Chapter 13

Data Visualization

Visualization, the graphic representation of data, has come to be an essential component of scientific computing. Visualization techniques range from a simple plotting of data points to sophisticated interactive simulations, but all allow researchers to analyze the results of their computations visually. One can literally "look at" the data to identify special areas of interest, anomalies, or errors that may not be apparent when scanning raw numbers. Visualization is often the only way to interpret the large data sets and complex problems common to the applications run on the Connection Machine system.

13.1 A Distributed Graphics Strategy

In keeping with its role as a network resource, the CM-5 uses a distributed graphics strategy to support a wide range of user applications. The key items in this strategy are

- the parallel processing power of the Connection Machine supercomputer
- the specialized power and interactive visualization environments, such as AVS, provided by dedicated graphics display stations
- the use of standard protocols, such as X11, to allow communication among a variety of hardware and software

A full range of interconnections is supported, from high-speed HIPPI interfaces through FDDI and Ethernet for longer-distance communications, to allow fast communication between the CM and graphics display stations.

Basically, the pattern is as follows: Computations carried out by the CM's parallel processing nodes manipulate data to create graphics primitives, which can then

Copyright © 1993 Thinking Machines Corporation

be sent to a graphics display station anywhere on the network. This strategy lets users maximize the value of existing hardware and software, while taking advantage of the computational speed and power of the CM, the high bandwidth of CM I/O, and the rendering power and speed of graphics workstations (such as those from Silicon Graphics and Sun), which implement many advanced rendering techniques in hardware and offer extensive visualization environments to make interactive rendering easy for the user.

Following this strategy, for example, a scientific visualization program can use the CM to compute image geometry (including, for example, polygon coordinates and color information) and then send it from the CM directly to local memory on the graphics workstation, where the results of simulations done on the CM can be displayed and analyzed interactively.

At the workstation, users benefit from the ability to create and use graphical user interfaces (GUIs). GUIs are widely used today and growing in popularity, as their use enhances productivity for applications programmers and users alike, allows tighter coupling of simulation and visualization, and allows such activities as simulation steering. Many tools exist for the creation of such interfaces, and all are now available to the CM programmer.

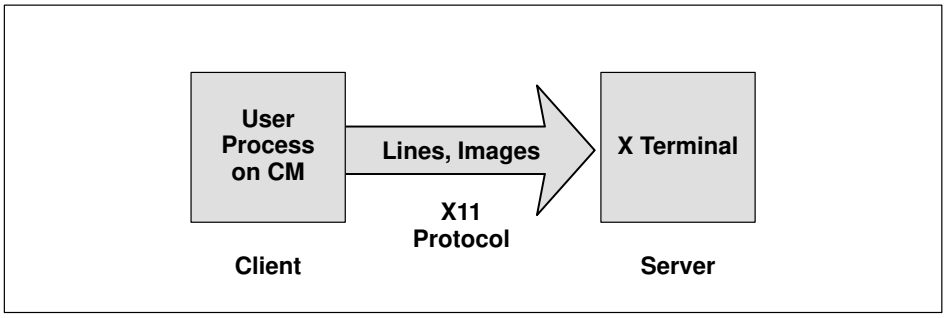

Figure 23. Distributed graphics using X11.

13.2 An Integrated Environment

By using the distributed graphics strategy described above, together with an underlying protocol such as X11 or an existing GUI, such as AVS, programmers can create and use a wide variety of integrated environments for their computational and visualization tasks. Connection Machine software provides an

November 1993
Copyright © 1993 Thinking Machines Corporation

environment that permits the exchange of very large data sets between the CM and framebuffers, workstations, or X window terminals.

13.3 The X11 Protocol

Support for the network-based X graphics protocol is integral to the CM distributed graphics strategy, since use of this protocol facilitates both data transfer and the use of GUIs, and allows considerable portability: data from a CM can be displayed on any X workstation.

But simple portability is not the only issue involved. As useful as graphics workstations are, the extra-large data sets typically used in CM applications frequently provide more data than such workstations can readily handle. The solution to this problem lies partly in using the CM's power to reduce the volume of information contained in the data sets so that the workstations can handle it rapidly, and partly in the successful integration of visualization environments, workstations, and high-speed framebuffers into a coherent system for rendering scientific data.

