13.19

THE WEATHER VISUALIZER, JAVA™, HABANERO™, AND THE FUTURE

Joel Plutchak*, Robert B. Wilhelmson, Mohan K. Ramamurthy, Steven E. Hall, and Vladimir Tokarskiy
University of Illinois at Urbana-Champaign, Urbana, Illinois

1. INTRODUCTION

The Department of Atmospheric Sciences at the University of Illinois Urbana-Champaign has developed a web-based visualization tool known as The Weather Visualizer (DAS, 1997), which allows users to interactively select and customize the presentation of meteorological data and images.—

Since its debut in 1995, the goals of the various versions of the Weather Visualizer have remained fairly consistent:

- Provide reasonably fast access to customizable user requests
- Present a useful yet manageable set of choices for data type and options
- Incorporate appropriate explanatory or tutorial information to complement the data
- Use a framework that is easily obtainable and familiar to a wide range of users

While these goals have largely been accomplished, computer and network technologies have been advancing at a rapid pace, allowing newer and better ways to meet the goals. Therefore, Weather Visualizer development has been proceeding along two lines, one using conventional World Wide Web (WWW) technology and another using object-oriented, distributed technology.

2. CONVENTIONAL IMPLEMENTATION

The initial, conventional implementation of the Weather Visualizer (Ramamurthy et al., 1996) uses standard WWW technology: HTML documents and forms displayed on the client computer coupled with Common Gateway Interface (CGI) scripts that run on the server computer.

The introductory WWW document contains categories of data from which the user chooses, e.g., surface observations, forecast products, upper air soundings, etc. Each option is linked to either further data subcategories or a form from which relevant options for a specific data set can be selected. Once the desired options are selected and the choices are submitted, a CGI script runs on the server. The script constructs the requested data product and returns it in the form of an HTML document. For example, from the surface observation page, a user might choose to view a map of the southwestern portion of the United States, with current temperature, cloud symbols, weather symbols, and dew point plotted for major cities, add pressure contour lines, and plot it all on top of a current infrared satellite image. A script on the server would get the request, execute the programs necessary to satisfy the request, and return an image having the requested elements embedded in an HTML document.

An important facet of the HTML documents are hyperlinks to explanatory and tutorial information (Hall et al., 1996a). These come in two basic forms. The WWW forms for each data category contain hyperlinks explaining each of the available options for that data set, as well as hyperlinks to related topics such as the definition of Greenwich Mean Time. The other type of explanatory information is in the form of hyperlinks from the returned data products. Clicking on the legend of an image product accesses information pertaining to the appropriate legend item. For example, the surface report image described above features a legend for surface contours, fronts, and radar summary. Clicking on the radar portion of the legend brings up an explanation of radar, as well as how to interpret a radar summary plot.

There are two significant limitations to the conventional WWW/CGI approach. One is that interactivity in user choice is quite limited. User input cannot be verified until submission of a request, so if, for example, data is not available for a given choice, the user is not informed until after submitting the request and waiting for a response from the server. Depending on network connectivity,
network load, and server load, the time lag can be substantial.

The most significant drawback to this approach is that since each request spawns a computationally intensive process, the ability to handle requests is limited by the computational capacity of the server. Experience has shown that under normal load, a single server (in this case a Hewlett-Packard 9000/715

* *Corresponding author address:* Joel Plutchak, Dept. of Atmospheric Sciences, Univ. of Illinois at Urbana-Champaign, 105 S. Gregory St., Urbana IL 61801; email <plutchak@uiuc.edu>.

workstation) can handle 10-20 simultaneous requests. During significant weather events, the large number of requests can easily saturate the server, adversely impacting system responsiveness.

Many solutions to the server saturation problem have been investigated, e.g., adopting the FastCGI interface, use of multiple servers, static allocation and control of system resources, etc. However, it is recognized that the conventional WWW/CGI method will always be limited by the capabilities and capacity of the server, and fails to take advantage of the computing resources of the client computer.

3. IMPLEMENTATION USING JAVA

The release of Sun Microsystems' Java programming language and environment (Sun, 1997) in early 1996 has significantly changed the rules for WWW programming. Java allows the creation of *applets*— small programs that are downloaded and executed on the client system as they are needed. This capability simultaneously presents a solution to the server saturation problem and allows a higher level of user interactivity with the programs and data.

The server load problem is the most obvious advantage the use of Java brings. In one scenario, the program itself is transmitted once, while data is transmitted as it is needed and then manipulated locally on the client. Since simple file transmission (whether Java applets or meteorological data) is all that occurs from the server viewpoint, the load on the server is greatly diminished in comparison with the conventional CGI implementation.

