World Wide Technology Conference, December 1, 1994.

NCSA HTTPd Development Team, The Common Gateway Interface, 2003, http://hoohoo.ncsa.uiuc.edu/cgi/intro.html .

Pierce, M., Youn, C., and Fox, G., Interoperable Web Services for Community Portals, SC02, November 2002.

Post, F.H., Vrolijk, B., Hauser, H., Laramee, R.S., and Doleisch, H., Feature extraction and visualization of flow fields, Eurographics 2002, http://www.cs.uct.ac.za/Research/CVC/-EuroGraphics2002/dl/conf/eg2002/stars/s4_flowvis_post.pdf .

Schmidt, D., Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects, John Wiley & Sons, 666 pages, 2000.

Stegmaier, S., Magallon, M. and T. Ertl, A Generic Solution for Hardware-Accelerated Remote Visualization, Joint Eurographics – IEEE TCVG Symposium on Visualization, 2002.

Yokokawa, M., Itakura, K., Uno, A., Ishihara, T. and Kaneda, Y., 16.4-Tflops Direct Numerical Simulation of Turbulence by a Fourier-Spectral Method on the Earth Simulator, 2002, http://www.ultrasim.info/sc2002-273.pdf .