

AVS: Visual Interface for Parallel Applications

Ian Curington
Advanced Visual Systems, Inc., UK

1 INTRODUCTION

Visual Application development including control of multiple heterogeneous architectures is discussed, with illustrations of coupling numerical analysis such as Computational Chemistry and Visualization. Parallel processing enhances the performance of application systems, exploiting modern computer architecture and networks. The extended complexity of parallel applications require the development of new programming and user interface methodologies. AVS provides application control over parallel processes residing on either local processors of a shared memory multiprocessor system, networks of interconnected uniprocessors, or advanced MPP architectures.

2 VISUAL COMPUTING

Visual Computing refers to the control of computers by direct manipulation of visual representations of appropriate tools and is a natural paradigm for the use of computers by people. The same elements of interactive graphics and user control of computation developed for data visualization can be applied in many areas, including the control of parallel processing. Application developers are able to build an object oriented application using icons to represent those objects (AVS Technical Overview, '92).

The Application Visualization System (AVS) is the leading 3D, multi-platform, data visualization environment, and the only one to include a complete data flow model with distributed parallel control with a visual specification interface allowing both synchronous and asynchronous process control. AVS provides a comprehensive set of development tools for the delivery and implementation of parallel applications.

The Visual Application Development environment of AVS contains several subcomponents:

Network Editor:	Visual Programming Tool
Module Generator	Integration Tool for User Code
Layout Editor	Graphical User Interface Design Tool
Command Language	Scripting Control and Callable API

Visual Application Development has two primary goals. The first goal is to shorten the time needed to take an application from prototype stage to final delivery of the completed application. The second goal is to support both application prototyping and production application development environments.

3 GRANULARITY: AVS MODULE ARCHITECTURE

The fundamental computation unit in AVS is the *module*. Modules are either C, C++, or FORTRAN-77 functions that process input data and generate output data. Modules are high-level subroutine functions, for instance a complete convolution, contouring algorithm, 2D-FFT, or a flow field particle integration. Modules have graphic icons, and to create an application modules are linked together into groups, called a *network*. The connections represent the flow of data among modules, using a data flow execution scheduler. Each module can be on a separate machine, or on the same machine, or even in the same process as other modules. Modules may be simple transfer functions, or complex multi-pass integration routines, such computing the particle path through a 3D flow field, or take control and steer the AVS flow executive, to specify explicit synchronization, as required for keyframe animation. AVS has its origins on multiprocessor vector architectures, and is able to take advantage of closely coupled MIMD architectures (Siewiorek '91).

4 FLOWEXECUTIVE

The purpose of constructing a network of modules is to provide a data processing pipeline in which, at each step, the output of one module becomes the input of another. In this way, data can enter AVS, flow through the modules of a network, and finally be rendered on a display or stored outside AVS. This process requires that each module in the network be invoked at the appropriate time.

Overall network control in AVS is the responsibility of the AVS Flow Executive. As such, the Flow Executive is responsible for managing the execution order of the modules. The Flow Executive also supervises *data flow* between modules, keeping track of where data is to be sent and by what method to send the data.

The Flow Executive is based on the Data Flow Architecture, which has been used in computer science for many years. Data flow is quite different from the architecture found in standard monolithic applications. For instance, data flow does not have a global memory space nor does it have a single program counter. An operation in data flow (*i.e.* a module) is enabled if and only if all the required input values have been computed. Enabled modules consume input values and produce output values, which are sent to downstream modules that need these values. An application based on the data flow architecture does not introduce sequencing constraints other than those imposed by data dependencies.

AVS has extended the basic Data Flow architecture into a form called the Adaptive Flow Architecture. Adaptive flow is optimized for a multi-process architecture, and includes direct module data communication pipes, upstream or feedback paths, shared memory configurations, inplace memory operators, and reentrant module instancing.

5 PARALLEL MODULE EXECUTION

In order to take advantage of the growing number of multi-processor workstations and heterogeneous computing environments, AVS supports parallel module execution. As long as a module has no input dependencies with other modules that would be executing at the same time, modules will execute in parallel. This improves overall performance of AVS network execution and allows more efficient use of available computational resources. The opportunity for parallel module execution is identified by those modules inside a fork-join region, with no internal dependencies (Hockney, '88).