Enhanced interactivity has been a key goal for the Weather Visualizer (Wojtowicz et al., 1996). The use of Java enhances interactivity in several ways. Data input and verification can be done on the client side by the Java program, so those conditions can be detected and corrected without the need for a network data transfer and response from the server. Also, once the requested data is transferred to the Java applet, it can be manipulated by the user with no further network transmission and resulting time lag. Furthermore, embedded information can be transmitted along with data. For example, by sending navigation and calibration modules along with a satellite image, the applet itself can display information about the image in meaningful units such as reflectivity or temperature for image pixel values and latitude/longitude for image locations.

The applets that have been developed to date take advantage of the above features (Hall et al., 1996b). One applet is used to obtain an interactive weather report. A series of current images is transmitted and assembled under user control on the client. The images used are provided as parameters to the applet, so any data that can be displayed as an image can be used with the applet. The initial set of data consisted of surface observations, three types of satellite images, radar summaries, frontal analysis plots, isobars, and isotherms. Since the data is time sensitive, images are updated automatically as newer data reaches the server. Geographic location is computed on the client system, as is cloud top temperature from an infrared satellite image. The applet also allows access to additional data from the server, displaying (both graphically and in tabular form) forecast data and current surface observations at any point on the displayed map. Finally, the applet contains hyperlinks to the same tutorial and explanatory information that is available from the conventional visualizer.

Another applet provides access to and animation of sequences of satellite image products generated for the department's *Daily Planet*™ WWW server (DAS, 1997b). The applet allows user control over animation of the image sequences, including specific image selection and animation speed, options that more conventional WWW animation techniques do not allow.

Currently under development are small general-purpose applets which are meant to be liberally but non-intrusively deployed within more conventional WWW documents. This will allow those who have the desire and capability to enrich their WWW browsing by using Java to do so, while not distracting those who wish to rely on the more conventional WWW functionality. These applets will implement common scientific needs— data plotting with interactive control of the display, viewing images with embedded information, the ability to subset or zoom while viewing data, etc.

Additional ongoing work with Java involves a further shift of computation from the server to the client. Rather than downloading data formatted as images, the source data itself can be transmitted and manipulated appropriately on the client. This will not only lessen the computation needed on the server, but allow the user more flexibility in formatting the information on the client system.

One such technique under development involves resolution-appropriate data transfer. A full resolution GOES data set combined with a vector map database and a meteorological station database is being used for testing the concept. The user will be presented with a greatly reduced version of the full image, with an appropriately sparse depiction of station locations and map boundaries. A region of interest would then be selected by zooming or otherwise selecting a subset of the image. The display of the zoomed data would take place on the client until the viewed resolution became degraded. At that time, a higher-resolution subset of the full image would be requested from the server, allowing a beneficial tradeoff between data download time and access to high resolution data. Simultaneously, the density of stations plotted as well as the granularity of the map boundaries would be adjusted appropriately. Starting with a quick-loading sub-sampled full hemisphere GOES image with continental boundaries depicted and a handful of stations plotted, the user could zoom down to, for example, Chicago and

surrounding suburbs, with county boundaries and local highways plotting on a 1-km resolution subset of the GOES image.

Java brings an additional strength to providing WWW functionality. As an object-oriented programming language, Java facilitates writing modular reusable objects, which can be used as the building blocks for subsequent related efforts. These modular objects also lend themselves particularly well to distributed programming, since they are by nature self-contained entities which can be easily transmitted over the network.

4. HABANERO

Although not an initial goal of the Weather Visualizer, we found that the ability to interactively collaborate with individuals at potentially far-flung locations on the Internet was a logical next step. The target use for such a capability is in a classroom setting, with a mentor from the meteorological community providing a real-time weather briefing for students in a distant city. Rather than verbally informing the students to bring up a certain image, the mentor would control a master application with slave applications on the students' desktops, allowing each student to see exactly what the mentor is viewing and describing.

The ability to add collaborative capability was made much simpler by the release of Habanero (NCSA, 1997) by a group at the National Center for Supercomputing Applications (NCSA). Habanero is a framework for sharing objects with colleagues distributed around the Internet. Included are all the networking facilities, routing, arbitration and synchronization mechanisms necessary to accomplish the sharing of state data and key events between collaborator's copies of a software tool. The Habanero project is investigating the enhancements in distributed interpersonal communication made possible when single-user computer software tools are recast as multi-user, collaborative work environments. With Java at the forefront of Internet application development, the Habanero group chose to provide the collaborative capability as a set of Java classes and objects.

Casting Habanero in terms of Java classes made adapting the Java version of the Weather Visualizer to the collaborative framework a natural and obvious choice. Indeed, the interactive weather report applet was adapted as a featured demonstration application for the premier release of Habanero. With a few modifications to the original applet, the scenario presented above became a reality. Further development along this path will depend upon feedback from the initial users.

Java Share, a package from Sun Microsystems which is scheduled for release sometime in 1997, will also provide collaborative capability to Java applications.

