Subject: Re: problems plotting LARGE amounts of 2D data?
Posted by jworley on Thu, 24 Feb 1994 21:51:56 GMT
View Forum Message <> Reply to Message

Hi

michael> I am trying to find ways to plot LARGE amounts of 2D data,
michael> and I like to know what is currently the state of the
michael> art. For the sake of this posting, let's say "large" means
michael> much more data than fits in real memory. From my own
michael> experience, using the virtual memory of the workstation to
michael> store large amounts of data impedes performance due to
michael> excessive paging. Here is what I have been able to gather
michael> so far:

michael> 2) Khoros 1: loads everything into virtual memory. Any
michael> updates from Khoros 2.0?

We are actively trying to solve the large data set problem. It is a technically challenging problem because the problems posed by large data sets present different problems in different applications. For example, in an isolated application, the problem is pretty much limited to how to get around the virtual memory limitation efficiently. However, in other environments, such as Cantata (the visual programming language in Khoros), large data sets also pose problems because intermediate stages in a processing pipeline can quickly chew up any temp space that may be available.

The first issue is being addressed by data services. Data services is a data abstraction that provides read and write data in many file formats via an application programmers interface (API). This API provides a means of storing and retrieving data in units that are convenient to your application area. Data services is responsible for managing memory by caching only the portions of the data set that you are processing. Thus, only a reasonable portion of your data set is in memory at any given time. Data processing programs (including graphical applications) that are distributed with Khoros 2.0 will be written to data services. People who use Khoros as a development platform will be strongly encouraged to write their applications to data services as well.

The second issue, that of multiple copies is a byproduct of the intermediate stages of the data flow program. Each operator in the data flow program is written to read in an input set of data, perform some processing on the data, and then write out an output set of data. So, the problem of multiple copies is not really related to the data flow program, but rather related to the data flow operators. Ideally, if the data could be passed between operators in some serial fashion

(such as streams or sockets), intermediate copies of the data would not be needed. Unfortunately, data flow operators often can not be written to accept serial input and produce serial output, but rather require the ability to access data non-sequentially (an N-dimensional FFT is an example). These operators require the ability to randomly access the data input and output. As you have pointed out, this presents a significant problem when operating on large data sets.

Our approach to addressing this problem is to provide functionality for automatically buffering streamed data so that it can be accessed in a non-serial fashion. By addressing this problem in the infrastructure, we can guarantee that only a minimal number of temporary copies are present at any time (typically this is two copies per data pathway).

There are probably some direct ways of "getting around" this problem. Forcing everyone to write stream-processing routines is one approach. However we don't see this as a reasonable solution since many algorithms don't lend themselves to this type of interaction. Our goal is to abstract low-level issues away from the users who will be creating their own modules so that they can focus on the problem of implementing their algorithms without having to worry about working around limitations in their hardware and operating system. In the context of large data sets and data flow environments such as Cantata, the cost of doing this is increased overhead in terms of both temporary storage and performance. The objective is to minimize these costs while also minimizing the complexity of the system.

Hope we have been helpful.

Jeremy Worley, Steve Kubica, and the Khoros Group

Jeremy Worley jworley@khoros.unm.edu
The Khoros Group (505)837-6500
--

Jeremy Worley jworley@khoros.unm.edu
The Khoros Group (505)837-6500