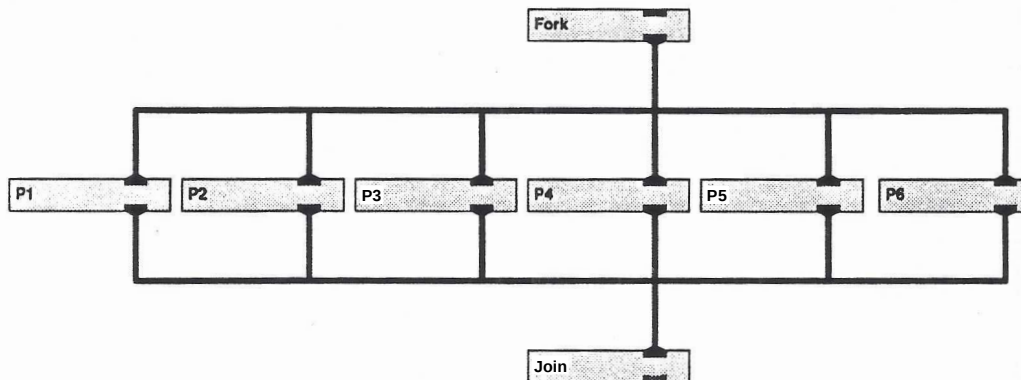


Figure 1, *Parallel Region Identification.*

6 REMOTE MODULE COMMUNICATION

AVS uses a remote procedure call mechanism to establish communications between modules that reside on different hosts. When the user starts up a module, AVS creates a new process in which that module runs. AVS sets up a connection between the module and the Flow Executive, using Berkely UNIX socket mechanisms. Both sides use remote procedure calls to communicate through this connection. On some MPP machines, notably the Meiko CS-2 and the Thinking Machines CM-5, where architecture specific high speed communication mechanisms exist, this transport layer of AVS is redefined to take advantage of the local resource.

Although each module process and data flow pipe can be directed to a remote processor, the Flow Executive remains the central controller, with knowledge of data dependencies and execution order. In this way execution order and data flow can be decoupled for optimal performance. The Flow Executive sends very small control packets to each remote process, queuing requests for execution or data transfer.

Any module can execute locally on the same host as the AVS program and flow executive, or on a remote system of the same or different type, as typical in a heterogeneous distributed network. Where each side of a data communication channel is a different class of system, data is sent through an XDR (external Data Representation) interface filter on each end.

7 SUPERCOMPUTER VERSIONS OF AVS

The AVS software system, with integral parallel system support is now available on all the major UNIX workstations, and supports most of the graphics options on these. AVS is also available for compute server configurations, using remote access over X-Windows. Versions of AVS are available or in progress on a number of supercomputer systems including:

Convex	C2/C3
Cray Research	YMP, Superservers
Intel	Multiprocessor Systems
Meiko	CS-2
Thinking Machines	CM-5
Kendall Square Research	KSR-1
Wavetracer	Tracer
Fujitsu	VP-xx
NEC	SX-3

The actual availability may vary from the above and should only be used as a general guide.

8 ILLUSTRATIONS OF PARALLEL DATA FLOW

8.1 Parallel Stock Option Price Model

Researchers at Syracuse University have developed a parallel implementation of several stock option price models, running on a heterogeneous network of workstations and MPP computers. The kernel of the model runs on a Connection Machine (CM-2 & CM-5), and a DECmpp-12000(sx), with an IBM-RS/6000 (GTO) as the front-end control and graphical presentation system. AVS is used both to control of remote parallel processes, and to manage the data flow to and from remote file server systems. Two orders of magnitude speed improvement over sequential workstation versions are reported (Cheng '92).

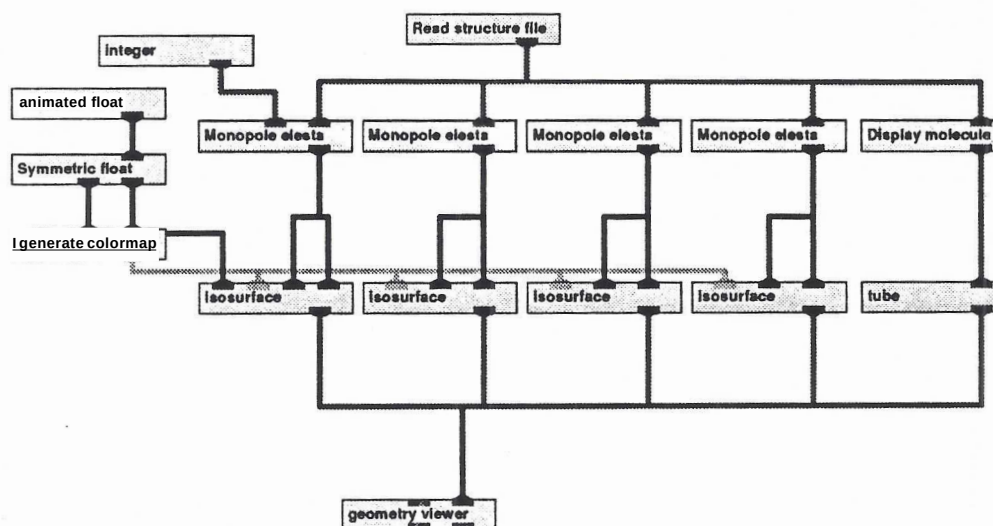


Figure 2, *Four Monopole Electostatic Charge Isosurfaces*

8.2 Parallel Electostatic Molecular Field Computation

The molecular data type and molecular analysis modules in AVS allow user defined energy functions over a structure file. By creating pairs of independent processing steps, a number of electrostatic field results and isosurface sampling operators can run in parallel on the same molecule, returning a 3D graphics display will all four results, including a ball and stick representation of the original structure (figure 2). The field calculation is dominated by local memory floating point calculations, while the isosurface is a mixture of floating point search, data structure traversal, and memory intensive geometric surface construction.