13.4 The CMX11 Library

The CMX11 library provides routines that allow the transfer of parallel data between the CM and any X11 terminal or workstation. The library is callable from CM Fortran and C*. It contains routines that draw text strings, polygons, and image-text strings; draw and fill points, lines, rectangles, and arcs; and draw and get images. The CMX11 library thus extends the X11 libraries by providing parallel network calls that use parallel variables instead of serial arrays. For example, where the X library offers an **`XDrawPoint`** routine, the CMX library offers **`CMXDrawPoint`**:

```
CMXDrawPoint(Display *display, Drawable *d,
             GC gc, int x, int y)
```

where *x* and *y* are pointers to parallel variables, and all other arguments are identical to the serial call.

Similarly, the CMX version of the X11 **`XPutImage`** routine uses the arguments and semantics of the original to provide a parallel transfer of an image that exists as a parallel array:

Copyright © 1993 Thinking Machines Corporation

```
CMXPutImage(display, d, gc, data, depth,
            src_x, src_y, dest_x, dest_y,
            width, height)
```

Note that no X protocol extensions are necessary, since the underlying CM socket mechanism makes the data source entirely transparent to the server. In most cases, the user simply makes the parallel version of the normal call, and the parallel data is inserted into the data stream in the same format and position as it would have been in the equivalent serial call. This greatly facilitates the user's task.

13.5 Visualization Environments — CM/AVS

CM/AVS is the first of the GUIs available on the CM-5. Other GUIs are expected to be adapted to the CM-5 in the future.

CM/AVS adapts and extends the Application Visualization System (AVS, from Advanced Visualization Systems, Inc.) to the realm of the CM-5. AVS provides a rich graphic programming environment in which a user builds a distributed visualization application. An application may involve diverse operations such as filtering, graphing, volume rendering, polygon rendering, image processing, and animation.

CM/AVS enables an application to operate on data that is distributed on CM-5 processing nodes and to interoperate with data from other sources. A user runs AVS normally on a local workstation and uses the modules and functions that CM/AVS provides to process data on the CM-5. That way, the advantages of user-interface–intensive workstation visualization are combined with the power of data-intensive CM-5 applications.

November 1993
Copyright © 1993 Thinking Machines Corporation

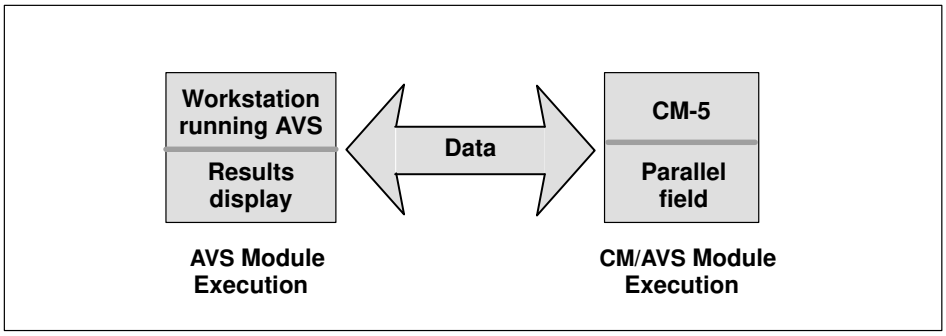

Figure 24. Distributed visualization.

The building blocks of an AVS application program are small, packaged units of code, called *modules*. Most modules process a set of inputs into a set of outputs. They provide functions such as volume rendering, isosurfacing, image processing, polygon rendering, fluid-flow visualization, graphing, statistical analysis, file I/O, and many others. Hundreds of visualization modules are available from AVS and Thinking Machines and in the public domain.

Modules are connected to form larger applications, called *networks*. In a network, information is passed between the modules using a small number of standard data types such as arrays and geometric objects. The small number of data types allows a wide variety of modules to be interconnected, allowing rich, custom environments to be created.

CM/AVS provides a parallel version of the AVS "field" data type. AVS fields are used to represent arbitrary arrays of data. CM/AVS's parallel field data is allocated on the CM-5 processing nodes as CM Fortran arrays or C* parallel variables.

In the AVS network, parallel fields appear identical to regular serial fields; the two may be used interchangeably. When CM/AVS modules that operate on parallel data are connected with AVS modules that operate on serial data, CM/AVS routines convert the data between parallel and serial fields as required. The conversion is transparent to the user and to the module writer.

Copyright © 1993 Thinking Machines Corporation