Subject: Re: Reading columns of binary data
Posted by JD Smith on Mon, 07 Aug 2006 21:54:44 GMT
View Forum Message <> Reply to Message

On Mon, 07 Aug 2006 10:12:16 -0700, Wayne Landsman wrote:

> This is probably more of a feature request than a question, though
> there is a chance the desired feature already exists within IDL.
>
> A FITS binary table might plausibly consist of 500 columns and 500,000
> rows of data in a fixed length binary format. To read the 32nd
> column there are 2 options:
>
> (1) Loop over the 500,000 rows, extracting the scalar value for
> the 32nd column for each row, and construct the 500,000 element ouput
> array
> (2) Read the entire 500,000 x 500 file into memory, and extract
> the 32nd column
>
> (In practice, one probably would use a hybird method of looping over an
> intermediate size buffer. Also note that an identical problem occurs
> when extracting every nth pixel from an extremely large image on disk.)
>
> I understand that the extraction of a column will never be as fast as
> reading a row of data, because the bytes to be read are not contiguous.
> But I am hoping that the heavy work can be done at a lower level than
> the IDL syntax.
>
> Erin Sheldon has recently written a C routine BINARY_READ linked to
> IDL via a DLM to efficiently read a binary column (
> http://cheops1.uchicago.edu/idlhelp/sdssidl/umich_idl.html#C CODE).
> (He also has routines ASCII_READ and ASCII_WRITE to do this for the
> less urgent problem of ASCII columns.) While I might adopt this
> routine, it would be nice for portability reasons if a DLM were not
> necessary. Say, a new keyword SKIP to READU
>
> IDL> a = fltarr(200)
> IDL> readu, 1, a, skip = 100
>
> to indicate to skip 100 bytes before reading consecutive elements.
>
> It appears that MATLAB already has a function FREAD to support reading
> columns of data.

But does it work in a vector fashion? Using POINT_LUN with READU in
IDL gives you the FREAD functionality, but requires you to loop
500,000 times (in your example). So, the only real problem you have
is the looping penalty (and it's a big problem).

I'd regard this as yet another example of why IDL needs a fast compiled "side-loop" primitive for generic looping operations which don't need the full conveniences of the interpreter loop (ability to hit Control-C to interrupt, etc.). I should be able to say:

```
a=fltarr(500000,/NO_ZERO)
f=0.0
for_noblock i=0L,500000L-1L do begin
  point_lun,un,i*100L
  readu,un,f
  a[i]=f
endfor
```

and not have it be 100x slower than the equivalent in C.

JD