5. THE FUTURE

Technology in the computer science sub-disciplines of networking, object-oriented programming, and distributed systems are advancing rapidly, and will continue to present better solutions to the goals of a tool such as the Weather Visualizer. We believe the best way to take advantage of such advances is to develop tools in a modular and forward-looking way.

As discussed earlier, Java provides an excellent environment for creating interactive WWW applications. However, it is still an immature language. Several planned extensions to the Applications Programming Interface (API) have potential for incorporation into the Weather Visualizer. The Java Media APIs will encompass 2-dimensional, 3-dimensional, and animation packages. Support for 2-dimensional graphics and imaging is primitive in the current release, and 3-dimensional and animation capabilities are not directly supported. The addition of these APIs will remove much of the complexity currently needed to implement basic graphical and imaging functions, and allow the use of Virtual Reality (VR) for visualizing data from Java programs. Use of VR and other extended graphics capabilities is planned for future versions of the Weather Visualizer.

Work done by the GeoLens project (USDAC, 1997) also holds great interest for use with the Weather Visualizer. Part of the project, known as GeoHarness, involves providing a standard framework for describing and locating earth science data sets and images. Using GeoHarness, the Weather Visualizer could query the user for geographic location, sensor type, etc., connect to a GeoHarness server to locate candidate data sets, and finally display the selected data. This would open up use of the application to a wide range of data, greatly increasing its utility.

Another area which could open up the possibilities of the Weather Visualizer involves other projects using distributed applications and object-oriented design. Instead of having a program running on a single machine with a copy of data that may have existed elsewhere, distributed objects and remote method invocations allow an application running locally to transparently access data and execute procedures that exist on other systems on the network. One would in effect transform the current standalone applets into part of a larger and more flexible distributed system. For example, an image represented by a remote object would be formatted or calibrated using a subroutine that exists on yet another system, displaying the result in a Weather Visualizer applet– with little additional work on the part of at applet. CORBA, the Common Object Request Broker Architecture, is one standard being developed that defines mechanisms for providing interoperability between applications on heterogeneous distributed environments. Sun Microsystems' forthcoming Remote Method Invocation (RMI) and

Object Serialization Java APIs also hold promise in this area.

Whatever the future holds, we will continue to make use of emerging technologies and remain positioned to respond to further advances.

6.ACKNOWLEDGEMENTS

The initial version of the Weather Visualizer as well as the development of the tutorial/explanatory information was performed under the Collaborative Visualization project, which is funded by NSF Grant #RED-9454729. The remainder of the work described was developed under the HORIZON Project, funded by NASA Information Infrastructure Technologies and Applications (IITA) Grant CAN-OA-94-1.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

NCSA and Habanero are proprietary trademarks owned by the Board of Trustees of the University of Illinois.

7. REFERENCES

Dept. of Atmospheric Sciences (DAS), cited 1997a: The Weather Visualizer. [Available on-line from <http://covis.atmos.uiuc.edu/covis/visualizer/>].

——, cited 1997b: The Daily Planet™. [Available on-line from <http://www.atmos.uiuc.edu/>].

Hall, S. E., Sridhar, M., Ramamurthy, M. K., and Wilhelmson, R. B, 1996a: The Design and Implementation of Multimedia Web-based Instructional Modules in K-12 Education. Preprints, *Fifth Symposium on Education*, Atlanta GA, Amer. Meteor. Soc., 128-131.

——, Ramamurthy, M. K., Wilhelmson, R. B., Plutchak, J., Wojtowicz, D., and Sridhar, M., 1996b: The Weather Visualizer: A Java Tool for Interactive Learning. *Proceedings of the First Symposium on Education*, Lincoln, Nebraska, Int. Geosci. and Remote Sensing Soc., 1498-1500.

NCSA, cited 1997: NCSA Habanero. [Available on-line from <http://www.ncsa.uiuc.edu/SDG/Software/Habanero/HabaneroHome.html>.]

Ramamurthy, M. K., Wilhelmson, R. B., Hall, S. E., Plutchak, J., and Sridhar, M., 1996: CoVis Geosciences Web Server: An Internet-based Resource for the K-12 Community. *12th International Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology*, Atlanta GA, Amer. Meteor. Soc., J27-32.

Sun Microsystems, Inc., cited 1997: Java™ – Programming for the Internet. [Available on-line from <http://java.sun.com/>.]

USDAC, cited 1997: Geolens Technical Information. [Available on-line from <http://usdac2.rutgers.edu/project/tech_info.html>.

Wojtowicz, D., Wilhelmson, R. B., Ramamurthy, M. K, 1996: IICE: Bringing Interactivity to Image-based WWW Products. *12th International Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology*, Atlanta GA, Amer. Meteor. Soc., 413-417.