Subject: Re: Large Image Handling FAQ?
Posted by Earl F. Glynn on Wed, 10 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote in message ...
> Chunking is your friend. That is, operating on a "large" but not "too
> large" subsegments of your data at once

"Chunking" is exactly what the Microsoft "Terraserver" does --
supposedly with 1.01 terabytes compressed the world's largest
database:

http://terraserver.microsoft.com/

I' guessing, but it appears that a request for data is turned into
SQL database calls that pull the desired "chunks" together.

efg

_____
efg's Computer Lab: www.efg2.com/lab
efg's Tech Book Store: www.efg2.com/lab/TechBooks

Earl F. Glynn E-Mail: EarlGlynn@att.net
Overland Park, KS USA

Subject: Re: Large Image Handling FAQ?
Posted by Craig Markwardt on Wed, 10 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

"Andy Sowter" <asowter@synopticsga.freeserve.co.uk> writes:
>
> I'm dealing with many large (130MB is a typical size - 16-bit integers)
> images at a time, doing matching and 3-d reconstruction. Are there any FAQs
> or links out there relating to tips to do this - I don't like just ramping
> up my RAM or virtual memory every time because, well, it doesn't feel right
> (and it only seems to cause problems in the long run).?
>

I don't have a FAQ, but I do handle large files. In typical X-ray
astronomy applications we don't have big images, but we do have large
database files.

Chunking is your friend. That is, operating on a "large" but not "too
large" subsegments of your data at once. That way you can take
advantage of IDL's vector processing, but not overflow the memory. In
X-ray astronomy this is facilitated by the fact that data elements are
independent.

Below is a summary of the techniques I use for batch processing, in pseudocode. The technique depends on the situation.

Pixel-by-pixel

```
FOR I = 0, N_PIXELS-1 DO BEGIN
  X = READ_PIXEL(I)
  Y = OPERATE_ON(X)
  WRITE_PIXEL, Y, I
END
```
Advantage: low memory usage
Disadvantage: loop-intensive, not vectorized

Entirely vectorized

```
IMG = DBLARR(NX_PIXELS, NY_PIXELS)
READU, UNIT1, IMG
NEWIMG = OPERATE_ON(IMG)
WRITEU, UNIT2, IMG
```
Advantage: entirely vectorized
Disadvantage: may take too much memory, slows down with swapping

Chunked

```
N = N_PIXELS
WHILE N GT 0 DO BEGIN
  BUFSIZE = N < BUFMAX ;; BUFMAX is the chunk size
  X = READ_PIXELS()    ;; Read one chunk at a time
  Y = OPERATE_ON(X)
  WRITE_PIXELS, Y
  N = N - BUFSIZE
END
```
Advantage: low memory footprint, fast because it is vectorized
Disadvantages: more complicated code, doesn't work if chunks must interact with each other.

You may also want to look in to associated variables. I don't use them, but they may provide some virtual-memory-like features. There is also a virtual memory array package by Eric J. Korpela which may do the trick, but that requires some compiling and LINK_IMAGEing.

Good luck,

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@astrog.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: Large Image Handling FAQ?
Posted by Mathew Yeates on Wed, 10 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message