Science Information Systems Newsletter, Vol. IV, No. 40, 1997.

Framework for Collaborative Steering of Scientific Applications

Beth Schroeder* Greg Eisenhauer Karsten Schwan
Fred Alyea Jeremy Heiner Vernard Martin
Bill Ribarsky Mary Trauner Jeffrey Vetter
Ray Wang Song Zou

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332

1 Interactive Computational Instruments

Difficult computations can be made more effective and efficient if users can easily contribute to the problems these computations address. This article's purpose is an introduction to the potential increases in functionality and performance gained by the online interaction of end users with high performance *computational instruments* on single and on networked parallel machines (also see [7]). Namely, we consider systems in which users interact with computations as if they were physically accessible laboratory instruments and in which entire *distributed laboratories* are constructed from sets of such computational instruments. Within this context, our intent is to facilitate both (1) online interactions with single computational instruments and (2) interactions among multiple scientists and multiple instruments located at physically distributed sites where scientists may have dissimilar areas of expertise.

With our research and with the larger-scale *Distributed Laboratories* project at the Georgia Institute of Technology[3], we aim to improve the state of the art of interactive high performance computing for parallel and distributed applications on the variety of heterogeneous platforms now in common use by HPC users and researchers. Particularly, our goal is to develop a general framework for enhancing the interactivity of high performance applications. This framework:

- contains general interactivity (i.e., online monitoring and steering) mechanisms with which high performance applications may be inspected and/or steered at runtime by algorithms, human users, or both;
- supports interchangeable visualizations run across heterogeneous and distributed hardware platforms, using a robust and portable data meta-format for transporting visualization content;
- supports the simultaneous interaction of multiple scientists with single large-scale computations; and
- permits collaboration via multiple computational instruments among scientists working in separate locations.

In the remainder of the article we introduce one of the parallel and distributed scientific applications used in our research. We then explore interactions with a single computational instrument during its execution, typically referred to as *interactive program steering*†. Finally we discuss the use of multiple computational instruments by sets of end users, thereby moving from issues addressed in previous work toward interesting topics in future research.

*The Email is: beths@cc.gatech.edu

†Interactive steering may be defined as the 'online configuration of a program by algorithms or by human users, where the purpose of such configuration is to affect the program's execution behavior'.

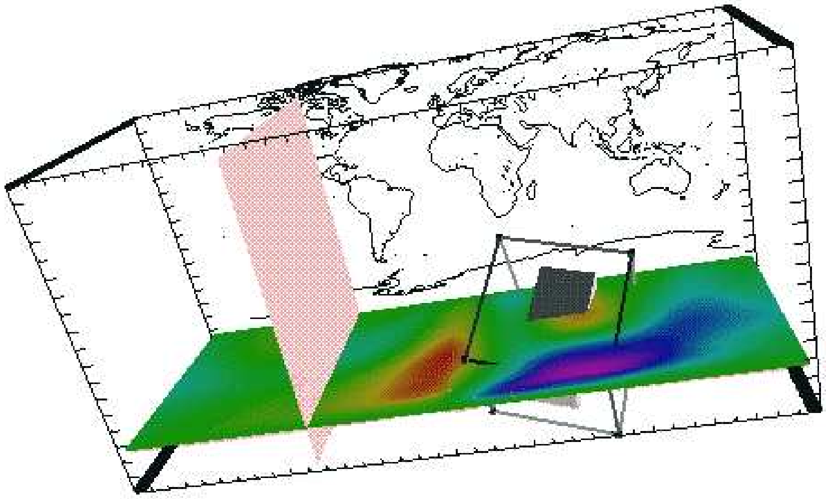

Figure 1: Three-dimensional visualization of atmospheric modeling data.

2 Atmospheric Modeling

In collaboration with atmospheric scientists at Georgia Tech, we have developed a parallel and distributed global chemical transport model. This model uses assimilated windfields (derived from satellite observational data) for its transport calculations, and known chemical concentrations also derived from observational data for its chemistry calculations. Models like these are important tools for answering scientific questions concerning the stratospheric-tropospheric exchange mechanism or the distribution of species such as chlorofluorocarbons (CFCs), hydrochlorofluorocarbon (HCFCs) and ozone. Our model contains 37 layers, which represent segments of the earth's atmosphere from the surface to approximately 50 km, with a horizontal resolution of 42 waves or 946 spectral values. The model's implementation uses a spectral approach to solve the transport equation for each species. Details of the model's solution approach, parallelization, performance results, and future plans concerning model development are described in [5].

A sample interface for collaborative model steering, depicted in Figure 1, displays a latitudinal and longitudinal slice of the distribution of N^2O atoms in the atmosphere. The highest concentrations of N^2O are represented in red; the lowest in blue. Steering using this interface is discussed in Section 3.3.