8.3 Parallel Edge Detection Image Analysis

An image analysis problem is shown where 12 variations of 2D spatial edge detection algorithms are compared, with input from the same source and a tiled window display shows each result. Each algorithm is a self-contained 2D operator, and can run independently on a separate processor. The original image feed can be identical as shown (figure 3), or sub-images from a high resolution source.

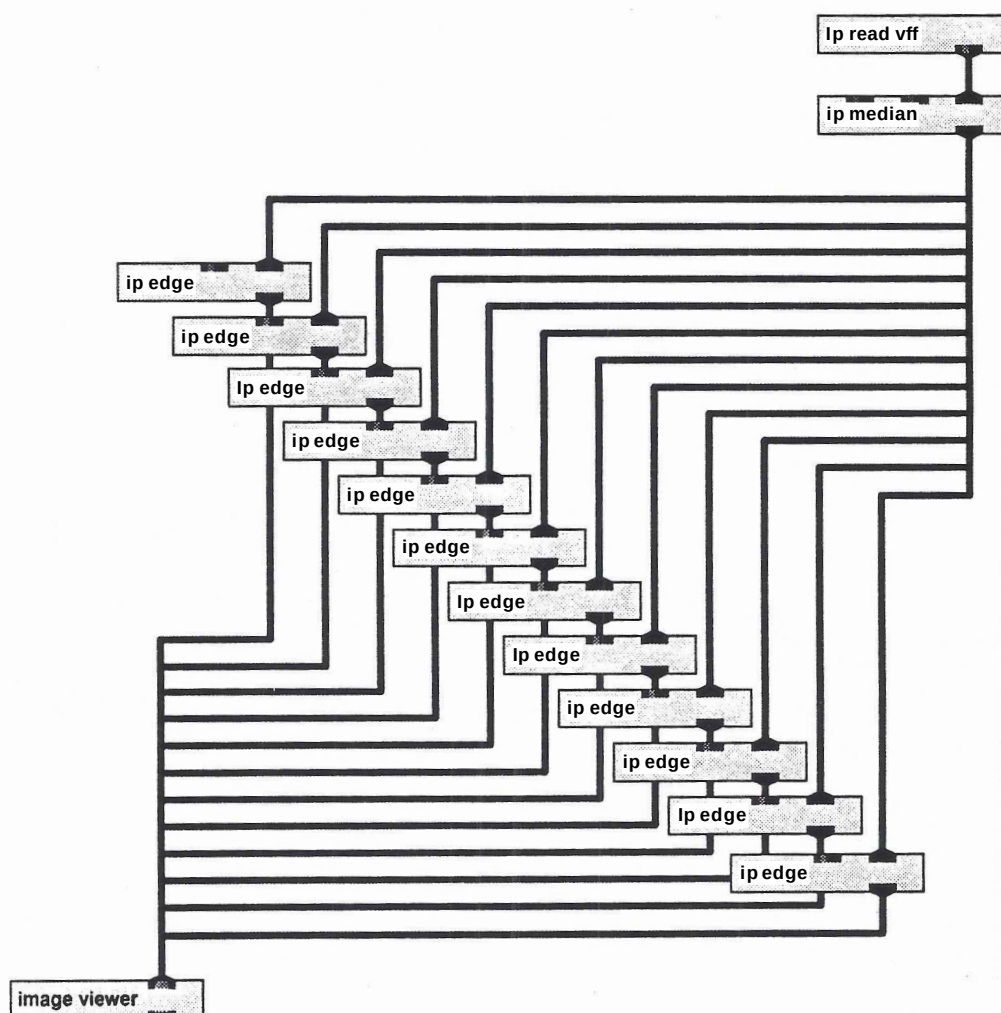


Figure 3, *Parallel Image Edge Detection.*

9 CONCLUSION

The specification and control of parallel applications requires complex strategies, and is often a non-intuitive aspect of advanced parallel architecture computing systems. The use of visual programming, with visual icons showing data dependencies, parallel fork-join regions, and visual feedback of data flow patterns enables the rapid prototyping and tuning of complex parallel applications. The AVS architecture meets this goal with support for networked heterogeneous workstations, and MPP supercomputers.

REFERENCES

- Cheng, G., Mills, K. and Fox, G.(1992) *An Interactive Visualization Environment for Financial Modeling on Heterogeneous Computing Systems*, Draft Report, Syracuse University, New York.
- Hockney, R.W. and Jesshope, C.R. (1988) *Parallel Computers 2*, Adam Hilger, Bristol.
- Siewiorek, D.P. and Koopman, P.J. (1991) *The Architecture of Supercomputers: Titan, a Case Study*, Academic Press, London.
- Staff, (1992) *AVS Technical Overview*, Advanced Visual Systems Inc., Boston.