Subject: Re: problems plotting LARGE amounts of 2D data?
Posted by thompson on Wed, 23 Feb 1994 20:20:02 GMT
View Forum Message <> Reply to Message

mcheng@dunlop.cs.wisc.edu (Michael Cheng) writes:

> In article <thompson.761937768@serts.gsfc.nasa.gov> thompson@serts.gsfc.nasa.gov (William Thompson) writes:

>> If a program such as IDL (or any of the others you mention) is putting things
>> into virtual memory, it can only be because you don't have enough real memory
>> available to you. The only thing you can do is buy more memory, or if your
>> operating system supports memory usage quotas (such as VMS) then you need to
>> increase your quotas.

> I argue that buying more memory is not always the best solution for the
> following reasons:
> 1) Some of us are poor. (arguably a weak reason)
> 2) Some data are always larger than the largest amount of memory
> reasonable amounts of money can buy.
> 3) Even a few medium sized data can overload real memory quickly.
> For example, working with five 25-meg data sets already requires 125 megs
> of memory.

> There has always been a mistmatch between virtual memory policy and
> the need to handle large amounts of data. This mismatch has been demonstrated
> time and again in database systems. I feel that this issue will come
> up again with respect to scientific data sets.

> So we go back to my original question: is there any software package
> for plotting large amounts of 2D data that does better than loading
> everything into virtual memory?

> Mike

Somebody else mailed me privately that he thought that you were probably talking about database management techniques, rather than memory management once data had already been read into arrays. I don't know how the other packages you mentioned work, but I can comment on how this applies to IDL.

IDL doesn't incorporate any kind of database management scheme, neither relational nor network nor object-oriented. It only reads in what you tell it to read in. Generally speaking you have to write an IDL routine to read your data files, although there is built-in support nowadays for certain kinds of commonly-used file formats such as FITS, HDF, CDF and netCDF. So, in that sense IDL does *not* read everything into memory. It reads in what you tell it, whether that's an entire file all at once or piece by piece, because you've written an old-fashioned program to do just what you wanted to do. You have

complete control and complete responsibility. :^)

On the other hand, if you wanted to read in a bunch of data and extract global properties from the data, then it would be easier to read the data into a single large array. For example if you had a 2048x2048 image that you wanted to display, then you would have to read that in as a single large array. Or if you wanted to read in 1000 X,Y traces of 500 points each and calculate the average trace, then it would in general be quicker to read those traces into a couple of big arrays than to process each trace individually in a loop. That's because loops are rather expensive in IDL.

I've certainly done things in IDL where I've handled large amounts of data by reading in the data bit by bit. For example, at one point I had a whole series of images and I wanted to get the mean and standard deviation as a function of pixel position. I didn't have enough memory to read all the data at once, so I read the images one-by-one and did a running calculation. Other situations is where I wanted to apply the same routine to a bunch of files. I just constructed a program with a FOR loop that read each file in turn and processed it separately. In general, though, it's more efficient to use as much memory as you can.

Bill Thompson

---