3 Interactivity Framework

Figure 2 depicts a general interactivity framework for a complex high performance application. The Distributed Laboratories project includes support for all aspects of such applications, including visualizations, steering interfaces, data communication and analysis middleware, collaboration support, and application monitoring and steering support. Before discussing the general issues in distributed collaborative applications we focus more narrowly on the framework required to support a single user interacting effectively with a single computational instrument. Specifically, we consider:

- the support required to efficiently monitor system- and application-level behavior in parallel and distributed programs;
- frameworks and support for steering such a program though external interfaces; and
- visualization interfaces for both monitoring and steering.

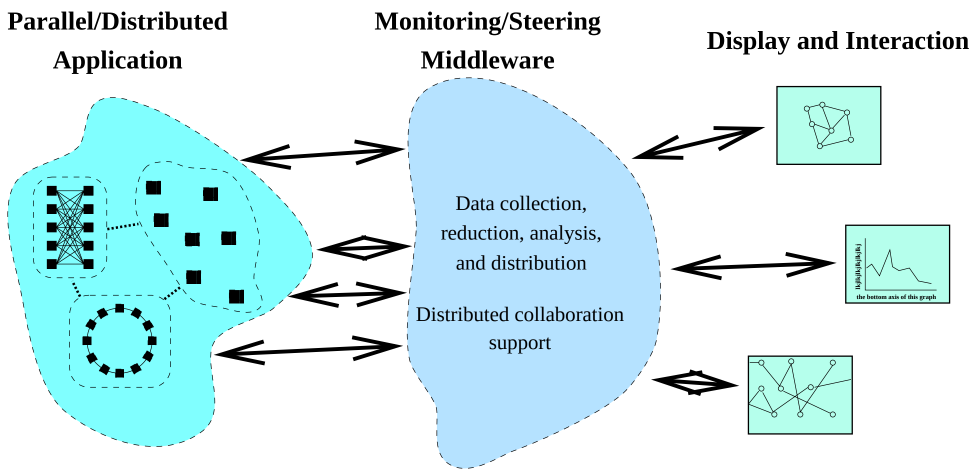

Figure 2: General Interactivity Framework

3.1 Monitoring

Interacting with computational instruments in terms meaningful to end users requires online monitoring – the dynamic gathering of application-specific information from an instrument as it executes. Falcon [4] is a set of tools and libraries supporting online, application-specific, event-based monitoring of parallel and distributed applications. Falcon consists of a sensor specification language and compiler for generating application sensors, and one or more local agents for online information capture, collection, filtering and analysis of event data. In addition, Falcon uses intermediate monitoring/steering middleware to disseminate monitoring events to the potentially large number of clients that wish to interact.

Falcon's implementation is based on a local agent, usually residing on the monitored program's machine, to capture event data. On shared memory architectures, this local agent may be an additional thread operating in the instrument's address space. Local agents, due to their proximity to the computational instrument, can gather monitoring data quickly while minimizing interference with the instrument's operation. For physically distributed instruments, multiple local agents are employed and each of the instrument components is treated as a separately monitored entity.

3.2 Interactive Steering

A steering system used by external agents must have the following functionality: (1) receiving the computational instrument's runtime information from the online monitoring system, (2) analyzing and then displaying the information to the end user or submitting it to a steering agent, (3) accepting steering commands from the user or agent, and (4) enacting those commands to affect the application's execution. In Falcon's implementation of computational steering, at least one local agent runs on the target machine with the application. This local steering agent performs any steering actions requested by external agents. External agents are driven by monitoring data and may request steering actions directly based on this data or they may support graphical interfaces interfaces and request steering actions in response to user manipulations.

One insight from our experience with interactive steering is that computational instruments differ in terms of the ease with which certain steering actions may be implemented. An instrument's implementation may permit some of its internal variables to be inspected and steered continuously, with little additional instrumentation of the code. In general, however, potential improvements in performance or functionality by the addition of steering depend largely on an instrument's implementation and on the steering and monitoring actions required.

Another insight from our experiences is that program steering must consider overheads not relevant to performance monitoring, which tends to focus on the effects of program perturbation. These overheads are: (1) the perturbation to the application due to instrumentation for monitoring and steering, (2) the latency of the monitoring to enactment feedback loop, and (3) the costs of decision making part of this latency. Specifically, for steering, the end-to-end latency of the monitoring to enactment feedback loop is a critical performance constraint when steering actions cannot be based on 'stale' monitoring data, or when such actions become inappropriate after some future program state has been passed.

3.3 Application-specific Visualization/Steering Interfaces

While monitoring and steering support are essential for efficient, low-impact interactivity, it is often the final visual interface that determines the success and effectiveness of interactivity. The interactive 3D visualization of atmospheric modeling data shown in Figure 1 is constructed and used with a set of modules from the GlyphMaker display and steering system [6] (originally constructed with the SGI Explorer environment, and now being supported in the OpenInventor framework). These modules implement the functionality required by certain application-specific displays, in this case modules that convert monitoring data extracted from the computational instrument from its spectral form to a gridded form more suitable for visualization, and a module acting as a reader for converting the data being displayed to be printed on high the high resolution output devices used by atmospheric scientists (using the PV-Wave visualization system).

Using the 3D visualization shown in Figure 1, the atmospheric model is steered by first positioning the latitudinal and longitudinal planes, sizing and moving a rectangle to intersect a plane, then entering a specific desired concentration increment/decrement. The resulting set of new concentration values is forwarded from the visualization interface to the computational instrument via the aforementioned monitoring/steering infrastructure. The new concentration values are used as part of the next timestep taken by the model to result (hopefully) in improved model behavior.

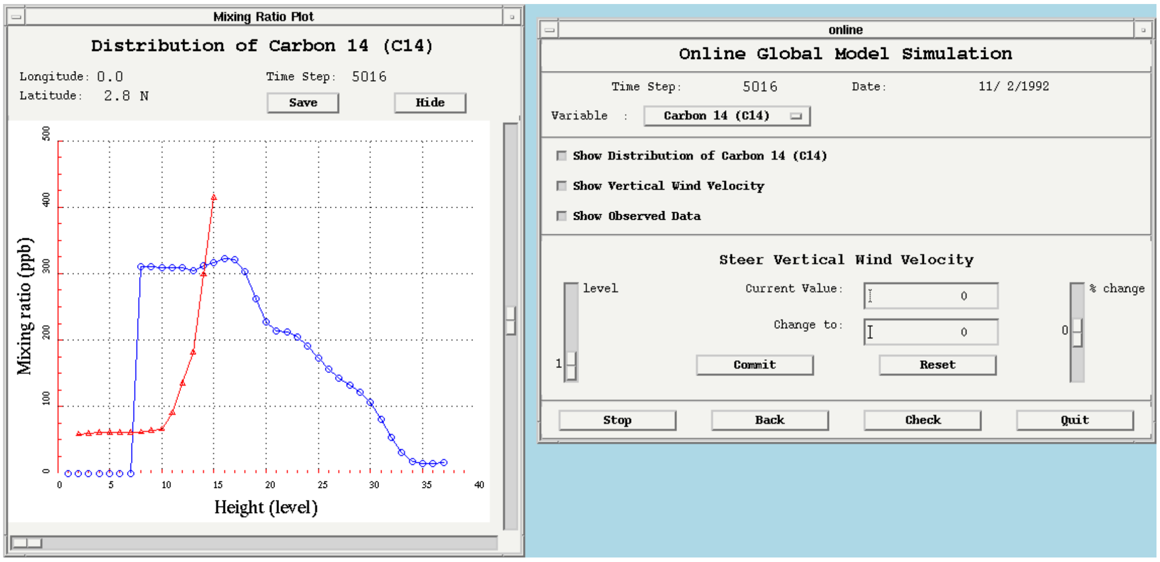

Figure 3: An application-specific display for online control of the atmospheric modeling code.

In contrast to the 3D visualizations providing excellent overviews of model behavior, a complementary 2D steering interface operating with subsets of the atmospheric modeling application's data is shown in Figure 3. This interface's display presents the distribution of C^{14} at the single latitude of 2.8° N. It has two logical parts: one for showing both the computed and the observed concentration values of C^{14} atoms in air to the end user, and the other for accepting steering requests from the user. The computed results of the C^{14} distribution are represented by the blue plotted curve from atmospheric layer 0 to 37, which is updated for every model time step. The concentration of C^{14} actually observed at this point is represented by the red plot. When noticeable discrepancies between the calculated values and the observed values are detected, the user can dynamically modify the application execution to 'correct' the computations.

4 Toward Distributed Laboratories

Distributed laboratories are environments where scientists and engineers working in geographically separated locations share access to interactive visualization tools and large-scale simulation computations, share information generated by such instruments, and collaborate across time and space to evaluate and discuss their results. The intent is to permit scientists, engineers, and managers at geographically distinct locations (including individuals telecommuting from home) to combine their expertise in solving shared problems by allowing them to simultaneously view, interact with, and steer sophisticated computation instruments executing on high performance distributed platforms.

The underlying, enabling technologies needed to support distributed laboratories research include:

- dynamic monitoring, adaption, and interactive steering of high performance computations, as discussed in Section 3;
- interconnectivity and data exchange infrastructure; and
- collaboration and shared visualization technologies.

Novel communications and data analysis middleware facilitating online monitoring and interactive steering in a multi-scientist/multi-instrument environment are discussed in the following sections.

4.1 Communications and Data Analysis Middleware

A characteristic of Distributed Laboratories is that tools and experimenters may come and go dynamically and may be interested in different types of data at different times. Additionally, the various programs that cooperate to make up a Distributed Laboratory may not all be under the control of a single group, compiled by the same compiler or written in the same language. These characteristics impose needs on the communications and data analysis middleware that are addressed by two novel communication libraries, called DataExchange and PBIO. These libraries provide a communications infrastructure that allows instruments, data analysis tools and interactive client displays and visualization displays to be plugged into the system dynamically. First, the PBIO (Portable Binary Input Output)[1] library supports the transmission of binary records between heterogeneous machines. PBIO is essentially a data meta-representation. Users register the structure of the data they wish to transmit or receive, and PBIO transparently masks the representation differences across heterogeneous machine architectures. In particular, PBIO handles differences in the sizes, locations and even basic types of the fields in the records to be exchanged.

While PBIO supports exchanging data between two clients, the DataExchange library[2] layered on top of PBIO provides support for establishing communication between agents, for resolving differences between data formats used by multiple agents, for forwarding data from agent to agent, and for processing data within an agent. DataExchange allows application handler functions to be bound to the arrival of new data so the DataExchange library augmented with a few application functions can thus serve as a configurable data filter, dramatically simplifying the task of creating networks of cooperating agents that gather, analyze and distribute the data required for displays.

4.2 Collaboration

A goal of Distributed Laboratories is to provide a mechanism for end users to jointly manipulate shared instruments. This goal is being realized in joint work with visualization and CSCW researchers, where we are developing (1) mechanisms for coordinating what is rendered on separate machines and (2) abstractions for manipulating shared complex entities. Our goal is to create a framework and collaboration library with which application-specific interaction and collaboration abstractions are easily constructed, thereby removing some of this burden from the developer. The method we use to attain this goal is to add means for constructing relevant 'interactors' as part of the data visualizations being employed by end users.

5 Conclusions

The Distributed Laboratories project aims to construct an infrastructure with which future scientists and engineers can interact with each other and with their computational instruments as if they were physically co-located in a single laboratory. This paper reports on some specific research efforts being undertaken at Georgia Tech that address the topic of interactive high performance programs:

- The Falcon steering and monitoring tools and infrastructure are used in the online observation and manipulation of scientific computations.
- The DataExchange and PBIO middleware serve to transport the events and their contents used when observing high performance applications and when steering them.
- The visualization support provided by the GlyphMaker tool permits the definition of appropriate visual abstractions and their efficient representation on 3D graphical displays.
- Collaboration infrastructure and abstractions are provided using the OpenInventor graphical display framework.

References

[1] Greg Eisenhauer. Portable self-describing binary data streams. Technical Report GIT-CC-94-45, College of Computing, Georgia Institute of Technology, 1994. http://www.cc.gatech.edu/tech_reports.

[2] Greg Eisenhauer and Beth Schroeder. The DataExchange library. Technical Report GIT-CC-96-17, Georgia Institute of Technology, 1996. http://www.cc.gatech.edu/tech_reports.

[3] Richard Fujimoto, Karsten Schwan, Mustaq Ahamad, Scott Hudson, and John Limb. Distributed laboratories: A research proposal. Technical Report GIT-CC-96-13, Georgia Institute of Technology, 1996. http://www.cc.gatech.edu/tech_reports.

[4] W. Gu, G. Eisenhauer, E. Kraemer, K. Schwan, J. Stasko, J. Vetter, and N. Mallavarupu. Falcon: On-line monitoring and steering of large-scale parallel programs. Report GIT-CC-94-21, College of Computing, Georgia Institute of Technology, 1994.

[5] Thomas Kindler, Karsten Schwan, Dilma Silva, Mary Trauner, and Fred Alyea. Parallelization of spectral models for atmospheric transport processes. Technical Report GIT-CC-95-17, Georgia Institute of Technology, College of Computing, Atlanta, GA 30332-0280, December 1994. http://www.cc.gatech.edu/tech_reports.

[6] W. Ribarsky, E. Ayers, J. Eble, and S. Mukherjea. Glyphmaker: creating customized visualizations of complex data. *Computer*, 27(7):57–64, 1994.

[7] Beth Schroeder, Greg Eisenhauer, Karsten Schwan, Jeremy Heiner, Peter Highnam, Vernard Martin, and Jeffrey Vetter. From interactive applications to distributed laboratories. *submitted to IEEE Computational Science and Engineering*, June 1996.